Conferences in Research and Practice in
Information Technology

Volume 168

Data Mining and Analytics 2015 (AusDM 2015)

,

Data Mining and Analytics 2015

Proceedings of the
Thirteenth Australasian Data Mining Conference
(AusDM 2015), Sydney, Australia,
8–9 August 2015

Kok-Leong Ong, Yanchang Zhao, Glenn Stone and
Md Zahid Islam, Eds.

Volume 168 in the Conferences in Research and Practice in Information Technology Series.
Published by the Australian Computer Society Inc.

Published in association with the ACM Digital Library.

Data Mining and Analytics 2015. Proceedings of the Thirteenth Australasian Data Mining Conference (AusDM 2015), Sydney, Australia, 8–9 August 2015

Conferences in Research and Practice in Information Technology, Volume 168.

Editors:

Kok-Leong Ong
La Trobe Business School
College of Arts, Social Sciences and Commerce
La Trobe University
P.O.Box 821, Wodonga Victoria 3689, Australia
Email: kok-leong.ong@latrobe.edu.au

Yanchang Zhao
Department of Immigration and Border Protection, Australia;
and RDataMining.com
5 Chan St
Belconnen, ACT 2617, Australia
Email: yanchang@rdatamining.com

Md Glenn Stone
School of Computing, Engineering and Mathematics
Western Sydney University
Locked Bag 1797
Penrith NSW 2751, Australia
Email: g.stone@westernsydney.edu.au

Md Zahid Islam
School of Computing and Mathematics
Faculty of Business
Charles Sturt University
Bathurst, NSW 2795, Australia
Email: zislam@csu.edu.au

Series Editors:
Vladimir Estivill-Castro, Griffith University, Queensland
Simeon J. Simoff, University of Western Sydney, NSW
Email:crpit@scem.uws.edu.au

Publisher: Australian Computer Society Inc.
PO Box Q534, QVB Post Office
Sydney 1230
New South Wales
Australia.

Conferences in Research and Practice in Information Technology, Volume 168.
ISSN 1445-1336.
ISBN 978-1-921770-18-0.

Document engineering by CRPIT, August 2015.

The *Conferences in Research and Practice in Information Technology* series disseminates the results of peer-reviewed research in all areas of Information Technology. Further details can be found at http://crpit.com/.

Table of Contents

Message from the General Chairs

On behalf of the AusDM 2015 organization committee, we are pleased to welcome you to the 13th Australasian Data Mining Conference, which will be held in Sydney, Australia.

The Australasian Data Mining Conference has established itself as the premier Australasian meeting for both practitioners and researchers in data mining. Since AusDM02 the conference has showcased research in data mining, providing a forum for presenting and discussing the latest research and developments. This year, the conference embodies a set of 17 papers, selected through careful and rigorous peer review by more than 40 Program Committee members as best of the submissions. It is our honour to have invited 12 most prominent scholars in the data mining area as our keynote speakers. Their dedication is highly appreciated.

None of this would have happened without the earnest efforts of the organizers behind the scenes. We had an excellent team that has worked very hard to organize AusDM 2015. First, we would like to thank Program Committee Co-Chairs: Zahid Islam, Kok-Leong Ong, Yanchang Zhao and Glenn Stone and their team of Program Committee members who have done an outstanding job in carrying out the paper review tasks. We are very grateful to the Steering Committee Co-chairs: Simeon Simoff and Graham Williams, and the Steering Committee members, including Peter Christen, Paul Kennedy, Jiuyong (John) Li, Kok-Leong Ong, John Roddick, Andrew Stranieri, Geoff Webb, and Yanchang Zhao, for their invaluable advisory roles. Special thanks to Jing Jiang, for her tireless efforts as the Local Chair. Last but not least, we thank our sponsor, QCIS at UTS, for the warm support. We must thank Chengqi Zhang, director of QCIS, for inviting the exceptionally strong team of keynote speakers.

Finally, we thank you all conference participants for making AusDM a success, and hope that you have an enjoyable and fruitful stay in Sydney.

Yours Sincerely,

Ling Chen
University of Technology Sydney

Richi Nayak
Queensland University of Technology

August 2015

Message from the Program Chairs

Welcome to the 13th Australasian Data Mining Conference, in Sydney, Australia.

A total of thirty nine (39) papers were submitted to the two conference tracks (research and industry). From these, each paper was rigorously reviewed by at least two reviewers and up to a maximum of four reviewers took part in providing an assessment of the papers' merits. After careful consideration, seventeen (17) papers were selected for inclusion in the final conference program.

Our Program Committee members have been pivotal to the success of this conference. Many have worked to provide timely reviews that are crucial to ensuring the success of the conference. On behalf of the entire organising committee, we express our appreciation to the committee for their cooperative spirit and extraordinary effort. Many members delivered every review requested, and more. It was a true privilege to work with such a dedicated and focused team, many whom were also active in helping with the publicity of the conference. We also wish to extend our appreciation to any of the external reviewers relied upon by the Program Committee members; they have played a part of making this conference possible.

Beyond the technical program in this proceedings, the conference has been enriched by many other items. These include the co-location with the 21st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) 2015 and the 2nd International Conference on Data Science (ICDS) 2015, and the availability of keynote speakers from both AusDM and ICDS conferences. We trust these programmes will provide insightful new research ideas and directions.

Lastly, we hope you enjoy the conference as much as we have enjoyed being part of delivering it.

Yours Sincerely,

Kok-Leong Ong
La Trobe University

Yanchang Zhao
Department of Immigration and Border Protection, Australia;
and RDataMining.com

Glenn Stone
Western Sydney University

Md Zahid Islam
Charles Sturt University

August 2015

Conference Organisation

General Chairs

Ling Chen, University of Technology Sydney
Richi Nayak, Queensland University of Technology

Program Chairs (Academic)

Md Zahid Islam, Charles Sturt University
Glenn Stone, Western Sydney University

Program Chairs (Industry)

Kok-Leong Ong, La Trobe University
Yanchang Zhao, Department of Immigration and Border Protection, Australia; and RDataMining.com

Sponsorship Chair

Andrew Stranieri, University of Ballarat

Local Chair

Yue Xu, Queensland University of Technology

Steering Committee Chairs

Simeon Simoff, Western Sydney University
Graham Williams, Australian Taxation Office

Steering Committee Members

Peter Christen, Australian National University
Paul Kennedy, University of Technology Sydney
Jiuyong Li, University of South Australia
Kok-Leong Ong, La Trobe University
John Roddick, Flinders University
Andrew Stranieri, University of Ballarat
Geoff Webb (advisor), Monash University
Yanchang Zhao, Department of Immigration and Border Protection, Australia; and RDataMining.com

Program Committee

Industry Track

Rohan Baxter, Australian Taxation Office
Edward Kang, Australian Customs and Border Protection Service
Clifton Phua, NCS Pte Ltd, Singapore
Ross Farrelly, Terradata ANZ, Datamilk
Ke Zhang, Department of Health and Ageing, Australia
Jin Li, Geoscience Australia
Kee Siong Ng, Pivotal
Yingsong Hu, Department of Human Services, Australia
Adriel Cheng, Defence Science and Technology Organization
Debbie Zhang, Australian Taxation Office
Wei Peng, Telstra Australia
Martin Rennhackkamp, PBT Group
Yogesh Nerurkar, Accenture (Digital, Data, Analytics)

Research Track

Xue Li, The University of Queensland
Lin Liu, University of South Australia
Ping Guo, Beijing Normal University
Xiaohui Tao, University of Southern Queensland
Md Anisur Rahman, Charles Sturt University
Adil Bagirov, Federation University of Australia
Paul Kwan, University of New England
Yee Ling Boo, RMIT University
Md Marwan Md Fuad, University of Tromsø
Robert Layton, University of Ballarat
Ting Yu, Transport for NSW
Guandong Xu, University of Technology Sydney
Brad Malin, Vanderbilt University
Christine O'Keefe, CSIRO Computational Informatics
Dinusha Vatsalan, Australian National University
Francois Poulet, University of Rennes 1 - IRISA
Xuan-Hong Dang, University of California at Santa Barbara
Yun Sing Koh, University of Auckland
Gang Li, Deakin University
Siamak Tafavogh, University of Technology Sydney
Sitalakshmi Venkatraman, Northern Melbourne Institute of TAFE
Mengjie Zhang, Victoria University of Wellington
Tom Osborn, University of Technology Sydney
Geng Li, ORACLE, USA
Md Geaur Rahman, Charles Sturt University
Junbin Gao, Charles Sturt University
Vinh Nguyen, Western Sydney University
Goce Ristanoski, NICTA
Peng Zhang, University of Technology Sydney

Additional Reviewers, Research Track

Samuel Fletcher
Md. Nasim Adnan
Michael Siers
Qing Huang

AusDM Sponsors

We wish to thank the following sponsors for their contribution towards this conference.

http://www.togaware.com

http://www.uts.edu.edu.au/research-and-teaching/our-research/
quantum-computation-and-intelligent-systems

http://www.uts.edu.au/

http://www.westernsydney.edu.au/

Keynotes

On Mining Heterogeneous Information Networks

Philip Yu
University of Illinois at Chicago, USA
psyu@cs.uic.edu

Abstract

The problem of big data has become increasingly importance in recent years. On the one hand, the big data is an asset that potentially can offer tremendous value or reward to the data owner. On the other hand, it poses tremendous challenges to distil the value out of the big data. The very nature of the big data poses challenges not only due to its volume, and velocity of being generated, but also its variety, where variety means the data can be collected from various sources with different formats from structured data to text to network/graph data, etc. In this talk, we focus on the variety issue and discuss the recent development in mining of heterogeneous information networks which can be applied to multiple disciplines, including social network analysis, World-Wide Web, database systems, data mining, machine learning, and networked communication and information systems. We will examine the problem of integration of multiple data sources using heterogeneous information network models. Fusion of multiple social networks will also be considered.

Big Data Algorithms and Clinical Applications

Yixin Chen
Washington University in St Louis, USA
chen@cse.wustl.edu

Abstract

In the era of big data, we need novel algorithms on top of the supporting platform. In this talk, I will first discuss some key aspects of big data algorithms in general. Then, I will talk about our recent medical big data project as a case study. Early detection of clinical deterioration is essential to improving clinical outcome. In this project, we develop new algorithms for clinical early warning by mining massive clinical records in hospital databases. The research focuses on the large population of patients in the general hospital wards, who are not in the intensive care units and suffer from infrequent monitoring. I will discuss the challenges this big data application poses to traditional machine learning and data mining algorithms, our recent progress, and the lessons we learnt. Promising results from a formal clinical trial at the Barnes-Jewish Hospital, the teaching hospital of the Washington University School of Medicine, will be discussed.

Algorithm Acceleration for High Throughout Biology

Wei Wang
University of California, Los Angeles, USA
weiwang@cs.ucla.edu

Abstract

High throughput sequencing technique has been demonstrated as a revolutionary means for modern biology because it provides deep coverage and base pair-level resolution. It produces vast amount of data which pose new computational challenges, because subsequent analyses often rely on a sequence alignment step that re-establishes the origin of each read, a process that is both time consuming and error prone. In this talk, we will present our latest accomplishment in algorithm advances that dramatically accelerate the analysis by removing the necessity of sequence alignment. We will demonstrate through a concrete example of RNASeq quantification, in which we are able to achieve two orders of magnitude speedup and deliver competitive accuracy.

Big Data for Everyone

Jian Pei
Simon Fraser University, Canada
jpei@cs.sfu.ca

Abstract

Big Data post grand opportunities and challenges for egocentric analytics on Big Data. In this talk, I will discuss several interesting problems centered on egocentric queries and analysis on Big Data. We want to answer a series of natural questions imperative in several killer applications, such as "How is this patient similar to or different from the other Type II diabetes patients in the database?", "How is University X distinct from the other universities?", and "How is this residential property distinct from the others available in the market?" To answer such questions on Big Data, we have to search data of high dimensionality and high volume, and possibly of high dynamics as well. I will present some preliminary research results and some application case studies we obtained recently, as well as more challenges we identified.

Big Data Mining and Data Science

Yong Shi
Chinese Academy of Sciences
yshi@ucas.ac.cn

Abstract

Big Data has become a reality that no one can ignore. Big Data is our environment whenever we need to make a decision. Big Data is a buzz word that makes everyone understands how important it is. Big Data shows a big opportunity for academia, industry and government. Big Data then is a big challenge for all parties. This talk will discuss some fundamental issues of Big Data problems, such as data heterogeneity vs. decision heterogeneity, data stream research and data-driven decision management. Furthermore, this talk will provide a number of real-life Big Data Applications. In the conclusion, the talk suggests a number of open research problems in Data Science, which is a growing field beyond Big Data.

.

Scaling Log-Linear Analysis to Datasets with Thousands of Variables

Geoff Webb
Monash University, Australia
geoff.webb@monash.edu

Abstract

Association discovery is a fundamental data mining task. The primary statistical approach to association discovery between variables is log-linear analysis. Classical approaches to log-linear analysis do not scale beyond about ten variables. By melding the state-of-the-art in statistics, graphical modeling, and data mining research, we have developed efficient and effective algorithms for log-linear analysis, performing in seconds log-linear analysis of datasets with thousands of variables and providing a powerful statistically-sound method for creating compact models of complex high-dimensional multivariate distributions.

Large Scale Metric Learning using Locality Sensitive Hashing

Ramamohanarao Kotagiri
University of Melbourne, Australia
kotagiri@unimelb.edu.au

Abstract

Metric learning tries discover mapping of features such that objects belonging a particular class each other in the new space. However, the current methods of discovering such matric mappings are computationally in feasible when the data set is huge with large number of features. My talk will describe the state of the art algorithms for metric learning. I will present our recent work on an efficient approach for discovering metric learning based mappings using Locality Sensitive Hashing (LSH). Our generic approach can accelerate state-of-the-art metric learning while achieving competitive classification accuracy, expanding feasibility by an order of magnitude. Our approach can accelerate Large Margin Nearest Neighbour (LMNN) to learn metrics on 1,000,000 samples in 3.6 minutes which is reduced from 5.8 hours.

Big Data Analytics in Business Environments

Hui Xiong
State University of New Jersey, USA
hxiong@rutgers.edu

Abstract

Recent years have witnessed the big data movement throughout all the business sectors. As a result, awareness of the importance of data mining for business is becoming wide spread. However, the big data are usually immense, fine-grained, diversified, dynamic, and sufficiently information-rich in nature, and thus demand a radical change in the philosophy of data analytics. In this talk, we introduce a set of scenarios for understanding and mining of business data in various business sectors. In particular, we will discuss the technical and domain challenges of big data analytics in business environments. The theme to be covered will include (1) the data mining problem formulation in different business applications; (2) the challenging issues of data pre-processing and post-processing in business analytics; (3) how the underlying computational models can be adapted for managing the uncertainties in relation to big data process in a huge nebulous business environment. Finally, we will also show some promising research directions.

Discovering Negative Links on Social Networking Sites

Huan Liu
Arizona State University, USA
huan.liu@asu.edu

Abstract

Social networking sites make it easy for users to connect with, follow, or "like" each other. Such a mechanism promotes positive connections and helps a social networking site to grow without direct belligerent or negative encounters. This type of one-way connections makes no distinction between indifference and dislike; in other words, two users have only, by default, positive connections. However, it is apparent that as one's network grows, some users might not be benevolent toward each other, or negative links could form, though not explicitly stated. In this talk, we assess the need for discovering such hidden negative links, explore ways of finding negative links, and show the significance of negative links in social media applications like data classification and clustering, recommendation systems, link prediction, and tie-strength estimation. *This presentation is based on Dr. Jiliang Tang's Doctoral Dissertation at ASU.

Resource Management in Cloud Computing Systems

Albert Y. Zomaya
The University of Sydney, Australia
albert.zomaya@sydney.edu.au

Abstract

The cloud is well known for its elasticity by leveraging abundant resources. Cloud data centres easily host thousands or even millions of multicore servers. Further, these servers are increasingly virtualized for the sake of data centre efficiency. However, the reality is that these resources are often relentlessly exploited particularly to improve applications performance. Although the elasticity facilitates achieving cost efficiency (or the performance to cost ratio), the ultimate efficiency in resource usage (or more broadly data centres) lies in scheduling and resource allocation strategies that explicitly take into account actual resource consumption. The optimization of resource efficiency in clouds is of great practical importance considering its numerous benefits in the economic and environmental sustainability. In this talk, we will discuss resource efficiency in cloud data centres with an example of large-scale distributed processing applications including scientific workflows and MapReduce jobs.

Defining Data Science

Yangyong Zhu
Fudan University, China

Abstract

In the age of big data, data science has become a hot occupation, supplanting traditional information science and big data engineering. This may indicate that data science has become its own branch of research. The term "data science" first appeared in CODATA Data Science Journal in 1990. So far, it has had several different interpretations. This talk aims to address what goals data science should seek to meet, and what data science itself is. We will present key connotations of data science: the first is the study of data itself. Its goal is to explore datanature and scientific issues related to datanature. The second is the study of the rules of the natural world as reflected by data, i.e., the study the natural world performed through the study of data.

.

Learning with Big Data by Incremental Optimization of Performance Measures

Zhihua Zhou
Nanjing University, China
zhouzh@nju.edu.cn

Abstract

A popular approach to achieve a strong learning system is to take the performance measure that will be used for evaluation as an optimization target, and then accomplish the learning task by an optimization procedure. Many performance measures in machine learning, however, are unfortunately non-linear, non-smooth and non-convex, leading to difficult optimization problems. With big data, the optimization becomes even more challenging because of the concerns of computational, storage, communication costs, etc. Particularly, it becomes almost impossible to collect all data at first and then perform optimization, and it is desired to be able to optimize performance measures incrementally, without accessing the whole data. In this talk we will introduce some studies along this direction.

Contributed Papers

On Ranking Nodes using kNN Graphs, Shortest-paths and GPUs

Ahmed Shamsul Arefin[*+] **Regina Berretta** **Pablo Moscato**[*]

Centre for Bioinformatics, Biomarker Discovery and Information-Based Medicine,
School of Electrical Engineering and Computer Science,
Faculty of Engineering and Built Environment,
The University of Newcastle, Callaghan, NSW 2308, Australia
Email: Ahmed.Arefin, Regina.Berretta, Pablo.Moscato}@newcastle.edu.au
[+]Currently at the ICT Services, University of Southern Queensland, Qld [*] Contact author

Abstract

In this paper, we present graphics processing unit (GPU) based implementations of three popular shortest-path centrality metrics- closeness, eccentricity and betweenness. The basic method is designed to compute the centrality on gene-expression networks, where the network is pre-constructed in the form of kNN graphs from DNA microarray data sets. The relationship among the genes in the kNN graph is determined by the similarity of their expression levels. The proposed method has been applied to a well known breast cancer microarray study and we highlighted the correlation of the highly ranked genes to the time to relapse of the disease. The method is readily applicable to other datasets, where the data points can be recognised in a multidimensional space. It can be applied to other networks (e.g., social networks, the Internet, etc.) with minimal modifications.

Keywords: Shortest paths, breadth first search, centrality, kNN, CUDA, microarrays, gene-expression.

1 Introduction

Centrality analysis measures the relative importance of the elements in a given network based on their connectivity within the network structure. In other words, centrality measures help to rank the network elements according to their importance within the network structure. Formally, the centrality of a network is defined as follows (Junker et al. 2006), let $G(V, E)$ be a directed or undirected graph (network), then the centrality on G is defined as a function $C : V \rightarrow \mathbb{R}$ that assigns a real number to each vertex. For a pair of two vertices, u and v, if $C(u) > C(v)$, one can say that u is more central than v. Although many of the popular centrality metrics are actually originated from the classical analysis of *social networks*, now they have successfully been investigated on many other practical networks, e.g., the Internet (Page et al. 1998, Gkorou et al. 2011), public transport networks (Kazerani & Winter 2009), power grid network (Jin et al. 2010), biological networks (Potapov et al. 2005, Bader & Madduri 2008), etc. A brief review of the existing centrality metrics and their applications can be found in (Junker & Schreiber 2011, Newman 2010). The main problem with many of the metrics is that their sequential implementations can often become very time consuming. For instance, the betweenness centrality computation of all nodes in a graph requires $\mathcal{O}(n^3)$ time with *Floyd-Warshall* algorithm, so for network with 1M nodes, a sequential method may take decades of computation on a general purpose computer. Even though there exists some faster approximate methods (Jacob et al. 2005, Eppstein & Wang 2001), their high error rates on larger networks can severely limit their applicability (Jia et al. 2008).

One feasible way to compute the centrality of such large-scale networks would be to parallelize the computation and interestingly, a number of parallelization approaches for such purpose have already been developed. Some of them are quite fast and scalable, but unfortunately, require highly sophisticated and expensive computer systems with parallel processing capabilities. For instance, Bader and Madduri (Bader & Madduri 2006) implemented several parallel shortest path based metrics using shared memory multiprocessors on CRAY MTA-2. Later, Madduri et al. (Madduri et al. 2009) presented a refinement of the same work by proposing a *lock-free* variant on CRAY-XMT system (Mizell & Maschhoff 2009). Jin et al. (Jin et al. 2010) utilized the same system for computing the betweenness of power grid contingency measurements utilizing the same set of algorithms. Edmonds et al. (Edmonds et al. 2010) presented a set of distributed memory algorithms for computing centralities using cluster computers with at least 100's of compute nodes. Alternatively, there exist a few GPU implementations, which can be considered as relatively inexpensive approaches. However, a common problem with these implementations is their relatively lower scalability, when compared with the CPU based parallel counter-parts.

In this work, we present fast methods for computing three shortest-path based centrality metrics, closeness, eccentricity and betweenness. We apply the methods on gene-expression networks constructed from DNA microarray data sets. We use a GPU-based fast and scalable method (*GPU-FS-kNN*) for constructing the network. The proposed centrality computation methods are adapted from the sequential Breadth First Search (BFS) and Brandes's (Brandes 2001) shortest-path computation method for a given graph.

2 Literature Review

The shortest path based centrality metrics are usually implemented by using the basic path finding algorithms (e.g., single source source shortest paths (SSSP), all-pair shortest paths (APSP), etc.). Along with the super-computer based implementations (as discussed above), there exist a few GPU implementations of the basic graph traversal algorithms (e.g.,

as breadth first search (BFS), Single Source Shortest Path (SSSP) (Harish & Narayanan 2007)), which can be used as building blocks for computing the known shortest path based centralities. In addition, Sriram et al. (Sriram et al. 2009) proposed the first GPU-based implementation of the shortest path betweenness centrality that "*parallelize the BFS wavefront from different source nodes at different thread blocks*". However, this so-called *node parallel* approach performs sub-optimally when some nodes have more neighbors than the others. For instance, if a block has less workload and finishes early, it must stay idle until all others are finished, which can lead to serious load-imbalances. Jia (Jia 2010) solved this problem by proposing an *edge-parallel* and achieved a comparatively better parallelism by "*exploiting neighbors of each wavefront nodes in parallel*". However, similar to (Sriram et al. 2009), they not only perform the BFS from different source nodes on different thread blocks but also duplicates several data structures to each of these blocks. This data replication severely limits the number of thread blocks that can be launched at a time and hence, their implementation can run with only atmost 10 – 30 CUDA blocks (the number of standard SMs in the current series of GPUs). To date, the best optimized GPU implementations of the shortest path based centralities are proposed by Shi and Zhang (Shi & Zhang 2011). Even though their implementations are the adaptations of those in (Jia 2010), they eliminated the need for data duplication (to each block) by maintaining a two dimensional flag matrix which keeps track of the traversed paths. However, their approach is still unscalable to very large-scale networks, as it requires duplicated edge lists (on device) to maintain the neighborhood structures. There are a few other approaches developed more recently (e.g., see (McLaughlin & Bader 2014)) that are relatively more scalable but at the expense of hundreds of GPUs and more expansive hardware setup.

3 Proposed Centrality Computation Method

The proposed centrality computation method is designed to identify central elements in DNA microarray gene-expression data sets. It works in two steps, first it constructs a k-nearest neighbor (kNN) graph from a given multi-dimensional (gene-expression) data set. Next, it computes the centrality metrics on that graph.

3.1 Construction of the kNN Graphs from Gene-expression Data Sets

The k nearest neighbor (kNN) graph is an important graph structure where each node is connected to its k nearest or, closest nodes and the closeness is defined by a distance *metric*. The computation of a *distance matrix* is a fundamental task in constructing a kNN graph, if it is not provided as the input. Although there exist several methods that do not require the complete (or half, as symmetric) distance matrix, they suffer from a different problem called *curse of dimensionality* (e.g., see *kd-Trees*). From gene-expression data sets, the kNN graphs can be constructed in many ways, e.g., by using exhaustive search techniques, such as brute-force kNN search. The basic construction is quite simple, for a set of data elements (in a given metric space) the kNN graph can be produced by creating an edge from each element to its k nearest elements.

However, the computation of the distances of the nearest neighbours for large-scale instances becomes very slow on general purpose computers. Fortunately, the nearest neighbours of each vertex can be computed and searched independently and hence, the brute force approach is highly parallelizable.

It may be noted that the most common problem with the existing GPU-based brute force kNN algorithms are two-fold, firstly, they can only work if all the distances between query and reference points, i.e., the distance matrix, can fit into GPU's in-memory (e.g., see (Garcia et al. 2008)); secondly, they assume that the value of k is relatively small in comparison with the instance size (Liang et al. 2009). In contrast, we utilized a scaled and parallelized variant of the simple brute force kNN algorithm that is implemented using a chunking-based approach called GPU-FS-kNN. It can efficiently utilize GPUs and can handle instances with more than one million objects and fairly larger values of k (e.g., tested with k up to 64) on a single GPU. On multiple GPUs, if data partitioning is applied, then the method is capable of handling much larger instances and higher dimension sizes. Details of our GPU-based kNN graph construction method can be found in (Arefin et al. 2012*b*) and its other applications in (Arefin et al. 2012*c*,*a*,*d*, 2013).

3.2 Representation of Graphs

Whilst the graph representation methods on the CPU have been studied extensively, many of these methods are not suitable for GPUs. The GPUs have limited on-board memory and their memory access patterns are completely different from CPU. Traditional methods, such as adjacency matrix, stores a graph $G(V, E)$ using $\mathcal{O}(|V|^2)$ space. Using this approach, a graph with millions of vertices will require terabytes of storage memory. Therefore, adjacency matrix-based representations are infeasible on GPUs. In contrast, adjacency lists are more relaxed and can store graphs using much less memory ($\mathcal{O}(|V| + |E|)$). There exist several approaches for storing graphs on GPUs, for instance, adjacency list based (Harish & Narayanan 2007, Leist et al. 2009) and the traditional adjacency matrix (Katz & Kider 2008), but by accelerating it using the device shared memory.

In this work, we store our kNN graph (noted by Gk) in a single dimensional array of edge structure, where the basic structure has at least 3 members {*source*, *target* and *weight*}. There may be some additional members of the structure, depending upon the problem in question. All the graphs maintain the following two properties, to facilitate our algorithm developments. First, edges are sorted by the source vertex indices, then by weights. Second, each distinct source vertex has exactly k neighbors and hence, source vertex id changes exactly after k edges. We store our the graphs in the device global memory, which is slower but much larger than any of the other device memory types. Therefore, a higher scalability in terms of input is expected. Moreover, for very large graphs, the proposed structure allows to load graphs from the host to device part by part (i.e., by chunks). Furthermore, for extremely large graphs, an external memory approach (Vitter 2001) may be used to store the chunks of the graph into the external hard-drives and then load back to host and subsequently to the device memory.

Algorithm 1: Single Source Shortest Path Algorithm

Input : Gk, s, k;
Output: $dist$ array is initialized;
1 **Initialize** $dist[v] \leftarrow -1, \forall v \in V$;
2 $dist[s] \leftarrow 0$;
3 $flag \leftarrow$`true`; $d \leftarrow 0$;
4 **while** $flag =$`true` **do**
5 **host** $\rightarrow$ **device** $(flag, d)$;
6 **BFS** $(Gk, dist, d, flag, n, k)$;
7 **device** $\rightarrow$ **host** $(flag, d)$;
8 $d \leftarrow d + 1$;
9 $flag \leftarrow$`false`
10 **device** $\rightarrow$ **host** $(dist)$;
11 **return** $dist$;

Algorithm 2: BFS (Parallel)

Input : $Gk, dist, d, flag, n, k$;
Output: $dist$ array is initialized;
1 $tid \leftarrow$ thread id;
2 **if** $tid < n \times k$ **then**
3 $source \leftarrow Gk[tid].source$;
$target \leftarrow Gk[tid].target$;
4 /* **Path discovery** */
5 **if** $dist[source] = d$ **then**
6 $found \leftarrow$`true`;
7 **else if** $dist[target] = d$ **then**
8 **swap**$(target, source)$ `// device function`;
9 $found \leftarrow$`true`;
10 **else** $found \leftarrow$`false`;
11 /* **Path Traversal** */
12 **if** $found =$`true` **then**
13 **if** $dist[target] = -1$ **then**
14 $dist[target] \leftarrow d + 1$;
15 $flag \leftarrow$`true`;

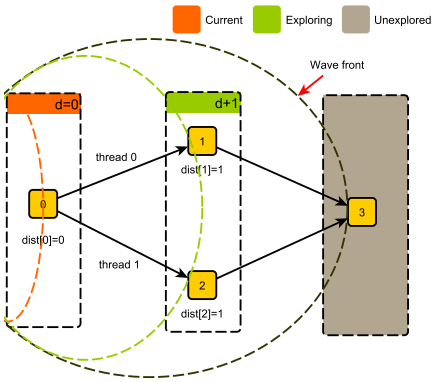
3.3 Single Source Shortest Paths (SSSP) for the kNN graphs

Given a graph $G(V, E)$ and a source vertex $s \in V$, the SSSP problem finds the shortest paths (or geodesic distance) from s to every other vertex $v \in V$. There exist several algorithms (e.g., *Dijkstra's*, *Bellman-Ford*, etc.) to compute the SSSP on directed graphs. We consider that the input (Gk) as unweighted, therefore we simply extend the BFS algorithm.

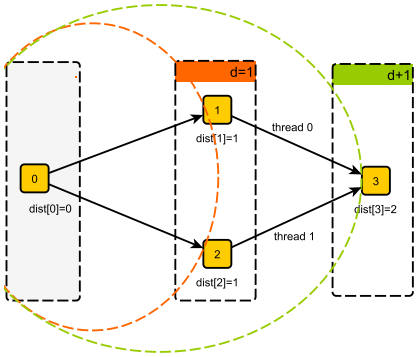
Table 1: Variables and arrays used in the shortest path based centralities.

Names	Descriptions
d	The shortest path distance
$dist$	An array to hold the distance from the source s to each node $v \in V$
σ	An array to hold the number of shortest paths from the source s to each node $v \in V$
δ	An array to hold the dependency of the source s on each node $v \in V$
P	A bit matrix to track the traversed paths
$flag$, $found$	Boolean flags

The data-parallel implementation of the SSSP problem presented in Algorithm 1 (variables are explained in Table 1, see also (Arefin 2013)) is an adaption of the parallel BFS algorithm presented in (Shi & Zhang 2011). However, there is a difference between their graph representation and ours. To facilitate the graph traversal in both directions (i.e., forward and backward, which is required by the betweenness centrality) they keep each edge twice using two longer arrays that maintain the one-to-one neighborhood correspondence with each other. In contrast, we achieve the same goal, but without performing any data replication. We incorporate a simple modification in the parallel BFS Algorithm (line 4 – line 10, Algorithm 2), so that we can use the kNN graph straight into the kernel. Each thread checks both ends of each kNN edge and investigate if any of them is in the current *wavefront* (BFS level) (see Figure 1). If the *source* node is in the current level, then the respective thread updates the distance of the *target*. Otherwise values of *source* and *target* are swaped and the corresponding distances are updated. It reduces the workload balance and hence, the execution times may slightly increase, but we consider this as a trade-off between the space and run-time complexity of the algorithm.



(a) Pass 1



(b) Pass 2

Figure 1: An illustration of the parallel BFS and SSSP algorithm on a network of four vertices, where node 0 is considered as the source vertex (i.e., $s \leftarrow 0$ and $d \leftarrow 0$). (a) First launch of the BFS kernel, thread 0 and 1 explore the node 1 and 2, respectively (in parallel). The respective locations in the $dist$ array are initialized by the distance, $d \leftarrow 1$ (b) Second launch of the kernel, now both threads explore node 3 in parallel and initialize the respective location in $dist$ by the node distance $d \leftarrow 2$.

Algorithm 3: k**NN Closeness Centrality**

Input : Gk, k;
Output: Closeness centrality of each vertex stored in cc array;

```
1  Initialize dist[t] ← −1, ∀t ∈ V;
2  flag ←true; d ← 0;
3  foreach s ∈ V do
4  |  dist[s] ← 0;
5  |  while flag =true do
6  |  |  host → device (flag, d);
7  |  |  BFS (Gk, dist, d, flag, n, k) ;
8  |  |  device → host (flag, d);
9  |  |  d ← d + 1;
10 |  |  flag ←false
11 |  device → host (dist);
12 |  Accumulate_Closeness (dist, n, s, cc);
13 return cc;
```

Algorithm 4: Accumulate_Closeness

Input : $dist, n, s, cc$;
Output: Closeness centrality of s is accumulated in $cc[s]$;

```
1  sum ← 0;
2  c = n;
3  for i ← 0 to n do
4  |  if dist[i] = −1 then c ← c − 1;
5  |  ;
6  |  else sum ← sum + dist[i];
7  |  ;
8  |  if c = 0 then cc[s] ← 0;
9  |  ;
10 |  else cc[s] ← ((c − 1)^2/(n − 1))/sum;
11 |  ;
```

3.4 Closeness Centrality

The **closeness centrality** measures the closeness between a pair of nodes in terms of their shortest path distance, in other words, for each node it computes the reciprocal of the sum of all pairwise distances within the network (Sabidussi 1966).

Therefore, for a given network and a source node s, closeness centrality is computed as follows,

$$cc(s) = \frac{1}{\sum_{t \in V} dist(s,t)} \tag{1}$$

The closeness centrality can only be applied to connected networks, yet it has a wide range of applications (Ma & Zeng 2003, Yang & Zhuhadar 2011). We compute the centrality by simply extending the proposed GPU-based SSSP algorithm (Algorithm 1). The modified algorithm is demonstrated in Algorithm 3 (k**NN Closeness Centrality**). The idea is to run the BFS kernel from each node in each separate iteration (line 3 – line 11, Algorithm 3) and subsequently, accumulate the closeness. For a given source node, the accumulation kernel (Algorithm 4) sums all the distances stored in *dist* array (computed by the BFS kernel) and finds closeness.

3.5 Eccentricity Centrality

The **eccentricity centrality** (Harary et al. 1965) computes for every vertex, the reciprocal of the maximum (shortest path) distance to all other nodes. For

Algorithm 5: Accumulate_Eccentricity

Input : $dist, n, s, ec$;
Output: Eccentricity centrality of vertex s is accumulated in $ec[s]$;

```
1  max ← −1; e = n;
2  for i ← 0 to n do
3  |  if dist[i] = −1 then e ← e − 1;
4  |  ;
5  |  else if dist[i] > max then max ← dist[i];
6  |  ;
7  if e = 0 then ec[s] ← 0;
8  ;
9  else ec[s] ← ((e − 1)^2/(n − 1))/max;
10 ;
```

a given network and a source node $s \in V$, it is defined as,

$$ec(s) = \frac{1}{\mathbf{max}\{dist(s,t) : t \in V\}} \tag{2}$$

This can be computed by updating the accumulation kernel using Algorithm 5 in Algorithm 3.

3.6 Betweenness Centrality

The **(shortest path) betweenness centrality** (Freeman 1977) counts the number of communications a vertex can monitor, or in other words, the rate of shortest paths experienced by an interior vertex. A node becomes more central not only for being on many shortest paths, but also on most of the shortest paths. It is based on a notion of *pair-dependency* (δ_{uv}(s)), which measures the fractions of shortest paths between u and v passing through the node s,

$$\delta_{uv}(s) = \frac{\sigma_{uv}(s)}{\sigma_{uv}} \tag{3}$$

The betweenness centrality of s is computed by summing up all the pairwise dependencies,

$$bc(s) = \sum_{u \neq s \neq v \in V} \delta_{uv}(s) \tag{4}$$

As the metric involves computation of shortest paths between all pairs of vertices, a straightforward implementation may require $\mathcal{O}(n^3)$ computations (e.g., using *Floyd-Warshall* algorithm). However, for a graph without loops or multiple edges, Brandes (Brandes 2001) proposed a *dynamic programming* approach that can reduce the search space and significantly lower the computational complexity. For a graph with n nodes and $|E|$ edges, the metric can be computed in $\mathcal{O}(n|E|)$ (unweighted) $\mathcal{O}(n|E| + n^2 \log n)$ (weighted) times. The author introduced a notion of *dependency* (in contrast to the pairwise dependency in (Freeman 1977)) of a vertex $u \in V$ on a single vertex $s \in V$, as follows,

$$\delta_{u\bullet}(s) = \sum_{v \in V} \delta_{uv}(s) \tag{5}$$

Which always obeys a recursive relation. For instance, the dependency of $u \in V$ on any $s \in V$ obeys following relation,

$$\delta_{u\bullet}(s) = \sum_{w : dist(u,w) = dist(u,s)+1} \frac{\sigma_{us}}{\sigma_{uw}} (1 + \delta_{u\bullet}(w)) \tag{6}$$

Therefore, following Brandes (Brandes 2001) the betweenness centrality of a node s can be computed by summing up all the dependencies as follows,

$$bc(s) = \sum_{u \neq s \in V} \delta_{u\bullet}(s) \quad (7)$$

This approach is the fastest known technique for computing the betweenness centrality. However, its sequential variant is still too costly for the large graphs. Therefore a number of CPU-based parallel variants are proposed so far and many of them require highly expensive hardware setup (discussed earlier). In contrast, we propose two data-parallel variants of the Brandes's approach on GPUs. Same as before, we consider that the input is an unweighted kNN graph and the edges are sorted by the source vertices.

3.6.1 Parallel Brandes's Betweenness Centrality 1

There are two major components of the Brandes's algorithm (Brandes 2001). A **SSSP component** that keeps track of the number of shortest paths (σ) arriving at each node and the predecessors ($Pred$), where the *predecessors* of a node is a group of adjacent nodes that are on the shortest paths to the node. An **accumulation component** that computes the dependency (δ) of each predecessor node on the source node (s) and sums the betweenness centrality from the dependencies (Equation 7).

Table 2: Variables in the Brandes's betweenness centrality (See (Brandes 2001))

Names	Descriptions
Q	A queue (initially empty)
S	A stack (initially empty)
$Pred[v]$	A list of predecessors on shortest paths from source
p, q	Two arrays of size k

In our naïve parallelization, we only parallelize the **BFS portion** of the SSSP component (Algorithm 6). Our aim is to explore only the neighbors of each source node in parallel, but process (i.e., identify the respective shortest paths and predecessors) the sources sequentially. Duplicate edges are not considered during this exploration process and hence, they are removed apriori. The adapted method is presented as Algorithm 7, which we term as **Parallel Brandes's Betweenness Centrality 1**. We call it naïve parallel, as it parallelizes the centrality computation partially and in an unoptimized way. Detailed working procedure of the method can be found in (Brandes 2001). Since, it utilizes the data structures proposed in the original approach (see Tables 1 and 2), it is applicable to both **directed** and **undirected graphs**.

3.6.2 Parallel Brandes's Betweenness Centrality 2.

To improve the performance of the naïve parallel implementation, we adapt the edge parallel approach proposed in (Shi & Zhang 2011). Same as in the closeness and eccentricity centrality, our SSSP computation takes kNN graphs straight as the input and does not require any lengthy pre-processing of neighbors (as required by the methods (Jia 2010, Shi & Zhang 2011)). However, we perform a slight modification in the **BFS kernel** (Algorithm 2) of the SSSP

Algorithm 6: Naïve Parallelization of BFS in Brandes (Brandes 2001)

Input : $Gk, dist, \sigma, p, q, v, k$;
Output: σ, p and q arrays are initialized;
1 $tid \leftarrow$ thread id;
2 $start \leftarrow v \times k$; $limit \leftarrow start + k$;
3 **if** $tid \geq start$ **and** $tid < limit$ **then**
4 $i = tid - start$;
5 $w \leftarrow Gk[tid].source$;
6 **if** $dist[w] < 0$ **then**
7 $dist[w] = dist[v] + 1$;
8 $q[i] \leftarrow w$; // keep the visited nodes
9 **if** $dist[w] = dist[v] + 1$ **then**
10 atomicAdd($\sigma[w], \sigma[v]$);
11 $p[i] \leftarrow w$; // keep the predecessors

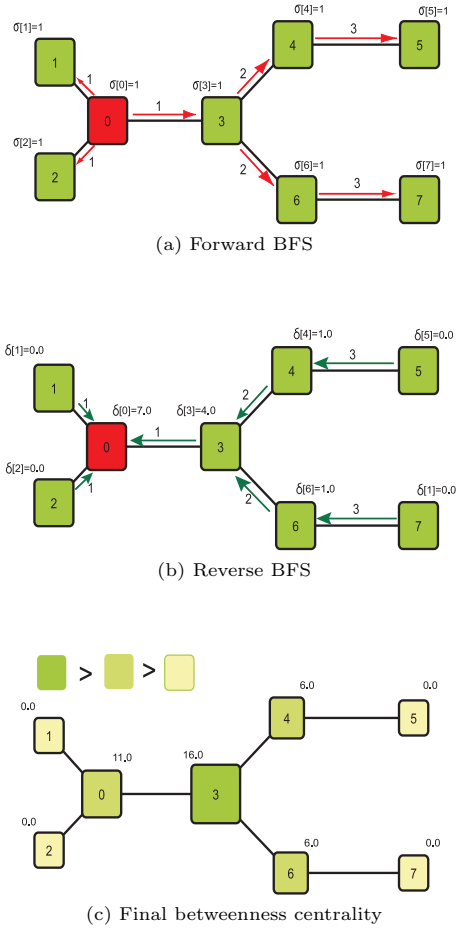
Algorithm 7: Parallel Brandes's (Brandes 2001) Betweenness Centrality 1

Input : Gk and k;
Output: Betweenness centrality of each vertex stored in bc array;
1 **Create** $Q, S, Pred, \sigma, \delta$;
2 **foreach** $s \in V$ **do**
3 **Initialize** $dist[t] \leftarrow \infty$, $\sigma[t] \leftarrow 0 \ \ \forall t \in V$;
4 $dist[s] \leftarrow 0$; $\sigma[s] \leftarrow 1$;
5 $Q.push(s)$;
6 **while** $Q \neq \varnothing$ **do**
7 $p[i], q[i] \leftarrow 0, \forall i \in k$;
8 $v \leftarrow Q.pop()$;
9 $S.push(v)$;
10 **host** $\rightarrow$ **device** (p, q);
11 **BFS** $(Gk, dist, \sigma, p, q, v, k)$ (Algorithm 6) ;
12 **device** $\rightarrow$ **host** (p, q);
13 $Q.push(q[i]), \ \ \forall i \in k$;
14 $p[i]$ **append** $\rightarrow Pred(v) \ \ \forall i \in k$;
15 **device** $\rightarrow$ **host** (σ);
16 **Initialize** $\delta[v] \leftarrow 0, \ \ \forall v \in V$;
17 **while** $S \neq \varnothing$ **do**
18 $w \leftarrow S.pop()$;
19 **for** $v \in Pred[w]$ **do**
20 $\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[w]} \times (1 + \delta[w])$;
21 **if** $w \neq s$ **then** $bc[w] \leftarrow bc[w] + \delta[w]$;
22 ;
23 **return** bc;

computation, as now we need to store the number of shortest paths (σ) arriving at each node and moreover, we need to keep track of the traversed paths. We add the following four lines at the end of the BFS kernel. The bitwise operations are explained in (Arefin et al. 2013).

if $dist[target] = d + 1$ **then**
  $bit = target \times n + source$;
  atomicOr(P+BIT_POS(bit), BIT(bit));
 /*P[source][target]=1*/
  atomicAdd($\sigma[target], \sigma[source]$)
 /*σ[target]+=σ[source]*/

The Brandes's algorithm requires a **backtrace** procedure (referred as *back propagation* in (Shi & Zhang 2011, Jia et al. 2012)) for accumulating the de-



(a) Forward BFS

(b) Reverse BFS

(c) Final betweenness centrality

Figure 2: Demonstration of BFS in forward and reverse direction and the computation of the betweenness centrality.

pendencies (δ) and computing the betweenness centrality (bc). This procedure can be seen as a **BFS in reverse direction**. For a given node, when the SSSP computation is finished along with the distances ($dist$) and shortest paths (σ) computations, we perform the backtracing by traversing the nodes from the farthest distance to the nearest distance and at the same time, we compute dependencies (δ) from σ. An illustration of this process is demonstrated in Figure 2, where we show the BFS (starting from node 0) in forward and reverse directions. During the forward phase (Figure 2(a)), it explores node 1, 2 and 3 in parallel at the first invocation, then node 4, 6 and 5, 7 in the subsequent invocations and the corresponding values of σ are also computed at the same time. Then during the reverse phase (Figure 2(b)), the exploration starts from the nodes in the farthest distance, i.e., node 5 and 7 and subsequently exploration of all the nodes in reverse order are performed along with the computation of corresponding dependencies (δ) (using Equation 6). The kernel to perform the BFS in reverse order and respective dependency computation is presented in Algorithm 8.

During the backtrace, we also need to accumulate the betweenness of the node in question. We utilize the accumulation kernel of (Shi & Zhang 2011) to perform this accumulation (Algorithm 9). The complete edge parallel variant of the Brandes's method is presented in Algorithm 10. The algorithm is optimized to have a better performance than the naïve parallel implementation, as the number of shortest paths and respective dependencies are computed in parallel.

Algorithm 8: BFS in Reverse Order

Input : Gk,$Pred$, σ, δ, n,k;
Output: δ is initialized;

```
1  tid ← thread id;
2  if tid < n × k then
3      source ← Gk[tid].source;
       target ← Gk[tid].target;
4      if dist[u] = d − 1 then
5          found ←true;
6      else if dist[w] = d − 1 then
7          swap(target, source) // device
           function;
8          found ←true;
9      else found ←false;
10     if found =true then
11         bit ← target × n + source;
12         if P[BIT_POS(bit)] & BIT(bit) then
13             atomicAdd(δ[i], σ[i]/σ[j] × (1 + δ[j]));
```

Algorithm 9: Accumulate Betweenness

Input : bc, δ;
Output: Betweenness centrality of each vertex is stored in `bc` array;

```
1  tid ← thread id;
2  if tid < n then
3      if tid ≠ s and dist[tid] = d − 1 then
4          bc[tid] ← bc[tid] + δ[tid];
```

Algorithm 10: Parallel Brandes's (Brandes 2001) Betweenness Centrality 2

Input : Gk and k;
Output: Betweenness centrality of each vertex stored in `bc` array;

```
1  Initialize bc[v] ← 0,  ∀v ∈ V;
2  foreach s ∈ V do
3      Single Source Shortest Path (Gk, s, k)
       (Algorithm 1, see BFS);
4      Initialize δ[v] ← 0,  ∀v ∈ V;
5      /* Backtrace and Accumulate*/;
6      while d > 1 do
7          host → device (d);
8          BFS_Reverse_Order (Gk, dist, σ, δ, P, d)
           (Algorithm 8);
9          Accumulate_Betweenness
           (s, d, dist, δ, bc) (Algorithm 9);
10         device → host (d);
11         d ← d − 1;
12 device → host (bc);
13 return bc;
```

However, the one limitation of this approach is that it was originally designed for sparser (e.g., scale-free) graphs (Shi & Zhang 2011, Jia 2010) and it can only work with undirected graphs (due to the mechanism of forward and reverse traversal). Thus, it assumes a directed kNN graph as an undirected graph and computes the centrality respectively. In contrast, the naïve parallel implementation can handle the directed and relatively denser graphs, as it follows the original data structures as proposed in (Brandes 2001).

4 Results

4.1 Test Environment

We implemented all the proposed algorithms on the following hardware setup. A total of four NVIDIA Tesla C2050 GPU cards were installed on a X8DTG-Q Supermicro server containing 2×Intel Xeon E5620 2.4GHz processors, 32GB DDR3 RAM and 800GB of Local Hard Disk. The programs were written in `C++` and `CUDA (toolkit 4.0)` and compiled using the `g++` v4.4.4 and `nvcc` compilers on a Linux x86_64 OS (kernel version 2.6.9). The computational times were measured using CUDA timer utility (NVIDIA 2007).

4.2 Comparison Tools

We compared the performance of the GPU implementations against respective CPU and GPU variants (a standard implementation of Brandes's algorithm is provided in BOOST Graph Library (Siek et al. 2000), however we used the simpler implementation given in GPU-FAN (GPU-based Fast Analysis of Networks) (Shi & Zhang 2011)). Its GPU variant and the CPU closeness and eccentricity centralities were also taken from the same source: *http://bioinfo.vanderbilt.edu/gpu-fan/*)

4.3 Data Collection and Preprocessing

To assess the methods performance, We utilized a renowned breast cancer gene-expression study contributed by (van de Vijver et al. 2002) (see also, (Van T Veer et al. 2002)). The original data set has a total of 24,479 probe sets (with 1,281 control probes) in 295 breast cancer patients. For each patient the published data set has five attributes: log(ratio), log(ratio) error, p-value for log(ratio) significance, log(intensity), and a flag for each spot (= 0 for control or bad spot, = 1 for valid measurement). In this experiment we only utilized the log(ratio) attribute for each valid measurement that resulted in a total of 24,158 probe sets (for all the 295 patients/ samples). To test the scalability of the proposed method, we extended our search space by creating an artificial data set: *BC_Expanded*, containing a total of 384,125 (details in (Arefin 2013)). The expansion was performed using 'difference' operator between each pair of genes. Then we created several sub data sets (n = 50 000, 75 000 and 100 000) from this extended data.

4.4 Performance Evaluation

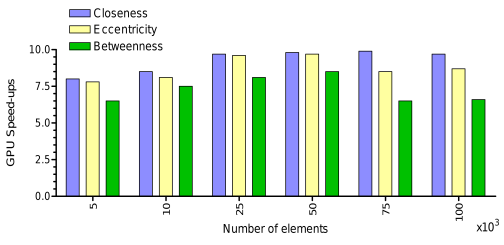


Figure 3: Speed-ups gained by the proposed GPU implementations on the original and expanded data sets.

First, we applied the CPU implementations on these data sets and computed the performance gains of each of the respective GPU implementation (Closeness, Eccentricity and Betweenness Centrality 2) (Figure 3). It can be noted that the CPU implementations were not optimized to handle the kNN graphs (which may contain a number of duplicate edges). Therefore, we created respective graph data structures for the removal of self-loops from the input graphs. The times required for the preprocessing have been added to the respective experimental evaluations. The speed-ups obtained at this stage is depicted in Figure 3.

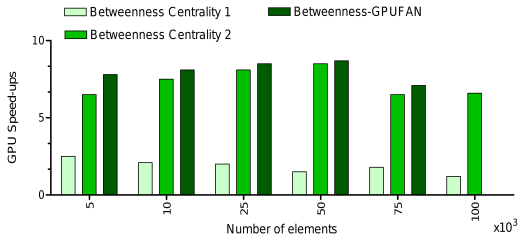


Figure 4: Speed-up gains achieved by the GPU variants of the betweenness centrality.

Next, using the same set of graphs, we computed the speed-up gains achieved by the two GPU variants of the Brandes's betweenness centrality, i.e., naïve parallel (adapted from (Brandes 2001)), edge parallel (adapted from (Shi & Zhang 2011)) and compared them against the speed-up gains achieved by the GPU variant proposed in (Shi & Zhang 2011). The results are presented in Figure 4.

We observed that the Betweenness Centrality 1 (naïve parallel) and the Betweenness-GPUFAN by Shi and Zhang (Shi & Zhang 2011) achieved the lowest and highest speed-ups, respectively. Even though the speed-ups achieved by the proposed **Betweenness Centrality 2** variant was as slightly lower than the original approach (due to the workload imbalance introduced among the threads), on the largest instance only the proposed approach could scale properly in our device (due to the elimination of edge data duplication and hence, data redundancy as explained in Section 3.2). The usage of the kNN graphs instead of the replicated adjacency edge list (as proposed in (Shi & Zhang 2011)) aided us to gain the extended scalability. We found the proposed edge-parallel approach as a preferable method for the large-scale kNN graphs. However, it can only be applied to the undirected graphs. In contrast, the naïve parallel approach can be applied to directed graphs, but it performed only twice faster than the original Brandes's algorithm (Brandes 2001).

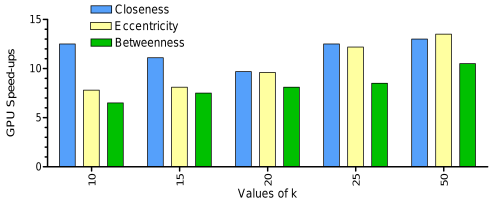


Figure 5: Variation of the GPU speed-ups due to the change in k (for a fixed value of n =25,158).

Next, we used the original data metrics (van de Vijver et al. 2002) (n = 24 158) to construct five different kNN graphs (k = 10, 15, 20, 25 and 50) again utilized the three centrality metrics (Betweenness Centrality 2). The results are shown in Figure 5. The centralities did not receive much affects by the

changes in the value of k . The reason behind this behavior can be understood from Section 3.6. For each source vertex, the parallel BFS spawn threads using the number of its nearest neighbors and hence, an increase in the value of k, in fact further improves the parallel hardware utilization.

4.5 Significance of the Highly Ranked Genes

We have plotted the intersection of top 1,000 central probes in Figure 6 that we identified using the three centrality measures (ECC - eccentricity centrality, CC - closeness centrality and BCC - betweenness centrality 2). The input gene co-expression network was constructed from the original data matrix (n = 25,158) from the kNN graph with fixed value of $k = \ln(n)$. This result brings some interesting insight about the data, as it may facilitate further research in the direction of finding genes that experience the most shortest paths over them (see Tables 3 and 4).

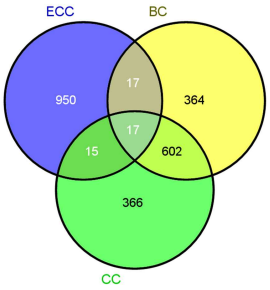


Figure 6: Intersection of the top 1,000 probes in the breast cancer data set (van de Vijver et al. 2002) identified by the three shortest path centralities, where the input gene co-expression network was constructed as a kNN graph for a fixed value of $k = \ln(n)$).

Table 3: Top 10 most central probes at the intersection of probes selected by two of the three centrality measures.

CC ∪ BC	**ECC ∪ CC**	**BC ∪ ECC**
D42055(NEDD4)	AA055642	AA044906
AW138427	AA101173(HESRG)	AA045749(CPXM2)
AI825936(B3GNT5)	AA131323(ITGA8)	AA053806
AB028998(TENC1)	AA176629	AA057596
NM_002051(GATA3)	AA189151	AA059342
NM_004496(FOXA1)	AA190858	AA205599(OTUD7A)
AK000604(COL4A3BP)	AA398575	AA210704
AA935783	AA400740	AA211418
AI123555(ADAMTS5)	AA401392(TAF7L)	AA284301
AA831836	AA434109(FBXL6)	AA406164

For instance, the top BC scoring gene TWISTNB - TWIST neighbor is also in correlation $\rho = 0.245141$ with patients' times to relapse (see Figure 7). We computed the correlation using expression levels of the selected genes against patients' times to relapse given in years (see (van de Vijver et al. 2002)), using a robust correlation function, $\rho = abs(Pearson(x, y) + Spearman(x, y)$. Further, descriptions of the identified genes are out of the scope of this work. However, apart from the hypothetical proteins, some of the genes e.g., FOXA1, GATA3 have previously been reported for their significance in causing tumors and cancers.

Table 4: Top 10 most central probes at the intersection of probes selected by the three centrality measures.

ECC ∪ BC ∪ CC	**Description**
AA001928	–
AA013349	–
AA034111	–
AA035279	–
AA086248	–
AA121481(TWISTNB)	TWIST neighbor
AA129725	–
AA150107(COBLL1)	COBL-like 1
AA192224(SMTNL1)	–
AA233912	–

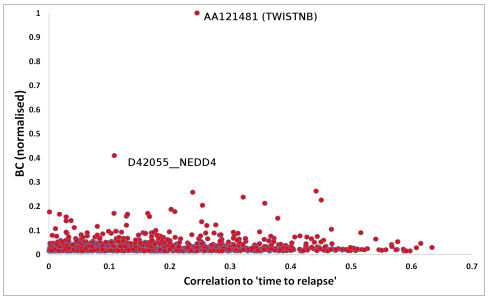


Figure 7: A scatter plot showing correlation of the top 1000 most central genes (betweenness centrality) against 'time to relapse' of cancer in the 295 patient (van de Vijver et al. 2002). The input gene co-expression network was constructed as a kNN graph for a fixed value of $k = \ln(n)$).

5 Conclusion

We proposed GPU implementations of three popular centrality metrics. Although, they are designed to work with the kNN graphs, simple modifications can make them applicable to other graphs. We choose the kNN graphs as the input, as our main goal is to identify the central elements from multi-variate data sets (e.g., microarrays, time series data etc.). Generally, it is not possible to make any distinction or ranking among the elements in a multi-variate data set if they are given in the form of a distance matrix (complete graph). We have shown that the kNN graphs can be successfully utilized in such case. Notably, they contain the most important proximity relations. We showed that the scalability of the existing GPU variant of a shortest path based centrality metrics (betweenness) can further be improved using kNN graphs. The proposed methods should be able to scale up to million nodes on the current series of GPUs (e.g., NVIDIA Tesla and the most recent Keplar architectures). However, an optimization of the bit matrix is required for a further scalability.

6 References

References

Arefin, A. S. (2013), 'An integrated, fast and scalable approach for large-scale biological network analysis— nova. the university of newcastle's digital repository'.

Arefin, A. S., Berretta, R. & Moscato, P. (2013), A gpu-based method for computing eigenvector centrality of gene-expression networks, *in* 'Proceedings

of the Eleventh Australasian Symposium on Parallel and Distributed Computing-Volume 140', Australian Computer Society, Inc., pp. 3–11.

Arefin, A. S., Riveros, C., Berretta, R. & Moscato, P. (2012*a*), Computing large-scale distance matrices on gpu, *in* Y. Jianming & X. Bin, eds, 'Proceedings of the 7th International Conference on Computer Science Education (ICCSE)', IEEE, Melbourne, Australia, pp. 576 –580.

Arefin, A. S., Riveros, C., Berretta, R. & Moscato, P. (2012*b*), 'GPU-FS-kNN: A software tool for fast and scalable ($k$nn) computation using GPUs', *PLoS ONE, in Press* .

Arefin, A. S., Riveros, C., Berretta, R. & Moscato, P. (2012*c*), kNN-Borůvka-GPU: A fast and scalable MST construction from kNN graphs on GPU, *in* B. Murgante, O. Gervasi, S. Misra, N. Nedjah, A. M. A. C. Rocha, D. Taniar & B. O. Apduhan, eds, 'ICCSA (1)', Vol. 7333 of *Lecture Notes in Computer Science*, Springer, pp. 71–86.

Arefin, A. S., Riveros, C., Berretta, R. & Moscato, P. (2012*d*), $k$nn-mst-agglomerative: A fast and scalable graph-based data clustering approach on gpu, *in* Y. Jianming & X. Bin, eds, 'Proceedings of the 7th International Conference on Computer Science Education (ICCSE)', IEEE, Melbourne, Australia, pp. 585 –590.

Bader, D. A. & Madduri, K. (2006), Parallel algorithms for evaluating centrality indices in real-world networks, *in* 'Proceedings of the 2006 International Conference on Parallel Processing', ICPP '06, IEEE Computer Society, Washington, DC, USA, pp. 539–550.

Bader, D. A. & Madduri, K. (2008), 'A graph-theoretic analysis of the human protein-interaction network using multicore parallel algorithms', *Parallel Comput.* **34**(11), 627–639.

Brandes, U. (2001), 'A faster algorithm for betweenness centrality', *Journal of Mathematical Sociology* **25**(2), 163–177.

Edmonds, N., Hoefler, T. & Lumsdaine, A. (2010), A Space-Efficient Parallel Algorithm for Computing Betweenness Centrality in Distributed Memory, *in* 'International Conference on High Performance Computing', pp. 1 – 10.

Eppstein, D. & Wang, J. (2001), Fast approximation of centrality, *in* 'Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms', SODA '01, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 228–229.

Freeman, L. C. (1977), 'A set of measures of centrality based on betweenness', *Sociometry* **40**(1), 35–41.

Garcia, V., Debreuve, E. & Barlaud, M. (2008), Fast k nearest neighbor search using GPU, *in* 'Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on', pp. 1–6.

Gkorou, D., Pouwelse, J. & Epema, D. (2011), Betweenness centrality approximations for an internet deployed p2p reputation system, *in* 'Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum', IPDPSW '11, IEEE Computer Society, Washington, DC, USA, pp. 1627–1634.

Harary, F., Norman, R. Z. & Cartwright, D. (1965), *Structural models : an introduction to the theory of directed graphs*, Wiley, New York.

Harish, P. & Narayanan, P. J. (2007), Accelerating large graph algorithms on the gpu using cuda, *in* 'Proceedings of the 14th international conference on High performance computing', HiPC'07, Springer-Verlag, Berlin, Heidelberg, pp. 197–208.

Jacob, R., Kosch, D., Lehmann, K. A. & Peeters, L. (2005), 'Algorithms for centrality indices', *Network* **3418**, 62–82.

Jia, Y. (2010), Large graph simplification, clustering and visualization, PhD thesis, Champaign, IL, USA. AAI3430988.

Jia, Y., Hoberock, J., Garland, M. & Hart, J. C. (2008), 'On the visualization of social and other scale-free networks', *IEEE Trans. Vis. Comput. Graph.* **14**(6), 1285–1292.

Jia, Y., Lu, V., Hoberock, J., Garland, M. & Hart, J. C. (2012), Edge v. node parallelism for graph centrality metrics, *in* W. mei W. Hwu, ed., 'GPU Computing Gems Jade Edition', Morgan Kaufmann, pp. 15–28.

Jin, S., Huang, Z., Chen, Y., Chavarría-Miranda, D. G., Feo, J. & Wong, P. C. (2010), A novel application of parallel betweenness centrality to power grid contingency analysis, *in* 'IPDPS', IEEE, pp. 1–7.

Junker, B., Koschutzki, D. & Schreiber, F. (2006), 'Exploration of biological network centralities with CentiBiN', *BMC Bioinformatics* **7**(1), 219+.

Junker, B. & Schreiber, F., eds (2011), *Analysis of Biological Networks*, 1st edn, Wiley-Interscience, Secaucus, NJ, USA.

Katz, G. J. & Kider, Jr, J. T. (2008), All-pairs shortest-paths for large graphs on the gpu, *in* 'Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware', GH '08, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 47–55.

Kazerani, A. & Winter, S. (2009), 'Can betweenness centrality explain traffic flow ?', *Traffic* (Gigerenzer 2008), 1–9.

Leist, A., Playne, D. P. & Hawick, K. A. (2009), 'Exploiting graphical processing units for data-parallel scientific applications', *Concurr. Comput. : Pract. Exper.* **21**(18), 2400–2437.
URL: *http://dx.doi.org/10.1002/cpe.v21:18*

Liang, S., Wang, C., Liu, Y. & Jian, L. (2009), Cuknn: A parallel implementation of k-nearest neighbor on cuda-enabled gpu, *in* 'Information, Computing and Telecommunication, 2009. YC-ICT '09. IEEE Youth Conference on', pp. 415 –418.

Ma, H.-W. & Zeng, A.-P. (2003), 'The connectivity structure, giant strong component and centrality of metabolic networks.', *Bioinformatics* **19**(11), 1423–1430.

Madduri, K., Ediger, D., Jiang, K., Bader, D. A. & Chavarría-Miranda, D. G. (2009), A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets, *in* 'IPDPS', IEEE, pp. 1–8.

McLaughlin, A. & Bader, D. A. (2014), Scalable and high performance betweenness centrality on the gpu, *in* 'Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis', SC '14, IEEE Press, Piscataway, NJ, USA, pp. 572–583.
URL: *http://dx.doi.org/10.1109/SC.2014.52*

Mizell, D. & Maschhoff, K. (2009), Early experiences with large-scale cray XMT systems, *in* 'Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing', IPDPS '09, IEEE Computer Society, Washington, DC, USA, pp. 1–9.

Newman, M. (2010), *Networks: An Introduction*, Oxford University Press, Inc., New York, NY, USA.

NVIDIA, C. (2007), *NVIDIA CUDA Compute Unified Device Architecture Programming Guide*, NVIDIA Corporation.

Page, L., Brin, S., Motwani, R. & Winograd, T. (1998), The PageRank citation ranking: Bringing order to the web, *in* 'Proceedings of the 7th International World Wide Web Conference', Brisbane, Australia, pp. 161–172.

Potapov, A. P., Voss, N., Sasse, N. & Wingender, E. (2005), 'Topology of mammalian transcription networks.', *Genome informatics. International Conference on Genome Informatics* **16**(2), 270–278.

Sabidussi, G. (1966), 'The centrality index of a graph', *Psychomatrika* **31**, 581–603.

Shi, Z. & Zhang, B. (2011), 'Fast network centrality analysis using gpus', *BMC Bioinformatics* **12**(1), 149.

Siek, J., Lee, L.-Q. & Lumsdaine, A. (2000), 'Boost random number library', Software Library.
URL: *http://www.boost.org/libs/graph/*

Sriram, A., Gautham, K., Kothapalli, K., J., N. P. & R, G. (2009), Evaluating centrality metrics in real-world networks on gpu, *in* 'Proceedings of 16th Internation Conference on High Performance Computing', HIPC '09, IEEE, pp. 578–585.

van de Vijver, M. J., He, Y. D., van't Veer, L. J., Dai, H., Hart, A. A. & et al. (2002), 'A gene-expression signature as a predictor of survival in breast cancer.', *The New England Journal of Medicine* **347**(25), 1999–2009.

Van T Veer, L. J., Dai, H., Van De Vijver, M. J., He, Y. D., Hart, A. A. M., Mao, M., Peterse, H. L., Van Der Kooy, K., Marton, M. J., Witteveen, A. T. & et al. (2002), 'Gene expression profiling predicts clinical outcome of breast cancer.', *Nature* **415**(6871), 530–6.

Vitter, J. S. (2001), 'External memory algorithms and data structures: dealing with massive data', *ACM Comput. Surv.* **33**(2), 209–271.

Yang, R. & Zhuhadar, L. (2011), Extensions of closeness centrality?, *in* V. A. Clincy, K. E. Hoganson, J. Garrido & V. Dasigi, eds, 'Proceedings of the 49th Annual Southeast Regional Conference, 2011, Kennesaw, GA, USA, March 24-26, 2011', ACM, pp. 304–305.

Link Prediction and Topological Feature Importance in Social Networks

Stephan A. Curiskis **Thomas R. Osborn** **Paul J. Kennedy**

Centre for Quantum Computation and Intelligent Systems
Faculty of Engineering and Information Technology
University of Technology, Sydney,
15 Broadway, Ultimo, NSW 2007,
Email: stephan.a.curiskis@uts.edu.au

Abstract

The problem of link prediction describes how to account for the development of connection structure in a graph. There are many applications of link prediction, such as predicting missing links and future links in online social networks. Much of the literature has focused on limited characteristics of the graph topology or on node attributes, rather than a broad range of measures. There is a rich spectrum of topological features associated with a graph, such as neighbourhood similarity scores, node centrality measures, community structure and path-based distance measures. In this paper we formulate a supervised learning approach to link prediction using a feature set of graph measures chosen to capture a wide range of topological structure. This approach has the advantage that it can be applied to any graph where the connection structure is known. Random forest learning models are used for their high accuracy and measures of feature importance. The feature importance scores reveal the strength of contribution of the topological predictors for link prediction in a variety of synthetically generated network datasets, as well as three real world citation networks. We investigate both undirected and directed cases. Our results show that this approach can deliver very high model precision and recall performance in certain graphs, and good performance generally. Our models also consistently outperform a simpler comparison model we developed to resemble earlier work. In addition, our analysis of variable importance for each dataset reveals meaningful information regarding deep network properties.

Keywords: social networks, link prediction, supervised learning, centrality, community structure, graph topology

1 Introduction

Link prediction describes how the likelihood for a link existing between two nodes in a complex network can be estimated. Many approaches have been proposed, but most require non-topological, node specific information to achieve high accuracy. A key goal of link prediction is the development of an accurate model that can be applied universally to any social network dataset (Shibata et al. 2012).

There are many applications of link prediction. For instance, these methods can be applied for link recommendation to users in online social networks. Another application identified for link prediction models is the evaluation of evolving social network models (Lu & Zhou 2011). Link prediction algorithms estimate the influence of a set of features on the likelihood of link formation. A wide range of topological features can provide information regarding emergent properties of a social network through their predictive importance. This information may provide an empirical basis for the derivation of the rule set of an evolving social network mode. The research question of using a link prediction model to discover information about the deep structure of a social network is still open in the literature.

In this paper, we present a novel link prediction framework that can be applied universally to any network where the topology is known. We define a set of topologically derived features which capture a wide range of network properties, and apply a random forest supervised learning model. Our method is tested against a variety of synthetic networks and real world datasets, for both the undirected and directed cases. We also evaluate whether feature importance scoring can provide information about global and emergent properties of a network.

2 Related Work

There are three common frameworks to link prediction: the similarity based approach, methods based on maximum likelihood estimation, and probabilistic modelling approaches.

The simplest framework for link prediction is the similarity-based approach. In this method, each pair of nodes i and j is assigned a score s_{ij}, defined as the similarity between i and j. All unobserved edges are then ranked according to their scores, with higher ranks having a greater link likelihood. Much of the early literature on link prediction focussed on the use of singular similarity features, or small sets of features. For example, Adamic & Adar (2003) developed a similarity measure based on common neighbours to predict connections amongst web pages. Liben-Nowell & Kleinberg (2007) experimented with a wider range of similarity based measures, but still used each in isolation to rank node pairs with the highest scores. Many similarity indices have been proposed, see Lu & Zhou (2011) for more details. As these methods are relatively simple, each similarity index only considers a limited amount of information regarding the graph topology. As such, the accuracy of these indices is generally quite low.

A more recent framework is to use algorithms based on maximum likelihood estimation. This approach assumes some topological form of the network, e.g. an exponential random graph (Zaccarin & Rivellini 2010). An algorithm then estimates the model parameters against the dataset. Another method was proposed by Clauset et al. (2008) and starts by inferring the hierarchical organisation of a network. The hierarchical structure is assumed to extend across the network, and is applied to predict missing links. A modelling approach known as a stochastic block model has also been developed. These models partition nodes into groups, which strongly influence link probabilty. However, this method is known to ignore heterogeneity in node degree. Zhang et al. (2014) recently extended the stochastic block model by correcting for variable node degree, with improved results. Heterogeneity across other network properties may still be unaccounted for in this approach, limiting its performance in real world datasets. For these methods to work well in practice, the network structure should be first understood and matched to the closest topological form before a model is be built.

The third common framework uses probabilistic modelling. This methodology aims to abstract the underlying link structure of the network through training a probabilistic model, commonly a supervised learning model. For instance, Backstrom & Leskovec (2011) developed a supervised random walk approach. This method combines node attributes as a supervised learning problem to guide a random walk to nodes which are more likely to be connected. There are many types of probabilistic link prediction models which can be applied. Wang et al. (2011) tested a range of supervised learning models on a social network derived from mobile phone data. They found that a decision-tree model performed most effectively when trained on a mixture of both network and data specific node attribute measures.

Recent publications in this area have focussed on wider sets of features used in a supervised learning framework to predict link formation. More diverse topological features can capture different types of complex structure. Shibata et al. (2012) applied a support vector machine learning model to a variety of features on citation networks, including similarity scores, some centrality measures, community classification and node attributes. This approach achieved high performance, and the model weights for each feature were provided as importance measures. The non-topological node attribute features strongly influenced most of their models, and it was found that different models were required for different citation networks. Bliss et al. (2014) analysed link prediction on a large Twitter social network using a wide variety of topological and node attribute similarity features. An evolutionary algorithm was used to estimate the coefficients for a linear combination of features, with good results. Both of these publications, however, utilised non-topological attributes. This limits the network datasets to which their application can be applied to, and are difficult to compare across other networks. There is a great deal of interest in this area, so for a detailed review see Lu & Zhou (2011) and, more recently, Wang et al. (2014).

The method we present in this paper is motivated by the above publications using the supervised learning framework. However, we only consider topological features to ensure that our approach can be applied universally to any network where the topology is known. We contribute to the existing literature by extending the range of topological features used. We also show that a model with a wider range of topological features consistently outperforms a reduced feature set model. A random forest model is used as it can deliver high accuracy and analysis of feature importance. The results outlined in this work provide information regarding emergent properties of synthetically generated networks, as well as three real world citation networks.

The rest of this paper is organised as follows. Section 3 outlines the research methods. Section 4 delivers results regarding the model performance on each dataset, and the analysis of feature importance. We conclude in Section 5, and outline the future directions for this work.

3 Research Methods

In this section, we outline our supervised learning framework for link prediction using a purely topological feature set. We proceed by describing the datasets used and the data preparation and modelling approach. The features are then defined, followed by the model evaluation methodology.

We define a graph G as an ordered pair $G = (V, E)$ comprising a set of nodes V and edges E. The graph is endowed with nodes v_i and edges e_{ij}, where $i, j = 1, \ldots, n$. Edges are symmetric for undirected networks, i.e. $e_{ij} = e_{ji}$, but $e_{ij} \neq e_{ji}$ for directed networks. For directed networks, we will refer to the "to" node as y, and the "from" node as x.

3.1 Data Description

Our model is applied over three types of synthetically generated networks: an Erdős-Rényi random graph, a small-world network, and a scale-free network. While idealised and simplistic, these three graphs can provide topology often observed in real world networks (Newman 2003). These networks are generated using the *igraph* package with *R* statistical software (Csardi & Nepusz 2006). For each model type, we define a variable m which varies the number of edges in the network. The parameter m roughly gives the number of edges as $m \times V$, and we take $m \in \{1, 2, 3\}$. Higher values of parameter m tend to give unrealistic properties, such as much higher graph densities than the real world networks. The model performance also does not vary substantially with $m > 3$.

- *Scale-free* networks are generated by the preferential attachment mechanism, where new nodes are connected preferentially to existing nodes with higher degrees. We generate scale-free networks SF_m with 2,000 nodes. Parameter m is defined as the number of edges added per node.
- *Small-world* networks start with a regular lattice, then proceed to rewire edges randomly across the network. We generate small-world networks SW_m with starting lattice dimension equal to 1, nodes equal to 2,000 and rewiring probability of 0.05. Parameter m is defined as the lattice connection neighbourhood distance.
- *Erdős-Rényi* random graphs start with a fixed number of nodes, and edges are created randomly with uniform probability. We generate Erdős-Rényi random graphs ER_m with 2,000 nodes. The uniform connection probability is defined as $\frac{m}{1{,}000}$.

We have chosen three real world network data sets to apply the link prediction model to: Cora, Citeseer and WebKB (Sen et al. 2008). These data sets

have been chosen as the full connection structure is provided, and they have been used in recent studies (De et al. 2013). All three data sets represent citation networks, are directed, and all contain the full connection structure and node attributes. Table 1 outlines the key properties of each graph. *Di* gives the diameter of the network, *APL* is the average path length, *Cls* gives the clustering coefficient of the network, and *Dns* describes the graph density.

Table 1: Key properties of graph datasets

Graph	V	E	Di	APL	Cls	Dns
Synthetic						
SF_1	2,000	1,999	17	7.8	0	0.001
SF_2	2,000	3,997	7	3.8	0.006	0.002
SF_3	2,000	5,994	6	3	0.01	0.003
SW_1	2,000	2,000	132	51.4	0	0.001
SW_2	2,000	4,000	20	10.1	0.362	0.002
SW_3	2,000	6,000	13	7	0.436	0.003
ER_1	2,000	1,913	33	11	0.001	0.001
ER_2	2,000	3,917	12	5.7	0.002	0.002
ER_3	2,000	5,951	8	4.5	0.003	0.003
Real World						
WebKB	878	1,388	8	3.1	0.036	0.004
Cora	2,709	5,278	19	6.3	0.093	0.001
Citeseer	3,328	4,552	28	9.3	0.13	0.001

In addition to the key graph measures, the networks are plotted with a Fruchterman-Reingold layout, which gives a visual indication of their structure. Figures 1(a) and 1(b) show the plots of the first two scale-free networks SF_1 and SF_2, respectively. SF_3 has been excluded as the structure becomes difficult to discern visually. The branch-like structure is clearly visible in SF_1. Similarly, Figures 1(c) and 1(d) depict the structure of the small-world networks SW_1 and SW_2, and Figures 1(e) and 1(f) plot ER_1 and ER_2. It is noted that these latter graphs are not fully connected, however the largest component makes up the majority of the network in both cases.

Figures 1(g), 1(h) and 1(i) show the structure of the real world datasets. Anecdotally, we can see some common patterns amongst the three real world graphs and our generated graphs. For example, the WebKB graph shown in Figure 1(g) appears to have a similar branching structure to the scale-free network in Figure 1(a). In terms of graph measures, WebKB possesses similar properties to SF_2 and SW_2, although the scale-free network has a lower clustering coefficient, and the small-world network has a smaller diameter.

While the link prediction method outlined in this paper can be applied to both directed and undirected networks, we note that there may be differences in performance in each case. We therefore consider both directed and undirected interpretations of the above networks in our application.

3.2 Data Preparation and Learning Method

Link prediction is known to be a very unbalanced classification problem (Wang et al. 2014). There are usually a large number of node pairs to predict over, and a small number of actual links. We address these issues through implementing a sampling process. We also adopt a random forest learning model. This modelling approach has been chosen because it can be effectively trained to distinguish between unbalanced classes (Breiman 2001). It is proven to be particularly robust to data outliers and also very accurate. Lastly, it can provide measures for the importance of each feature on the likelihood of link formation, which we evaluate for each data set.

Table 2: Link prediction features

Feature	Category
1. Jaccard coefficient	Similarity
2. Adamic-Adar index	
3. Dice similarity	
4. Degree	Centrality
5. Betweenness	
6. Closeness	
7. Eigenvector	
8. PageRank	
9. In same community	Community
10. Community density	
11. Community clustering	
12. Cross-community edge weight	
13. Node pair geodesic	Distance
14. Community pair geodesic	

In all data sets, we have prepared the data for modelling in the following way:

- We create the feature set for all node pairs in G, as defined in Section 3.3
- The data is split into 70%/30% training/testing sets using random sampling.
- In the training data set, we select all the link observations (True class), and sample the remaining set of node pairs (False class) such that there is a 1:100 ratio between the True and False link classes. Through experimentation, we found that oversampling the False cases in the training data produces a more accurate model. 1:100 provides a reasonably representative sample of False cases, while still providing a proportion of True cases high enough for the random forest to model accurately.
- The random forest model is trained using 50 trees, since we found through experimentation that the model accuracy did not improve with additional trees. We also set the number of features included in each tree to the floor of the square root of the total number of features.

3.3 Model Features

We present four categories of measures in this paper: similarity scores, node centrality, community structure and distance measures. While the real world datasets have available certain node attributes, our aim is to use only topological features in our model. This allows for our method to be applied in a standard way to a wide variety of networks in future. We also consider directed cases of each measure. Table 2 outlines the features used by their category. We present the directed version of the features as the generalisation to undirected graphs is trivial.

We refer to node pairs $x, y \in G$, where $x \neq y$ as xy, and classify x as the *from* node and y as the *to* node.

3.3.1 Similarity Measures

Similarity measures have been applied extensively to link prediction. A simple approach to link prediction is to rank all pairs of nodes by their score according to a specific similarity index, and take those node pairs with the highest score to be the most likely connected pairs (Lu & Zhou 2011). However, these

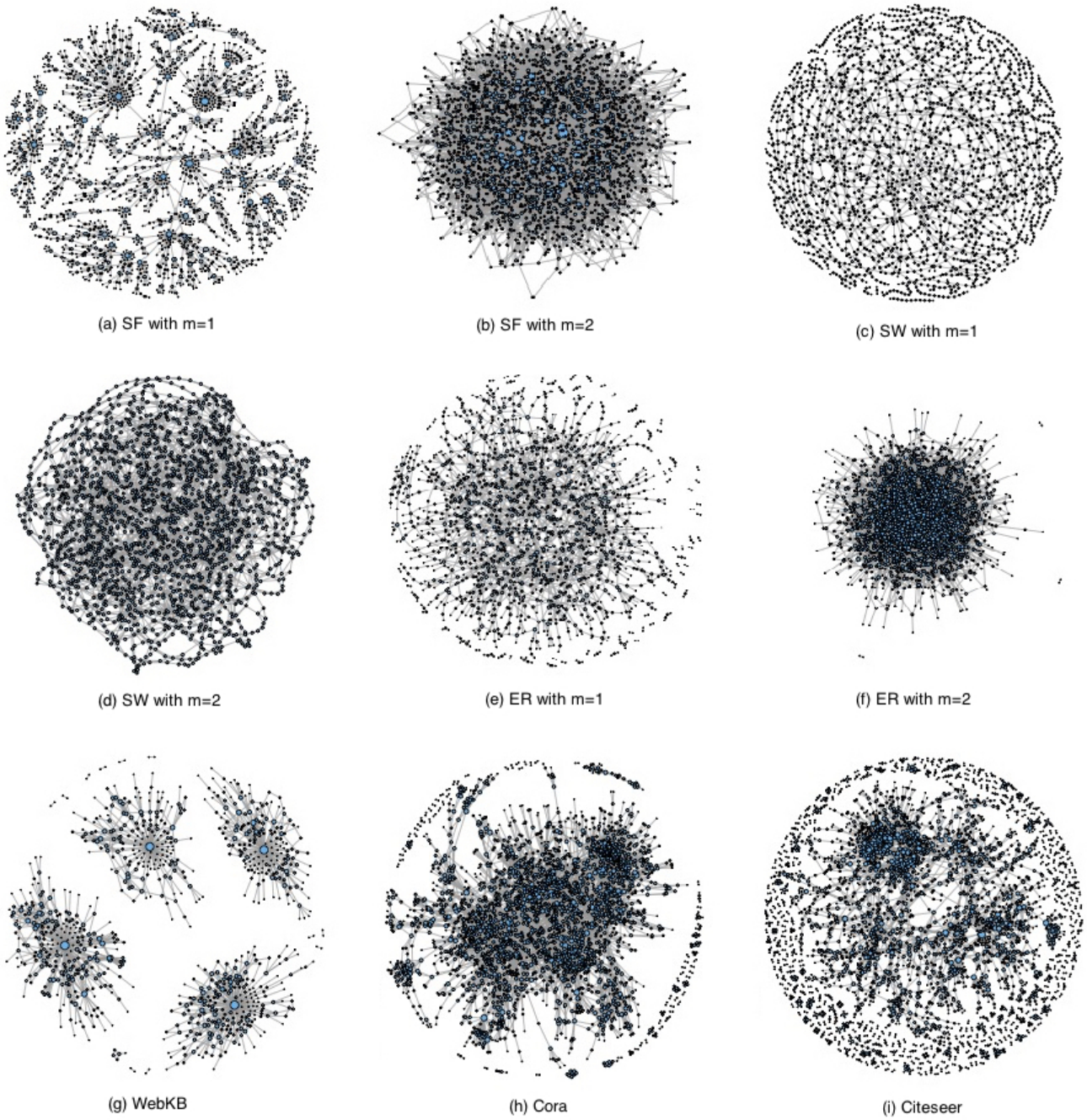


Figure 1: Plots of all network datasets used, in the undirected interpretation. Node size is scaled to represent the degree of the node. (a) represents the scale-free network SF_1, and SF_2 is shown in (b). The branch-like structure is clearly visible in (a), where each node is only connected to one existing node. (c) and (d) depict the plots of the small world networks, SW_1 and SW_2. (e) and (f) show the plots of the Erdős-Rényi networks with $m = 1$ and $m = 2$ respectively. It is clear that there is a large connected component in ER_1, with a number of smaller components around the edges. With $m = 2$ in (f), most of the network is connected. (g), (h) and (i) show plots of the real world networks WebKB, Cora and Citeseer, respectively. It is evident that WebKB consists of four connected components, each with a highly connected hub node, indicating scale-free structure. Cora and Citeseer, shown in (h) and (i), both show a large connected component with a number of smaller components, similar to (e).

measures can also be used as features in the supervised learning approach to link prediction. We consider the following three similarity measures to use as features in the supervised learning problem.

(1) Jaccard similarity coefficient (Sim_{xy}^{Jacc}):

The Jaccard similarity coefficient of two vertices is the number of common neighbours divided by the union of the neighbours of both vertices. For a node x, let $\Gamma(x)$ denote the set of neighbours of x. For directed networks, $\Gamma(x)$ defines the set of nodes with a link *from* node x. The Jaccard similarity coefficient is then defined as

$$Sim_{xy}^{Jacc} = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}.$$

(2) Adamic-Adar index (Sim_{xy}^{AA}) (Adamic & Adar 2003):

This index extends the simple counting of common neighbours to include a term which gives less

connected neighbours higher weight. Letting $k_{out,z}$ be the out-degree of node z, the Adamic-Adar Index is defined as

$$Sim_{xy}^{AA} = \sum_{z \in \Gamma(x) \cup \Gamma(y)} \frac{1}{\log k_{out,z}}.$$

(3) Dice similarity coefficient (Sim_{xy}^{Dice}):

This similarity coefficient, also known as the Sørensen index, is similar to the Jaccard index in form, but has been applied to ecological communities and is known to be robust to outliers (Sørensen 1948). The Dice similarity coefficient is given by

$$Sim_{xy}^{Dice} = 2\frac{|\Gamma(x) \cap \Gamma(y)|}{k_{out,x} + k_{out,y}}.$$

3.3.2 Node Centrality Measures

Centrality measures have been used for a long time within the field of social network analysis to describe the relative influence of a node over the network. Indeed, scale-free networks arose from the insight that individuals may connect preferentially to more highly connected individuals, i.e. those with high degree centrality (Barabási & Albert 1999). Shibata et al. (2007) showed that in citation networks, the betweenness centrality measure was positively correlated between pairs of nodes where a connection existed, and proved to be a significant predictor for future connections. Further to these results, we expect that in social networks individuals may connect preferentially to other individuals according to a variety of centrality measures. We therefore include a broad range of features derived from centrality measures into our supervised learning model and evaluate their predictive power and importance.

The question of how the centrality measures for each node pair are used as features requires consideration. A number of approaches have been applied previously. For example, Shibata et al. (2007) derive a feature based on the difference between the in-degrees of the two nodes, $Cn_{xy}^{PA_2} = k_{in,x} - k_{in,y}$. This measure will capture differences in the in-degree between the nodes, but may not distinguish between high and low in-degree nodes. Another feature derivation commonly used for preferential attachment is defined as the product of each node's degree, i.e. $Cn_{xy}^{PA_1} = k_x \times k_y$ (Lu & Zhou 2011, Barabási & Albert 1999). This method gives more weight to either node having high degree, however the formula does not distinguish between the *to* and *from* nodes. Additional consideration of this measure is therefore required for directed networks. However, this approach is more common in the literature (Wang et al. 2014) so we adopt the same convention. To accomodate the directed network case we also include as a separate feature the centrality measure for the *to* node, y. The undirected centrality product features are labelled as $Cn_{x\times y}^{Measure}$, and the additional directed centrality features are labelled $Cn_{y}^{Measure}$. The list of centrality features are outlined as follows.

(4) Degree centrality ($Cn_{x\times y}^{Dgre}, Cn_{y}^{Dgre}$):

The degree centrality is defined simply as the number of connections of each node in an undirected network, or the in-degree in a directed network. We construct a feature for the product of the in-degrees of both nodes, and another feature for the in-degree of the *to* node:

$$Cn_{x\times y}^{Dgre} = k_{out,x} \times k_{in,y},$$
$$Cn_{y}^{Dgre} = k_{in,y}$$

We expect that a higher degree product indicates high connection likelihood in the undirected case. In the directed case, we also expect that *from* nodes x with a higher out-degree and *to* nodes y with a higher in-degree are more likely to be connected. The in-degree of the *to* node y as a separate feature for directed networks.

(5) Betweenness centrality ($Cn_{x\times y}^{Btwn}, Cn_{y}^{Btwn}$):

Previous studies have revealed that betweenness centrality can be a useful predictor of links in citation networks (Shibata et al. 2012). This measure represents the extent to which a node lies on the shortest paths (geodesics) between other nodes, which can be a useful indicator of influence on network flow. Nodes with high betweenness centralities tend to bridge otherwise unconnected subsets of a network. Formally, for nodes i, s and t, let

$$n_{st}^{i} = \begin{cases} 1 & \text{if } i \text{ lies on the geodesic path from } s \text{ and } t \\ 0 & \text{otherwise.} \end{cases}$$

The betweenness centrality of a node i is then defined as:

$$Cn_{i}^{Btwn} = \sum_{st} \frac{n_{st}^{i}}{g_{st}},$$

where g_{st} gives the total number of geodesic paths from s to t. Our features are then defined as:

$$Cn_{x\times y}^{Btwn} = Cn_{x}^{Btwn} \times Cn_{y}^{Btwn},$$

with Cn_{y}^{Btwn} included in the directed case.

(6) Closeness centrality ($Cn_{x\times y}^{Clse}, Cn_{y}^{Clse}$):

Closeness centrality measures the inverse of the mean distance from a node to all other nodes (Newman 2010). Letting d_{ij} be the length of the geodesic path from node i to j, closeness centrality is defined as:

$$Cn_{i}^{Clse} = \frac{n}{\sum_j d_{ij}}.$$

We therefore construct our features as

$$\begin{aligned} Cn_{x\times y}^{Clse} &= Cn_{x}^{Clse} \times Cn_{y}^{Clse} \\ &= \frac{n}{\sum_i d_{xi}} \times \frac{n}{\sum_i d_{yi}}, \end{aligned}$$

with Cn_{y}^{Clse} included as a separate feature in the directed case.

(7) Eigenvector centrality ($Cn_{x\times y}^{Egnv}, Cn_{y}^{Egnv}$):

Eigenvector centrality extends from degree centrality with the acknowledgement that not all neighbours are equal. This measure awards each node with a score proportional to the sum of the scores of its neighbours. Derivations of eigenvector centrality have been applied effectively to link prediction (Symeonidis et al. 2013), so we expect that it

will be a useful feature. The eigenvector centrality measure for node i, v_i, is defined as:

$$v_i = \kappa_1^{-1} \sum_j A_{ij} v_j,$$

where κ_1 is the leading eigenvalue of A, A_{ij} is the ij^{th} element of A, and v_j is the eigenvector centrality of node j. We construct our feature as:

$$\begin{aligned} Cn^{Egnv}_{x\times y} &= Cn^{Egnv}_x \times Cn^{Egnv}_y \\ &= \left(\kappa_1^{-1} \sum_i A_{xi} v_i\right) \times \left(\kappa_1^{-1} \sum_i A_{yi} v_i\right). \end{aligned}$$

As before, we also use Cn^{Egnv}_y as a feature in the directed case.

(8) PageRank ($Cn^{PgRk}_{x\times y}, Cn^{PgRk}_y$):

PageRank was originally designed as a measure of web page importance (Page & Brin 1998). It is similar to the eigenvector centrality in form, however the neighbour centrality score for each neighbour i is divided by that node's out-degree, k_i^{out}. This penalises the influence of neighbours with very high out-degree. We construct our PageRank features as follows:

$$\begin{aligned} Cn^{PgRk}_{x\times y} &= Cn^{PgRk}_x \times Cn^{PgRk}_y \\ &= \left(\alpha \sum_i A_{xi} \frac{v_i}{k_i^{out}} + \beta\right) \left(\alpha \sum_i A_{yi} \frac{v_i}{k_i^{out}} + \beta\right), \end{aligned}$$

where α and β are constants. We also include Cn^{PgRk}_y as before in the directed case.

3.3.3 Community Measures

Community detection describes the problem of partitioning a graph into densely connected subsets, commonly referred to as communities. A graph's community structure has been shown to be predictive of link formation, as nodes within the same community are more likely to be connected. For example, Shibata et al. (2012) include a feature for whether two nodes are in the same community in their supervised learning model for link prediction, with good performance.

The problem of community detection has received a great deal of interest, see Fortunato (2010) for a comprehensive review. However, many of the community detection methods have been developed for undirected graphs, and some have a high computational cost. We utilise the infomap community detection algorithm as it has computational time $O(V(V \times E))$ and can handle directed graphs (Rosvall et al. 2009). Once the graph is partitioned into communities, we create the community graph G_C by aggregating the vertices of G to their community partitions. Each node in G_C is therefore a community partition of G, and we use the symbols μ and ν to refer to the *from* and *to* nodes in G_C, respectively. We also let $\mu, \nu = 1, \ldots, k$, where k is the total number of community partitions in G. We assign edge weights according to the number of links between each community pair in G. The community features are defined as follows:

(9) In same community (Cm^{Comm}_{xy}):

We expect that a pair of nodes xy in the same community should have a higher likelihood of being connected given that the communities are partitioned according to connection density. This measure is defined for node pairs simply as:

$$Cm^{Comm}_{xy} = \begin{cases} 1 & \text{if } x \text{ is in the same community as } y \\ 0 & \text{otherwise.} \end{cases}$$

Including a feature for two links being in the same community should effectively reduce the link likelihood space dramatically. This will yield more accurate link prediction in networks where the community clustering is strong.

(10) Community density (Cm^{Dens}_{xy}):

The density of a graph is simply the number of edges divided by the number of possible edges. We apply this measure to each community μ, represented as induced subgraphs of G, $\mu \subset G$. Letting $E(G)$ define the set of edges in G, we define $|E(\mu)|$ as the number of edges xy, with $x, y \in \mu, G$. The density of μ is then defined as

$$Dens_\mu = \frac{|E(\mu)|}{\sum_{x,y\in\mu,G} 1}.$$

We then construct our feature vector for each node pair in G as

$$Cm^{Dens}_{xy} = \begin{cases} Dens_\mu & \text{if } x \text{ and } y \text{ share community } \mu \\ 0 & \text{otherwise.} \end{cases}$$

We expect that two nodes in the same community with higher density will have a greater likelihood of being connected than those in a different community with lower density, or in separate communities.

(11) Community clustering coefficient (Cm^{Clst}_{xy}):

The clustering coefficient, also known as transitivity, for community subgraph μ is defined as

$$Clst_\mu = \frac{\text{(number of closed paths of length two)}}{\text{(number of paths of length two)}}.$$

Similarly to the community density, we expect that nodes in the same community with higher clustering are more likely to be connected. The feature vector is constructed as

$$Cm^{Clst}_{xy} = \begin{cases} Clst_\mu & \text{if } x \text{ and } y \text{ share community } \mu \\ 0 & \text{otherwise.} \end{cases}$$

(12) Cross-community edge weight (Cm^{Ewgt}_{xy}):

The community graph G_C provides a more coarse network from which we can take attributes to use for link prediction of node pairs in G. One straightforward measure is the edge weight between each community pair. We construct a feature vector for the node pairs in G based on the edge weights between their respective communities, where the nodes are not in the same community. Let A_C denote the adjacency matrix for G_C, and let $\mu, \nu \in G_C$ denote the community classifications for nodes $x, y \in G$ respectively. The cross-community edge weight is then defined as:

$$Cm^{Ewgt}_{xy} = \begin{cases} \sum_{xy\in G} A_{C_{\mu\nu}} & \text{if } \mu \neq \nu \\ 0 & \text{otherwise.} \end{cases}$$

3.3.4 Distance Measures

Distance measures indicate the number of links between a pair of nodes. Our expectation is that nodes which are closer together are more likely to be connected. For the purposes of link prediction, we only consider distance greater than one for each node pair. This ensures any direct connections are not counted, as links in the graph have distance of one. We construct distance measures for both the individual node pairs, and the distance between their communities in G_C.

(13) Node pair geodesic (D_{xy}^{Node}):

For a pair of nodes $x, y \in G$, let $d_{xy,2}$ be the shortest path (geodesic) from x to y of length greater than or equal to 2. Our feature vector is then simply given by

$$D_{xy}^{Node} = d_{xy,2}$$

(14) Community pair geodesic (D_{xy}^{Comm}):

For a pair of nodes $x, y \in G$ lying in communities $\mu, \nu \in G_C$ respectively, our feature vector for each node pair is defined as the distance from μ to ν:

$$D_{xy}^{Comm} = d_{\mu\nu,2}$$

An issue that arises with the distance features, particularly on small networks, is that there may not be a path with length greater than one for node pairs which are connected. If the network is not fully connected, then there will also be node pairs that do not have a geodesic. These issues effectively introduce missing values into the observations. To account for this, we simply replace any missing distance values with the mean across the dataset which allows for these observations to be included in the model. More sophisticated approaches may be developed to account for missing distance values, but we leave this to future work.

3.4 Model Evaluation

Given the unbalanced nature of the link prediction problem, measuring the performance of a model requires consideration. A common approach is to use the model precision, recall, and the associated F_1 measure (Wang et al. 2014). With our model trained on the training data set, we apply the following evaluation measures on the testing data set only. We abbreviate true positives to TP, false positives to FP, and false negatives to FN.

$$Precision = \frac{TP}{TP + FP},$$

$$Recall = \frac{TP}{TP + FN},$$

$$F_1 = \frac{TP}{TP + FP + FN}.$$

We have chosen precision, recall and F_1 specifically because they are useful in unbalanced problems since they ignore true negatives. There are likely to be a very large number of records classified as true negatives due to the high number of node pairs that do not have links, which distort the performance results. In addition to these evaluation measures, we also provide the precision and recall charts.

In this paper, we expect that a wider range of topological features can produce a more accurate link prediction model, as well as revealing diverse information about deeper graph properties. To demonstrate the performance improvement, we create a comparison model based on the same topological feature set used by Shibata et al. (2012), with a random forest model as the learning algorithm rather than the support vector machine approach. The list of features for the comparison models is given in Table 3. It is noted that the approach by Shibata et al. (2012) includes non-topological features, so we are not making a direct comparison between the two different approaches. We also modify the comparison model features to be applicable to undirected networks. The directed and undirected cases are treated in the same way as outlined in Section 3.3. We compare the performance of models trained with the full feature set to the comparison set by producing precision recall charts for both.

The last aspect of model evaluation we consider is to determine the relative importance of the topological features. We provide an analysis of the random forest mean decrease accuracy importance measure, and discuss the results with respect to model accuracy.

Table 3: Link prediction features for comparison model

Feature	Category
1. Jaccard coefficient	Similarity
2. Adamic-Adar index	
3. Dice similarity	
4. Degree	Centrality
5. Betweenness	
6. In same community	Community
7. Community density	

Table 4: Model performance on all undirected network datasets

Network	Precision	Recall	F_1	AUC
Synthetic				
$\mathbf{SF}_1$	1	1	1	1
$\mathbf{SF}_2$	0.552	0.729	0.628	0.9932
$\mathbf{SF}_3$	0.529	0.527	0.528	0.9729
$\mathbf{SW}_1$	1	0.998	0.999	1.000
$\mathbf{SW}_2$	0.548	0.851	0.666	0.9932
$\mathbf{SW}_3$	0.56	0.829	0.669	0.9858
$\mathbf{ER}_1$	0.82	0.964	0.886	0.9999
$\mathbf{ER}_2$	0.622	0.476	0.54	0.951
$\mathbf{ER}_3$	0.463	0.267	0.339	0.8206
Real World				
WebKB	0.709	0.791	0.748	0.9867
Cora	0.402	0.727	0.518	0.983
Citeseer	0.406	0.876	0.555	0.9969

4 Results

4.1 Model Performance: Undirected Graphs

Table 4 outlines the model performance on all data sets, interpreted as undirected networks. The models trained on the synthetic networks with parameter $m = 1$ give very high performance. The model trained on the scale-free network with $m = 1$ actually classifies every node pair correctly. However, as we increase the value of parameter m, the model perfor-

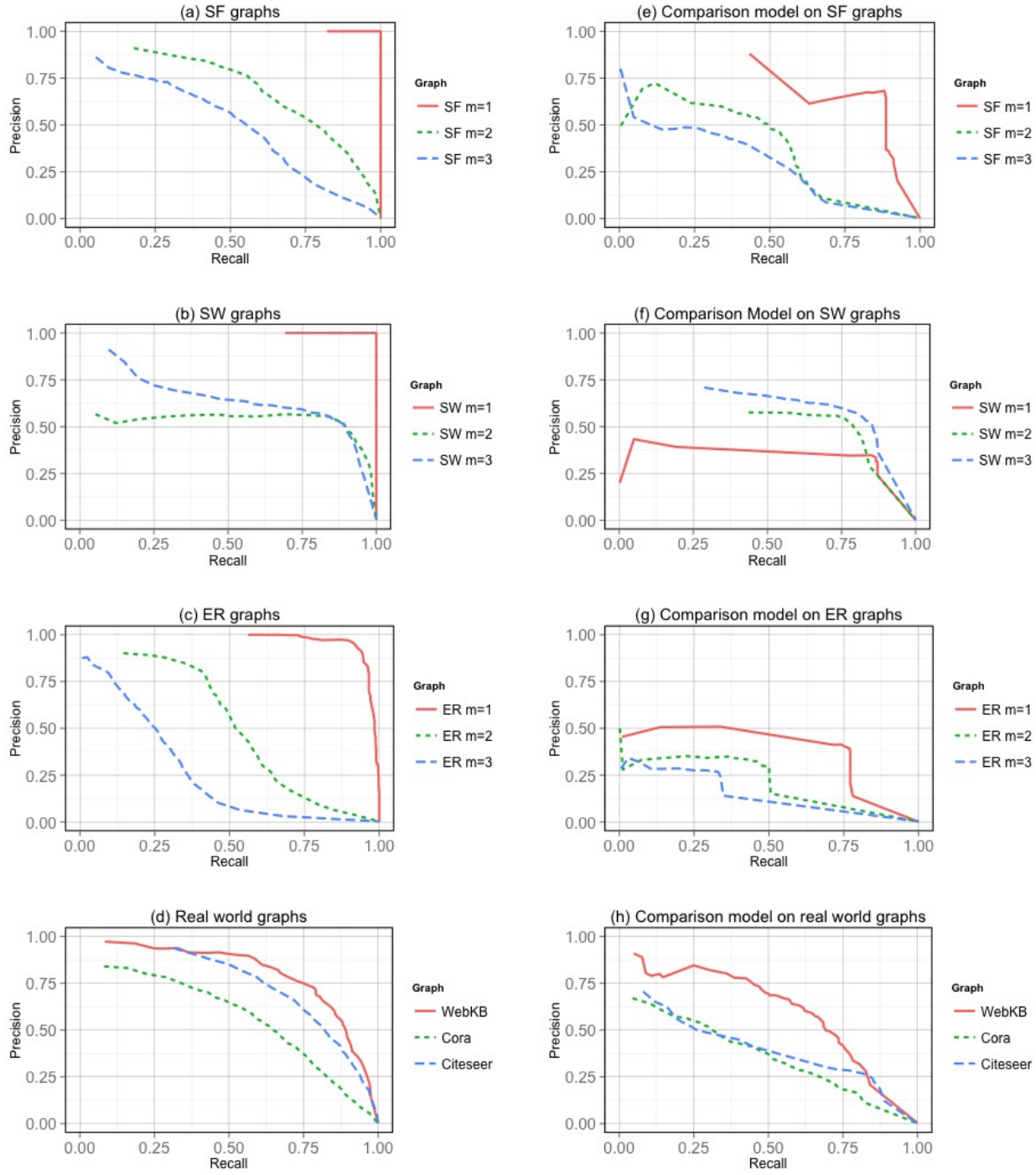


Figure 2: Plot of the precision and recall charts for the undirected scale-free networks (a), small-world networks (b), Erdős-Rényi random graphs (c), and the real world networks (d). Plots (e) to (h) depict the performance of the comparison models trained on the same network as opposite with a smaller range of topological features.

mance decreases in all cases. The scale-free network models lose precision with higher values of m, and recall drops substantially as well. The small-world network maintains good recall and F_1, although precision does drop with $m > 1$. The Erdős-Rényi random graph loses precision as m is increased, but the recall drops much more rapidly to 0.267 with $m = 3$.

Figure 2(a-c) depicts the precision and recall curves for the undirected synthetic networks, where recall is plotted on the x-axis and precision on the y-axis. It is clear from Figure 2(a) that the model performs well for the scale-free graphs with $m = 1, 2$ and 3, given that these curves remain in the top triangle of the plot, and their concavity is down. The small-world networks in Figure 2(b) retain good performance as well. It is interesting that the model on the network with $m = 3$ actually seems to outperform that with $m = 2$. We can see the former model achieves a slightly higher precision of 0.56, compared to 0.548. The Erdős-Rényi random graphs in Figure 2(c) show good performance for the case with $m = 1$, but a large difference to the case with $m = 2$, and again with $m = 3$. All models, except ER_3, deliver AUC higher than 0.97. These results indicate that our learning model can accurately describe the network generating processes of these three undirected

synthetic networks, however this accuracy diminishes as nodes are added to the network with more edges.

On the real world datasets the model gives consistently high recall and AUC values. This indicates that a high proportion of the actual links are being classed correctly. Precision gives how many of the predicted links are actually links, and our model performs well on the WebKB dataset with precision of 0.709. However, the precision values for Cora and Citeseer are much lower at around 0.4. Figure 2(d) gives the precision against recall curves for the real world datasets. We can see that our model performs the best on WebKB, followed closely by Citeseer. The model classification gives a low precision score to the model trained on Citeseer, however the recall is much higher than Cora. The precision recall curve for Citeseer indicates that the model could deliver higher precision with a small drop in recall.

Table 5: Model performance on all directed network datasets

Network	Precision	Recall	$\mathbf{F}_1$	AUC
Synthetic				
$\mathbf{SF}_1$	0.201	0.926	0.33	0.9994
$\mathbf{SF}_2$	0.237	0.522	0.326	0.9443
$\mathbf{SF}_3$	0.206	0.194	0.2	0.7293
$\mathbf{ER}_1$	0.29	0.45	0.353	0.9855
$\mathbf{ER}_2$	0.345	0.154	0.213	0.9401
$\mathbf{ER}_3$	0.35	0.075	0.123	0.8391
Real World				
WebKB	0.487	0.796	0.604	0.9988
Cora	0.297	0.798	0.433	0.9988
Citeseer	0.237	0.955	0.379	0.9997

Figure 2(e) to 2(h) show the precision and recall curves for the comparison model with a reduced topological feature set, trained on the same graphs as the full model. The key observation from these plots is that these models fail to achieve high performance in both recall and precision. Many of these curves are also not smooth or monotonic. This is likely due to the feature sampling of the random forest over the reduced feature set; the curve may change sharply when an important feature is excluded or included. We therefore conclude that by including a wider range of topological features in our supervised learning model, we achieve much higher performance than a smaller set of features.

4.2 Model Performance: Directed Graphs

When we apply the same methodology to the directed versions of these networks, including the centrality measures for the *to* nodes y, the results differ substantially. Table 5 gives the model performance on the directed interpretation of the networks. *igraph* does not currently support a directed version of the small-world network, so we have removed it from this analysis. It is clear that the approach is not as accurate on directed networks as their undirected interpretation. Precision is down significantly for all networks, however recall remains high for the real world network models and the synthetic graphs with $m = 1$. This result suggests that the model can still identify the node pairs that have links, but cannot accurately distinguish the direction of the link. We will explore this more through an analysis of the feature importance on each network.

Figure 3(a-c) show the precision and recall charts for the directed networks. We can clearly see that precision is not as high in the undirected case for the real world graphs, although recall is slightly improved. However, it is clear that the synthetic networks have not performed as well. To compare, Figure 3(d-f) show the precision and recall charts for the comparison model with the reduced feature set. It is clearly evident that the full model outperforms the simpler model. However, it is noted that the comparison model seems to perform slightly better on the real world networks than the synthetic. This may be due to a more balanced feature importance. We explore feature importance in Section 4.4.

4.3 Feature Importance: Undirected Graphs

As mentioned earlier, one of the advantages of using a random forest model is that features which are of lesser use in prediction tend to be effectively downweighted. One measure of the importance of the features is the *mean decrease accuracy*, which is a scaled average of the prediction accuracy of each feature. It effectively measures the decrease in model accuracy when values of each feature are randomly permuted (Breiman 2001). Figure 4 shows a matrix of the mean decrease accuracy for each feature and graph, for both the undirected and directed cases. To save space, we have only shown the synthetic networks with $m = 2$ as these networks are the most similar to the real world graphs.

The undirected models and graphs are given in Figure 4(a). In the model for Citeseer, community based features and distance measures hold the highest importance. The most important feature in this dataset is the community density, followed by the same community flag and the node geodesic. The importance of this feature set may be due to the large number of unconnected components in the Citeseer graph; nodes in these minor components are likely to be attributed to the same community. The model has therefore limited the possible connection space dramatically by assigning a higher importance to community based features. The analysis is very similar for the Cora dataset as well, however the node geodesic is flagged as the most important feature. In both graphs, the centrality measures are also important.

Of all the networks analysed, the model trained on WebKB delivered the best performance. In this dataset, we see a broader distribution of predictive importance across the feature classes. All the features have a mean decrease accuracy in the same order of magnitude. Community based features have the strongest importance, however centrality measures are very close in magnitude, as are the similarity features. These results are not surprising, given the existence of four key connected components, each with a strongly connected hub node.

The Erdős-Rényi random graph model produced a low score for the recall measure, and we see centrality measures strongly influencing the predictions. Degree centrality was the most important, followed by eigenvector and closeness centralities. It seems that the model has trained closely to the node influence features, given the lack of strong neighbourhood and community structure. This may also explain the low model performance, since node influence is not likely to show huge variation in a random graph.

The feature importance in the small-world network is assigned primarily to the community and node geodesic measures. Again, this makes sense given that local connections are far more common in this network type. Centrality measures are marked with low importance, given that highly influential nodes are rarer.

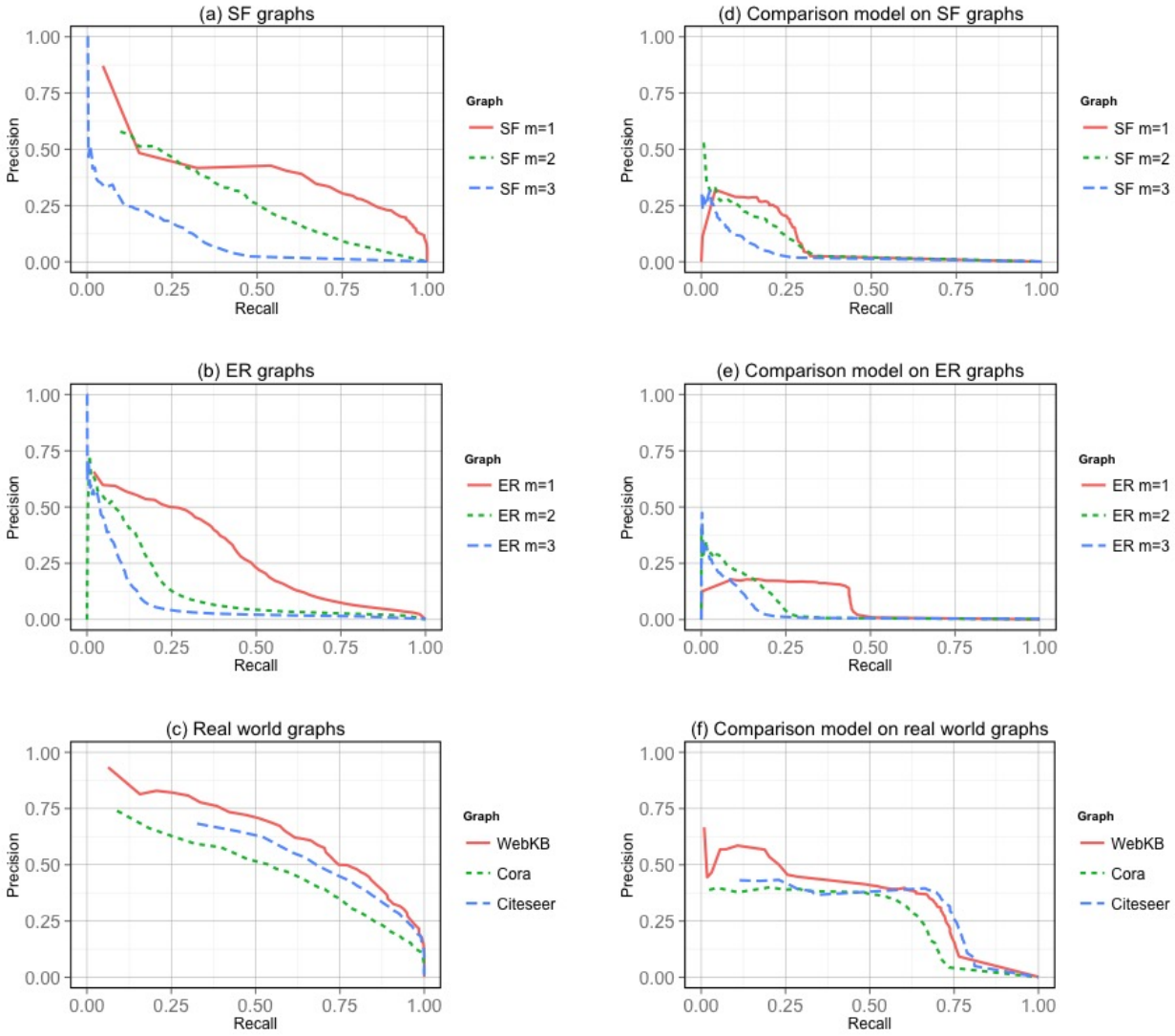


Figure 3: Plot of the precision and recall charts for the directed scale-free networks (a), Erdős-Rényi random graphs (b), and the real world networks (c). Plots (d) to (f) depict the performance of the comparison models trained on the same network as opposite with a smaller range of topological features.

In the scale-free network centrality measures are important as we expect given the preferential attachment growth mechanism. Closeness, PageRank and degree centralities all score highly. However, the most important feature is actually the node geodesic. Community features are also flagged with high importance. These results suggest that community structure and local, densely connected sets of nodes have self-organised in this artificial network.

4.4 Feature Importance: Directed Graphs

For the directed case, Figure 4(b) depicts the feature importance per network. As discussed in Section 4.2, the drop in precision described in Section 3.1 may be due to the model assigning more importance to measures that don't necessarily discriminate between the two nodes. In other words, they will identify the node pairs most likely to be connected, but will not accurately determine the link direction.

For the Cora and Citeseer directed networks in Figure 4(b), the model has scored the community based measures highly, particularly the community density feature. This implies that many of the predicted links will likely be in high density communities. However, this measure does not carry any information regarding the direction of the link within a community. These models all deliver high recall, which shows that the community based features are predictive of a link existing between two nodes in either direction. The only other highly important feature is the *to* node's in-degree. It seems that with only one important centrality measure carrying link direction, these models deliver low precision, but manage to correctly classify a majority of the links.

Similarly to the undirected case, the WebKB model achieved the best performance. Again, there is a distribution of predictive importance across the feature categories. Community structure and node centrality measures are assigned high scores, for both the centrality products and the *to* node centralities. The higher precision for this model relative to the models for Cora and Citeseer can be explained by the fact that most of the centrality features are important. However, the node neighbourhood features are not important in the directed case, which likely explains the drop in precision relative to the undirected model.

The synthetic networks, on the other hand, show strong influence of node centrality product features.

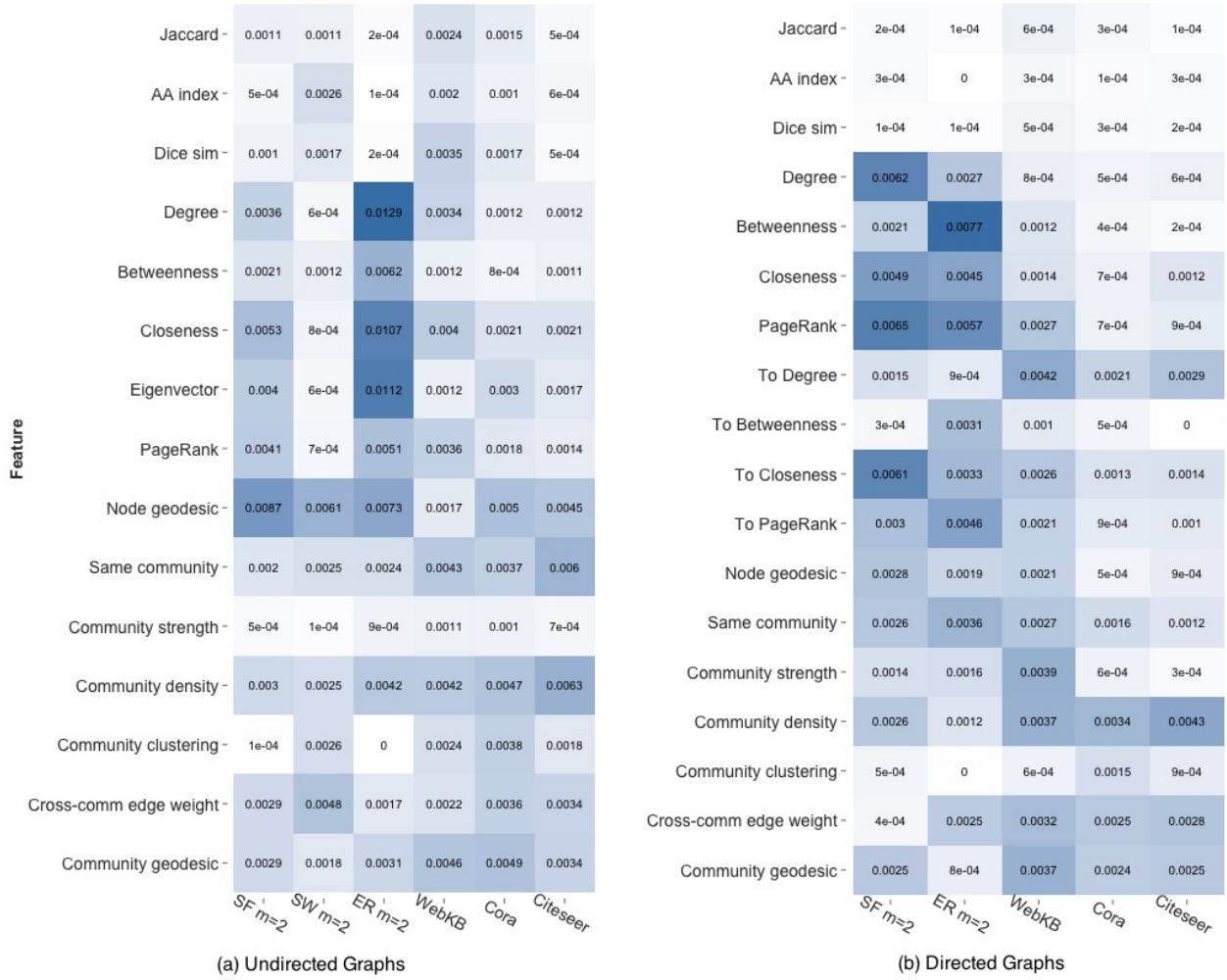


Figure 4: Plot of the feature importance for both the undirected and directed models. Feature importance is determined by the Random Forest mean decrease accuracy measure. Higher mean decrease accuracy is given darker colour.

Betweenness is highly important for the ER_2 model, and both degree features in the SF_2 model. Relative to the undirected feature importance for these models, similarity scores receive very low importance. Community based features are also less important in the directed version. The lower precision and recall for these models is likely due to the bias towards centrality features.

The results for the directed case clearly suggest that more work is required on the feature set. Particularly, more features are required that carry information regarding the direction of the link. As well as this, additional variations of measures based on in-links and out-links should be considered.

5 Conclusion

We have described a new approach to link prediction using a broad range of graph topological features and a random forest learning model. This approach has the distinct advantage of being applicable to any network where the connection structure is known. It also can discover global or emergent properties of the network through analysis of feature importance.

Our method was tested on three types of synthetically generated datasets, as well as three real world citation networks. In the undirected case, the model performs very well in terms of precision, recall, F_1 and AUC. It was shown that the model can achieve perfect classification of classes on a scale-free network with one edge added per node. However, our approach clearly performs more accurately on synthetic networks with less complex structure, i.e. where the number of edges added per new node is small. We also found that the model in the directed case delivers lower precision as it does not accurately distinguish the direction of the link. Modifying the feature set in the directed case to account for the link direction more strongly may address this issue. To demonstrate that including a broader range of topological features can give higher performance, we compared our approach to a model with fewer features. It was shown that a larger set of features consistently gives higher performance.

It was found that the importance of the input features vary significantly with the different networks. The model performs best when the features are more evenly important across the feature categories, as in the WebKB network. Generally, the model tends to perform very well when strong neighbourhood and community structure is present, in addition to high centrality importance. Finally, the analysis of feature importance provides a method for the discovery of complex, emergent properties within a social network. This was demonstrated through the varying importance of neighbourhood and community-based features in many of the datasets considered.

5.1 Further Work

In future, we aim to extend our approach and apply it to more network datasets, particularly large social networks with complex structure. To reduce the computational cost associated with several of the features used, we will consider reducing the number of features used to those with the highest predictive importance, while retaining as much accuracy as possible. Alternative supervised learning methods will be considered as well. The directed version of our approach also needs further development so that the model can more accurately distinguish the link direction.

References

Adamic, L. A. & Adar, E. (2003), 'Friends and neighbors on the web', *Social Networks* **25**, 211–230.

Backstrom, L. & Leskovec, J. (2011), Supervised random walks: predicting and recommending links in social networks, *in* 'WSDM Conference', Hong Kong, pp. 635–644.

Barabási, A. & Albert, R. (1999), 'Emergence of scaling in random networks', *Science* **286**(5439), 509–512.

Bliss, C., Frank, M., Danforth, C. & Dodds, P. (2014), 'An evolutionary algorithm approach to link prediction in dynamic social networks', *Journal of Computational Science* **5**, 750–764.

Breiman, L. (2001), 'Random forests', *Machine Learning* **45**(1), 5–32.

Clauset, A., Moore, C. & Newman, M. E. J. (2008), 'Hierarchical structure and the prediction of missing links in networks', *Nature* **453**, 98–101.

Csardi, G. & Nepusz, T. (2006), 'The igraph software package for complex network research', *InterJournal* **Complex Systems**, 1695.
URL: *http://igraph.org*

De, A., Ganguly, N. & Chakrabarti, S. (2013), Discriminative link prediction using local links, node features and community structure, *in* '2013 IEEE 13th International Conference on Data Mining', pp. 1009–1014.

Fortunato, S. (2010), 'Community detection in graphs', *Physics Reports* **486**, 75–174.

Liben-Nowell, D. & Kleinberg, J. (2007), 'The link-prediction problem for social networks.', *Journal of the American Society for Information Science and Technology* **58**(7), 1019–1031.

Lu, L. & Zhou, T. (2011), 'Link prediction in complex networks: a survey', *Physica A* **390**, 1150–1170.

Newman, M. E. J. (2003), 'The structure and function of complex networks', *SIAM Review* **45**(2), 167–256.

Newman, M. E. J. (2010), *Networks: an introduction*, Oxford University Press.

Page, L. & Brin, S. (1998), 'The anatomy of a large-scale hypertextual web search engine', *Proceedings of the Seventh International World Wide Web Conference* **30**(1–7), 107–117.

Rosvall, M., Axelsson, D. & Bergstrom, C. T. (2009), 'The map equation', *The European Physical Journal Special Topics* **178**(1), 13–23.

Sen, P., Namata, G. M., Bilgic, M., Getoor, L., Gallagher, B. & Eliassi-Rad, T. (2008), 'Collective classification in network data', *AI Magazine* **29**(3), 93–106.

Shibata, N., Kajikawa, Y. & Matsushima, K. (2007), 'Topological analysis of citation networks to discover the future core articles', *Journal of the American Society for Information Science and Technology* **56**(6), 872–882.

Shibata, N., Kajikawa, Y. & Sakata, I. (2012), 'Link prediction in citation networks', *Journal of the American Society for Information Science and Technology* **63**(1), 78–85.

Sørensen, T. (1948), 'A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons', *Biologiske Skrifter* **5**(4).

Symeonidis, P., Iakovidou, N., Mantas, N. & Manolopoulos, Y. (2013), 'From biological to social networks: Link prediction based on multi-way spectral clustering', *Data and Knowledge Engineering* **87**, 226–242.

Wang, D., Pedreschi, D., Song, C., Giannotti, F. & Barabási, A.-L. (2011), Human mobility, social ties, and link prediction, *in* 'Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '11, ACM, New York, NY, USA, pp. 1100–1108.

Wang, P., Xu, B., Wu, Y. & Zhou, X. (2014), 'Link prediction in social networks: the state-of-the-art', *Science China* **58**(1–38).

Zaccarin, S. & Rivellini, G. (2010), Modelling network data: An introduction to exponential random graph models, *in* F. Palumbo, C. N. Lauro & M. J. Greenacre, eds, 'Data Analysis and Classification', Studies in Classification, Data Analysis, and Knowledge Organization, Springer Berlin Heidelberg, pp. 297–305.

Zhang, X., Wang, X., Zhao, C., Yi, D. & Xie, Z. (2014), 'Degree-corrected stochastic block models and reliability in social networks', *Physica A* **393**, 553–559.

AWST: A Novel Attribute Weight Selection Technique for Data Clustering

Md Anisur Rahman and Md Zahidul Islam
School of Computing and Mathematics
Charles Sturt University
Panorama Avenue, Bathurst, NSW 2795
Australia.
{arahman, zislam}@csu.edu.au

Abstract

In this paper we propose a novel attribute weight selection technique called AWST that automatically determines attribute weights for a clustering purpose. The main idea of AWST is to assign weight on an attribute based on the ability of the attribute to cluster the records of a dataset. The attributes with higher abilities get higher weights for clustering. We also propose a novel discretization approach in AWST to discretize the domain values of a numerical attribute. The performance of AWST is compared with three other existing attribute weight selection techniques. We compare the performance of AWST with the three existing techniques namely SABC, WKM and EB in terms of Silhouette Coefficient using nine (9) natural datasets that we obtain from the UCI machine learning repository. The experimental results show that AWST outperforms than the existing techniques on all datasets. The computational complexities and the execution times of the techniques are also presented in the paper. Note that, AWST requires less execution time than many of the existing techniques used in this study.

Keywords: Clustering, Fuzzy Clustering, Hard Clustering, Cluster Evaluation, Data Mining, Attribute Weight Selection.

1 Introduction

Clustering is a process of grouping similar records in a cluster and dissimilar records in different clusters (Rahman, 2014, Tan et al., 2005, Han and Kamber, 2006, Rahman and Islam, 2014, Rahman et al., 2014, Rahman et al., 2015). It extracts hidden patterns, from large datasets, that helps in decision making processes in various fields including medical research, crime detection\prevention, social network analysis and market research (Zhao and Zhang, 2011, Oatley and Ewart, 2003, Adderley et al., 2007, Li et al., 2010, Pirim et al., 2012, Sun et al., 2012, Chan et al., 2012). Therefore, it is important to produce good quality clusters from a dataset.

There are many existing clustering techniques that consider all attributes of a dataset as equally important for the clustering purpose (Redmond and Heneghan, 2007, Chatzis, 2011, Lee and Pedrycz, 2009).

However, all attributes of a dataset may not be equally important for clustering (Rahman and Islam, 2011, Ahmad and Dey, 2007, Rahman, 2014, Rahman and Islam, 2012, Hung et al., 2011). Hence, it is important to find the appropriate attribute weights for clustering. There are many existing techniques that automatically identify attribute weights/importance (Ahmad and Dey, 2007, Bai et al., 2011, Hung et al., 2011, Chen et al., 2012, Cordeiro de Amorim and Mirkin, 2012, Huang et al., 2005, Niu et al., 2008, Boongoen et al., 2011, Gançarski et al., 2008, He et al., 2011).

It is assumed in some literatures on clustering (Rahman and Islam, 2012, Rahman, 2014, Rahman and Islam, 2011, Islam and Brankovic, 2011) that the user knows their dataset well and therefore would be able to assign weights to the attributes to meet their clustering purposes. However, this may not always be the case and the user may prefer to use automatic weights for clustering.

Therefore, in this paper we propose a technique called AWST for selecting attribute weights automatically. The main idea of AWST is to assign weight to an attribute based on the ability of the attribute to cluster the records. The attributes with higher abilities get higher weights. By using AWST a user can assign weights automatically on the attributes or can get an idea of the weights of the attributes so that they can assign weights manually for clustering the records of a dataset. We also propose a novel discretization technique in AWST in order to find the attribute weights. Therefore, the contributions of this paper are as follows.

- A novel discretization technique
- A novel attribute weight selection technique

We compare the performance of AWST with three other existing attribute weight selection techniques namely SABC (Ahmad and Dey, 2007), WKM (Hung et al., 2011) and EB in terms of Silhouette Coefficient using nine (9) natural datasets that we obtain from the UCI machine learning repository (Bache and Lichman, 2013). The performance of AWST is better than the three existing attribute weight selection techniques on nine (9) datasets. We also present the computational complexities and execution times of the techniques. Note that, the execution time of AWST is lower than some of the existing techniques used in this paper.

The structure of the paper is as follows: in section 2 we discuss background study; in section 3 we present our proposed technique called AWST; in section 4 an empirical analysis on our discretization approach is presented; in section 5 the experimental results and

discussion are presented; and the conclusion of the paper is presented in section 6.

2 Background study

In this study, D denotes a dataset, n denotes the number of records of dataset D i.e. D= $\{R_1, R_2 \ldots, R_n\}$, and m denotes the number of attributes of dataset D, i.e. $A= \{A_1, A_2 \ldots A_m\}$. The attributes of a dataset can be numerical and/or categorical (Tan et al., 2005, Han and Kamber, 2006). The numerical and categorical attributes are also known as continuous and nominal attributes, respectively. There is a natural ordering among the domain values of a numerical attribute, whereas there is no natural ordering among the domain values of a categorical attribute. In Table 1, we present an example dataset that has ten records ($R_1, R_2, \ldots, R_{10}$) and four attributes (Age, Marital-Status, Qualification, and Occupation), where Marital-Status, Qualification, and Occupation are categorical attributes and Age is a numerical attribute. The domain values of the numerical attribute Age range from 30 to 65. The domain values for the categorical attribute Marital-Status are {Single, Married}. Similarly, the domain values of all the other categorical attributes can be learnt from Table 1.

Record	Age	Marital-Status	Qualification	Occupation
R_1	65	Married	PhD	Academic
R_2	30	Single	Master	Engineer
R_3	45	Married	Master	Engineer
R_4	30	Single	Bachelor	Physician
R_5	55	Married	PhD	Academic
R_6	35	Single	Bachelor	Physician
R_7	60	Married	PhD	Academic
R_8	45	Single	Bachelor	Physician
R_9	35	Single	Master	Engineer
R_{10}	42	Married	Master	Engineer

Table 1: A synthetic dataset

Many existing clustering techniques consider that all attributes in a dataset have equal weights (significance levels) meaning that all attributes are equally important for clustering (Redmond and Heneghan, 2007, Lee and Pedrycz, 2009, Chatzis, 2011). They do not allow the data miner to assign different weights to different attributes. In these techniques, the data miner can either ignore (i.e. assign a weight equal to 0) or consider (i.e. assign a weight equal to 1) an attribute while clustering the records.

There are of course a number of clustering techniques that automatically (not user defined) assign weights to attributes (Ahmad and Dey, 2007, Bai et al., 2011, Hung et al., 2011, Chen et al., 2012, Cordeiro de Amorim and Mirkin, 2012, Huang et al., 2005, Niu et al., 2008, Boongoen et al., 2011, Fan et al., 2009, Chan et al., 2004, Gançarski et al., 2008, He et al., 2011, Huang, 1998). Since the weights are calculated automatically the user does not have the opportunity to assign different weights and explore various clustering results. The weight of the attributes is often calculated using the pair-wise distance of the values belonging to the attribute (with respect to other values belonging to other attributes), where a higher pair-wise distance (on average) indicates a greater ability to separate/cluster the records (Ahmad and Dey, 2007, He et al., 2011). The weight of an attribute can also be calculated from its variation within the clusters. If the total distance between the values of an attribute within a cluster, for all clusters, is low then it shows a low variation of attribute values. In this paper we call the attribute weight selection based clustering proposed by Ahmad and Dey (2007) as SABC. An attribute weight is considered to be inversely proportional to the variation of the attribute (Huang et al., 2005, Cordeiro de Amorim and Mirkin, 2012). The entropy of the values of an attribute is sometimes used for calculating weight where high entropy indicates a low variation and high weight (Hung et al., 2011). The attribute weight (based on entropy,) based K-Means clustering technique is called Weighted K-Means (WKM) (Hung et al., 2011). WKM does not work on a dataset that has both categorical and numerical attributes whereas our proposed attribute weight selection technique called AWST works on a dataset that has both categorical and numerical attributes.

Attribute weights are often calculated separately within each cluster from an initial set of clusters. This approach is generally called Subspace Clustering, which can be an effective clustering method, especially for high dimensional datasets, in order to avoid the curse of dimensionality (Huang et al., 2005, Bai et al., 2011, Chen et al., 2012). Unlike many other techniques, Boongoen et al (Boongoen et al., 2011) proposed a technique which is applicable with various clustering techniques rather than just K-Means (Niu et al., 2008). The technique first finds the k-nearest records of a record and then finds the weight of an attribute with respect to the nearest records. The attribute may have different weights for different records.

Instead of using k-nearest records, many existing techniques rely on an initial set of clusters (or nearest records) for estimating attribute weights through the calculation of the variation of the attribute values within each cluster. If the initial clustering quality is bad then the attribute weight estimation is also likely to be bad. If the initial clustering quality is good then it appears to be arguably unnecessary to find attribute weights and again find the clusters. Additionally, there are often some attributes which are not relevant to the dataset and these can cause noise in the initial clustering and in the weight estimation. These attributes need to be identified and removed before estimating attribute weights.

3 AWST: Our Novel Attribute Weight Selection Technique

AWST estimates the significance/weight of an attribute according to the ability of the attribute to cluster the records. That is, the attributes that have a greater ability to cluster the records are given a higher weight. Clustering ability is tested based on the well-known evaluation criterion called the Xie-Beni (XB) Index (Mukhopadhyay and Maulik, 2009, Chou et al., 2004). Note that our technique does not depend on the class attribute of the dataset as we realize that datasets used for clustering generally do not have any class attributes. We

now discuss the basic steps which we use in the AWST algorithm as follows:
Step 1: Divide the dataset into clusters based on the domain values of an attribute. Numerical attributes are discretized automatically using the proposed approach;
Step 2: Calculate the XB of the clusters; and
Step 3: Calculate attribute weights for all attributes based on the XB values.

Step 1: Divide the dataset into clusters based on the domain values of an attribute

In order to calculate the weight of an attribute, AWST divides the dataset into mutually exclusive horizontal segments based on the values of the attribute where within a segment all records have the same value for the attribute. If the attribute is categorical then all records of a segment will have the same categorical value for the attribute. If the attribute is numerical then we first discretize the attribute and divide the dataset in segments in such a way that all the records within a segment have the same category of the attribute. The dataset is divided into segments for each attribute one by one. If there are |A| attributes in a dataset it is divided into segments |A| times, where each time it is divided based on a different attribute.

The values of a categorical attribute are clearly categorized in the dataset. However, finding categories for a numerical attribute may not be so intuitive. If we divide the values into B categories (where B could be the square root of the domain size or any other constant number) with equal ranges then we do not take into account natural properties such as the distribution of the values and instead we discretize them artificially.

Adjacent numerical values are typically similar to each other, but the boundaries of the categories need to be determined carefully considering the distribution of the values so that a category represents a concentration of values that can essentially be thought of as a category by itself. Our initial empirical results also indicate that categorizing the values of a numerical attribute using a predetermined number (B) of equal ranges did not give us a sensible result to indicate the clustering ability of a numerical attribute. The initial empirical results are presented in Section 4.

Therefore, for a numerical attribute X = [x_1, x_n] (having domain size = n), AWST discretize the values of the attribute using a novel approach, which is inspired by intelligent K-Means (IKMeans) (Cordeiro de Amorim and Mirkin, 2012). Note that IKMeans originally dealt with all the attributes of all records aiming to find initial seeds, whereas we deal with the values of a single attribute and we aim to find natural categories for the values of the attribute instead of the seeds of the records.

We find the average of all values and call it the grand average, which is $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$. We also find the value x_j having the maximum distance from the grand average as follows:

$$x_j = x_a : \left|x_a - \frac{\sum_{i=1}^{n} x_i}{n}\right| > \left|x_b - \frac{\sum_{i=1}^{n} x_i}{n}\right| ;\ \forall b \neq a \qquad (1)$$

All values are then divided into two partitions P_1 and P_2, where in one partition, P_1, we have values that are closer to the most distant value from the grand average and in the other partition, P_2, we have the remaining values. In equation 2, the value x_l is closer to x_j than the grand average $\bar{x}$ and in equation 3, the value x_p is closer to the grand average $\bar{x}$ than x_j.

$$P_1 = \left\{x_l : |x_j - x_l| \leq \left|\frac{\sum_{i=1}^{n} x_i}{n} - x_l\right|\right\} \qquad (2)$$

$$P_2 = \left\{x_p : |x_j - x_p| > \left|\frac{\sum_{i=1}^{n} x_i}{n} - x_p\right|\right\} \qquad (3)$$

We then go to the next iteration and find the average (called partition average, P_a) of partition P_1 having all values closer to the most distant value than the grand average.

$$P_a = \frac{\sum_{i=1}^{|P_1|} x_i}{|P_1|} ;\ \forall x_i \in P_1 \qquad (4)$$

Two partitions are then again created using the partition average and the grand average, where in one partition we have all records closer to the new partition average P_a than the grand average, and in the other partition we have the remaining values. We continue the iterations while the difference between two consecutive partition average values is greater than a small default threshold ε.

We then remove the partition with values closer to the partition average than the grand average. Among the remaining values we next choose a new most distant value from the same grand average. The whole process is then repeated for the remaining values. We continue this until the partition contains more than a user defined number of threshold t. The partitions are finally used as the categories of the numerical attribute.

We argue that a partition represents a natural concentration of values since in our approach a partition average stabilises when there is a reasonable gap between the values belonging to two partitions.

Step 2: Calculate XB of the clusters

The dataset is then divided into mutually exclusive segments/clusters where, in a cluster, all records have the same value for the attribute (if it is a categorical attribute) or same category of the attribute (if it is a numerical attribute). The Xie-Beni Index (XB) (Chou et al., 2004, Mukhopadhyay and Maulik, 2009) of the clusters for an attribute is then calculated. Note that while calculating XB, we need to calculate the distance between the records and the seeds. For calculating the distance between records, we use similarity between categorical values(Giggins and Brankovic, 2012) and normalized numerical values. We repeat step 1 and step 2 in order to calculate the XB Index for all attributes (see the algorithm as shown in Figure 1). Note that during the XB calculation, the sequence or order of the attributes of the dataset does not have any impact.

Step 3: Calculate attribute weights for all attributes based on XB values

We then calculate the attribute weights based on the XB values for the attributes. An attribute having a lower XB has higher cluster ability than attributes having higher XBs. We calculate normalized XB for the i^{th} attribute.

$$N(\mathrm{XB}_i) = \frac{XB_i}{\sum_{a=1}^{|A|} XB_a} \qquad (5)$$

Where XB_i is the XB value of the i[th] attribute and |A| is the total number of attributes in the dataset, we then calculate the weight of the i[th] attribute as follows:

$$W_i = 1 - N(XB_i) \quad (6)$$

The user can assign the W_i values (see Eq. 6) as the weights of the attributes in clustering. Alternatively, they can sort the attributes according to the XB values and assign higher weights as they like on attributes having lower XB values than attributes having higher XB values. We present the algorithm for AWST in Figure 1 above.

Algorithm: Attribute Weight Selection Technique (AWST)
Input: A dataset D having A number of attributes, a user defined number of values t and a user defined threshold ε
Output: Weights of the attributes W

Set $XB \leftarrow \emptyset$ /*XB stores Xie-Beni (XB) Index of each attribute where XB_i is the XB of i[th] attribute */

Set $W \leftarrow \emptyset$ /*W is a set of weights where W_i is the weight of the i[th] attribute */

FOR (i=1 to |A|) Do

  IF A_i is numerical attribute DO

    A_i ←Categorize attribute A_i /*Categorize numerical attribute A_i */

    S ←FindCluster (D, A_i) /*divide the dataset into clusters based on the categories of the attribute */

    XB_i ←CalculateXB (D, S) /*calculate the Xie-Beni Index of the clusters */

    XB ← XB ∪ XB_i

  END IF

  ELSE

    S ←FindCluster (D, A_i) /*divide the dataset into clusters based on the domain values of the attribute s */

    XB_i ← CalculateXB (D, S) /*calculate the Xie-Beni (XB) Index of the clusters */

    XB ← XB ∪ XB_i

  END ELSE

END FOR

FOR (i=1 to |A|) Do

  $N(XB) \leftarrow \frac{XB_i}{\sum_{j=1}^{|A|} XB_j}$ /*normalized the Xie-Beni value for each attribute */

  $W_i \leftarrow 1 - N(XB)$

  W ← W ∪ W_i

END FOR

Return W

Figure 1: Algorithm for AWST

4 An Empirical Analysis on Discretization

We perform an empirical analysis to evaluate the quality of discretization by using our approach and another approach that discretize using the square root of the domain size of the values of the numerical attributes. We discretize the values of each numerical attribute by using our approach as discussed in Step 1 of Section 3. We next calculate the weight of each attribute by AWST. Based on the attribute weights, we next divide the attributes of the dataset into three equal categories, namely best attributes (BA), medium attributes (MA) and worst attributes (WA). We then assign equal weights (0.4) to the BAs for clustering the records using CRUDAW-F (Rahman, 2014). The clusters produced by CRUDAW-F are evaluated in terms of silhouette coefficient. The silhouette coefficient values of CRUDAW-F in the PID, CA and CMC datasets that we obtained from the UCI machine learning repository (Bache and Lichman, 2013) are presented in Table 2.

We next discretize the values of the numerical attributes by using the square root of the domain size (SRDS) of the numerical attribute and calculate the weight of each attribute by AWST. We next divide the dataset into three categories in the same way that we did above and apply CRUDAW-F by using 0.4 weights for the BAs. The clusters produced by CRUDAW-F are also evaluated in terms of silhouette coefficient. In the PID, CA and CMC datasets, the silhouette coefficient values of CRUDAW-F are presented in Table 2. From Table 2, by using our discretization approach, the silhouette coefficient value of CRUDAW-F is better than the silhouette coefficient value of CRUDAW-F by using the discretization by the square root of the domain size (SRDS).

Datasets	Silhouette coefficient (higher the better)	
	Discretization using our approach (OA)	Discretization using square root of the domain size of a numerical attribute (SRDS)
Pima Indian Diabetes (PID)	0.2706	0.1910
Credit Approval (CA)	0.5284	0.2372
Contraceptive Method Choice (CMC)	0.6607	0.3612

Table 2: The Silhouette coefficient (SC) of CRUDAW-F with the discretization by our approach and square root over of domain size

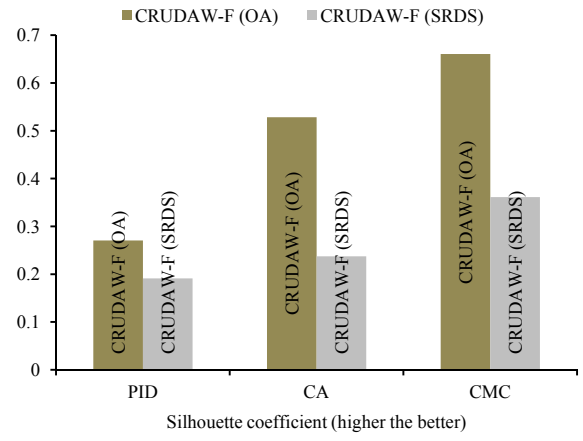


Figure 2: The Silhouette coefficient of CRUDAW-F with the discretization by our approach (OA) and square root over of domain size (SRDS)

For both discretization approaches, we also present the silhouette coefficient values of CRUDAW-F in Figure 2. From Figure 2, we see that our approach for discretization performs better than the discretization by SRDS.

5 Experimental Results and Discussion

We compare the performance of our proposed attribute weight selection technique called AWST with three other existing attribute weight selection techniques namely SABC (Ahmad and Dey, 2007), W-K-Means (WKM) (Hung et al., 2011) and the entropy-based (EB) approach (Rahman and Islam, 2012, Rahman, 2014, Rahman et al., 2015).

SABC is a clustering technique that uses its own attribute weight selection method to assign weights on the attributes prior to clustering. In this study we implement the attribute weight selection method of SABC. Once the weights are selected we use the weights in an existing technique called CRUDAW-F (Rahman and Islam, 2012, Rahman, 2014) for clustering the records, as illustrated in Figure 3. Note that CRUDAW-F uses a weight selection approach for first selecting the weights of attributes and then using them for clustering records. CRUDAW-F is a Fuzzy C-Means based clustering technique where it requires weights of attributes for clustering. In our experiment we replace the original weight selection approach of CRUDAW-F by the weight selection approach of SABC. In Figure 3 we refer to this arrangement as "SABC+CRUDAW-F".

Similarly, WKM is also a clustering technique that uses its own attribute weight selection technique to first compute the weights of the attributes of a dataset. It then uses the weights for clustering the records. In our experimentation, we only implement the attribute weight selection approach of WKM to compute the weights of the attributes. The weights are then fed into CRUDAW-F for clustering the records. In Figure 3 we refer to this arrangement as "WKM+CRUDAW-F".

EB computes the attribute weights by using the entropy of each attribute of a dataset (Rahman and Islam, 2012, Rahman, 2014). We then feed these weights on the attributes into CRUDAW-F in order to get the clustering result. In Figure 3 this arrangement is called "EB+CRUDAW-F".

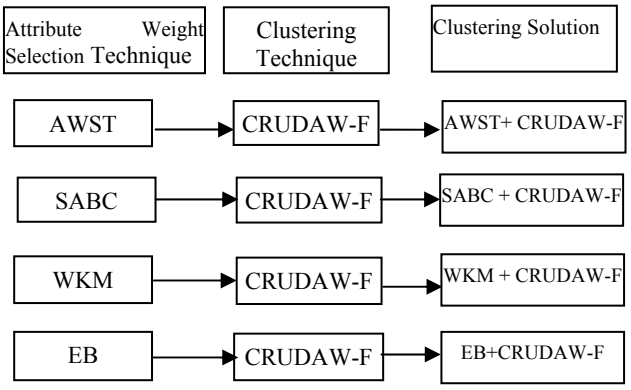


Figure 3: Interaction between an attribute weight selection technique and CRUDAW-F to produce clustering solution

Finally, we use our proposed attribute weight selection technique called AWST and then feed the weights into CRUDAW-F to get the final clustering result. This arrangement has been called as "AWST+CRUDAW-F" in Figure 3. Our main goal in the experiments is to use the same clustering technique (which is CRUDAW-F) for different weight selection methods so that we can compare the performance of the weight selection methods.

The Datasets

We use nine natural datasets, namely Mushroom (MR), Blood Transfusion (BT), Credit Approval, (CA) Breast Cancer (BC), Pima Indian Diabetes (PID), Liver Disorders (LD), Contraceptive Method Choice (CMC), Chess and Adult. All of these datasets were available at the UCI Machine Learning Repository (Bache and Lichman, 2013). A brief introduction to the datasets is presented in Table 3.

We first remove all records with missing values. After removing these records, the MR, CA, Adult and BC datasets had 5644, 653, 30162 and 277 records respectively. We also remove the class attributes from the datasets before we apply the clustering techniques to them.

Datasets	Records with any missing values	Records without any missing values	No. of categorical attributes	No. of numerical attributes	Class Size
Mushroom (MR)	8124	5644	22	0	2
Blood Transfusion (BT)	748	748	0	4	2
Credit Approval (CA)	690	653	9	6	2
Breast Cancer (BC)	286	277	9	0	2
Pima Indian Diabetes (PID)	768	768	0	8	2
Liver Disorders (LD)	345	345	0	6	2
Contraceptive Method Choice (CMC)	1473	1473	7	2	3
Chess	28056	28056	3	3	18
Adult	32561	30162	8	6	2

Table 3: A brief introduction to the datasets

The Parameters Used in the Experiments

In our proposed discretization approach, we use two user defined parameters: 1) the difference between two consecutive partition averages ε; and 2) the number of required values around a grand average t. In the experiments we consider the value of $\varepsilon = 0.00005$ and $t = 1$. The number of iterations for CRUDAW-F is considered as 50 that is mentioned in the original study (Rahman, 2014).

The Experimental Results

We first calculate the attribute weights using our proposed AWST. Based on the attribute weights obtained by AWST, we next divide the attributes of the datasets into three equal categories, namely best attributes (BA), medium attributes (MA) and worst attributes (WA). We then assign equal weights (0.4) to the BAs for clustering the records using CRUDAW-F (Rahman, 2014). Similarly, we also calculate attribute weights using SABC and assign equal weights (0.4) to the BAs (according to SABC) for clustering the records using CRUDAW-F (Rahman, 2014). We also repeat this process for WKM and EB of finding attribute weights and assigning 0.4 weights to the BA attributes.

So CRUDAW-F clusters the records using the 0.4 weights of the BA attributes four times. The first time the BA attributes were chosen using AWST. Second time the BA attributes were chosen using SABC. Similarly in the third and fourth times the BA attributes were chosen using WKM and EB respectively. Finally, the clustering quality of each of the four CRUDAW-F runs is evaluated using silhouette coefficient. The clustering result producing the best silhouette coefficient indicates the best selection of the attribute weights. Note that, the clustering results produced by CRUDAW-F based on each attribute weight selection techniques are deterministic.

While exploring attribute weights automatically, AWST discretizes the values of numerical attributes using the novel approach explained in Section 3. Although SABC and EB can identify attribute weights automatically, they do not have any techniques for the categorisation of numerical values.

Datasets	AWST + CRUDAW-F	SABC + CRUDAW-F	EB + CRUDAW-F	WKM + CRUDAW-F
PID	**0.2706**	0.1814	0.2156	0.2156
LD	**0.3806**	0.2365	0.3398	0.3137
BT	**0.6163**	**0.6163**	0.3826	0.384
CMC	**0.6607**	0.6273	0.2697	NA
CA	**0.5284**	0.4164	0.2758	NA
BC	**0.6562**	0.5286	0.4083	NA
MR	**0.7649**	0.6478	0.5329	NA
Adult	**0.4917**	0.356	0.3554	NA
Chess	**0.9215**	0.867	**0.9215**	NA

Table 4: The Silhouette Coefficient of CRUDAW-F based on each attribute weight selection techniques on nine datasets

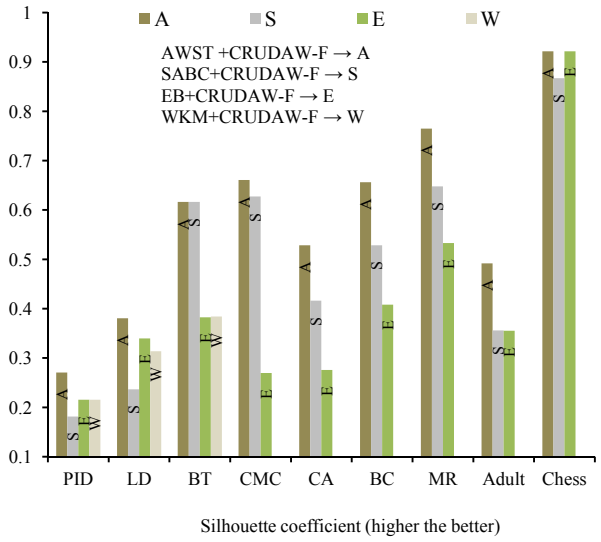


Figure 4: The Silhouette Coefficient of CRUDAW-F based on each attribute weight selection technique on nine datasets

In order to favour SABC and EB, we use our discretization approach for them. Therefore we use the same discretization approach used by AWST for the SABC and EB techniques, basically to favour them and make the experiment a tough evaluation of AWST. The WKM technique was applicable to numerical attributes only. Therefore, we could not evaluate WKM for the datasets having any categorical attributes. The silhouette coefficient (the higher the better) results are presented in Table 4 and Figure 4 for all datasets. From Table 4 and Figure 4 we can see that the performance of AWST is better than the existing techniques in all datasets.

Complexity and Execution Time of the Techniques

We now calculate the complexity of AWST. The overall complexity of AWST is $O(nm^2)$ whereas the overall complexity of WKM is $O(nm)$ (Hung et al., 2011) and the overall complexity of SABC $O(nm^2 + m^2S^3)$ (Ahmad and Dey, 2007), where n is the number of records, m is the number of attributes, and S is the average number of distinct categorical values in a dataset.

We also calculate the total execution time required by CRUDAW-F including the weight selection technique (see Table 5). Table 5 shows that for the PID dataset CRUDAW-F required 0.158 seconds when the attribute weights are determined by AWST. Similarly, for the same dataset, CRUDAW-F required 0.626 seconds when the attribute weights were determined by SABC. For other datasets, the execution time of the techniques can be learnt from Table 5. We use a shared computer system with 4x8 core Intel E7-8837 Xeon processors, 256 GB of RAM and 23 TB of disk storage.

Datasets	AWST + CRUDAW-F	SABC + CRUDAW-F	EB + CRUDAW-F	WKM + CRUDAW-F
PID	0.158	0.626	0.431	0.041
BT	0.048	0.385	0.315	0.014
LD	0.046	0.299	0.228	0.009
BC	0.105	0.037	0.005	NA
CA	0.395	0.537	0.343	NA
CMC	0.234	0.394	0.296	NA
MR	2.547	2.474	0.084	NA
Adult	25.918	116.197	73.601	NA
Chess	1.920	30.621	29.333	NA

Table 5: The execution time (in seconds) of the techniques for all datasets

6 Conclusion

In this paper, we present a novel attribute weight selection technique called AWST. In AWST we discretize the numerical attribute values using our novel approach which was inspired by IKMeans (Cordeiro de Amorim and Mirkin, 2012). AWST calculates the clustering ability of an attribute through the Xie-Beni Index (XB) of the clusters obtained by the categories of the attribute. However, to find the clustering ability of an

attribute, any other internal cluster evaluation criteria such as the Davies-Bouldin Index or Dunn Index could be used (Dunn, 1974, Davies and Bouldin, 1979).

We experimentally compare the performance of AWST with the performances of the SABC, WKM and EB techniques. In the experiments, we select the best attributes (BA) using each technique. We next apply CRUDAW-F (Rahman, 2014) to the datasets by considering the best attributes obtained by each technique separately. Based on each technique, we produce the clusters using CRUDAW-F and calculate the silhouette coefficient of the clusters. The experimental results indicate the superiority of AWST over the existing techniques in all nine datasets.

One of the advantages of AWST is that it requires less execution time when compare with many existing attribute weight selection techniques. The performance of AWST is also shown to be better than many existing attribute weight selection techniques.

7 References

Adderley, R., Townsley, M. & Bond, J. 2007. Use of data mining techniques to model crime scene investigator performance. *Knowledge-Based Systems,* 20**,** 170-176.

Ahmad, A. & Dey, L. 2007. A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering,* 63**,** 503-527.

Bache, K. & Lichman, M. 2013. UCI Machine Learning Repository University of California, Irvine, School of Information and Computer Sciences.

Bai, L., Liang, J., Dang, C. & Cao, F. 2011. A novel attribute weighting algorithm for clustering high-dimensional categorical data. *Pattern Recognition,* 44**,** 2843-2861.

Boongoen, T., Changjing, S., Iam-On, N. & Qiang, S. 2011. Extending Data Reliability Measure to a Filter Approach for Soft Subspace Clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* 41**,** 1705-1714.

Chan, E. Y., Ching, W. K., Ng, M. K. & Huang, Z. 2004. An optimization algorithm for clustering using weighted dissimilarity measures. *Pattern Recognition,* 37**,** 943-952.

Chan, K. Y., Kwong, C. K. & Hu, B. Q. 2012. Market segmentation and ideal point identification for new product design using fuzzy data compression and fuzzy clustering methods. *Applied Soft Computing,* 12**,** 1371-1378.

Chatzis, S. P. 2011. A fuzzy c-means-type algorithm for clustering of data with mixed numeric and categorical attributes employing a probabilistic dissimilarity functional. *Expert Systems with Applications,* 38**,** 8684-8689.

Chen, X., Ye, Y., Xu, X. & Huang, Z. 2012. A feature group weighting method for subspace clustering of high-dimensional data. *Pattern Recognition,* 45**,** 434-446.

Chou, C. H., Su, M. C. & Lai, E. 2004. A new cluster validity measure and its application to image compression. *Pattern Analysis and Applications,* 7**,** 205-220.

Cordeiro De Amorim, R. & Mirkin, B. 2012. Minkowski metric, feature weighting and anomalous cluster initializing in K-Means clustering. *Pattern Recognition,* 45**,** 1061-1075.

Davies, D. L. & Bouldin, D. W. 1979. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Dunn, J. 1974. Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*.

Fan, J., Han, M. & Wang, J. 2009. Single point iterative weighted fuzzy C-means clustering algorithm for remote sensing image segmentation. *Pattern Recognition,* 42**,** 2527-2540.

Gançarski, P., Blansché, A. & Wania, A. 2008. Comparison between two coevolutionary feature weighting algorithms in clustering. *Pattern Recognition,* 41**,** 983-994.

Giggins, H. & Brankovic, L. VICUS - A Noise Addition Technique for Categorical Data. *Proc. Data Mining and Analytics 2012 (AusDM 2012)*, 2012 Sydney, Australia. ACS, CRPIT 134: 139 - 148.

Han, J. & Kamber, M. 2006. *Data Mining Concepts and Techniques,* San Francisco, Morgan Kaufmann.

He, Z., Xu, X. & Deng, S. 2011. Attribute value weighting in k-modes clustering. *Expert Systems with Applications,* 38**,** 15365-15369.

Huang, Z. 1998. Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Mining and Knowledge Discovery*.

Huang, Z., Ng, M. K., Hongqiang, R. & Zichen, L. 2005. Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 27**,** 657-668.

Hung, W.-L., Chang, Y.-C. & Stanley Lee, E. 2011. Weight selection in W-K-means algorithm with an application in color image segmentation. *Computers and Mathematics with Applications,* 62**,** 668-676.

Islam, M. Z. & Brankovic, L. 2011. Privacy preserving data mining: A noise addition framework using a novel clustering technique. *Knowledge-Based Systems,* 24**,** 1214-1223.

Lee, M. & Pedrycz, W. 2009. The fuzzy C-means algorithm with fuzzy P-mode prototypes for clustering objects having mixed features. *Fuzzy Sets and Systems,* 160**,** 3590-3600.

Li, S.-T., Kuo, S.-C. & Tsai, F.-C. 2010. An intelligent decision-support model using FSOM and rule extraction for crime prevention. *Expert Systems with Applications,* 37**,** 7108-7119.

Mukhopadhyay, A. & Maulik, U. 2009. Towards improving fuzzy clustering using support vector machine: Application to gene expression data. *Pattern Recognition,* 42**,** 2744-2763.

Niu, K., Zhang, S. & Chen, J. 2008. Subspace clustering through attribute clustering. *Frontiers of Electrical and Electronic Engineering in China,* 3**,** 44-48.

Oatley, G. C. & Ewart, B. W. 2003. Crimes analysis software: 'pins in maps', clustering and Bayes net prediction. *Expert Systems with Applications,* 25**,** 569-588.

Pirim, H., Eksioglu, B., Perkins, A. D. & Yuceer, C. 2012. Clustering of high throughput gene expression data. *Computers & Operations Research,* 39**,** 3046-3061.

Rahman, M. A. 2014. *Automatic Selection of High Quality Initial Seeds for Generating High Quality Clusters without Requiring any User Inputs.* PhD thesis in Computer Science, School of Computing and Mathematics, Charles Sturt University, Australia.

Rahman, M. A. & Islam, M. Z. Seed-Detective: A Novel Clustering Technique Using High Quality Seed for K-Means on Categorical and Numerical Attributes *Proc. Nineth Australasian Data Mining Conferecne (AusDM 11)*, 2011, Ballarat, Australia, ACS, CRPIT 121: 211-220

Rahman, M. A. & Islam, M. Z. CRUDAW: A Novel Fuzzy Technique for Clustering Records Following User Defined Attribute Weights. *Proc. Tenth Australasian Data Mining Conference (AusDM 2012)*, 2012, Sydney, Australia, ACS, CRPIT 134: 27 - 42.

Rahman, M. A. & Islam, M. Z. 2014. A Hybrid Clustering Technique Combining a Novel Genetic Algorithm with K-Means. *Knowledge-Based Systems,* 71**,** 345-365.

Rahman, M. A., Islam, M. Z. & Bossomaier, T. DenClust: A Density Based Seed Selection Approach for K-Means. *Proc. 13th International Conference on Artificial Intelligence and Soft Computing (ICAISC 2014)*, 2014, Poland.

Rahman, M. A., Islam, M. Z. & Bossomaier, T. 2015. ModEx and Seed-Detective: Two Novel Techniques for High Quality Clustering by using Good Initial Seeds in K-Means. *Journal of King Saud University-Computer and Information Sciences,* 27**,** 113-128.

Redmond, S. J. & Heneghan, C. 2007. A method for initialising the K-means clustering algorithm using kd-trees. *Pattern Recognition Letters,* 28**,** 965-973.

Sun, J., Chen, W., Fang, W., Wun, X. & Xu, W. 2012. Gene expression data analysis with the clustering method based on an improved quantum-behaved Particle Swarm Optimization. *Engineering Applications of Artificial Intelligence,* 25**,** 376-391.

Tan, P.-N., Steinbach, M. & Kumar, V. 2005. *Introduction to Data Mining*, Pearson Addison Wesley.

Zhao, P. & Zhang, C.-Q. 2011. A new clustering method and its application in social networks. *Pattern Recognition Letters,* 32**,** 2109-2118.

Multiple Imputation on Partitioned Datasets

Michael Furner[1] **Md Zahidul Islam**[2]

[1] Center for Research in Complex Systems
School of Computing and Mathematics,
Charles Sturt University,
Bathurst, NSW 2795, Australia,
Email: mfurner@csu.edu.au

[2] Center for Research in Complex Systems
School of Computing and Mathematics,
Charles Sturt University,
Bathurst, NSW 2795, Australia,
Email: zislam@csu.edu.au

Abstract

This paper discusses the impact of making modifications to partition-discovering missing value imputation techniques, and through this process develops a novel imputation algorithm which makes use of partition discovering and multiple imputation - two state of the art techniques. We discuss the difference between *global* and *partition-discovering* imputation techniques and show how the techniques have been developed over time through making modifications to existing techniques in the literature.

Beginning by examining the role of missing value imputation as it relates to the world's increasing desire for data analysis, we proceed to review the current state of the art in regards to *global* and *partition-discovering* imputation techniques, and categorise a variety of existing algorithms into these classes. Provided in this section is an in-depth discussion of an algorithm from each of these categories (EMI and SiMI) in order to gain a greater understanding of how each one works before developing novel techniques.

This is followed by the presentation of several variants to the SiMI algorithm, which are used as a launchpad to our discussion of our proposed technique, the MultiSiMI algorithm, which is shown to improve SiMI's quality of imputation on 6 of 7 datasets tested. This technique is the major contribution of this paper. Each section with a variant of SiMI presents experimental results for the variant discussed in order to gain an understanding of how intelligent modifications to existing algorithms can result in superior novel techniques such as MultiSiMI. We conclude by reviewing the contributions of the paper and recommending some future research directions.

Keywords: missing value imputation, data mining, missing values, decision trees, data cleansing, data munging

1 Introduction

As the world becomes increasingly dominated by digital media and technology, so too have we advanced our methods of analysing data for various purposes. In the 21st century technology is so ubiquitous that we are collecting more data than ever before. Data mining algorithms such as decision trees and artificial neural networks give researchers, marketers, and analysts the ability to peer deeply into the patterns that exist in huge datasets, allowing them to find unprecedented levels of new information within the collected records.

It may often be thought, however, that data collection is perfect. Being surrounded by the numerous devices we use everyday builds a false sense of security about their reliability. It is in fact the case that whether by fault of the collection hardware, or some information being deliberately omitted by the data subject - datasets are commonly incomplete (i.e. some attribute values are missing). In situations where this is the case it is important that the missing values are preprocessed (for example *cleansed*) in order to ensure that the analysis undertaken on the data is more accurate and provides more meaningful results than the analysis on the unprocessed data. If there is noise in the dataset, the noisy values need to be identified as such so that we can either correct for the noise or mark them as if they were missing. When the values are missing or marked as such, the values need to be dealt with - often with a missing value imputation strategy.

Historically, a common approach to handling when a record in a dataset has missing values for one or more of its attributes has been to completely remove the record. This trivial solution is known as *Complete Case Analysis* (Schafer & Graham 2002), and while it seems like an intuitive idea it has several problems. Simply removing records may cause biases towards particular values in attributes, causing a skewed analysis (Schafer & Graham 2002). Substituting attribute means for missing values is another simple approach for dealing with this problem, but it has been shown in the literature to create biased estimates (Tresp et al. 1995). Also, if one attribute is missing in a record and we delete the whole record, we lose the data in the rest of the record. This data may be important or have ramifications for the analysis of the dataset, and in a world where data is money it is hardly an economical approach. Also, it has been shown in the literature that the accuracy of prediction using decision trees improves when imputing the missing values in a dataset rather than leaving them un-imputed (Wang et al. 2014).

Because of the limitations of such simplistic approaches to handling missing data, many algorithms have been produced to make a well reasoned estimate

for the missing values in a dataset. Amongst these methods is the Expectation Maximisation Imputation (EMI) algorithm (Schneider 2001), which performs an iterative imputation on the numerical missing values within a record by using the mean values of the numerical attributes and the correlation matrix of the dataset. By using a cycle of parameter estimation and imputation until convergence, the algorithm makes use of the common and well documented EM technique (Dempster et al. 1977) for the purposes of imputation.

Due to EMI's reliance on the correlation matrix between attributes, several techniques have been developed in order to find horizontal segments of the dataset in which intra-attribute correlation and similarity among the attributes are high. An early one of these was DMI (Rahman & Islam 2011), which used the leaves of a decision tree in order to find these horizontal segments. A more advanced interpretation of a similar concept can be found in SiMI (Rahman & Islam 2013*a*), which instead finds the intersections between leaves from the different trees of a decision forest and uses these as the segments. SiMI proposes that the records belonging to these intersections are expected to be more similar than those of a single leaf found by DMI, and therefore will provide better results for the EMI imputation (Rahman & Islam 2013*a*).

Another regression-based technique for missing value imputation is IBLLS (Iterative Bi-clustering Local Least Squares) (Cheng et al. 2012). Originally designed for use with microarray gene expression data, this algorithm iteratively finds the nearest neighbours of a record and imputes the missing values within the record using a least squares equation, only taking into account of those attributes within the record that are correlated highly enough with the target attribute (i.e. the attribute with a missing value that must be imputed). This, as well as EMI, can only be used to impute numerical attributes, so another technique must be undertaken to impute categorical ones.

As seen in the evolution of SiMI from DMI, and DMI from EMI, intuitive improvements to existing algorithms provide fertile grounds for further research within the field of missing value imputation. This paper aims to build on this tradition by combining the demonstrably powerful approach we identify as *partition-discovering* with the advanced techniques of multiple imputation. We present three variants of existing techniques, each identifying a key component in previous literature as a basis on which to modify existing algorithms. The first two of these provide examples of how different methods of modification result in a drastically different imputation accuracy result, with a clear improvement from the first to the second based on the nature of the change outlined. Through the process of developing these first two, we lead into our third - in which we show the strength of our proposed technique MultiSiMI, which provides a significant improvement upon the algorithm on which it is based and is the major contribution of the paper.

The paper will begin by discussing the different categories of missing value imputation techniques, with an explanation of some techniques that fall within them. The techniques are identified as *global* and *partition-discovering*. A large number of modern missing value imputation techniques are placed into these categories to provide a better understanding of the way they work in relation to each other. We then proceed to identify areas for change in existing techniques, and show how our modifications work to achieve a different result.

Each section provides experimental results which we use to further understand the impact of the modifications proposed. These results are gathered by running the algorithms in question on datasets we create by inserting missing values in a variety of patterns, ratios and models (Junninen et al. 2004) into publicly available clean datasets (i.e. datasets with no missing values) from the UCI Machine Learning Repository (Bache & Lichman 2013). The patterns used are as follows: a simple pattern, where a record can have at most one missing value; a medium pattern, where if there are missing values in a record then a minimum two and a maximum of 50% of the attributes will be missing; a complex pattern which has a minimum of 50% and maximum 80% attributes in a record with missing values and a blended pattern which has missing records with a mixture of records from the other three patterns, with 25% of the records with missing values being simple, 50% being medium, and 25% being complex (Rahman & Islam 2013*a*). We also use 4 ratios of missing values (1%, 3%, 5% and 10%) which determine the percentage of total attribute values in the data set that are missing (Rahman & Islam 2013*a*). In each section, we compare the section's modifications to SiMI, the algorithm that has been modified in each case.

It is important to note that there are multiple mechanisms through which missing values can occur. Missing at random (MAR) refers to when the missing attribute value depends on the other attributes of the record that are missing (Schafer & Graham 2002). Missing Completely at Random (MCAR) makes no such assumptions, meaning we assume that the probability of the value being missing is in no way related to anything else in the data set (Schafer & Graham 2002). Missing Not at Random (MNAR) implies that missing values depend on *other* missing values so it is subsequently impossible to estimate from those values we have access to (Aydilek & Arslan 2013) (Schafer & Graham 2002). The advanced techniques to be discussed in this paper make the assumption that the mechanism for missing values in the dataset is MCAR - an important factor to note, as a different approach would need to be undertaken for other patterns.

The models of missingness used are Overall and Uniformly Distributed (UD). In the UD model, missing values are spread equally over all of the attributes, wherein Overall they are not. With the four missing patterns, four missing ratios and two missing models we have 32 combinations of missingness. Each of these combinations is used to generate 10 missing value data sets with any given clean data set in order to compensate for the randomness in generating the missing value data sets. The 320 missing value datasets created for each original dataset have missing values in an MCAR pattern due to this process, so the application of our missing value imputation algorithms is suitable. We use this methodology for testing as it has been used previously in literature of a similar nature (Rahman & Islam 2011, 2013*a*, 2014) due to the characteristics of missing data in a dataset having an impact on the performance of the missing value imputation techniques (Junninen et al. 2004).

We use the index of agreement (d_2)(Junninen et al. 2004) evaluation criteria to evaluate the use of the algorithms. Other evaluation criteria such as *RMSE*, *MAE* and R^2 (Willmott 1982) have been used in the literature, but due to space restraints we will use the aforementioned d_2 criteria. For this evaluation criteria, we display the average over all 32 combinations, each of which has been averaged over 10 missing value datasets. All of the data sets used are freely available

Symbol	Meaning
D	Dataset
$\|D\|$	No. Records in Dataset
A	Set of Attributes
A_j	jth attribute in A
r_i	Record i in D
r_{ij}	Value of attribute j in r_i
f	Fitness function
k	Number of initial centroids in k-means
τ	Min. no. records in an intersection/cluster
μ	Mean vector
Σ	Covariance matrix
w	Weight scalar
δ	Correlation threshold
R	Correlation Matrix
C	Set of Clusters
C_i	ith cluster in C
C_{ij}	ID of the jth record of the ith cluster in C
$\hat{c_i}$	Cluster centre of C_i

Table 2: Symbol Table

from the UCI machine learning repository. See Table 1 for a summary of the data sets used in the experiments. A list of commonly used symbols (Table 2) has been provided to aid in the understanding of the many algorithms discussed in the paper.

2 Background Research: What are Partition-Discovering Imputation Techniques?

One of the most effective algorithms for missing value imputation is EMI (Nelwamondo et al. 2007). EMI works using the expectation-maximisation algorithm, which iteratively updates parameters based on previous iteration's results. This algorithm is designed to work on a whole dataset, and provides a novel, global technique to imputation.

It accomplishes this by computing the deviation of the available attributes in a record with missing values from their means, weighted by the correlation between the available attributes and the missing attributes. This correlation is derived from the covariance matrix Σ, and we split the mean vector μ into those attributes that are available in the record (μ_a) and those whose values are missing in the record (μ_m). We index the covariance matrix Σ as follows: Σ_{aa} is the covariance matrix between available attributes in the record, Σ_{am} is the covariance matrix between those attributes available and those missing, Σ_{mm} is the covariance matrix between missing attributes in the record, and Σ_{ma} is the covariance matrix between missing attributes in the record and available attributes in the record.

A vector of missing attributes in a target record r_i, x_m is estimated using the following equation:

$$x_m = \mu_m + (x_a - \mu_a)B + e \tag{1}$$

Where x_a is the available attributes in the target record, B is defined as $\Sigma_{aa}^{-1}\Sigma_{am}$ and e is a random residual vector with 0 mean and covariance matrix $\Sigma_{mm} - \Sigma_{ma}\Sigma_{aa}^{-1}\Sigma_{am}$ considered only on the first iteration (Schneider 2001). After each iteration of the algorithm, EMI recalculates the mean vector μ and the covariance matrix Σ, allowing the next iteration to use a more accurate estimate for the true mean vector and covariance matrix in its imputation. This process continues until there is no longer any change between iterations, meaning we have found the imputation with maximum likelihood based on the process.

A quick inspection of the equation indicates two things. First, the term $(x_a - \mu_a)$ will be minimised by highly similar records in the dataset, as the mean vector will be very close to the available attribute values. Secondly, the higher the correlation between attributes, the more accurate the result based on the regression coefficient matrix B. It is from these observations that justification for a new collection of algorithms was developed.

SiMI (Rahman & Islam 2013*a*) and its predecessors take a completely different approach by improving imputation accuracy from previous techniques - not by directly altering a basic algorithm or proposing a trivial imputation solution, but by providing an existing algorithm with an environment in which it can perform its imputation better. These algorithms justify this approach by claiming that certain segments of a dataset typically have higher correlation between attributes than their correlation within the whole dataset. They also argue that this property improves the efficacy of EMI as EMI uses the correlation between attributes as a primary component of its imputation calculation (Rahman & Islam 2011), as we previously observed. An example of this property would be the correlation between age and height. Within a dataset representing people, records with an age below 20 will likely have a high correlation between age and height that does not exist in the rest of the dataset. Similarly, SiMI proposes that these groups will also have highly similar records, providing even further justification.

SiMI's process works as follows. First, the dataset is divided into two parts; in one part we have all the clean records that have no missing values, and in the other we have all records that do have missing values. Then, SiMI builds a decision forest (Islam & Giggins 2011) on the clean records in the dataset, and once we have the rules for the decision trees in the forest, we assign the missing value records to their appropriate leaves. Each tree in the forest will have leaf whose logic rule satisfies the attribute values of a record r_i. We say that this record r_i belongs to the leaf, so therefore each leaf represents a set of records whose attribute values are satisfied by the leaf's logic rule. The record r_i belongs to one and only one leaf of each tree, but since there are n trees in a decision forest, r_i belongs to n leaves, one from each tree. SiMI will then take the intersection of each of these n sets of records. Now, r_i belongs to one and only one intersection, and this intersection is considered by SiMI to consist of highly similar records to r_i.

As some of these intersections may be very small, SiMI uses a merging algorithm to merge intersections that have less than a user defined value τ records. In considering which intersection that the smallest intersection with less than τ records should merge with, SiMI implements a user defined weight λ. They consider two criteria for selecting the best intersection to merge with, and use λ to determine the strength of these criteria on the selection process. The first of these criteria is similarity between intersections (Sim), in which we calculate the normalised record-to-record distance from one intersection to the other (d_j) and find the similarity via $(1 - d_j)$. The second criteria is correlation (Cor), which is determined by the L^2 norm of the correlation matrix for the new intersection that would be created if the two candidate intersections were merged. These criteria are combined as follows:

$$V = Sim \times \lambda + Cor \times (1 - \lambda) \tag{2}$$

SiMI merges the smallest intersection with the intersection that provides the highest V value. This

Dataset Name	No. of Records ($\|D\|$)	No. Numerical Attr.	No. Categorical Attr.	Total Attr.
Yeast	1484	8	1	9
Pima	768	8	1	9
Credit Approval (CA)	653	6	10	16
Contraceptive Method Choice (CMC)	1473	2	8	10
Heart	270	6	8	14
German CA (numeric)	1000	24	1	25
Auto MPG	392	5	3	8

Table 1: Summary of data sets used

process iterates until there are no more intersections with less than τ records. Once this merging is completed, SiMI performs an imputation with EMI to deal with missing numerical values, and a mode imputation (from just within the intersection) in order to deal with categorical attributes. Figure 1 shows how SiMI finds intersections with which to perform these operations on.

Missing value imputation algorithms can be grouped into two sometimes overlapping categories. The first category is *global* techniques, which use the entire dataset provided in order to impute the missing values. Algorithms such as EMI, least-squared imputation (Cai et al. 2006) (another regression technique), FIMUS (Rahman & Islam 2014), mean imputation (Schafer & Graham 2002), mode imputation (Schafer & Graham 2002), hot deck imputation (Schafer & Graham 2002), and Support Vector Regression Imputation (Mallinson & Gammerman 2003) fall into this category. Algorithms that divide the dataset in order to find a better environment to perform a global imputation technique within can be described as *partition-discovering*, and include DMI (Rahman & Islam 2011), IBLLS (Cheng et al. 2012), ILLS, LLS (Cai et al. 2006), k-NNI, SVDImpute (Troyanskaya et al. 2001), and SiMI (Rahman & Islam 2013*b*).

SiMI was shown in its original paper to provide better results than EMI over many datasets (Rahman & Islam 2013*a*), and as such we compare against SiMI only in our experiments (as the purpose of the experiments is to improve the imputation quality of SiMI). It is important to preface the following study with a note however - parametric partition-discovering techniques can be temperamental. The following section seeks to address this with the following question: "are decision trees really the best way to find high similarity horizontal segments?"

3 Using Alternative Methods for Finding High Similarity Horizontal Segments

SiMI and its predecessors use a decision tree or decision forest in order to find horizontal segments (i.e. subsets of records) of a dataset where within each subset the records have high similarity and the attributes are highly correlated. This is in order to increase the effectiveness of the EMI algorithm for imputing numerical attributes, and to provide sets of similar records for a better mode imputation on categorical attributes (Rahman & Islam 2013*b*).

A major issue with the use of decision trees for finding horizontal segments with high similarity is that they are a complex solution to a relatively simple task. The popular decision tree algorithm C4.5 has been extensively studied, and is known to have a complexity of $O(|D||A|^2)$ (Su & Zhang 2006) where $|D|$ is the number of records in the dataset and $|A|$ is the number of attributes. For a high dimensional dataset the complexity of building a tree can be very high. Moreover, not all datasets have well defined class attributes, so this can also be problematic as if the decision trees have a low prediction accuracy they may not be finding highly similar records within their leaves. SiMI also requires the specification of several parameters, including the parameters for decision tree and decision forest algorithms. This amounts to at least 8 parameters when we take into account SiMI's own λ and τ values (Rahman & Islam 2013*b*). Having these parameters set to non-optimal values (which vary from dataset to dataset) can drastically reduce the quality of the imputation produced. The process that SiMI undertakes in order to find horizontal segments is explained in more detail in the previous section, but this startling fact can be a motivating factor in developing new missing value imputation techniques.

Clustering algorithms are designed to find distinct sets of close records within a dataset (Jain 2010), and due to this may yield the potential for a better imputation result when combined with a regression algorithm such as EMI as has been done in DMI and its successors, or the least squares regression used in IBLLS. We propose that in the context of missing value imputation, the need to find distinct clusters is less important (i.e. those with high intra-cluster similarity and low inter-cluster similarity) than the need to find groups of highly similar records (i.e. considering high intra-cluster similarity and ignoring inter-cluster similarity), meaning we can also use a different method for determining the fitness of the set of clusters found.

We present a modification to SiMI for the purposes of testing the hypothesis that a clustering algorithm can be used in order to find high similarity horizontal segments, which will be further referred to as k-Means SiMI. For a clustering algorithm, we will use k-means, as it is widely used and has a low complexity of $O(|D|)$ (Jain 2010). The k-means algorithm uses a simple iterative process to find cluster, and as the name suggests takes a parameter for the number of expected clusters, k. The algorithm begins by selecting k centroids from random points from within the dataset, and assigns each records to one of these centroids. The centroids are then recalculated by taking the mean of the attributes of each of the records in their cluster, and this process of assignment and recalculation is repeated until there is no change in the cluster boundaries. These operations are all performed on a normalised version of the dataset. Due to the normalisation process it is possible to naively include categorical attributes in the k-means process by considering a distance of 1.0 to non-matching values and 0 to matching values. However, in this experimentation we do not include categorical attributes in

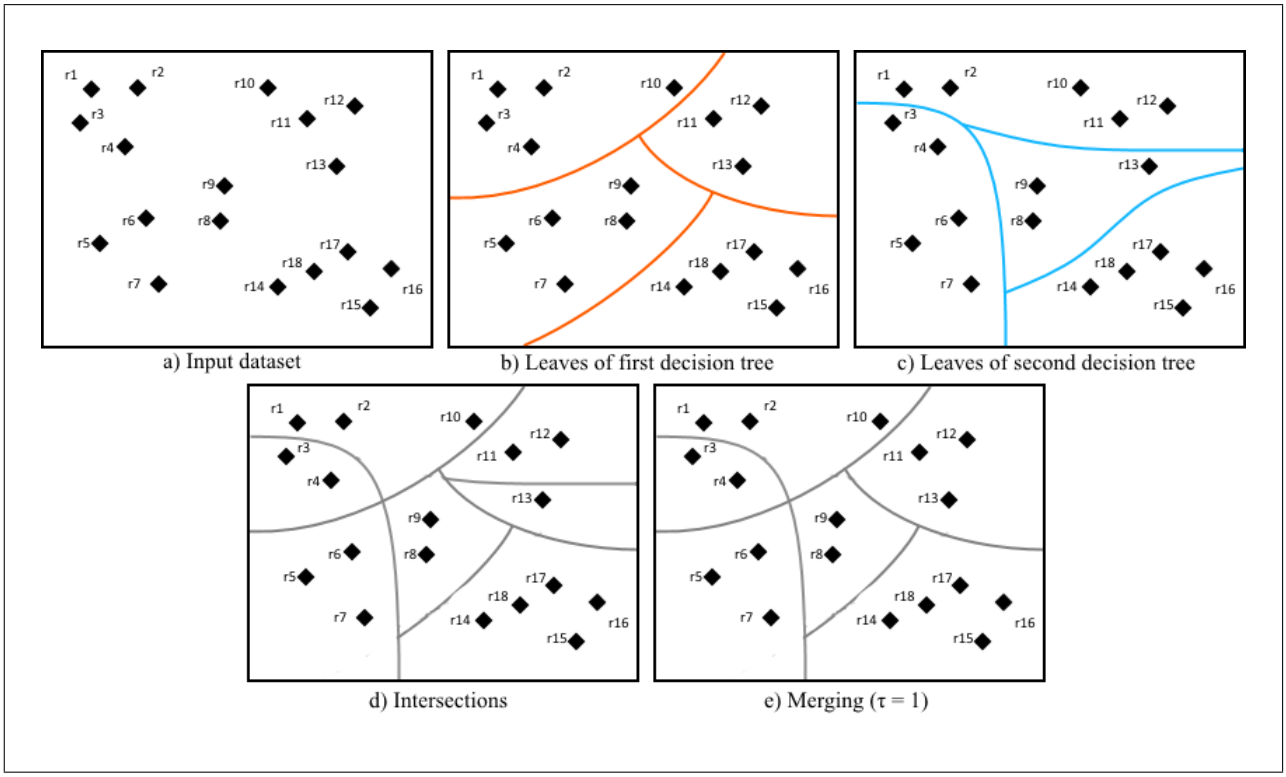


Figure 1: SiMI finding intersections on a basic dataset (Rahman & Islam 2013*a*).

the k-means process since it would place an imbalance onto categorical attributes in the distance calculations.

In order to set the k value and to deal with the non-deterministic nature of k-means, we run the algorithm multiple times (10 in our experiments) with various k values (between 2 and $\sqrt{|D|}$) to get several sets of clusters, and select the set of clusters that provides the best score with a fitness function f. This fitness function can be designed in many ways, and will have an impact on both the complexity and effectiveness of the algorithm. One way of determining f is to take into account the intra-cluster similarity between records and their cluster centre, weighted by the ratio of amount of records inside the cluster. Given this, we have:

$$f_1 = \frac{\sum_i^{|C|} \sum_j^{|C_i|} sim(r_{C_{i,j}}, \hat{c}_i)}{|D|} \tag{3}$$

Another potential fitness function we could use is based on Pearson correlation. This potential fitness function works by taking the average of the norms of the correlation matrix of the clusters.

$$f_2 = \frac{\sum_{j=1}^{|C|} \|R_j\|^2}{|C|} \tag{4}$$

Where R_j is the correlation matrix of the jth cluster.

After running k-means multiple times, we select the set of clusters with the highest f value and use this to proceed to the next step. Any cluster in this set that has less than τ records must be merged with another cluster, and we use SiMI's intersection merging procedure for this. As with SiMI, these horizontal segments have their numerical values imputed using EMI, and their categorical attributes imputed via mode imputation. All tests performed on SiMI in the experiments of this study we use a minimum of 7 decision trees, and a minimum of 100 records in each leaf. In cases where the training dataset (i.e. the clean records of the dataset) is too small to accommodate these parameters, we use a minimum record number between 20 and 5. Forests are generated using the SysFor (Islam & Giggins 2011) algorithm, and trees are generated using C4.5 (Quinlan 1993). The results are shown in Figure 2, with k-Means SiMI (shortened to kSiMI) indicating its fitness function as either f_1 (kSiMi f1) or f_2 (kSimi f2) from Equations 3 and 4.

As mentioned previously, the results in this paper are presented using *index of agreement* (d), defined for a record r_i in (Junninen et al. 2004) as:

$$d = 1 - \left[\frac{\sum_{j=1}^{|A|} (r_{ij} - r'_{ij})^z}{\sum_{j=1}^{|A|} (|r_{ij} - \mu_j| + |r'_{ij} - \mu_j|)^z} \right] \tag{5}$$

where z is either 1 or 2,r_{ij} is the original value for A_j (before missing values are added), and r'_{ij} is the imputed value for A_j. We use $z = 2$, thus the designation d_2. We present the average d_2 over the whole dataset, where a higher value is better.

Our results show three important characteristics. Firstly, the new variant only provides a consistently better result on the AutoMPG dataset. This may be due to AutoMPG having a small number of records, or due to the dataset having clearly defined clusters. Secondly, on the German CA dataset, the fitness function f_1 when used with k-Means SiMI outperforms SiMI, whereas f_2 does not. This is in contrast to on the AutoMPG dataset, where in the situation that both fitness functions with k-Means SiMI, f_2 outperformed f_1 Finally, and as a consequence of the previous two characteristics, we can see that the fitness function appears to have a large impact on the performance of the k-means SiMI algorithm.

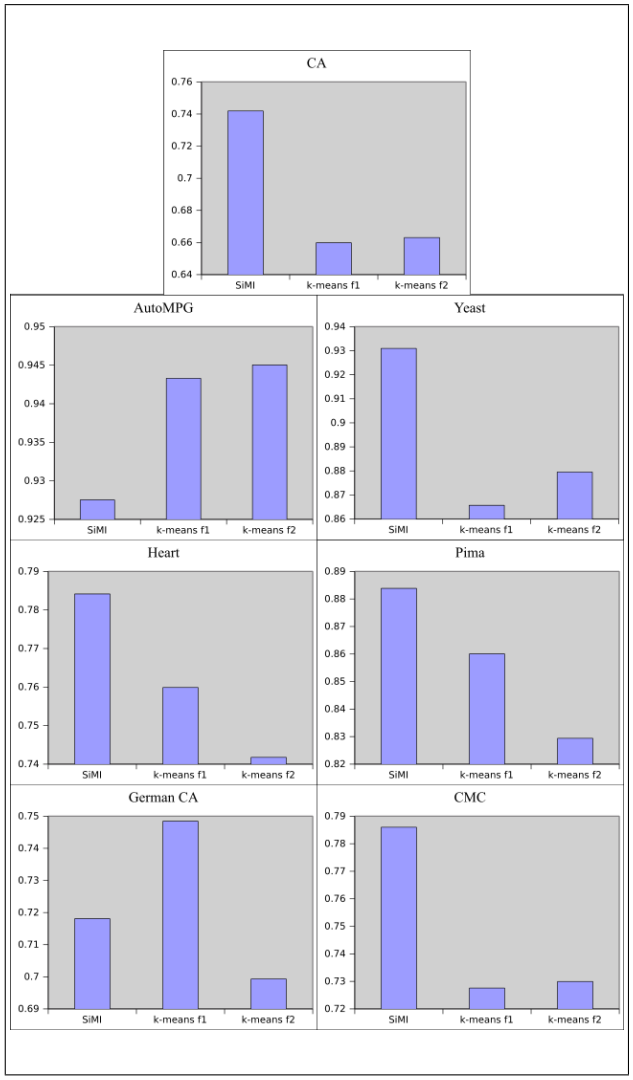


Figure 2: k-means SiMI d_2 results (higher the better)

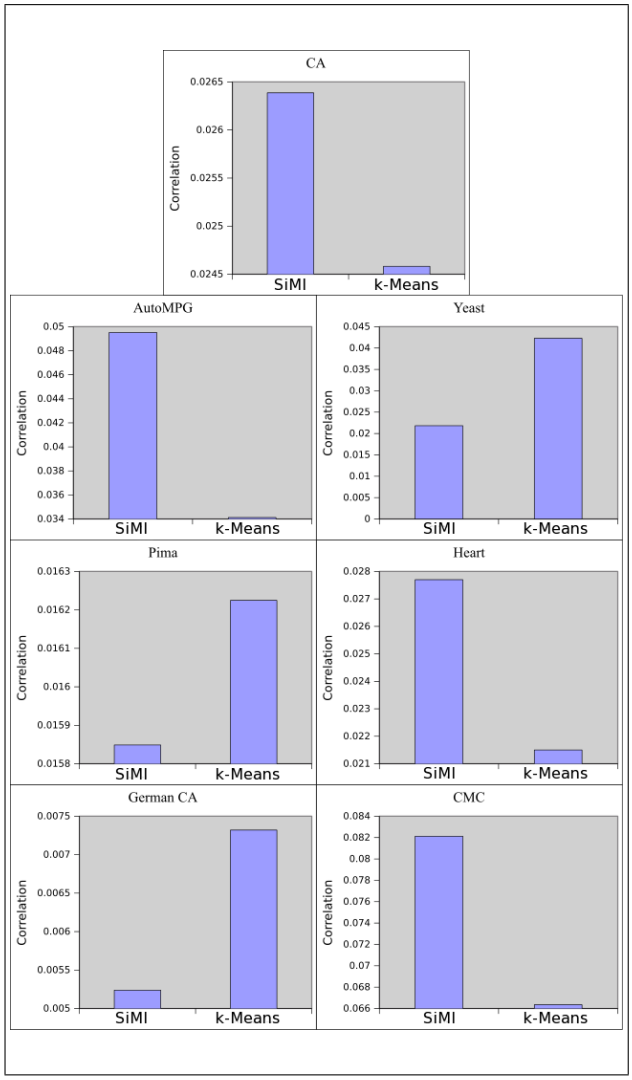


Figure 3: Results of correlation tests within partitions

The results obtained from testing this modification compared with the existing algorithm SiMI are thoroughly interesting, and from them we can derive several meaningful and justified directions for further research. We see that k-means partitioning performs with less efficacy than SysFor partitioning, likely due to the enhanced ability of a decision forest for taking into account categorical attributes for partitioning. Further tests should be completed using categorical attributes during clustering, although we advise the reader to use distance measures which are better suited to categorical attributes rather than the naive approach described earlier for this. Some of these include Eskin (Xiang & Islam 2014), IOF (Xiang & Islam 2014), and Gambryan (Xiang & Islam 2014), each of which has been shown to work with differing degrees of efficacy depending on the nature of the dataset (Xiang & Islam 2014). Perhaps a new, more powerful fitness function needs to be used, taking into account a combination of both similarity and correlation. At any rate, this seems to indicate that further research into the field of missing value imputation with regards to finding horizontal partitions needs to be focused on using *more sophisticated* methods of finding the partitions, rather than just k-means on numerical attributes with the specified fitness functions.

In order to gain an understanding of the results, we decided to test the average correlation within SiMI intersections and the k-means clusters our algorithm generates (using the norm of the correlation matrix for the subsets). The results of this can be seen in Figure 3. Not surprisingly, SiMI had a higher correlation on most datasets, although there are some peculiarities. Firstly, AutoMPG, the dataset on which the new algorithm performs best, does not have a higher average correlation between attributes inside the k-means clusters. Also, on German CA, Pima and Yeast we find that the k-means clusters have a higher average correlation than SiMI intersections. This may point to something else providing the enhancement in imputation accuracy for EMI, rather than the correlation being high as was suggested previously (Rahman & Islam 2013a).

Certainly there are upsides to the k-Means SiMI algorithm. Firstly, in that it is less complex than an algorithm such as SiMI. Secondly, it requires only 3 parameters in comparison to the at least 8 required by SiMI. In the past, in order to improve the horizontal segments found by DMI, a pre-imputation step was used. In the next section we discuss the use of this technique to improve the quality of clustering performed by k-Means SiMI to achieve an improved imputation results.

4 Consecutive layers of imputation

Techniques such as EDI (Rahman & Islam 2013*b*) have effectively used multiple consecutive layers of imputation in order to provide a better final imputation result. IBLLS similarly uses a self updating threshold to iteratively perform imputations that improve over time (Cheng et al. 2012). Subsequently, one may expect that by applying the same logic to the dataset before running the clustering algorithm k-means may improve the clusters and thus allow for a better imputation result.

EDI works by running a pre-imputation step before creating decision trees. In this step, the entire dataset has its numerical attributes imputed by EMI, and this imputed dataset is passed onto C4.5 to create a decision tree (Rahman & Islam 2013*b*). The theory behind this technique is that the imputed dataset should provide leaves more suitable for use in the DMI algorithm. The leaves found by the decision tree have the imputed numerical values replaced with their original missing values and are imputed using EMI and categorical attributes are imputed using a mode imputation.

This technique can also be applied to the k-Means SiMI algorithm described earlier in order to potentially achieve a better clustering result by allowing us to cluster all of the records at once. We call this new algorithm EKSiMI. We impute numerical attributes using EMI before using k-means to cluster the records, and then merge clusters that are below user defined minimum size τ. We then replace the records that originally had missing values within the final clusters. Finally, we impute as before, using EMI for numerical attributes within clusters and mode imputation using the modes of records within clusters for the categorical attributes. For the k-Means SiMI cluster fitness function, we have used f_1 described earlier.

The results found on EKSiMI (using the distance based fitness function) provide an interesting result (Figure 4). On most of the datasets, using EMI before the clustering process as a pre-imputation step provides a better final imputation results than standard k-Means SiMI, and makes it significantly more competitive with SiMI. This is likely due to the pre-imputation step allowing all records to be clustered at once, rather than clustering the clean records and assigning missing value records to the closest cluster afterwards.

We see the results of EKSiMI as promising, and indicative of the potential of using cluster-based partition-discovering techniques in the future. Future cluster-based partition-discovering techniques should however attempt to use a pre-imputation step for potentially better results. Perhaps with this pre-imputation step and a better fitness function, k-Means SiMI could become a very strong contender for missing value imputation. This technique, like EDI before it, suffers from a key problem however: if the initial imputation is incorrect or of poor quality, the second imputation will suffer dramatically. Unfortunately, while ever we use consecutive imputations, we create a dependency on the quality of the first imputation - something which can never be guaranteed, especially when there is a high volume of missing values in the original dataset.

The following section continues the idea of using several imputations, but we remove this dependency on a single pre-imputation by performing several imputations simultaneously in order to define our proposed technique.

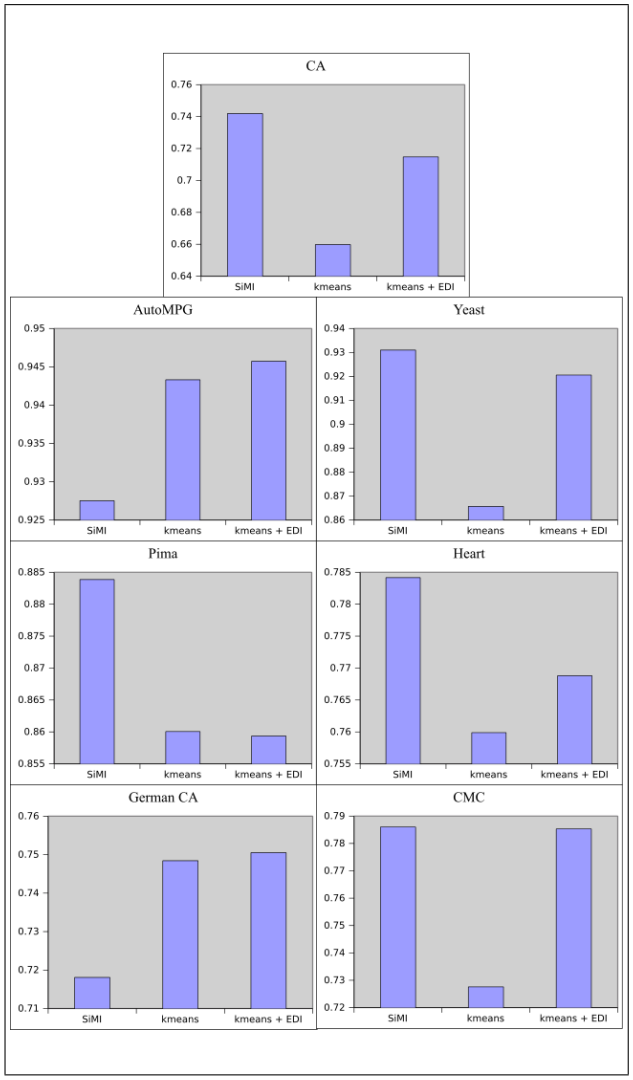
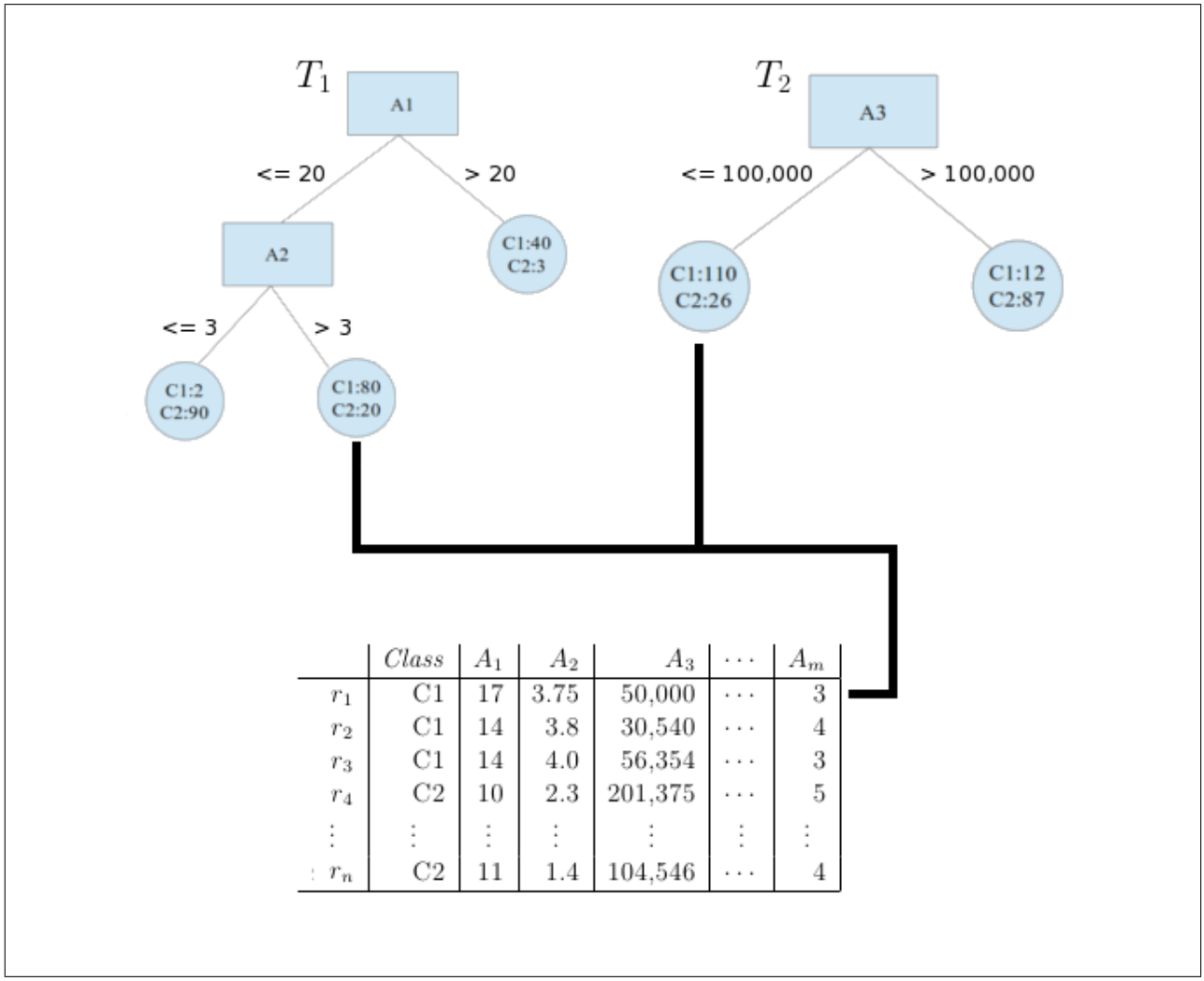


Figure 4: k-Means + EDI d_2 results (higher the better)

5 The Proposed Technique: Multiple Parallel Imputations

Multiple imputation (not to be confused with multiple consecutive layers of imputation) is a powerful technique which relies on taking multiple independent plausible imputed values for a missing value and combining them to find an improved imputation result (Schafer & Graham 2002). SiMI and the new technique k-Means SiMI both provide the ability to intuitively get such an imputation result. A decision forest (such as SysFor (Islam & Giggins 2011)) creates multiple trees based on considering different attributes as the root node and selecting different splitting points when generating the trees. If we consider each of the subsets found in these decision tree leaves to be a plausible subset for classifying a record, and instead perform EMI on the subset in order to impute missing values, then we create a system where we use several samples of the dataset to generate imputation results which should be close to the truth, as we already know that EMI performs better within the leaves of a decision tree (Rahman & Islam 2013*b*).

Figure 5 provides an example of how a single record within a dataset of size $n \times m$ falls into a single leaf in a tree, but several in a forest. Each of the rectangular nodes represents a splitting point based



	Class	A_1	A_2	A_3	$\cdots$	A_m
r_1	C1	17	3.75	50,000	$\cdots$	3
r_2	C1	14	3.8	30,540	$\cdots$	4
r_3	C1	14	4.0	56,354	$\cdots$	3
r_4	C2	10	2.3	201,375	$\cdots$	5
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
r_n	C2	11	1.4	104,546	$\cdots$	4

Figure 5: Example of leaves in a decision forest.

on one of the m attributes in the dataset. The lines which proceed from these rectangular nodes explain the rules upon which the forest has split the data to create the subsets used in its child nodes. Circular nodes represent leaves, and the amount of records of each class that falls within each leaf is printed inside the node. The forest is built on the clean records of the dataset, of which there are 235 in this example. Let us assume that r_1 has a missing value for attribute A_j. In order to impute the value for A_j, we first must see which leaf r_1 falls into for each tree. In the figure, we see that record r_1 falls into the middle leaf of T_1 and leftmost leaf of T_2 (this is indicated by the thick line linking the record to the leaves). We take the subset found from T_1 containing 100 records, 80 of which are in class C1 and 20 of which are in class C2, and append our missing-value record r_1. With this subset we perform EMI to impute missing numerical attributes, and perform a mode imputation to impute missing categorical attributes. This gives us an imputation we store as r_{ij}^1, as this is the imputation for value r_ij with the 1st tree. This process is repeated for each of the T trees in the forest. The second imputation r_{ij}^2 will use a subset with 136 other records based on the leaf r_1 falls into for T_2.

In the process of modifying SiMI to use this technique we can skip the step of intersection, but still use SiMI's merging strategy amongst the leaves of individual decision trees in order to ensure we have leaves of sufficient size for our imputations. Each imputation for attribute A_j in r_i can be represented as r_{ij}^l, and we express the full imputation as:

$$r_{ij} = \frac{\sum_{l=1}^{T} r_{ij}^l}{T} \tag{6}$$

Making such a simple change may seem trivial, however the results speak for themselves (Figure 6). MultiSiMI (as we have dubbed this algorithm) outperforms SiMI on 6 of the 7 datasets we have presented. Of all the modification techniques attempted so far, it would appear that multiple imputation can provide the key to unlocking the potential of *partition-discovering* imputation techniques in the future. We can see in MultiSiMI the true potential for improvement that exists when making intelligent novel modifications to existing *partition-discovering* imputation algorithms. This technique could be applied to many other algorithms in both creative or trivial ways. An example of a trivial solution would be to perform a k-NN imputation multiple times with many k values, and average the results.

6 Conclusion

In this paper we have proposed a new technique based on combining a partition-discovering approach with

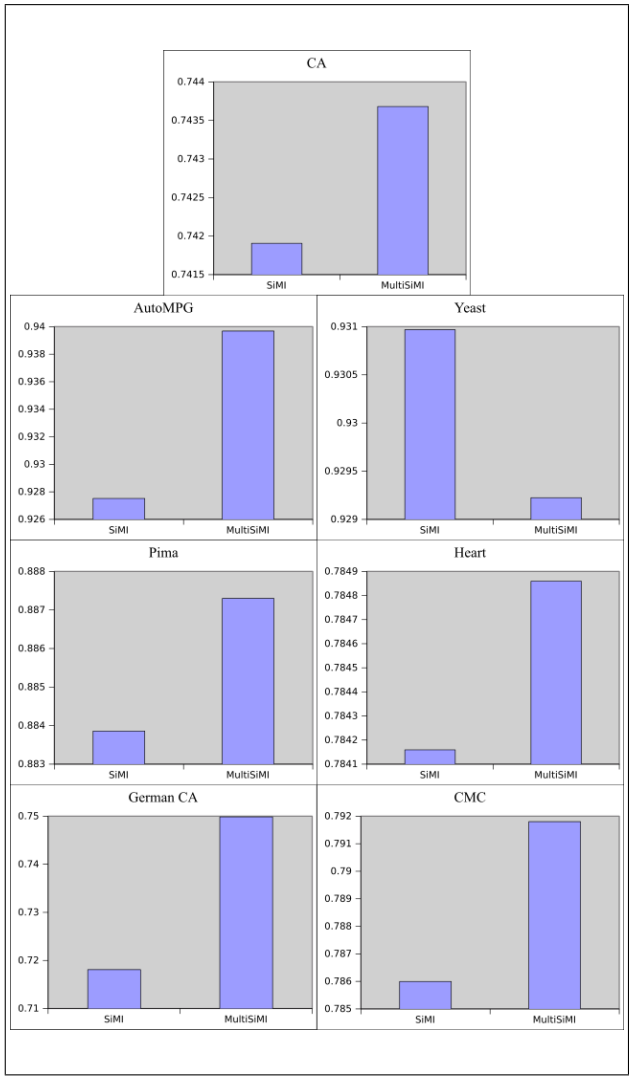


Figure 6: MultiSiMI d_2 results (higher the better)

multiple imputation, discussed variants to partition-discovering algorithms and how they effect imputation accuracy, and classified several existing techniques into two classes: *global* and *partition-discovering*. We have thoroughly examined the inner-workings of some of these existing algorithms and shown through the development of our variants to SiMI and final proposed technique MultiSiMI how the process of making intelligent changes to existing missing value imputation algorithms provides grounds for future research and stronger imputation results. Our proposed algorithm, MultiSiMI, performs better than the original algorithms it is based on in six out of seven tested situations and is the major contribution of the paper. The others perform better on certain patterns of missingness, and particularly perform well on the AutoMPG dataset. A comprehensive analysis should be undertaken in the future to see what exactly makes these algorithms perform to a higher degree of success on particular datasets. The issue of parameters having a wildly unpredictable impact on decision tree based partition-discovering techniques is addressed and related to this issue. This is another potential field in which further research should be conducted.

This paper achieves its aim of understanding the state of the art for missing value imputation and showing how that understanding can be translated into successful new algorithms. By examining the impact on imputation accuracy caused by the proposed changes, we get an even better understanding of how the state of the art will change in the near-future. The process of data cleansing is essential in the field of data analysis, and so it is important that the processes used are practical and make use of the most appropriate techniques. We show how altering the method for partition-discovery in existing algorithms effects the imputation result through k-Means SiMI, show how using an early imputation step can be used to increase the quality of the final imputation result using EKSiMI, and explore the use of multiple imputation in order to remove any dependencies between consecutive imputation steps and find a higher quality overall imputation result, giving us our final proposed technique. The design philosophy for MultiSiMI can easily be translated to create many other novel techniques based on the multiple imputation paradigm.

In many situations, it will be difficult to ascertain the correct parameters for missing value imputation algorithms, so we believe that an important step in the future is to develop techniques that require as little user input as possible. Complexity is also a major issue which is not addressed in many existing algorithms, with those algorithms instead being focussed on providing accuracy by any means necessary. Big data is said to consist of "three v's" - velocity, variety, and volume (Zikopoulos et al. 2011). The issue of complexity becomes increasingly important when we consider the volume of the data we are dealing with and the velocity at which it arrives - which of course requires it to be cleansed in a timely manner. None of the existing techniques discussed take this into serious account - so we see this as a strong contender for future research.

The problem of missing values in datasets is by no means solved. Extensive changes to the field are expected to take place over the next few years, as better techniques are discovered and developed. This paper demonstrates the huge potential of the field's future development and the practicality of employing these techniques.

References

Aydilek, I. B. & Arslan, A. (2013), 'A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm', *Information Sciences* **233**, 25–35.

Bache, K. & Lichman, M. (2013), 'UCI machine learning repository'.
URL: *http://archive.ics.uci.edu/ml*

Cai, Z., Heydari, M. & Lin, G. (2006), 'Iterated local least squares microarray missing value imputation', *Journal of bioinformatics and computational biology* **4**(05), 935–957.

Cheng, K.-O., Law, N.-F. & Siu, W.-C. (2012), 'Iterative bicluster-based least square framework for estimation of missing values in microarray gene expression data', *Pattern recognition* **45**(4), 1281–1289.

Dempster, A. P., Laird, N. M., Rubin, D. B. et al. (1977), 'Maximum likelihood from incomplete data via the em algorithm', *Journal of the Royal statistical Society* **39**(1), 1–38.

Islam, M. Z. & Giggins, H. (2011), Knowledge discovery through sysfor: a systematically developed

forest of multiple decision trees, *in* 'Proceedings of the Ninth Australasian Data Mining Conference-Volume 121', Australian Computer Society, Inc., pp. 195–204.

Jain, A. K. (2010), 'Data clustering: 50 years beyond k-means', *Pattern Recognition Letters* **31**(8), 651–666.

Junninen, H., Niska, H., Tuppurainen, K., Ruuskanen, J. & Kolehmainen, M. (2004), 'Methods for imputation of missing values in air quality data sets', *Atmospheric Environment* **38**(18), 2895–2907.

Mallinson, H. & Gammerman, A. (2003), 'Imputation using support vector machines'.

Nelwamondo, F. V., Mohamed, S. & Marwala, T. (2007), 'Missing data: A comparison of neural network and expectation maximisation techniques', *arXiv preprint arXiv:0704.3474* .

Quinlan, J. R. (1993), *C4. 5: programs for machine learning*, Vol. 1, Morgan kaufmann.

Rahman, G. & Islam, M. Z. (2011), A decision tree-based missing value imputation technique for data pre-processing, *in* 'Proceedings of the Ninth Australasian Data Mining Conference-Volume 121', Australian Computer Society, Inc., pp. 41–50.

Rahman, M. G. & Islam, M. Z. (2013*a*), 'Missing value imputation using decision trees and decision forests by splitting and merging records: Two novel techniques', *Knowledge-Based Systems* **53**, 51–65.

Rahman, M. G. & Islam, M. Z. (2013*b*), 'A novel framework using two layers of missing value imputation', *Conferences in Research and Practice in Information Technology* **146**.

Rahman, M. G. & Islam, M. Z. (2014), 'Fimus: A framework for imputing missing values using co-appearance, correlation and similarity analysis', *Knowledge-Based Systems* **56**, 311–327.

Schafer, J. L. & Graham, J. W. (2002), 'Missing data: our view of the state of the art.', *Psychological methods* **7**(2), 147.

Schneider, T. (2001), 'Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values.', *Journal of Climate* **14**(5).

Su, J. & Zhang, H. (2006), A fast decision tree learning algorithm, *in* 'Proceedings of the National Conference on Artificial Intelligence', Vol. 21, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, p. 500.

Tresp, V., Neuneier, R. & Ahmad, S. (1995), Efficient methods for dealing with missing data in supervised learning, *in* 'Advances in neural information processing systems', MORGAN KAUFMANN PUBLISHERS, pp. 689–696.

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D. & Altman, R. B. (2001), 'Missing value estimation methods for dna microarrays', *Bioinformatics* **17**(6), 520–525.

Wang, Y., Wang, L., Yang, D. & Deng, M. (2014), 'Imputing missing values for genetic interaction data', *Methods* **67**(3), 269–277.

Willmott, C. J. (1982), 'Some comments on the evaluation of model performance', *Bulletin of the American Meteorological Society* **63**(11), 1309–1313.

Xiang, Z. & Islam, M. Z. (2014), The performance of objective functions for clustering categorical data, *in* 'Knowledge Management and Acquisition for Smart Systems and Services', Springer, pp. 16–28.

Zikopoulos, P., Eaton, C. et al. (2011), *Understanding big data: Analytics for enterprise class hadoop and streaming data*, McGraw-Hill Osborne Media.

Designing a Knowledge-based Schema Matching System for Schema Mapping

Sarawat Anam[1], Yang Sok Kim[2], Byeong Ho Kang[1] and Qing Liu[3]

[1]{Sarawat.Anam,Byeong.Kang}@utas.edu.au
School of Engineering and ICT, University of Tasmania, Hobart, Australia

[2]yangsokk@gmail.com
Department of Management Information Systems, Keimyung University, Korea

[3]Q.Liu@csiro.au
Autonomous Systems, CSIRO Computational Informatics, Hobart, Australia

Abstract

Schema mapping that provides a unified view to the users is necessary to manage schema heterogeneity among different data sources. Schema matching is a required task for schema mapping that finds semantic correspondences between entity pairs of schemas. Semi-automatic schema matching systems were developed to overcome manual works for schema mapping. However, such approaches require a high manual effort for selecting the best combinations of matchers and also for evaluating the generated mappings. In order to avoid such manual works, we propose a Knowledge-based Schema Matching System (KSMS) that performs schema mapping both at the element and structure level matching. At the element level matching, the system combines different matching algorithms using a hybrid approach that consists of machine learning and knowledge engineering approaches. At the structure level matching, the system considers hierarchical structure that represents different contexts of a shared entity. The system can update knowledge if schema data changes over time. It also gives facilities to the users to verify and validate the schema matching results by incremental knowledge acquisition approach where rules are not predefined. Our experimental evaluation demonstrates that our system is able to improve the performance and to generate the accurate results.

Keywords: Schema matching, schema mapping, knowledge-based approach, element level and structure level matching.

1 Introduction

Schema matching is necessary to overcome semantic heterogeneity problem as the schemas are designed by different people. It finds mappings between semantically related entity pairs of schemas. These mappings are used to integrate data residing in different sources, and to make knowledge discovery easy and systematic. Schema matching can be done at the element level and structure level. Element level matching only considers matching names of the entity pairs, and it can be done by string similarity metrics and text processing techniques. Different string similarity metric and text processing technique perform well for different schema data. This is because the schema data contains different characteristics such as identical, abbreviated, synonym and combined words. In addition, the techniques generate schema matching problems: false positive (if reported match by expert is false and predicted match by algorithm is true), and false negative (if reported match by expert is true and predicted match by algorithm is false). Therefore, it is necessary to combine these techniques effectively, and to handle the matching problems.

Some solutions have been proposed in the literature to combine the techniques. YAM (Duchateau et al., 2009) uses machine learning approach for combining the techniques at the element level. The system shows if false positives are high, then precision becomes low. If false negatives are high, then recall becomes low. Precision can be 1.0 if false positive becomes zero. If precision becomes high but recall becomes very low, then overall performance becomes very low. For this, it is very important to increase the value of recall. The system runs much iteration until the similarity scores between entities become stable and it removes some incorrect mappings (pre-defined). However, it takes much time to iterate many times, and it needs to rebuild a training model if schema data changes overtime.

Incremental knowledge engineering approach, Censored Production Rules (CPR) based Ripple-Down Rules (RDR) has been used by (Anam et al., 2014) for schema mapping. The approach uses the features created by the combination of string similarity metrics and text processing techniques for creating rules. However, the limitation of the approach is that it is time-intensive to create rules for mapping entity pairs one by one at the element level. In order to overcome the limitations of the above approaches, it is necessary to use a hybrid approach that combines both machine learning and knowledge engineering approaches at the element level. Element level matching does not only give proper results for schema mapping as it only considers matching names of the entities. So it is important to do structure level

matching to get the accurate results. Structure level matching uses the result of element level matching and considers the hierarchical structure that represents different contexts of a shared entity. In order to get the final mapping results, it is necessary to combine the results of the element level and structure level matching using some aggregation functions. For determining the best suitable aggregation function, it is necessary to compare the performance of the functions.

In this research, we introduce a Knowledge-based Schema Matching System (KSMS) that matches schemas both at the element level and structure level and produces the final result. We use the following processes in the system:

- We use hybrid-RDR (Anam et al., 2015) approach at the element level matching. The approach consists of decision tree, J48 and incremental knowledge engineering approach, Censor Production Rules (CPR) based Ripple-Down Rules (RDR). We combine the similarity values of different string similarity metrics and text processing techniques for constructing features. These features are fed into J48 to generate matching results. If J48 generates some wrong matchings, then CPR based RDR is used for correcting and validating the matching results.
- We use graph matching algorithm, Similarity Flooding (Melnik et al., 2002) that matches schemas considering structural information to discover additional mappings.
- We combine the results of the element level and structure level using aggregation functions to get the final results. We compare the performance of the aggregation functions and choose the best one.

2 Basic Definitions

In this section, we give some basic definitions of the foundations of schema matching and mapping.

A **schema** is defined as a formal structure that represents a set of entities. Each schema entity has a name, a data type, a description (called annotation) as well as instances. The kind of schemas can be database schemas, XML-schemas, entity-relationship diagram, and ontology description.

Schema mapping takes as input two schemas, each consisting of a set of discrete entities, and determines as output the relationships holding between these entities (Cate et al., 2013).

Schema matching is a process that discovers mappings between similar or same entity for a given entity using matching algorithms. We give an example of schema matching and mapping in the following:

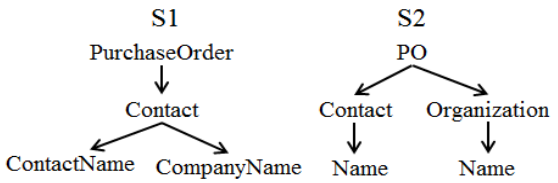


Fig. 1. Example of two schemas

For illustrating schema matching and mapping problem, we use two schemas, S1 and S2 representing the information of purchase order domain and the schemas are shown in **Fig.1**. These schemas contain different types of characteristics such as identical, abbreviated, synonym and combined words. Each schema consists of a set of schema entities. Similar types of schema entities are found in these datasets. For example, *PO* is an abbreviation of *PurchaseOrder* and *Company* is synonym of *Organization*. Schema matching is done at the element and structure level.

Element Level Matching considers only matching names of the entities. The basic techniques of this matching are string similarity metrics and text processing techniques. **String similarity metrics** compare the names of the schemas in order to produce a degree of similarity. **Text processing techniques** such as tokenization, abbreviation expansion and synonym lookup processes the names of the entities before matching. For example, *PO* is expanded to *PurchaseOrder* using abbreviation expansion.

Similarity measures produce numeric value ranging from 0 to 1 in normalized similarity metrics, schema mapping decision is Boolean – TRUE or FALSE. In order to take decision whether or not the source and the target entities are matched, a threshold value is specified. For example, Levenshtein string metric produces similarity value 0.4 between *ContactName* of S1 and *Name* of S2. If the threshold value is 0.4 for determining correct mapping that means the algorithm considers that all the pairs of entities with a confidence measure greater than or equal to 0.4 as correct mapping entities. Then the matching algorithm returns mapping decision to the user is TRUE. Another matching algorithm matches *CompanyName* in S1 and *Organization* in S2 using the combination of tokenization and synonym look up. First, *CompanyName* is tokenized as *{Company, Name}* and then *Company* and *Organization* are matched according to the meaning of the entities using synonym lookup and returns to the user that mapping decision is TRUE.

Only string similarity metrics and text processing techniques do not produce good performance for schema mapping. Therefore, it is necessary to use some combination functions such as machine learning algorithms, knowledge engineering approaches, neural network and hybrid approaches.

Structure level matching considers matching hierarchical structure of a full graph. In **Fig.1**, the hierarchical structure matching such as PurchaseOrder.Contact.contactName→ PO.Organization.Name is FALSE.
However, PurchaseOrder.Contact.companyName→ PO.Organization.Name is TRUE. This is because company and organization are matched according to the hierarchical context.

3 Related Works

There are some systems for schema mapping in the available literature. Lee and Doan developed a machine learning based approach, eTuner (Lee et al., 2007) to automatically tune schema matching systems to the problems. The approach handles relational schemas and considers only 1:1 mappings between schema pairs. It uses name matchers such as edit-distance and q-gram as

terminological matchers. It can match source schema against synthetic schemas, for which the ground truth mapping is known, and can find a tuning in order to improve the matching performance of source schema against real schemas. It needs user assistance to improve the tuning quality by getting suggestion about the domain-specific perturbation rules. As the perturbation rules are known, so the mapping between original source and perturbed schema is also known. The approach is used for semantic matches and maintaining wrappers. However, the approach only considers source schema and ignores target schema, and tunes only small to moderate size schemas. Another problem is that the perturbation rules are static and so for different mapping problems, the generated gold standard does not differ much (Peukert et al., 2012).

Meta level learning (Eckert et al., 2009) is the first to recognize the need to have more schema features for creating adaptive processes. For this, the authors combine different matchers using machine learning techniques. They use the output of different matchers and additional features about the nature of the entities to be matched, as input for the learning approach. However, no suitable gold mappings are available for learning, and for this learned models often are not able to return results with a good quality. Besides, the learning approach easily overfits with the learning base, and the performance decreases significantly with increasing sizes of decision trees.

Duchateau et al. present an approach, MatchPlanner (Duchateau et al., 2008) for schema matching which uses a decision tree to combine the best suitable match algorithms. The approach inputs a set of schemas and a decision tree which is composed of match algorithms, and outputs a list of mappings which are validated by experts to find out whether the matching is correct or not. The feedback is used to feed into another decision tree for learning. YAM (Duchateau et al., 2009) is a machine learning based schema matching factory. In the learning phase, YAM considers users' requirement such as a preference for recall or precision, provided expert correspondences. It uses a Knowledge Base (KB) that consists of a set of classifiers, a set of similarity measures, and pairs of schemas which have already been matched. In the matching phase, the KB is used to match unknown schemas. In the system, users are asked to select appropriate classifiers. If the users do not have proper knowledge, then they depend on the default classifiers. However, the default classifiers often do not produce good performance. In addition, without proper knowledge, it is not easy to provide the preference between precision and recall. Machine learning techniques are promising for element similarity, but they need to rebuild the training model if schema data changes over time. Inversely, knowledge engineering approach encodes human knowledge directly, such that knowledgebase can be constructed with limited data.

Some systems have used knowledge engineering approach for schema matching. (Peukert et al., 2012) propose a self-configuring and adaptive schema matching system. It uses different terminological matchers such as name, datatypes, annotations, and synonyms using WordNet. In the structure level matching, it uses similarity propagation approach. The system depends on some features that are computed from input schemas and from intermediate mapping results. The features are then used in matching rules to select matchers, aggregation and selection operators. The rules represent expert knowledge on how to define or adapt schema matching processes. The matching process is iteratively extended, rewritten and executed in order to correct matching problems. However, the system predefines mapping rules such as starting, aggregation, rewrite, refine and selection. Therefore, the system faces problems when the viewpoint of two schemas is highly different. Second problem is that if some pre-defined mappings are incorrect and these methods are run only one time to produce new mappings, then the accuracy of new results will be unconfident. Third problem is that the system tunes matching processes manually and it does not split the process control flow based on the type of entities to be matched. Traditional rule-based systems require time-consuming knowledge acquisition as in those systems a highly trained specialist, the knowledge engineer, and the time-poor domain expert are necessary in order to analyze domain (Richards, 2009).

AMC (Peukert et al., 2011) is a schema and ontology matching framework where it is necessary for the users to provide an appropriate operator from different types of operators such as matcher, combination, selection, analyzer and blocking operators as input and to investigate individual results of individual operator. For this, users need to gather knowledge about the operators. If users want to use the default operator, then the operator may not handle different schemas of different domains.

In this research, we use Hybrid-RDR (Anam et al., 2015) approach that combines decision tree, J48 and incremental knowledge engineering approach, Censor Production Rules (CPR) based Ripple Down Rules (RDR) at the element level. In the approach, the KB is empty at the beginning, and the first rule is added to the KB by classifying a dataset using decision tree classification model. Then rules are added incrementally in order to solve schema matching problems such as false positives and false negatives. There are some advantages of the approach. First, only one classification model of decision tree is used in the approach, so it does not generate any over fitting problem. Second, rules are not pre-defined. Rules are created based on the features constructed from string similarity metrics and text processing techniques. Third, the approach does not need time consuming knowledge acquisition as rules are only created to correctly classify the wrongly classified cases produced by decision tree model. At the structure level matching, Similarity Flooding algorithm is used to match the hierarchical structure of a full graph.

4 KSMS Overview

The main components of KSMS system are described in **Fig.2**. The system discovers mappings between two schemas by element level and structure level matchers. The final mapping results are produced by using aggregation function. The functionalities of the system are described below:

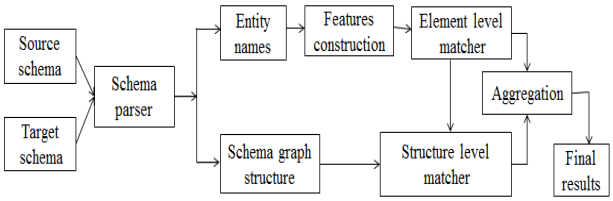


Fig.2. KSMS architecture

KSMS has been implemented in Java. It supports Graphical User Interface (GUI) for selecting schemas, displaying mapping knowledge created by feature construction process, classifying entities using J48 training model, creating rules for knowledge acquisition using features, checking satisfaction of rules, validating rules and also for saving rules to the Knowledge Base (KB).

In the system, any two schemas are first selected from repository. At the element level, input source and target schemas are parsed to extract names of the entities.

4.1 Feature Construction

Features of the entities are constructed using terminological matchers: text processing techniques and string similarity metrics. Feature construction processes are: **Step 1**, Cartesian product of the entities is generated. **Step 2**, three text processing techniques such as tokenization, abbreviation and acronym expansion, and synonym lookup are applied on the entities. **Step 3**, string similarity metrics are applied on the features of the attributes computed from the above two steps. We use string similarity metrics developed by two open source projects. For Levenshtein, JaroWinkler, Jaro Measure, TFIDF and Jaccard, we use open source library SecondString[1] and for Monge-Elkan, Smith-Waterman, Needleman-Wunsch, Q-gram and Cosine, we use SimMetric open source library[2]. Similarity values are normalized, such that the value within from 0 to 1, where 0 means strong dissimilarity and 1 means strong similarity. The threshold values for deciding schemas matching (true/false) are increased with 0.1 from 0 to 1. Another feature is created by using expert manual mapping (true/false). These features and features values are termed as attributes and cases respectively.

4.2 Element Level Matching

The extracted features including cases are fed into Hybrid-RDR approach. In the approach, knowledge base (KB) is empty at the beginning. First decision tree, J48 constructs a classification model using a small number of cases and uses the model for classifying the new cases. The decision tree rule is added in the KB as a first rule. Then users verify the results based on the expert manual mapping. If any case is wrongly classified (false positive/ false negative), new stopping rule is added to the KB to make the classification as NULL. The rule is created based on the features using a knowledge acquisition process of CPR based RDR (Kim et al., 2012). Knowledge acquisition is a process which transfers knowledge from human experts to knowledge based systems. The rule consists of one or more than one conditions. The condition has the form:

Attribute operator value

Where *attribute* is the feature, *operator* can be '=, !=, <, >, <=, >=', and *value* is the feature value. The conditions are added into a condition list to make rule. The rule is checked to determine whether it is satisfied by the current case or not. If the rule is satisfied, then the rule is validated on all the wrong classified cases to check whether other cases also satisfy the rule. The rule is saved in the KB as censor node which provides the classification of the wrongly classified cases as NULL. In order to correctly classify the NULL classified cases, alternative rules are added to the KB as child rules of the root rule for correctly classifying the cases as TRUE/FALSE.

The inference process is based on searching the KB represented as a decision list with each decision possibly refined again by another decision list. Once a rule is satisfied by any case, the process evaluates whether or not the exception rules are matched to the given case. If any exception rule is not satisfied, then the process stops with one path and one conclusion. However, if any exception rule is satisfied, the fired rule becomes zero according to censored conditions (Kim et al., 2012). Then other rules below the rule that was satisfied at the top level is evaluated. The process stops when none of the rules can be satisfied by the case in hand. The inference algorithm is the following:

1. Set lastFiredRule and CurrentRule as null
2. Get exceptionRule of rootRule
3. If exceptionRule is not null, set exceptionRule as currentRule
4. Evaluate inputCase with currentRule
   i. If inputCase satisfies currentRule, set currentRule as lastFiredRule and get exceptionRule of currentRule
      a. If exceptionRule is not null, set exceptionRule as currentRule and go to 4
   ii. Else get alternativeRule of currentRule
      a. If alternativeRule is not null, set alternativeRule as currentRule and go to 4
5. Stop inference process and return lastFiredRule

The mapping results produced by this approach at the element level are stored in a repository.

4.3 Structure Level Matching

At the structure level matching, input schemas are parsed and converted into graph data structure. Structure matching is used to adjust incorrect matches from matching phase, and it finds additional mappings. KSMS uses the results of element level to match schema graph structures based on a graph matching algorithm called Similarity Flooding (Melnik et al., 2002). The approach converts schemas into directed labelled graphs and uses

[1] http://secondstring.sourceforge.net
[2] http://sourceforge.net/projects/simmetrics

fix point computation to determine the matches between corresponding nodes of the graphs. It uses the concept that two nodes are matched based on the matching of neighborhood.

4.4 Final Results of Mapping

In this phase, we combine the mappings discovered from element level and structure level matching by weighted, average, minimum, maximum and harmonic mean aggregation methods. Different systems have used different aggregations function for combining mappings. In order to determine the best one, we compare the performance of all the aggregation functions. We define the similarity values found from element level matching and structure level matching by *esim* and *ssim* respectively. The aggregation functions are described below:

- **Weighted**: This strategy returns a weighted sum of the similarity values. The similarity value found from structure level matching is used as the threshold value which is the weight of element level matching, and the weight for structure level matching, W_{struct} is (1-threshold) (Ngo et al., 2011b). The weighted similarity of the entity pair, $e1$ and $e2$ is calculated as:

$$wsim(e1, e2) = W_{struct} \,.\, ssim(e1, e2) + (1 - W_{struct}).\, esim(e1, e2)$$

This combination strategy is used in some matching systems (Do and Rahm, 2002, Ngo and Bellahsene, 2012, Madhavan et al., 2001).

- **Average:** The average similarity is calculated by dividing the sum of the similarity values of two string metrics for each name pair by the total number of similarity functions. Average value is calculated by the following function:

  Avg= (*esim*+*ssim*)/2

The matching systems which use this strategy are (Do and Rahm, 2002, Volz et al., 2009, Jimenez et al., 2009).

- **Minimum:** This strategy returns the minimum similarity value between two string metrics. Minimum value is calculated by using the following function:

  Min=Math.min (*esim*, *ssim*)

- **Maximum**: This strategy returns the maximum similarity value between two string metrics. Maximum value is calculated by using the following function:

  Max=Math.max (*esim*, *ssim*)

The combination strategies, minimum and maximum are used in some matching systems (Do and Rahm, 2002, Volz et al., 2009, Massmann and Rahm, 2008).

- **Harmonic mean:** Harmonic mean is calculated by the following function:

  Harmonic mean=2**esim***ssim*/ (*esim*+*ssim*)

This combination strategy is used in the systems (Do and Rahm, 2002, Ngo et al., 2011a).

5 Experimental Design

5.1 Datasets

Five XDR schemas of purchase order domain, such as CIDX, EXCEL, NORIS, PARAGON and APERTUM obtained from www.biztalk.org are used for this evaluation study. We denote the schema datasets CIDX, EXCEL, NORIS, PARAGON and APERTUM by C, E, N, P, and A respectively. These schema datasets are used for schema mapping evaluation and terminological matching evaluation (Peukert et al., 2011). These schema datasets contain different types of characteristics such as identical words, combined words, abbreviated words and synonym words. Each schema dataset contains 35 (E), 30 (C), 46 (N), 82 (A), 59 (P) entities.

5.2 Experimental Procedure

In this research, we experiment ten matching tasks one-by-one using all combinations of five schema datasets such as C-E (first matching task is to deal with two datasets, CIDX and EXCEL), C-N, C-P, C-A, E-N, E-P, E-A, N-P, N-A and P-A. We take the Cartesian product of the schema datasets for ten matching tasks separately. The sizes of Cartesian product of the matching tasks are 1050 (C-E), 1380(C-N), 1770(C-P), 2460(C-A), 1610(E-N), 2065(E-P), 2870(E-A), 2714(N-P), 3772(N-A) and 4838(P-A) entity pairs respectively. We denote the matching tasks C-E, C-N, C-P, C-A, E-N, E-P, E-A, N-P, N-A and P-A by D1, D2, D3, D4, D5, D6, D7, D8, D9 and D10 respectively.

In the evaluation approach, we feed the datasets in to the static decision tree, dynamic decision tree (DT) approaches and Hybrid-RDR approach. The approaches learn a new model by including newly available data. We use the decision tree to compare the performance to the existing approaches as some systems, YAM (Duchateau et al., 2009) and MatchPlanner (Duchateau et al., 2008) use decision tree as a combination method. Here we divide the decision tree into static and dynamic decision tree. In the static decision tree, one dataset is used for building a training model and another dataset is used for testing. In the dynamic decision tree, one dataset is used for building a training model and test the test dataset. Then two datasets are combined and used for building a training model and test the test dataset. Incrementally, all the datasets except the test dataset are used for building a training model and test the test dataset.

We perform ten experiments to get the performances (precision, recall and F-measure) of the static decision tree, dynamic decision tree (DT) and Hybrid-RDR approaches. In all experiments, we randomly select datasets for training and testing. For example, we select D1 for training and D10 for testing, D7 for training and D3 for testing, D4 for training and D9 for testing. In such a way, we select the datasets for training and testing. The evaluation processes of the approaches are described below:

5.2.1 Static DT

In the static decision tree approach, we create decision tree model, $\boldsymbol{ML_0}$ for D1 and test D10. Then we create

$\boldsymbol{ML_1}$ for D2 and test D10. In this way, we create $\boldsymbol{ML_2}$ for D3 to $\boldsymbol{ML_8}$ for D9 and test D10. For other combination, we create $\boldsymbol{ML_0}$ for D7 and test D3, $\boldsymbol{ML_1}$ for D8 and test D3. In this way, we create $\boldsymbol{ML_8}$ for D1 and test D3.

5.2.2 Dynamic DT

In the dynamic decision tree approach, we create decision tree model, ML_0 for D1 and test D10. Then we incrementally add other datasets like D1+D2, D1+D2+D3 for creating decision tree models, ML_1, ML_2 respectively and test D10. In this way, we add all nine datasets for creating decision tree model, ML_8 and test D10.

For all decision tree approaches, we consider 10-fold cross validation. 10-fold cross validation means that the data is split into 10 groups where nine groups are considered for training and the remaining one group is considered for testing. This process is repeated for all 10 groups. For all experiments using decision tree, we use WEKA (Hall et al., 2009) data mining and machine learning toolbox.

5.2.3 Hybrid-RDR

In the Hybrid-RDR approach, we create decision tree model, ML_0 for D1 and test D10. We also test D2 and find some wrong classified cases. Then we refine the decision tree rule by adding censor/exception/stopping rule, $Rule_0$ and again classify the cases by adding alternative rule, $Rule_0$. The censor rules are added as censor nodes of decision tree in the KB and alternative rules are added as parent rules in the KB. The ML_0+$Rule_0$ is then used for testing D10 and also for testing D3. We add rule, $Rule_1$ again for the wrong classified cases of D3, and ML_0+$Rule_0$+$Rule_1$ is used for testing D10. In such a way, we incrementally add rules for all nine datasets, ML_0+$Rule_0$+$Rule_1$+…+$Rule_8$ and test D10.

5.3 Evaluation Metrics

As this task is a classification task, we use the following conventional metrics: precision = $\frac{TP}{TP+FP}$, recall = $\frac{TP}{TP+FN}$ and F-measure = $\frac{2*precision*recall}{precision+recall}$, where TP is True Positive (hit), FP is False Positive (false alarm, Type I error) and FN is False Negative (miss, Type II error). For a specific threshold value, we calculate TP, FP and FN by comparing manually defined matches (R) with the predicted matches (P) returned by the matching algorithms according to (Jimenez et al., 2009).

6 Evaluation Results

Performance of the static decision tree, dynamic decision tree and Hybrid-RDR approaches depends on the features of the datasets which are created using string similarity metrics and text processing techniques. The performance of Hybrid-RDR approach also depends on the efficient knowledge acquisition. We compute performance in terms of precision, recall and F-measure. Precision estimates the reliability of the match predictions and recall specifies the share of real matches. During schema mapping, manually matching schemas of two heterogeneous data sources and false identified matches by algorithms are handled by humans. The burden of deleting false identified matches is much easier than creating manual matches among thousands of schemas (Stoilos et al., 2005). As for calculating recall value, manually identified matches are necessary, so recall value is very important. Only precision or recall cannot estimate the performance of match algorithms (Cheng et al., 2005). So it is necessary to calculate the overall performance or F-measure of Hybrid-RDR approach and both static and dynamic decision tree using both precision and recall. For this, we determine the best performing classification system based on the optimized F-measure (Marie and Gal, 2008) for almost all experimental datasets.

6.1 Schema Mapping Results at the Element Level

At the element level, the names of the entities are matched by static decision tree, dynamic decision tree and Hybrid-RDR approaches. For all the above three approaches, we perform ten experiments and compute average performance of the experiments. In all experiment, we randomly select datasets for training and testing. We compare the performance of the approaches to other approaches, AMC (Peukert et al., 2011), COMA (Do and Rahm, 2002), FALCON (Hu et al., 2008), RONDO (Melnik et al., 2003) based on F-measure. The performance, F-measures of these approaches are found from AMC. All F-measure of the approaches are described in **Table 1**. The *Datasets* column describes the datasets used for the experiments. The other columns, *AMC*, *COMA*, *FALCON* and *RONDO* represent F-measure of these approaches. We denote static decision tree, dynamic decision tree and Hybrid-RDR by *S_DT*, *D_DT* and *HRDR* respectively. The schema mapping result found from element level matching is described in **Table 1**.

Datasets	AMC	COMA	FALCON	RONDO	S_DT	D_DT	H-RDR
D1	0.44	0.42	0.38	0.41	0.81	0.85	0.90
D2	0.71	0.63	0.62	0.43	0.74	0.87	0.89
D3	0.59	0.51	0.55	0.53	0.65	0.78	0.85
D4	0.52	0.46	0.35	0.47	0.62	0.76	0.87
D5	0.45	0.42	0.42	0.41	0.74	0.82	0.86
D6	0.65	0.60	0.70	0.60	0.67	0.84	0.90
D7	0.51	0.48	0.44	0.45	0.66	0.79	0.88
D8	0.55	0.50	0.54	0.55	0.64	0.76	0.85
D9	0.41	0.34	0.39	0.28	0.68	0.75	0.83
D10	0.30	0.31	0.25	0.25	0.56	0.60	0.80
AVG	**0.51**	**0.48**	**0.47**	**0.44**	**0.68**	**0.79**	**0.86**

Table 1. F-measures compariosn of the approaches

In **Table 1**, we compare performance, F-measure of some previous approaches to the static decision tree, dynamic decision tree and Hybrid-RDR. We find that our approaches show better performance compared to AMC, COMA, FALCON and RONDO. The average performances of these approaches are 0.51, 0.48, 0.47 and 0.44 respectively, whereas for static decision tree, average performance is 0.68. Though static decision tree improves performance compared to the previous approaches, but the performance is still low. F-measure is calculated from precision and recall. The reason of low

precision means high false positive values, and low recall means that the false negative numbers are very high. In order to increase the performance, we use dynamic decision tree which adds datasets gradually to the previous datasets for building training model and use the model for handling some false positives and false negatives. The approach improves the average performance up to 11% compared to static decision tree, but it is necessary to handle more false positives and false negatives to increase the performance. For this, we use Hybrid-RDR that handles the problems by efficient knowledge acquisition. The performance of Hybrid-RDR is reasonably high compared to other approaches for all datasets. The average performance of Hybrid-RDR is 0.86 which improves 18% and 7% compared to static and dynamic decision tree respectively.

The performance of the algorithms depends on the characteristics of the datasets such as identical, abbreviated, and synonym and combined words. If training dataset contains large number of abbreviated words, but test dataset contains large number of synonym words, then performance becomes low. For increasing the performance of dynamic decision tree, it is necessary to build models again with more datasets to correctly classify the schema data. Sometimes building model with a large amount of datasets may not improve the performance by classifying the schemas correctly because the learning approach easily overfits with the learning base. However, for the Hybrid-RDR approach, performance is improved by incrementally adding rules for solving false positives and false negatives.

6.2 Schema Mapping Results at the Structure Level

Only element level matching does not produce good results. In order to improve the performance and produce accurate results, we have performed structure level matching. The mapping result of structure level matching is shown in **Table 2**.

Datasets	Precision	Recall	F-measure
D1	0.98	0.94	0.96
D2	0.94	0.91	0.92
D3	0.93	0.95	0.94
D4	0.97	0.94	0.95
D5	1.00	0.89	0.94
D6	0.96	0.91	0.93
D7	0.95	0.93	0.94
D8	0.91	0.94	0.92
D9	0.95	0.91	0.93
D10	0.90	0.92	0.91
AVG	**0.95**	**0.92**	**0.93**

Table 2. Performance of KSMS at the structure level matching

In **Table 2,** we show that the performance of structure level matching in terms of precision, recall and F-measure. Precision is higher than recall in most of the datasets. This is reasonable when we consider structure level instead of element level. We compare this F-measure to the F-measure of the element level matching, and we find that average F-measure has been improved up to 7% when we consider the hierarchical structure at the structure level matching. The average precision, recall and F-measure of all the datasets in the purchase order domain are 0.95, 0.92 and 0.93 respectively.

6.3 Final Mapping Results by Aggregation functions

In order to combine the schema mapping results produced by element level and structure level matchers, and to produce the final results, we use aggregation functions on the F-measure. The final schema mapping results are shown in **Table 3** where the columns *Datasets, Harm, Avg, Min, Max, Weighted* describe information about datasets, HARMONIC MEAN, AVERAGE, MINIMUM, MAXIMUM and WEIGHTED aggregation results.

Datasets	Harm	Avg	Min	Max	Weighted
D1	0.93	0.93	0.90	0.96	0.90
D2	0.90	0.91	0.89	0.92	0.89
D3	0.89	0.90	0.85	0.94	0.86
D4	0.91	0.91	0.87	0.95	0.87
D5	0.90	0.90	0.86	0.94	0.86
D6	0.91	0.92	0.90	0.93	0.90
D7	0.91	0.91	0.88	0.94	0.88
D8	0.88	0.89	0.85	0.92	0.86
D9	0.88	0.88	0.83	0.93	0.84
D10	0.85	0.86	0.80	0.91	0.81
AVG	**0.90**	**0.90**	**0.86**	**0.93**	**0.87**

Table 3. Final mapping results

In **Table 3,** we find that MAXIMUM gives the highest and MINIMUM gives the lowest schema mapping results compared to other aggegation functions. As MAXIMUM takes the highest value and MINIMUM takes the lowest value between two values, we do not consider the results. We compare the results among other three functions. We find that WEIGHTED provides the lowest aggregation result. AVERAGE gives slightly better mapping results compared to HARMONIC MEAN for some datasets such as D2, D3, D6, D8 and D10. Therefore, the final average mapping performance, F-measure is 0.90.

7 Discussion

Schema mapping can be done by machine learning or knowledge engineering approaches at the element level. Machine learning approach is promising for element similarity, but it needs to rebuild a training model if schema data changes over time. Inversely, knowledge engineering approach encodes human knowledge directly such that knowledge base can be constructed with limited data, but it needs time consuming knowledge acquisition. In order to overcome the limitations, we have used Hybrid-RDR approach that combines machine learning algorithm, J48 and knowledge engineering approach, CPR based RDR in our system, KSMS. The advantage of Hybrid-RDR is that it needs only one training model to classify new schema data. If the model gives wrong classification, then rules are added incrementally in order to handle the problem. The approach increases performance incrementally with the help of knowledge acquisition and decreases rule addition over time.

However, only element level matching is not sufficient for schema mapping. This is because it is necessary to consider the hierarchical structure of a full graph in order to improve the performance and produce accurate results. For this, we have added the features of performing structure level matching in KSMS. Finally, we have used some aggregation functions for combining the results of both element level and structure level matching.

8 Conclusion and Future Works

In this research, we have presented a Knowledge-based Schema Matching System (KSMS) which has performed schema mapping both at the element and structure level. In order to show the ability of the system, we have used 5 XDR datasets from purchase order domain. Experimental results have shown that the system determines good performance both at the element and structure level. The final schema mapping result is determined by the average aggregation function. There are some advantages of our system compared to the existing systems. First, it is not necessary to select the best combination of matchers. Second, Knowledge base is empty at the beginning. That means the system does not need any initial expert correspondences from the users. Third, rules are not predefined. Rules are created based on the features constructed from element level matchers. Fourth, over fitting problem does not occur in the system as only one decision tree model is used for classifying schemas. Fifth, the system does not need time consuming knowledge acquisition as rules are only created to correctly classify the wrongly classified cases produced by decision tree model. Finally, the system can handle the schema matching problems: false positives and false negatives using knowledge acquisition. So users do not need to add, delete or modify schema mapping results manually.

In future, we will adapt our system for ontology mapping. Then we will experiment more datasets from other domains such as conference, bibliography and anatomy.

Acknowledgement

Autonomous Systems, Digital Productivity and Service Flagship, and the Tasmanian node of the Australian Centre for Broadband Innovation are assisted by a grant from the Tasmanian Government which is administered by the Tasmanian Department of Economic Development, Tourism and the Arts.

References

Anam, S., Kim, Y. and Liu, Q. (2014): Incremental Schema Mapping. *Knowledge Management and Acquisition for Smart Systems and Services.* Springer International Publishing.

Anam, S., Kim, Y.S., Kang, B.H. and Liu, Q. (2015): Schema Mapping Using Hybrid Ripple-Down Rules. *the Thirty-Eighth Australasian Computer Science Conference, ACSC 2015.* Sydney, Australia: CRPITT.

Cate, B.T., Dalmau, V. and Kolaitis, P.G. (2013): Learning schema mappings. *ACM Transactions on Database Systems (TODS),* 38**,** 28.

Cheng, W., Lin, H. and Sun, Y. (2005): An efficient schema matching algorithm. *Knowledge-Based Intelligent Information and Engineering Systems*, Springer, 972-978.

Do, H.-H. and Rahm, E. (2002): COMA: a system for flexible combination of schema matching approaches. *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB Endowment, 610-621.

Duchateau, F., Bellahsene, Z. and Coletta, R. (2008): A flexible approach for planning schema matching algorithms. *On the Move to Meaningful Internet Systems: OTM 2008.* Springer.

Duchateau, F., Coletta, R., Bellahsene, Z. and Miller, R.J.(2009): Yam: a schema matcher factory. *Proceedings of the 18th ACM conference on Information and knowledge management*, ACM, 2079-2080.

Eckert, K., Meilicke, C. and Stuckenschmidt, H. (2009): Improving ontology matching using meta-level learning. *The Semantic Web: Research and Applications.* Springer.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. (2009): The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter,* 11**,** 10-18.

Hu, W., Qu, Y. and Cheng, G. (2008): Matching large ontologies: A divide-and-conquer approach. *Data & Knowledge Engineering,* 67**,** 140-160.

Jimenez, S., Becerra, C., Gelbukh, A. and Gonzalez, F. (2009): Generalized mongue-elkan method for approximate text string comparison. *Computational Linguistics and Intelligent Text Processing.* Springer.

Kim, Y.S., Compton, P. and Kang, B.H. (2012): Ripple-down rules with censored production rules. *Knowledge Management and Acquisition for Intelligent Systems.* Springer.

Lee, Y., Sayyadian, M., Doan, A. and Rosenthal, A.S. (2007): eTuner: tuning schema matching software using synthetic scenarios. *The VLDB Journal—The International Journal on Very Large Data Bases,* 16**,** 97-122.

Madhavan, J., Bernstein, P.A. and Rahm, E. (2001): Generic Schema Matching with Cupid. *Proceedings of the 27th International Conference on Very Large Data Bases.* Morgan Kaufmann Publishers Inc.

Marie, A. and Gal, A. (2008): Boosting schema matchers. *On the Move to Meaningful Internet Systems: OTM 2008.* Springer.

Massmann, S. and Rahm, E.(2008): Evaluating Instance-based Matching of Web Directories. WebDB, 2008. Citeseer.

Melnik, S., Garcia-Molina, H. and Rahm, E. (2002): Similarity flooding: A versatile graph matching algorithm and its application to schema matching. *18th International Conference on Data Engineering*, IEEE, 117-128.

Melnik, S., Rahm, E. and Bernstein, P.A. Rondo (2003): A programming platform for generic model management. *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, ACM, 193-204.

Ngo, D., Bellahsene, Z. and Coletta, R. (2011a): A generic approach for combining linguistic and context profile metrics in ontology matching. *On the Move to Meaningful Internet Systems: OTM 2011*. Springer.

Ngo, D.H. and Bellahsene, Z. (2009): YAM++:(not) Yet Another Matcher for Ontology Matching Task. BDA'2012: 28e journées Bases de Données Avancées, 2012.

Ngo, D.H., Bellahsene, Z. and Coletta, R. (2011b): YAM++--Results for OAEI 2011. ISWC'11: *The 6th International Workshop on Ontology Matching*, 228-235.

Peukert, E., Eberius, J. and Rahm, E. (2011): AMC-A framework for modelling and comparing matching systems as matching processes. *27th International Conference on Data Engineering (ICDE)*, IEEE, 1304-1307.

Peukert, E., Eberius, J. and Rahm, E. (2012): A self-configuring schema matching system. *28th International Conference on Data Engineering (ICDE)*, IEEE, 306-317.

Richards, D. (2009): Two decades of ripple down rules research. *The Knowledge Engineering Review*, 24**,** 159-184.

Stoilos, G., Stamou, G. and Kollias, S. (2005): A string metric for ontology alignment. *The Semantic Web–ISWC,* Springer.

Volz, J., Bizer, C., Gaedke, M. and Kobilarov, G. (2009): *Discovering and maintaining links on the web of data*, Springer.

Detection of Structural Changes in Data Streams

Ross Callister **Mihai Lazarescu** **Duc-Son Pham**

[1] Department of Computing
Curtin University,
Kent St, Bentley WA 6102,
Email: Ross.Callister@postgrad.curtin.edu.au

Abstract

We propose new methods for detecting structural changes in data streams. Significant changes within data streams, due to their often highly dynamic nature, are the main cause in performance degradation of many algorithms. The primary difference to previous works related to change detection in data streams is our usage of an algorithmic process to define the changes. We focus on RepStream, a powerful graph based clustering algorithm, which has been shown to perform well in a stream clustering context. RepStream, like many other algorithms, operates according to parameters which are set by the user. Primarily, RepStream uses the K value to determine the degree of connectivity in its K Nearest Neighbour graph structure. RepStream requires that its K value be set suitably in order to achieve optimal clustering performance, which we measure in terms of F-Measure. Since real-world data streams are dynamic, with classes appearing and disappearing, and moving and shifting, this requires the K value to be varied according to the current state of the stream. However, such a problem in a data stream mining context is largely unexplored. We first consider this challenge by addressing the research question: when K needs to be changed. From a change detection perspective, our proposed method measures the structural variation of the underlying data stream using five different statistical and geometrical features which can be extracted whilst RepStream performs its clustering. We show that combining these features into a detection method gives promising results in regards to early detection of structural changes in data streams. We use the well known KDD Cup 1999 intrusion detection benchmark dataset, and show that our proposed method was able to identify many of the changes within the stream.

Keywords: Concept Drift, Change Detection, Stream Detection, Anomaly Detection,

1 Introduction

In this paper we propose a new method of detecting changes in a data stream by analysing features extracted from the memory contents of the graph-based stream clustering algorithm, RepStream. Data streams can vary greatly over time. This can reflect anything from a shift in customer buying habits, to anomalies in a sensor network, to attacks on a network which is being monitored (Kifer et al. 2004, Silva et al. 2013).

Given the nonstationary and complex nature of data streams, clustering them effectively has been the subject of much research recently. A major problem that this paper concentrates on is detecting *when* change occurs in the stream. Knowing when change occurs can be incredibly valuable information as it can be used to inform a stream clustering algorithm, for example, when to adjust its operating parameters to maintain optimal clustering, or when to drop previous data from memory in order to adjust to new patterns faster.

Our approach uses RepStream(Lühr & Lazarescu 2009) as a basis for our analysis as it is a graph-based clustering algorithm. RepStream constructs a directed K- nearest neighbour graph, which it used for clustering. The K-nearest neighbour graph structure has been used in other works previously as a method of clustering. It produces a graph in which vertexes which are close together (according to the chosen distance measure) are more likely to be connected than those which are further apart. This K-nearest neighbour system reflects the shape, and nature of the data, as represented in a multi-dimensional space. As such we can take advantage of this intrinsic arrangement of the data, and analyse features related to it to gain information about the dataset.

The most important operating parameter of RepStream is the number of outgoing edges each vertex has, which is often denoted as K. This K value determines total connectivity: a higher K produces more total edges in the graph and vice versa. Since RepStream uses connectivity as a fundamental part of its clustering process this results in the situation where a higher K value results in fewer, more connected clusters, while a lower K value results in more, less connected clusters.

In practice, it is often not trivial to determine an optimal value for K. This matter is further complicated by the dynamic nature of data streams. Clusters in a data stream may appear, disappear, merge, or split over time, their shapes may change, they may become more or less dense, or shift (in the sense of data points being represented by points in some n-dimensional space), as well as other sorts of changes. Due to the unpredictable and sometimes dramatic changes in data streams a single static K value is not always guaranteed to produce optimal clustering results. Ideally, the K value should be varied to match changes in the stream, as well as other possible adaptations - like increasing or decreasing the number of data points which are being stored in memory to, respectively, allow the algorithm to form a better model in the case of a stable period in the stream, or allow

the algorithm to adapt more quickly to changes in the case of a rapidly changing stream. This has applications in the real world as it would allow an algorithm to dynamically adjust to a data stream that shifts over time.

In a practical sense our research will allow us to locate points in a data stream that correspond to changes in the underlying distribution which directly map to the performance of RepStream. This is important to our goal of determining when parameters must be varied to maintain optimal clustering performance. Having parameters set sub-optimally can dramatically affect even the most recent state-of-the-art algorithms. Our goal is to make this a non-issue, initially by determining when to change the parameters, and then in future work, to address the problem of selecting the correct values of the parameters at these change points. While this work concentrates on RepStream, the concept of extracting and examining features for use in change detection can be applied to other algorithms as well.

In this work, we first address the research question of *when* K needs to be varied in order for RepStream to be more optimally tuned to the structural changes in the underlying data stream. Unlike most previous works on change detection which are totally limited to the statistical nature of the data stream, our critical argument here is that the parameters of the clustering algorithm should only be varied if that brings in considerable clustering benefits. Thus, the notion of changes in this work not only means the shifts in the statistical properties of the stream, but also depends on the specific algorithm being considered. Our contributions in this work include

- A novel perspective on structural changes that are algorithm specific, opening up future research directions for many stream clustering algorithms;
- A novel detection scheme consisting of five graph-based features and a window-based decision algorithm;
- A comprehensive analysis on the KDD Cup 1999 intrusion detection dataset.

The paper is organized as follows. Section 2 reviews related stream clustering algorithms and other change detection methods in the literature. Section 3 details our proposed method. Section 4 explains the measures used to evaluate our method. Section 5 gives a comprehensive analysis of the proposed method on the KDD Cup 1999 dataset (Stolfo et al. 2000). Finally, Section 6 concludes.

2 Related Works

There are many approaches to clustering data streams. To face the specific challenges of handling evolving data streams various approaches have been proposed. For example, by using micro clusters to record statistics about recent points, which can be clustered according to specified time-windows (Aggarwal et al. 2003, Kranen et al. 2011). Or by using subcluster structures which fade over time to make new data more relevant (Aggarwal et al. 2004, Zhou et al. 2008). Yet other approaches use grid based density structures, which decay to keep up with the new data arriving in the stream (Chen & Tu 2007).

Unfortunately, despite some novel and effective methods for handling streaming contexts, most stream clustering algorithms do not have mechanisms or methods for detecting change. Instead they often use sliding windows, where older data is discarded once it has become too old (Silva et al. 2013).

However, change in distributions of data streams over time has been a topic of research for some time. One approach maintains a reference window, and a sliding window, and compares the two using a distribution similarity function (Kifer et al. 2004). Sliding windows do ensure that newer data is used to make clustering decisions over older data, as older data may contain patterns that are no longer representative of the stream's distribution if it changes. However, sliding windows can not solve all the challenges that a data stream can present.

Another approach to change detection in a data stream uses a minimum description length to generate code tables which 'compress' the distribution of a stream. When the stream is no longer optimally compressed by the code table then this marks a change in the stream (Van Leeuwen & Siebes 2008).

Yet another approach proposes a method for detecting the appearance of new classes in a data stream clustering context by deferring classification of outliers and placing them in a buffer, then analysing the points in the buffer for cohesion representing a novel class (Masud et al. 2011). Along a similar vein (Bhatnagar et al. 2014) proposes the algorithm ExCC, a grid based method. New points added to ExCC which do not fit into already populated cells in the grid are added to a hold queue, and when a sufficient amount points are outside expected regions then a 'change' in the cluster distribution is recorded, and the grid is updated. This is an example where there is a sort of change detection in the operation of the algorithm. Unfortunately it can only detect changes related to the movement of cluster boundaries.

Other topics related to change detection in data streams include recording and tracking change over time for the identification of temporal change (Hahsler & Dunham 2011), or tracking change in a noisy stream through cluster density analysis(Nasraoui & Rojas 2006). The field of anomaly detection in data streams is also related, for example a paper by (Pham et al. 2014) which uses residual subspace analysis to detect anomalies in a compressed form of the data stream. Such anomaly detection is a form of change detection in a context where training data can be made available to determine a 'normal' stream state. Our approach, however, can not make assumptions about the normal state of a stream as it is likely to be unknown.

Other algorithms mention evolution in streams as a major issue (Bhatnagar et al. 2014, Forestiero et al. 2013), but as far as we are aware there are no existing algorithms that attempt to specifically locate when a data stream changes in a way that significantly affects the performance of a given clustering algorithm. Our approach differs from existing methods in that it seeks to detect arbitrary change in a data stream by using geometric features present within the K-Nearest neighbour graph structure of RepStream. That is, our approach seeks to detect locations in the stream which are likely to negatively impact the performance of clustering, and which are candidate locations for parameter adjustment.

Our approach differs from existing methods in that it seeks to detect arbitrary change by using geometric features present within the K-Nearest neighbour graph structure of RepStream.

3 Proposed Method

We propose that by extracting features from the data as it is processed by our chosen algorithm - RepStream - we can learn about when there are shifts in the underlying dataset. That is, by looking for changes in these fundamental features, we know when to expect the dataset to have shifted. By knowing when the distribution and nature of the dataset changes we can use this information to improve our clustering results.

RepStream was selected because it has been demonstrated to outperform other stream clustering approaches. Additionally, it uses both K-nearest neighbour arrangements of the data, and also the relative density of points when making its decisions. This hybrid method allows a wider range of the stream's properties to be captured.

3.1 The RepStream Clustering Algorithm

RepStream uses a combination of graph based and density based approaches to clustering (Lühr & Lazarescu 2009). It constructs a directed K-nearest neighbour graph, adding each data point it receives from the data stream one at a time. Due to memory limitations - one can not expect an algorithm to maintain every data point from a continuous data stream in memory simultaneously - thus it uses a first-in-first-out window, maintaining only the most recent points in its K nearest neighbour graph.

The K-nearest neighbour graph is a directed graph, in which each vertex has K outgoing edges to the K nearest other vertices in the graph. Two vertices in the graph are considered to be reciprocally connected if each vertex has an outgoing edge that connects to the other vertex.

When a new data point is inserted as a vertex into RepStream's K-nearest neighbour graph and it does not have a reciprocal connection to an already existing representative point, then that vertex becomes a new Representative point. Representative points, as their name suggests, act like representatives for other nearby vertexes. The closest Representative point that a vertex has a reciprocal connection to is the Representative point which represents that vertex. A vertex will be a member of the cluster that its Representative point is a member of.

Clustering in RepStream is done using representative points. RepStream maintains a second K nearest neighbour graph, which only its representative points are a member of. Two Representative points belong to the same cluster if they have a reciprocal connection in the representative K nearest neighbour graph, and are also density related to each other.

Density in RepStream is not absolute, it is rather the relative density that is used to determine clustering. The density relation radius of a representative point is equal to $\alpha \times AvgDist$ where $AvgDist$ is the average distance to its K nearest neighbours, as shown in Figure 1. The parameter α is set by the user, and has the value of 1.5 in our experiments in this paper, as suggested by RepStream's original paper.

Two representatives R_1 and R_2 are density related if R_2 is within the density relation radius of R_1, and R_1 is within the density relation radius of R_2.

Figure 2 shows an example of two representative points which fulfil the conditions to belong to the same cluster. Figure 3 shows an example where the representative points do not meet these requirements due to R_3 not being within the density relation radius of R_4.

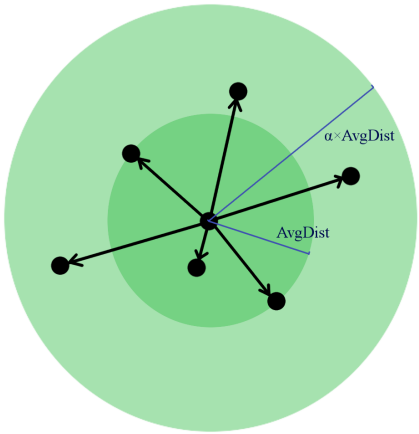


Figure 1: The density relation radius of a representative point is equal to $\alpha \times AvgDist$ where $AvgDist$ is the average distance to its K nearest neighbours.

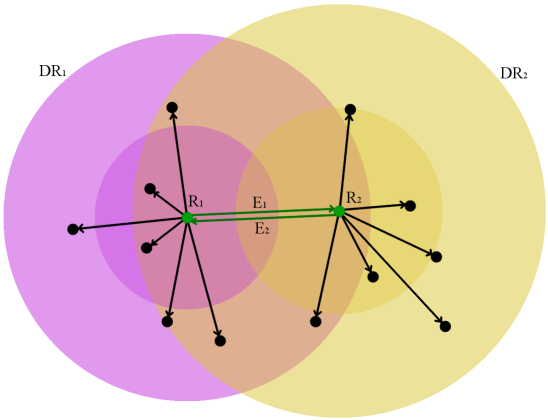


Figure 2: An example of two representative points which are both reciprocally connected, and density related

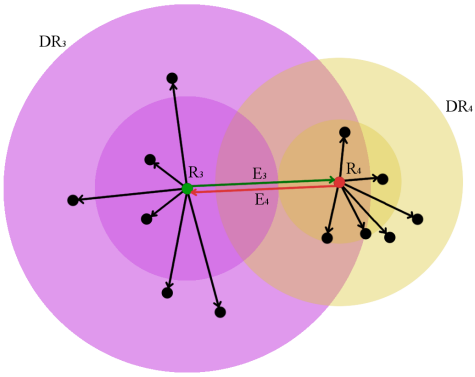


Figure 3: An example of two representative points which are reciprocally connected, but not density related

3.2 Structural Changes

First, we define a stream clustering algorithm as being stable if its clustering performance (which is measured in terms of F-Measure in this work) varies little as a number of samples arriving from a stationary data stream is sufficiently large. Then, we define structural changes associated with a stable clustering algorithm as statistical or geometrical changes in the data that lead to a significant deviation in clustering outputs. In the case of RepStream, our hypothesis is that it is reflected in the structure of the data points, specifically in regards to their geometric properties within the k-NN graph of RepStream. We note the following:

- From the definition, it follows that structural changes may correlate strongly with distributional or statistical changes in the data stream. A data stream represents samples taken from a distribution of data over time. This distribution may change as the stream progresses, in a way such that the structure of the data also changes, for example concept drift (Tsymbal 2004). It can refer to new parts in the data distribution appearing, or disappearing, or the changing of existing parts of the distribution.
- Structural changes depend on the specific algorithm and its sensitivity against the changes in the data stream. This is an important aspect because change detection would not be useful if it does not lead to a need to adjust the underlying clustering algorithm.

3.3 Features Extraction

We have examined various intrinsic features within RepStream's graph based clustering approach to determine whether they can be used to identify changes in the underlying data stream. The features we have concentrated on are: the cluster count over time, the number of edges created and removed over time, the number of cluster merges and splits over time, and the variation in the length of the edges over time. We note important that in order to extract these features, we need an active instance of RepStream with some value of K, which may not be optimal. This value of K is a proxy for the detection only. This proxy K value is desirably small as it leads to more efficient RepStream computation. Each of the features are extracted periodically, in our case after every 100 points processed by RepStream.

The following features are each extracted from RepStream as we believe there to be a correlation between them and the structural properties of the dataset.

Cluster Count Extracting the cluster count is computationally efficient. Since RepStream produces a clustering result for every data point inserted into the algorithm, this feature is simply the number of clusters that RepStream locates at each time step. However, it is important to note that this feature is dependant on the K value used. A higher K value will typically result in a lower number of clusters found by the algorithm, due to the higher connectivity of the k-NN graph. We found that lower values of K – approximately $K = 10$ – are more sensitive to changes in the data stream. This is possibly because higher K values are likely to have many strong connections, and be more stable than the lower K values. Cluster Count, as well as being relevant to the clustering result, also correlates to the stability of the stream, as the number of clusters changes less when the stream is relatively stable.

Edge Change Count The number of edges created and removed at the representative layer over time - the K-nearest neighbour change count - was chosen as a feature due to its ability to reflect the degree of change that the graph requires when data points are inserted. The idea is that when the data stream is stable then the representative points will also remain stable. Thus, the amount of changes at the representative level will be stable. On the other hand, when the data stream is shifting then it is expected that the number of edges that need to be updated at the representative level will vary, due to representatives needing to be created or destroyed. For example a graph vertex that is inserted outside existing clusters will be more likely to become a representative point, and need to cause updates in other representative points. On the contrary, a vertex inserted among a group of existing vertices can often be represented by an existing representative point, thus it does not cause any representative edge changes. This feature is extracted by counting the number of edge updates which are on representative points since the last measurement. In our experiments, it is every 500 points, or half the sliding window size.

Cluster Merges and Splits Counting the cluster merges and splits over time is used as a feature because it is likely to correlate with the stability of the data. RepStream creates clusters by considering the connectivity of representative points, as well as the local density of each representative point. Due to nodes being inserted, the density or connectivity of representative points may change and this results in the clusters splitting apart or merging together. When the dataset is stable, inserting new points is less likely to result in such changes. However, it may become more likely when the dataset shifts then points occurring in new locations, and being removed due to the first-in-first-out window. Based on this observation, we have selected the combined number of cluster merges and splits as a feature to examine when searching for change in the dataset. Similar to the k-NN change count, this feature can also be efficiently extracted by counting the number of times clusters merge and split since the last measurement.

Cluster Merges and Splits occur more rapidly when the structure of the stream changes due to data point distributions shifting outside established cluster boundaries. This causes the clusters to become unstable and causes splits and merges to occur until the algorithm can arrange the datapoints into a stable configuration

Edge Length Variation Edge length variation is measured as follows. Each vertex in RepStream's graph maintains outgoing connections to its K nearest neighbours. The standard deviation of the length of these outgoing edges is calculated for each vertex. The standard deviations are then added together and divided by the total number of points in the K-NN graph to find the average standard deviation over every point in memory. The idea behind this feature is that one would expect a relatively consistent edge length variation when the dataset is stable. If new clusters were to form outside existing clusters the edge length might increase since longer edges would need to be formed to maintain the K-NN graph. If the density of an existing cluster were to increase then

the total edge lengths might decrease, which would similarly lead to an increase in the standard deviation in the edge lengths. This feature, therefore, is selected as a candidate for tracking changes in the dataset.

History Count History count represents the number of times a point returns to a previous cluster. As the stream progresses individual data points change cluster membership. Even adding a single point can result in many points changing from one cluster to another. We keep track of the number of times each point in the first-in-first-out queue has returned to a cluster that it was previously in. Our hypothesis is that when structural changes are taking place within the dataset the clustering results will be unstable. This instability leads to a higher rate of individual points jumping between clusters over time. When the dataset is stable, on the other hand, the rate of change in cluster membership will be lower, due to new points not changing the dataset's structure significantly.

3.4 Detection Scheme

We propose a simple scheme using the extracted features to determine whether a change has occurred or not. Each individual feature is examined separately using a time-series change detection algorithm. The algorithm is shown as Algorithm 1. The inputs are listed and given as $(feature, M, H, \lambda)$. $feature$ represents the given feature as a time series. The parameter M is a multiplier which affects how sensitive the algorithm is to change; a higher value will cause the algorithm to require a larger shift in the time series before a change is detected. The parameter H is the number of previous points over which to track a moving average. The parameter λ is a parameter that determines how quickly the algorithm updates to match newer observations. The algorithm returns *changes* which contains a list of time indexes where changes have occurred in the time series $feature$.

Data: $(feature, M, H, \lambda)$
Result: Changes: a list of indexes where changes have been detected
$changes = \emptyset$;
$X = mean(feature(i - M : M))$;
$\sigma = standardDeviation(feature(i - M : M))$;
for $i = M$: *size(feature)* **do**
  $ma = mean(feature(i - M : M))$;
  $s = standardDeviation(feature(i - M : M))$;
  if $abs(ma - X) > M \times \sigma$ **then**
    $changes = changes + \{i\}$;
    $X = ma$;
    $\sigma = s$;
  else
    $X = (1 - \lambda) \times X + \lambda \times ma$;
    $\sigma = (1 - \lambda) \times X + \lambda \times s$;
  end
end

Algorithm 1: Algorithm for feature change detection

While this algorithm is written for a batch dataset rather than a stream, it can easily be modified for use in a stream since it only requires the past m data points to be stored in memory, as a sliding window.

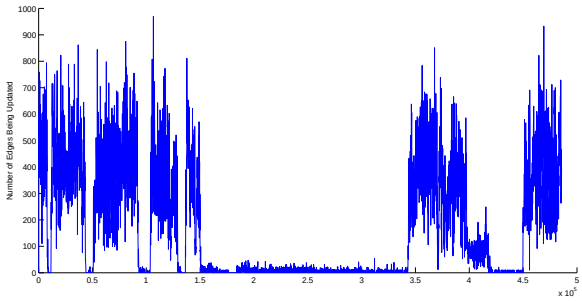


Figure 4: Number of edge changes (Change Count) over time for the KDD dataset at $K = 10$

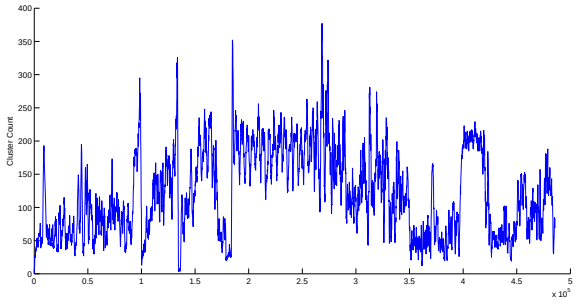


Figure 5: Number of clusters present over time for the KDD dataset at $K = 10$

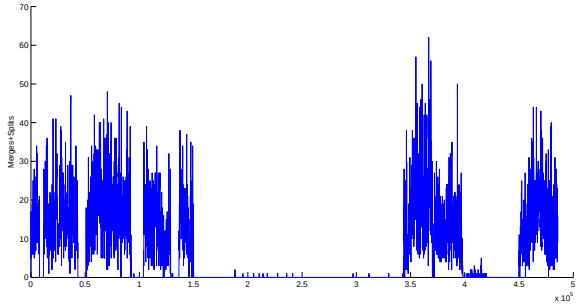


Figure 6: Number of cluster merges and splits over time for the KDD dataset at $K = 10$

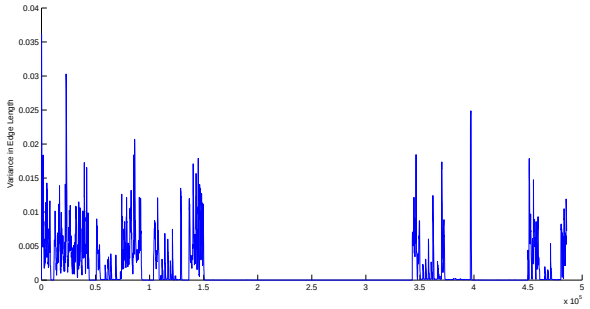


Figure 7: Variation in the edge lengths over time for the KDD dataset at $K = 10$

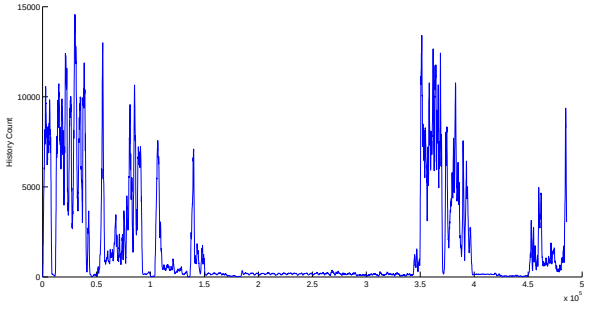


Figure 8: 'History Count' for the KDD Dataset at $K = 10$

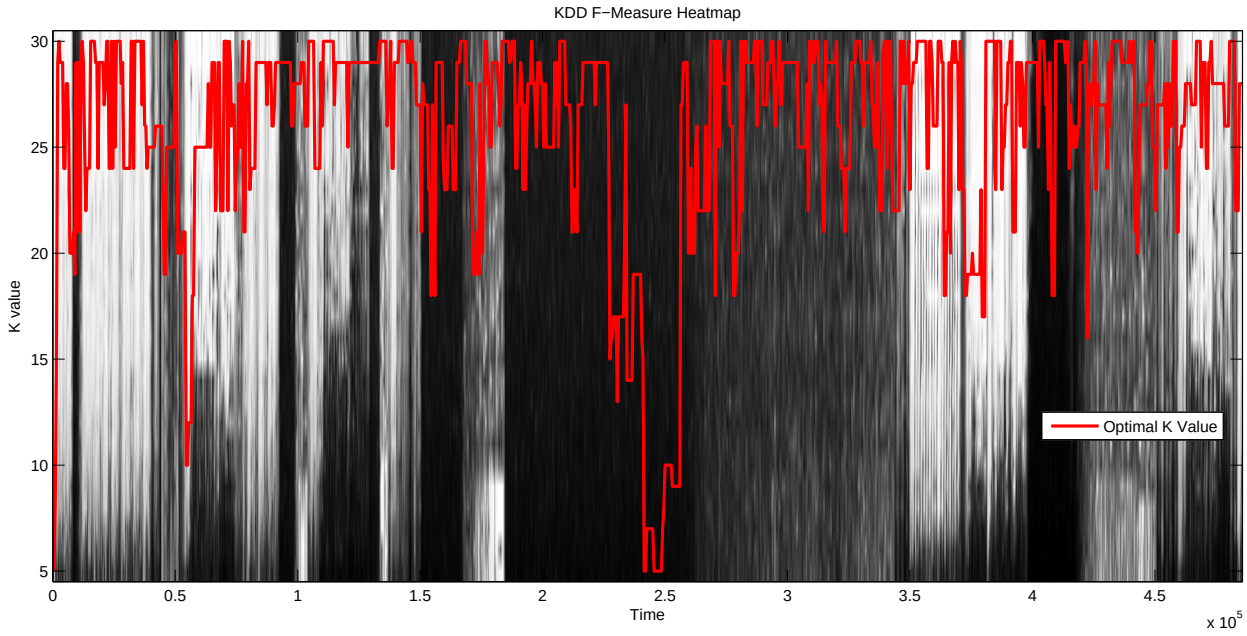


Figure 9: The optimal K value over time produced by RepStream with respect to F-Measure over time for the KDD dataset. Best viewed in colour.

Since there are 5 features that we are testing we use a system where multiple features must agree before a structural change is detected. At least N features must agree that there has been a change within the last T samples for the algorithm to detect a 'change'. Where N is greater than half the number of features then it is simply majority voting.

Figure 10 shows where the changes are detected in the case of the Edge Change Count feature, where $M = 1$ $H = 20$ and $\lambda = 0$. Edge Change Count was selected for illustrative purposes as it gives a clear idea of what varying the parameters does. The red shows the raw value of the feature, which varies significantly from point to point, blue shows the X value over time, as well as $X \pm \sigma$, and green shows exactly where the change points are detected. Figure 11 shows the same feature, but the M value has been changed to 30, while Figure 12 has the $\lambda = 0.001$, so that the X and σ values slowly adjust over time.

4 Algorithm Evaluation

In this paper we use an evaluation measure known as MTR - Mean Time Ratio (Bifet et al. 2013).

MTR is a combination of several important metrics, and is meant to evaluate a change detection algorithm in a single number. It is a combination of several important metrics. The formula for MTR is:

$$MTR = \frac{MTFA}{MTD} \times (1 - MDR).$$

Where MTFA is the mean time between false alarms, MTD is the mean time to detection, and MDR is the missed detection rate. A higher MTR is desirable, and this measure can be used to directly compare two change detection results.

MTR is chosen as our evaluation metric because of its specific design for use with change detection algorithms. Looking simply at the number of successful detections or the number of false alarms does not give a clear picture of how well an algorithm performs, because it is trivial to maximise either of those scores individually. Typically compromising between a high rate of detection, and a low rate of false alarms is desirable in practice. Mean Time Ratio takes both the false alarms and detection rate into account, and is ideal for evaluating the differences between detection results.

Also included in our evaluations are the individual values for Mean Time Between False Alarms (MTFA), Mean Time to Detection (MTD), and Missed Detection Rate (MDR). These measures indicate, respectively, the rate of false alarms, the time taken to detect changes, and the detection rate.

5 Experiments

5.1 KDD Cup 1999 Dataset

We select the well known KDD Cup 1999 intrusion detection dataset (Stolfo et al. 2000) to demonstrate the proposed method. It is made up of data extracted from a computer network being monitored during various simulated and controlled network attacks. We use the availably subsampled version of the dataset which contains approximately 500,000 data points, as well as ground truth class labels for evaluation purposes.

The KDD data contains data which represents normal traffic as well as data points representing 22 different types of attacks with varying durations from few to hundreds of thousands of points.

KDD has been used previously as an example of a real-world data stream used in evaluating stream clustering algorithms(Lühr & Lazarescu 2009)(Cao et al. 2006)(Ruiz et al. 2009). The varied attacks over time simulate the dynamic and unpredictable nature of a data stream, making it ideal to test our change detection methods on. Unfortunately, however, there remain very few real world benchmark datasets available for evaluation purposes. A recent survey by (Kaur et al. 2015) indicates that a major issue in stream clustering literature is the lack of availability of benchmark datasets. As such, KDD remains perhaps the only publicly available real-world stream

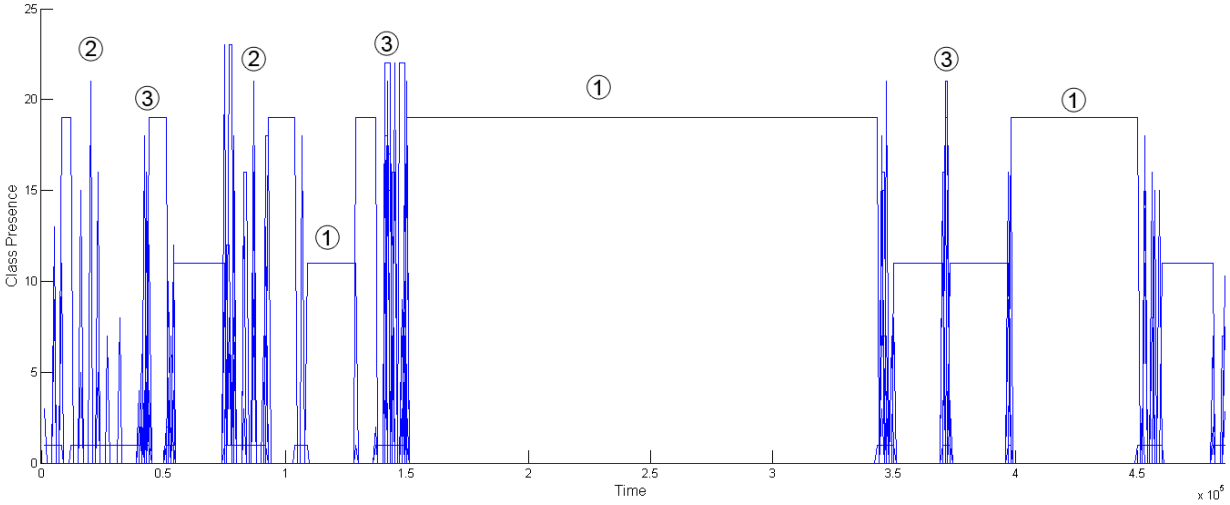


Figure 13: Presence of classes during the KDD data stream. Type 1 attacks are prolonged single classes, type 2 are quick single attacks interspersed by normal, and type 3 are rapid clusters of attacks.

dataset which has the necessary traits to evaluate our methods.

Classes in KDD Figure 13 shows the class presence for the KDD dataset (we note that all figures are best viewed in colour). For each class in the dataset a line was plotted. The value of the line is the class label if that class is present during that time during the stream, and zero otherwise. Class 1 is the 'normal' traffic, and every other class represents different attacks. The spikes in Figure 13 correspond to when attacks occur, and the plateaus represent prolonged denial-of-service (DOS) attacks. The most important part of this is when attacks occur, i.e. - the spikes and beginnings and ends of the DOS attacks in the class presence plot - as this tells us when the data stream changes due to an attack occurring.

Figure 13 also shows examples of the various types of attacks. The attacks labelled with type 1 are prolonged attacks made up of only a single class. Though the actual data contained within each successive data point may vary all the instances belong to the same class. There are the relatively rare instances of type 2, which contain very short attacks from a single class interspersed with data from the normal class. Then there are type 3 attacks, which are occurrences where many types of attacks occur in rapid succession, interspersed with the normal.

Optimal RepStream K Value Figure 9 shows the K value which produces the highest F-Measure score over time, in blue, superimposed over the F-Measure of each K value. In the background is a grey-scale collection of cells, which we call a 'heatmap'. At each time step (along the X axis) the K value is represented along the Y axis, and the brightness of that individual cell represents the F-Measure score at that time with that K value. That is, the brightness is between 0 (black) and 1 (white), and matches the F-Measure of that K value at that time step. The range of K values is between $K = 5$ and $K = 30$. This figure shows that the K value which produces the optimal clustering results (the brightest cell at a given time step) may vary considerably over the course of the stream. Therefore, the best performance can not be achieved by using a single K value.

Furthermore, Figure 9 also shows that at some times the range of K values which produce F-Measure results near the optimal is very large - when the bright regions are large vertically, and at other times the K value is optimal within a very specific range - when the bright regions are more constrained. This figure, when combined with the information in Figure 13, shows that when attacks occur in the data set, and when the classes shift, the optimal K value does change in response to the stream, and that to get optimal performance the stream must be monitored for such changes, and adapted to when they occur.

Ground Truth Change Points The class presence is what we use as the ground truth for evaluation of our technique. Whenever a class appears or disappears we define a ground truth change point. These change points are weighted according to how much they affect the clustering performance of RepStream. Where the distribution of F-Measure values changes significantly the ground truth is more heavily weighted than if the distribution changes less. The Bhattacharyya distance between the distributions is used to weight the ground truth, and any below a threshold are omitted for the sake of clarity.

The heatmap is important because it shows at which points during the data stream the RepStream algorithm performs well, and when it performs poorly. This gives an idea of when changes occur in the distribution data points over time. It is intuitive to see where major changes occur, thus it is used as as the background to give context to the results.

5.2 Results

Figure 1 shows a table of the results for our detection algorithm. It contains values for the Mean Time between False Alarms (MTFA), Mean Time to Detection (MTD), the Missed Detection Rate (MDR), and the Mean Time Ratio (MTR).

Figure 14 shows a visual representation of the detection rate for our algorithm when optimised to give the highest possible MTR value. The parameters used were $M = 1.4$, $H = 15$ and $\lambda = 0.001$, and the MTR was 39.46, mostly due to the low rate of false alarms.

Figure 15 has the parameters optimised for a higher detection rate. The inputs were $M = 1.2$, $H = 9$ and $\lambda = 0.001$, which gives a lower MTR

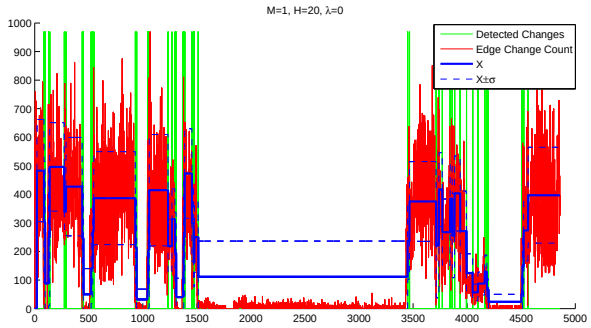


Figure 10: Detection on the single feature Edge Change Count, with $M = 1$, $H = 20$, and $\lambda = 0$

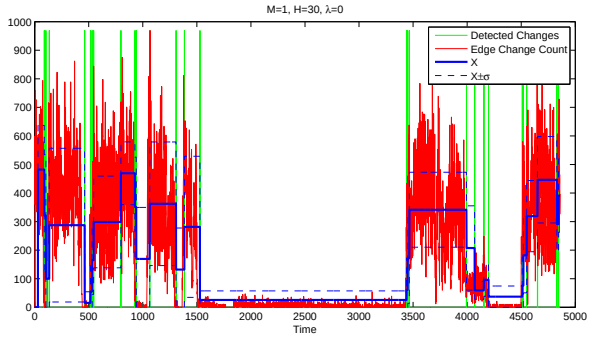


Figure 11: Detection on the single feature Edge Change Count, with $M = 1$, $H = 30$, and $\lambda = 0$

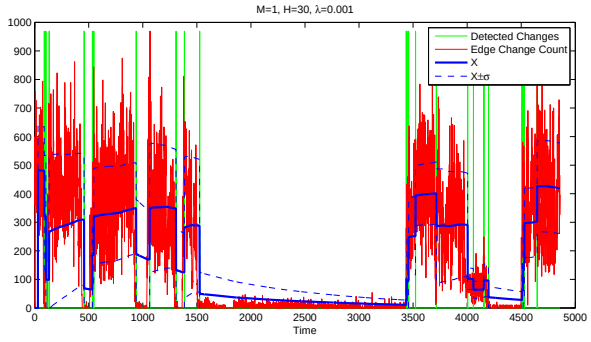


Figure 12: Detection on the single feature Edge Change Count, with $M = 1$, $H = 30$, and $\lambda = 0.001$

value of 9.16, but a higher rate of detection, at the expense of a higher rate of false alarms.

	MTFA	MTD	MDR	MTR
Moving Average Detection optimised for MTR	**803.3**	**5.42**	0.73	**39.46**
Moving Average Detecion optimised for lower MDR	219.0	19.13	**0.20**	9.16
OSVM detection optimised for MTR	186.2	12.64	0.50	7.36

Table 1: Results of our experiments on KDD with the Mean Time between False Alarms, Mean Time to Detection, Missed Detection Rate, and Mean Time Ratio

5.3 Comparison To OSVM Classification Method

We also tested the features using a different detection schema, for the sake of comparison. Using a Support Vector Machine we performed the detection as a simple two-class classification. Using a sliding we trained a One-class SVM with H datapoints, where the datapoints were simply 5 dimensional vectors representing each of our 5 features at a given time step. The next $\frac{H}{2}$ datapoints were then classified using the OSVM, and if at least half of them were not classified into the OSVM's trained class then it would mark a change at that point.

Figure 16 shows the results of our test using the One-class SVM detection approach. Optimising the algorithm to give the highest MTR resulted in a MTR value of 7.36, and had a much higher rate of false alarms compared to our proposed approach.

The SVM detection method results in a larger number of correctly detected changes, as evident from the lower MDR value, however this is at the expense of far more false alarms, with the relatively low value ot 186.15 as the MTFA. This combination results in a lower MTR score compared to our proposed detection method.

Our method when optimised for MTR yields a significantly higher total MTR score, as well as comparatively fewer false alarms. A different optimisation for our proposed method optimises to reduce the Missed Detection Rate as much as possible while retaining a comparable MTR (Figure 15). The lower MDR of 0.20 means that 80% of changes are successfully detected, with the drawback of a higher rate of false alarms, leading to a decreased MTFA score for this method.

5.4 Results Discussion

Our algorithm detects points where the features change, as in Figures 10-12. When all features are combined into a single detection method they produce the change points, marked in green on Figures 14 and 15. Both of these methods are optimised differently.

Our method of evaluation is limited in the sense that ground truth changes only take into account when classes appear and disappear. This is a limitation of the dataset used,

Whilst our proposed algorithm performs well, it still has problems, notably in the form of false alarms and changes which are not detected. For example the false alarms at around $Time = 1.85e + 05$ and $Time = 1.92e + 05$. At these times there are no ground truth changes, as it is during a time where there is only a single class present (shown in Figure 13). However, at that time the performance of RepStream does change, so it could be explained by a change in the distribution of that single class at that time.

Despite the false alarms our algorithm detects a high amount of the ground truth changes (Table 1). Many of the changes that are not detected, too, are missed due to the close proximity of the changes. Our algorithm is limited in the case where dramatic changes happen very rapidly.

6 Conclusion

There are significant challenges in detecting changes in a dataset, particularly when the dimensionality is

high. By extracting features of the data from a K-nearest neighbour graph we reduce the problem to detecting changes in a smaller number of time series, representing structural properties of the data.

We have presented a novel method of detecting concept changes in data streams by examining structural properties of graph based arrangements of the data. This approach has been shown to work even in high dimensional data, as with KDD which was treated as a 34 dimensional dataset. We take features in RepStream's K-nearest neighbour structure and use a time-series change detection algorithm with the goal of identifying when major changes in the underlying dataset have occurred.

Our detection algorithm outperforms a similar approach using OSVM techniques with respect to MTR on the well known KDD intrusion detection dataset. The KDD dataset was selected due to its use as a benchmark in prior literature, as well as the fact that it mirrors a real-world application of change detection.

Whilst the approach does have limitations, particularly when the changes occur rapidly, and when the changes are very subtle, it produces good results when tested on a real world dataset.

References

Aggarwal, C. C., Han, J., Wang, J. & Yu, P. S. (2003), A framework for clustering evolving data streams, *in* 'Proceedings of the 29th International Conference on Very Large Data Bases', VLDB Endowment, pp. 81–92.

Aggarwal, C. C., Han, J., Wang, J. & Yu, P. S. (2004), A framework for projected clustering of high dimensional data streams, *in* 'Proceedings of the 30th International Conference on Very Large Data Bases', VLDB Endowment, pp. 852–863.

Bhatnagar, V., Kaur, S. & Chakravarthy, S. (2014), 'Clustering data streams using grid-based synopsis', *Knowledge and information systems* **41**(1), 127–152.

Bifet, A., Read, J., Pfahringer, B., Holmes, G. & Ciobait, I. (2013), Cd-moa: Change detection framework for massive online analysis, *in* 'Proceedings of the 12th International Symposium, IDA 2013', pp. 92–103.

Cao, F., Ester, M., Qian, W. & Zhou, A. (2006), Density-based clustering over an evolving data stream with noise., *in* 'Proceedings of the SIAM Conference on Data Mining 2006', SIAM, pp. 326–337.

Chen, Y. & Tu, L. (2007), Density-based clustering for real-time stream data, *in* 'Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '07, ACM, New York, NY, USA, pp. 133–142.

Forestiero, A., Pizzuti, C. & Spezzano, G. (2013), 'A single pass algorithm for clustering evolving data streams based on swarm intelligence', *Data Mining and Knowledge Discovery* **26**(1), 1–26.

Hahsler, M. & Dunham, M. H. (2011), Temporal structure learning for clustering massive data streams in real-time., *in* 'Proceedings of the 2011 SIAM International Conference on Data Mining', SIAM, pp. 664–675.

Kaur, S., Bhatnagar, V. & Chakravarthy, S. (2015), Stream clustering algorithms: A primer, *in* 'Big Data in Complex Systems', Springer, pp. 105–145.

KDD Cup 1999 Dataset (n.d.).
URL: *https://archive.ics.uci.edu/ml/machine-learning-databases/kddcup99-mld/kddcup99.html*

Kifer, D., Ben-David, S. & Gehrke, J. (2004), Detecting change in data streams, *in* 'Proceedings of the 13th International Conference on Very Large Data Bases', VLDB Endowment, pp. 180–191.

Kranen, P., Assent, I., Baldauf, C. & Seidl, T. (2011), 'The clustree: indexing micro-clusters for anytime stream mining', *Knowledge and information systems* **29**(2), 249–272.

Lühr, S. & Lazarescu, M. (2009), 'Incremental clustering of dynamic data streams using connectivity based representative points', *Data & Knowledge Engineering* **68**(1), 1–27.

Masud, M., Gao, J., Khan, L., Han, J. & Thuraisingham, B. (2011), 'Classification and novel class detection in concept-drifting data streams under time constraints', *IEEE Transactions on Knowledge and Data Engineering* **23**, 859–874.

Nasraoui, O. & Rojas, C. (2006), *Robust Clustering for Tracking Noisy Evolving Data Streams*, chapter 72, pp. 619–623.

Pham, D.-S., Venkatesh, S., Lazarescu, M. & Budhaditya, S. (2014), 'Anomaly detection in large-scale data stream networks', *Data Mining and Knowledge Discovery* **28**(1), 145–189.

Ruiz, C., Menasalvas, E. & Spiliopoulou, M. (2009), C-denstream: Using domain knowledge on a data stream, *in* 'Procedings of the 12th International Conference on Discovery Science', Springer, pp. 287–301.

Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., de Carvalho, A. C. & Gama, J. (2013), 'Data stream clustering: A survey', *ACM Computing Surveys (CSUR)* **46**(1), 13.

Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A. & Chan, P. K. (2000), Cost-based modeling for fraud and intrusion detection: Results from the jam project, *in* 'Proceedings of the DARPA Information Survivability Conference and Exposition, 2000', Vol. 2, IEEE, pp. 130–144.

Tsymbal, A. (2004), 'The problem of concept drift: definitions and related work', *Technical Report from the Computer Science Department, Trinity College Dublin* .

Van Leeuwen, M. & Siebes, A. (2008), Streamkrimp: Detecting change in data streams, *in* 'Machine Learning and Knowledge Discovery in Databases', Springer, pp. 672–687.

Zhou, A., Cao, F., Qian, W. & Jin, C. (2008), 'Tracking clusters in evolving data streams over sliding windows', *Knowledge and Information Systems* **15**(2), 181–214.

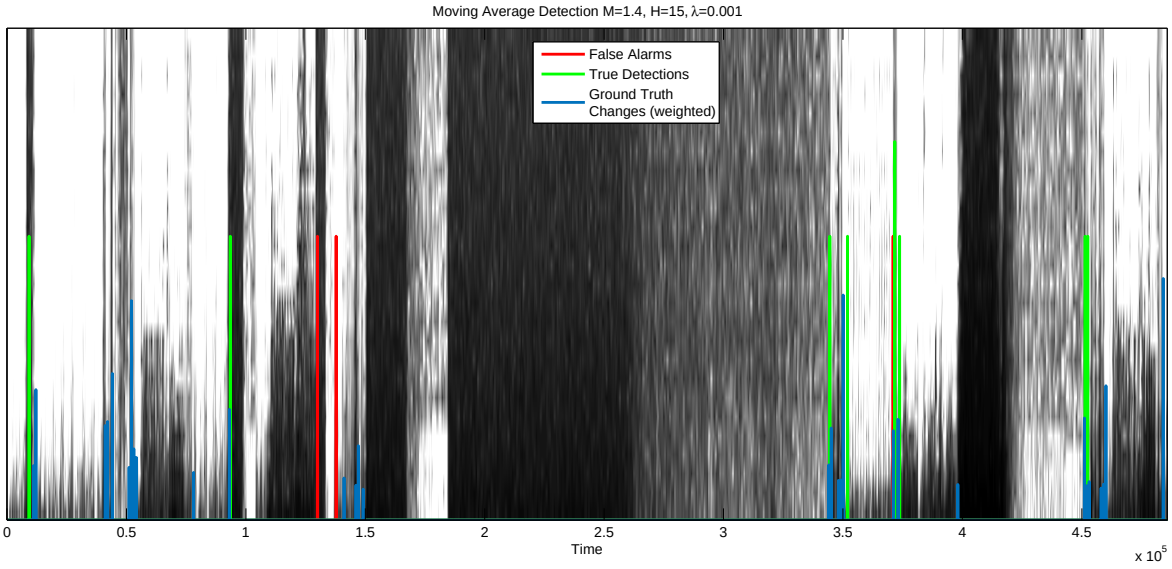


Figure 14: Our moving average detection algorithm on the KDD dataset, with parameters optimised for the highest MTR value

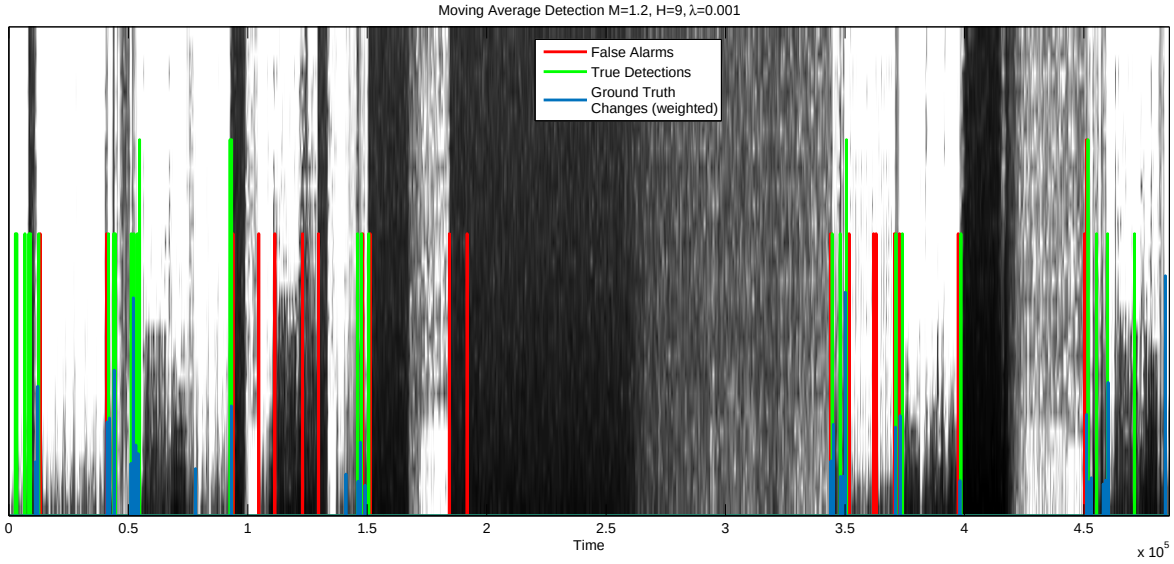


Figure 15: Our moving average detection algorithm on the KDD dataset, with parameters optimised for lower Missed Detection Rate

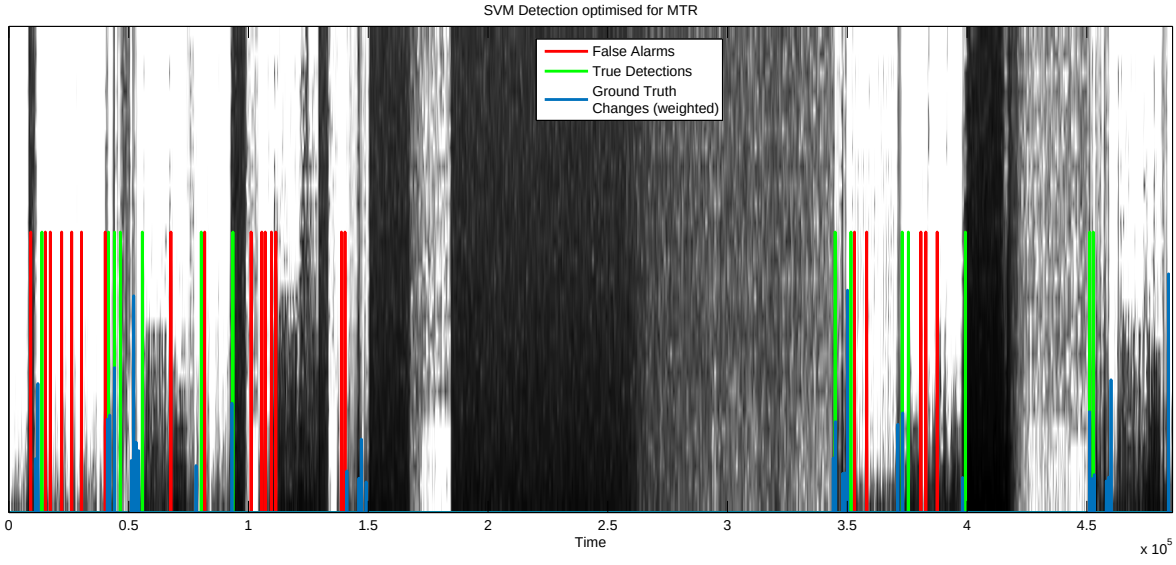


Figure 16: The OSVM comparison algorithm on the KDD dataset with parameters optimised for the highest MTR value

Complement Random Forest

Md Nasim Adnan **Md Zahidul Islam**

School of Computing and Mathematics
Charles Sturt University,
Panorama Avenue, Bathurst, NSW 2795,
Email: madnan@.csu.edu.au, zislam@csu.edu.au.

Abstract

Random Forest is a popular decision forest building algorithm which focuses on generating diverse decision trees as the base classifiers. For high dimensional data sets, Random Forest generally excels in generating diverse decision trees at the cost of less accurate individual decision trees. To achieve higher prediction accuracy, a decision forest needs both accurate and diverse decision trees as the base classifiers. In this paper we propose a novel decision forest algorithm called Complement Random Forest that aims to generate accurate yet diverse decision trees when applied on high dimensional data sets. We conduct an elaborate experimental analysis on seven publicly available data sets from UCI Machine Learning Repository. The experimental results indicate the effectiveness of our proposed technique.

Keywords: Decision Forest, High Dimensional Data Set, Prediction Accuracy, Random Forest.

1 Introduction

Organizations all over the world are interested in data collection. The collected data are generally analyzed for knowledge discovery and future prediction. For example, an insurance company may collect various data on their clients whom they already know about; whether *good* or *bad* clients. From the collected data the insurance company then discovers knowledge/pattern which is used for the prediction of potential future clients; whether *good* or *bad*. In data mining this is also known as the classification and prediction task (Tan, Steinbach & Kumar 2006).

Classification aims to generate a function (commonly known as a classifier) that maps the set of non-class attributes $\{A_1, A_2, ..., A_m\}$ to a predefined class attribute C (Tan et al. 2006), where a data set D can be seen as a two dimensional table having columns/attributes (i.e. $\{A_1, A_2, ..., A_m\}$ and C) and rows/records i.e. $D = \{R_1, R_2, \ldots R_n\}$. A class attribute is the labeling attribute of a record R_i. For example, *Client Status* can be the class attribute of a *Client* data set where a record of the data set can be labeled as *good* (which is a domain value of *Client Status*) while another record can be labeled as *bad*.

A classifier is first built from a training data set where records are labeled with the class attribute values such as *good* and *bad*, and then applied on future/unseen records (for which the class attribute values are not known) in order to predict their class values. There are different types of classifiers including Decision Trees (Breiman, Friedman, Olshen & Stone 1985),(Quinlan 1993),(Quinlan 1996), Bayesian Classifiers (Bishop 2008),(Mitchell 1997), Artificial Neural Networks (Jain & Mao 1996),(Zhang 2000),(Zhang, Patuwo & Hu 1998) and Support Vector Machines (Burges 1998).

Typically an ensemble of classifiers is found to be useful for unstable classifiers such as a decision tree (Tan et al. 2006). A decision forest is an ensemble of decision trees where an individual decision tree acts as a base classifier and the classification is performed by taking a vote based on the predictions made by each decision tree of the decision forest (Tan et al. 2006),(Polikar 2006),(Ho 1998),(Islam & Giggins 2011).

A decision forest overcomes some of the shortcomings of a decision tree. For example, a single decision tree discovers only one set of logic rules (i.e. a pattern) from a data set while there can be many other equally valid sets of logic rules (i.e. patterns) in the data set. Different decision trees of a forest generally extract different patterns. The pattern extracted by a single tree may fail to correctly predict/classify the class value of a record, but some other patterns could correctly classify the record. Therefore, the classification/prediction accuracy of a single tree can be improved by using a decision forest.

In order to achieve a higher ensemble accuracy a decision forest needs to have both accurate and diverse individual decision trees as base classifiers (Polikar 2006),(Ho 1998). An accurate individual decision tree can be obtained by applying a decision tree algorithm such as CART (Breiman et al. 1985) on a data set. However, we also need diversity among the trees in a forest in order to achieve higher accuracy. If the individual decision trees generate similar classification results (i.e. no or low diversity), there is no point of constructing a decision forest as all trees will commit similar errors. Therefore, we can not apply a decision tree algorithm many times in order to get many trees of a forest.

There are many decision forest algorithms (Breiman 1996),(Ho 1998),(Breiman 2001),(Rodriguez, Kuncheva & Alonso 2006),(Ye, Wu, Huang, Ng & Li 2014). We briefly introduce some of the algorithms and their limitations in Section 2. It is clear that there is room for further improvement in achieving a higher prediction and/or classification accuracy of a forest through achieving a higher individual accuracy of the trees and greater diversity among the trees.

In this paper we propose a decision forest algorithm called *Complement Random Forest* (CRF) that aims to build a set of diverse decision trees with high individual accuracies in order to achieve a high overall ensemble accuracy. Our proposed technique first creates a number of new "good attributes" (i.e. attributes with high classification capacity) from the existing attributes and thereby extends the attribute space of a data set D. The attribute space extension is expected to improve the classification accuracy of a decision tree obtained from the extended data set D_E. CRF next randomly creates a number of data sets $\{D_E^1, D_E^2, \ldots D_E^k\}$ (commonly known as bootstrap sample data sets (Breiman 1996),(Breiman 2001),(Han & Kamber 2006)) from D_E in order to eventually build two decision trees T_1^i and T_2^i from each data set D_E^i. While both D_E^i and D_E have the same the number of records, D_E^i may contain duplicate records since it is created by randomly picking the records (one by one) from D_E. Therefore, two randomly created data sets D_E^i and D_E^j are likely to be different to each other contributing to the diversity of the decision trees. That is, T_1^i and T_2^i (obtained from D_E^i) are likely to be different from T_1^j and T_2^j (obtained from D_E^j) due to the difference between D_E^i and D_E^j.

Moreover, since D_E^i contains real records (i.e. does not contain synthetic records created through approaches such as interpolation (Chawla, Bowyer, Hall & Kegelmeyer 2002)) and is extended with good quality attributes the trees T_1^i and T_2^i that are obtained from D_E^i are expected to be of high quality. When a forest building algorithm (such as SysFor (Islam & Giggins 2011)) builds trees *from a single data set* D then the trees can be overly attached to D, resulting in an over fitting problem. Whereas, trees built from different bootstrap sample data sets are expected to overcome this over fitting problem.

Before building the trees T_1^i and T_2^i from D_E^i, CRF first divides the attributes of D_E^i into two groups: good attributes and bad attributes. It then randomly selects half of the (total number of) good attributes and half of the bad attributes to form a new set of attributes S_1^i. Using the remaining good attributes and bad attributes it then creates the second set of attributes S_2^i. S_1^i and S_2^i are two mutually exclusive sets of attributes. A random sub-spacing is then applied on S_1^i in such a way so that we have proportionate number of good and bad attributes in the subspace $S_1^{i\prime} \subset S_1^i$. Similarly another subspace $S_2^{i\prime}$ is created from S_2^i. Note that, each subspace is thus mutually exclusive and guaranteed to contain some good attributes.

Two decision trees T_1^i and T_2^i are then built from D_E^i using two mutually exclusive sets of attributes $S_1^{i\prime}$ and $S_2^{i\prime}$, respectively. Since T_1^i and T_2^i are built from mutually exclusive sets of attributes they are completely different, contributing further to the diversity of the trees. The use of the random subspaces $S_1^{i\prime}$ and $S_2^{i\prime}$ have advantages over S_1^i and S_2^i in promoting diversity among the trees, especially when we build many trees from a number of bootstrap samples. Moreover, the presence of good attributes in both subspaces contribute to achieve higher individual accuracy for both T_1^i and T_2^i.

The remainder of this paper is organized as follows: In Section 2 we discuss the main motivation of the study. Section 3 presents the proposed technique. Section 4 discusses the experimental results. Finally, we offer some concluding remarks in Section 5.

2 Motivation

The accuracy of a decision forest typically depends on both individual tree accuracy and diversity among the decision trees (Polikar 2006),(Ho 1998). Diversity among the decision trees helps defusing the classification/prediction errors encountered by a single decision tree. The Random Forest algorithm (Breiman 2001) applies the Random Subspace algorithm (Ho 1998) on the bootstrap samples $D' = \{D^1, D^2, \ldots D^T\}$ (Breiman 1996),(Breiman 2001),(Han & Kamber 2006) of a training data set D. A bootstrap sample D^i is generated by randomly selecting the records from D where the total number of records in D^i typically remains the same as in D. Each record R_i from D has equal probability of being selected in D^i. A record R_i from D can be chosen multiple times in a bootstrap sample. It is reported (Han & Kamber 2006) that on an average 63.2% of the unique records from D are generally selected in a bootstrap sample. The Random Subspace algorithm as explained below is then applied on each bootstrap sample D^i $(i = 1, 2, \ldots, T)$ in order to generate $|T|$ number of trees for the forest, where a tree T_i is built from a bootstrap sample D^i.

The Random Subspace algorithm (Ho 1998) randomly draws a subset of attributes (also called subspace) f from the entire attribute space A. While drawing the subspace it does not ensure the presence of good attributes in the subspace. A decision tree is then built on the data set using only the subspace f. The individual tree accuracy and the diversity among the decision trees heavily depend on the size of f (i.e. $|f|$). If f is sufficiently small then the chance of having the same attributes in different subspaces is low. Thus, the trees in a forest tend to become more diverse. However, a sufficiently small f may not guaranty the presence of adequate number of good attributes, which may decrease individual tree accuracy. On the other hand, big $|f|$ may increase the individual tree accuracies but is likely to sacrifice high diversity.

The number of attributes in f is commonly chosen to be $int(\log_2 |A|) + 1$ (Breiman 2001). If we have a high dimensional data set consisting of 100 attributes (i.e. $|A| = 100$) then the number of attributes in a randomly selected subspace is $int(\log_2 100) + 1 = 7$. While building a decision tree, the test attribute of a node of the tree is then selected from the set of the 7 attributes. We argue that the small subset (only 7) of the original attributes may not contain any (or sufficient number of) good attributes, resulting in a low individual tree accuracy.

As a solution, the authors (Ye et al. 2014) applied the stratified sampling method to select the attributes in a subspace. The key idea behind the stratified sampling is to divide the attributes (A) into two groups. One group will contain the good attributes A_G and the other group will contain the bad attributes A_B. The attributes having the informativeness capacity higher than the average informativeness capacity are considered (Ye et al. 2014) to be the set of good attributes A_G and all other attributes are considered to be the bad attributes A_B. Then, the attributes for f are selected randomly from each group in proportion to the size of the groups. Unlike the Random Subspace algorithm (Ho 1998) the stratified sampling method (Ye et al. 2014) guarantees the presence of some good attributes in a subspace.

However, we realize that if a data set D contains a low number of good attributes (i.e. $|A_G|$ is low) and high number of bad attributes then the stratified sampling method also encounters a low number of good attributes in a subspace f. Let us assume that there are 10 good attributes and 90 bad attributes in a data set D. Since there are altogether 100 attributes the size of f would be $int(\log_2 100) + 1 = 7$. The proportionate number of the good attributes would be $\frac{(7\times10)}{100} = 0.7 \approx 1$ and bad attributes would be 6. The presence of just one good attribute in the pool of 7 attributes can decrease the individual tree accuracy. Additionally, since there are altogether 10 good attributes and every subspace is having only one good attribute, at least 2 trees in every 11 trees will have the same attribute at the root node. Therefore, the trees in a forest may have low diversity.

We believe that an eventual solution to this problem can be increasing the number of the good attributes through a space extension approach. A recent study (Amasyali & Ersoy 2014) demonstrated that decision forest building algorithms such as Bagging (Breiman 1996), Random Subspace (Ho 1998), Random Forest (Breiman 2001), and Rotation Forest (Rodriguez et al. 2006) achieve higher prediction accuracy when applied on extended attribute space.

A space extension technique (Amasyali & Ersoy 2014) works as follows. Any attribute pair say A_i and A_j is randomly selected from $A = \{A_1, A_2, \dots A_m\}$ and combined using the difference operator (i.e. by subtracting the values of A_j from the values of A_i) to form a new attribute $A'_k = A_i - A_j$. This process iterates $|A|$ times and thus $|A|$ new attributes $A_{rnd} = \{A'_1, A'_2, \dots A'_m\}$ are generated. Finally, the newly generated attribute space is added to the original attribute space to form the extended attribute space $A_E = A \cup A_{rnd}$. The Average Individual Accuracy (AIA) of the decision trees in a forest built by Bagging, Random Subspace and Random Forest slightly increases when they use an extended space data set compared to the original data set (Amasyali & Ersoy 2014).

However, we argue that a limitation of the technique (Amasyali & Ersoy 2014) is the random selection of the pair of attributes A_i and A_j for the attribute space extension. If two different attributes are *randomly selected* the chance of creating a good attribute is not as high as it would be in the case where attributes are selected and combined *systematically*.

We present an empirical analysis in Table 1 and Table 2. The gain ratio (i.e. the classification capacity of an attribute) (Quinlan 1993),(Quinlan 1996) values of the original categorical attributes of the Car Evaluation data set (*UCI Machine Learning Repository* n.d.) are presented in Table 1. The gain ratio of each attribute pair is then presented in Table 2. Since there are six original attributes we get ${}^6C_2 = 15$ new attributes (i.e. attribute pairs). For a new attribute pair, all values belonging to the attributes are concatenated. Therefore, the domain of a new attribute A'_k (that combines two original attributes A_i and A_j) is $|A_i| \times |A_j|$, where $|A_i|$ is the domain size of A_i. It is clear from Table 2 that if the new attributes are randomly picked then we may miss the attributes with the best gain ratio values.

A recent attribute extension technique (Adnan, Islam & Kwan 2014) handles this problem by selecting the attribute pairs that have the high gain ratio values. It empirically demonstrates that the addition of the best (according to the gain ratio values) $|A|/2$ new attributes with $|A|$ original attributes results in increased accuracy for a single decision tree. Following the technique (Adnan et al. 2014), three new attributes (as shown in Table 3) are selected from the attributes shown in Table 2.

Table 1: Gain Ratio of all original attributes for the Car Evaluation data set

Attribute Name	Gain Ratio
buying	0.0482
maint	0.0369
doors	0.0022
persons	0.1386
lug_boot	0.0189
safety	0.1654

Table 2: Gain Ratio of all newly generated attributes for the Car Evaluation data set

Attribute Name	Gain Ratio
buying_maint	0.0685
buying_doors	0.0254
buying_persons	0.0907
buying_lug_boot	0.0370
buying_safety	0.1047
maint_doors	0.0196
maint_persons	0.0838
maint_lug_boot	0.0294
maint_safety	0.0963
doors_persons	0.0635
doors_lug_boot	0.0119
doors_safety	0.0752
persons_lug_boot	0.0803
persons_safety	0.1666
lug_boot_safety	0.1038

Table 3: Selection of attributes

Attribute Name	Gain Ratio
maint_safety	0.0963
persons_safety	0.1666
lug_boot_safety	0.1038

However, we argue that the technique (Adnan et al. 2014) may suffer from a lack of variation in the set of new attributes. For example, all three new attributes in Table 3 have been generated from the original attribute *safety*, which has the highest gain ratio value in Table 1. Thus, the set of the best $|A|/2$ new attributes is likely to be strongly influenced by a few original attributes with high gain ratio value. The multiple appearance of very few attributes in the form of new attributes may cause a drop in diversity among the trees.

3 Our Technique

In order to address the issues discussed in Section 2 and achieve high prediction accuracy, we now propose a decision forest algorithm called Complement Random Forest (CRF) that has the following four steps (also see Algorithm 1).

Step 1: Extend the attribute space.

FOR i**=1 to** $|T|/2$: /* $|T|$ is the number of trees*/

Step 2: Generate a bootstrap sample data set.
Step 3: Create mutually exclusive subsets of attributes.
Step 4: Build two trees from the sample data set.
END FOR.

***Step 1*:Extend the attribute space**. It is evident from the literature that the attribute space extension improves the classification/prediction accuracy of a decision tree classifier (Amasyali & Ersoy 2014),(Adnan et al. 2014). However, if we extend the space randomly (Amasyali & Ersoy 2014) then we may not get the good attributes (new) in the extended space. This may not help building accurate classifiers. Moreover, if we only pick the best attributes (Adnan et al. 2014) then the extended space can be overly dominated by a few original good attributes reducing the diversity among the extended attributes. The space extension technique (Adnan et al. 2014) was originally proposed for a decision tree and not for forest.

Therefore, in this study we extend the attribute space by adding $|A|/2$ new attributes. We select a set of diverse, but good attributes as shown in Algorithm 1. The reason for extending the space by adding $|A|/2$ new attributes is the previous experimental analyses (Amasyali & Ersoy 2014),(Adnan et al. 2014),(Adnan & Islam 2014) that indicate a greater accuracy improvement through the $|A|/2$ extension than $|A|$ (or $2 \times |A|$ or $3 \times |A|$) extension.

We first generate new attributes by combining every pair of original attributes. Thus, we get a set of candidate attributes A_C, where the size of the set $|A_C| = {}^{|A|}C_2$. The obvious process of combining categorical attributes is the concatenation of the categorical values. On the other hand, for numerical attributes many different possible approaches for concatenation exist such as addition, subtraction, division, and multiplication (Amasyali & Ersoy 2014). We use the subtraction operator as it appears to be the most effective one (Amasyali & Ersoy 2014).

The attribute space is then extended by adding $|A|/2$ newly generated attributes. First we calculate the gain ratio (Quinlan 1993),(Quinlan 1996) of all attributes (original and newly generated), and compute the average gain ratio (*avg_GR_O*) of the original attributes. All new attributes $A'_k \in A_C$ that have gain ratio values higher than *avg_GR_O* are then stored in a set A'_C and sorted in the descending order of their gain ratio values.

The attributes from A'_C are selected and added in the set A_E one by one, in such a way so that no two (new) attributes in A_E have any common original attribute/s. That is, the attribute $A'_k \in A'_C$ with the maximum gain ratio is first included in A_E and removed from A'_C. Then another attribute $A'_l \in A'_C$ is selected and added in A_E, where A'_l has the highest gain ratio among the attributes in A'_C and none of the original attributes in A'_l is the same as any original attribute in A'_k. The inclusion of attributes in A_E continues until either all possible attributes from A'_C are added in A_E or $|A_E| > |A|/2$.

If $|A_E| < |A|/2$ even after adding all possible attributes then the attribute $A'_m \in A'_C$ having the highest gain ratio among the attributes in A'_C is added in A_E even if A'_m has original attribute/s that are same as the original attribute/s of any attribute $A'_k \in A_E$. Once A'_m is added in A_E it is removed from A'_C. This process continues until either $|A'_C| < 1$ or $|A_E| > |A|/2$. Finally A_E is added with A and thereby an extended space data set D_E is created.

Table 4: All newly generated attributes with Gain Ratio greater than *avg_GR_O*

Attribute Name	Gain Ratio
persons_safety	0.1666
buying_safety	0.1047
lug_boot_safety	0.1038
maint_safety	0.0963
buying_persons	0.0907
maint_persons	0.0838
persons_lug_boot	0.0803
doors_safety	0.0752
buying_maint	0.0685

We now illustrate the step with examples. The *avg_GR_O* value (for the original attributes of the Car Evaluation dataset) is calculated from Table 1 as 0.0683. Therefore, the new attributes that have the gain ratio values greater than 0.0683 are selected in A'_C as shown in Table 4. The new attribute *person_safety* (which is a combination of two original attributes *person* and *safety*) has the highest gain ratio 0.1666 and therefore is included in A_E. Since all remaining attributes except *buying_maint* have either the original attribute *safety* or the original attribute *person*, *buying_maint* is added in A_E as the second attribute. We still need one more attribute to include in A_E (since $|A| = 6$ and $\frac{|A|}{2} = 3$) and there is no other attribute with unique original attribute pair. Therefore, we select *buying_safety* as the third attribute to be included in A_E. Finally, an extended space data set D_E is created having nine $(6 + 3 = 9)$ non-class attributes $A \cup A_E$, and the class attribute C. Note that the number of records in D_E and D are exactly the same (i.e. $|D_E| = |D|$), but each record in D_E has $|A \cup A_E|$ attributes instead of $|A|$ attributes.

***Step 2*: Generate a bootstrap sample data set.** The steps from Step 2 to Step 4 are iterative (see Algorithm 1). For the i-th iteration, we create a bootstrap sample data set D^i_E (Han & Kamber 2006) from the extended space data set (D_E). Each record in D^i_E is selected randomly from D_E. $|D^i_E|$ is equal to $|D_E|$. However, D^i_E may contain duplicate records.

***Step 3*: Create mutually exclusive subsets of attributes.** We calculate the gain ratio of each attribute in the bootstrap sample data set D^i_E. The average gain ratio *avg_GR_E* is calculated. The attributes are then divided into two mutually exclusive groups: group of the "good attributes" A_G ($A_G \subset A \cup A_E$) and group of the "bad attributes" A_B ($A_B \subset A \cup A_E$). Any attribute having a gain ratio greater than or equal to *avg_GR_E* belongs to A_G and other attributes belong to A_B.

***Step 4*: Build two trees from the bootstrap sample data set.** The set of good attributes A_G is divided into two equal size (mutually exclusive) subsets A_{G1} and A_{G2} ($A_{G1} \cup A_{G2} = A_G$), where the attributes in A_{G1} and A_{G2} are randomly selected. Similarly, the set of bad attributes A_B is also divided into two equal sized subsets A_{B1} and A_{B2}.

Two mutually exclusive subsets of attributes S_1 and S_2 are now created combining $A_{G1} \cup A_{B1}$ and $A_{G2} \cup A_{B2}$, respectively (see Step 4 of Algorithm 1).

A random sub-spacing (Ho 1998) is then applied on S_1 in such a way so that we have proportionate number of good and bad attributes in the subspace S'_1, where the total number of attributes in the subspace is $int(log_2|S_1|) + 1$. Similarly another subspace S'_2 is created from S_2. Each subspace is guaranteed to contain good attributes since in the subspace we take proportionate number of good and bad attributes from S_1 or S_2. Note that the original random subspace algorithm (Ho 1998) does not take the proportionate number of good and bad attributes.

Two decision trees T_1^i and T_2^i are built from D_E^i using the attribute sets S'_1 and S'_2, respectively. Note that T_1^i and T_2^i are completely diverse since S'_1 and S'_2 are mutually exclusive subsets. The sub-spacing further increases the diversity of the trees obtained from different bootstrap sample data sets. Moreover, due to the presence of good attributes in both subsets both T_1^i and T_2^i are expected to have high individual accuracy. The extension of the attribute space in Step 1 also contributes to have greater number of good attributes in D_E^i resulting in trees with high individual accuracy. Therefore, for each bootstrap sample data set we get a pair of diverse but accurate decision trees.

Step 2, Step 3 and Step 4 are iterative. Step 2 produces different bootstrap sample data sets over different iterations contributing to the diversity of the decision trees without causing a drop of individual accuracy since a bootstrap sample data set consists of real (not synthetic) records. Although it is highly likely to have different sets of good attributes from different bootstrap sample data sets, there is a possibility of having similar (or same) set of good attributes A_G from different bootstrap sample data sets. However, the random selection of subsets A_{G1} and A_{G2} is expected to overcome the problem and produce diverse decision trees. Moreover, the application of the random subspace algorithm is expected to further ensure diversity among the trees.

The individual accuracy of the trees is promoted by Step 1 where we inject a number of newly generated good attributes. Moreover, in Step 4 the attribute space is divided into good and bad attributes in order to ensure the presence of good attributes in each subspace. Unlike the Stratified Random Forest algorithm (Ye et al. 2014), the set of good attributes are selected from the bootstrap sample data set D_E^i (instead of from the original data set D or D_E) for which the pair of trees T_1^i and T_2^i are built. Had the set of good attributes been selected from D_E and applied on D_E^i then there would not be any guarantee that the set of good attributes obtained from D_E were also the set of good attributes for D_E^i. Hence, we get the most specific set of good attributes for D_E^i which increases the possibility of a high individual tree accuracy and diversity among the trees.

Algorithm 1: Complement Random Forest

Input : The Training Data Set D. Number of trees T.
Output: Complement Random Forest (C_RF).

Required
  $C\_RF \leftarrow \phi$;
end
begin
  Step 1:
    $A_C \leftarrow$ *produce_Candidate_Attributes* (D, A);
    /* A is the set of attributes in D*/
    $G \leftarrow$ *compute_Gain_Ratio* (D, A_C, A);
    $A'_C \leftarrow$ *get_Good_Attributes* (G, A_C, A);
    $A_E \leftarrow$ *get_Extended_Space* (A, A'_C, G);
    $D_E \leftarrow$ *extend_Data_Set* (D, A_E);
  end
  for $i \leftarrow 1$ **to** $T/2$ **do**
    $\boldsymbol{D}_E^i \leftarrow \phi$;
    Step 2:
      for $j \leftarrow 1$ **to** $|\boldsymbol{D}_E|$ **do**
        curr_rec $\leftarrow$ *random_record*(D_E);
        $D_E^i \leftarrow D_E^i \cup$ curr_rec;
      end
    end
    Step 3:
      $A_G \leftarrow$ *get_Good_Attributes* (D_E^i);
      $A_B \leftarrow$ *get_Bad_Attributes* (D_E^i);
    end
    Step 4:
      A_{G1}, $A_{G2} \leftarrow$ *split_Attributes*(A_G);
      A_{B1}, $A_{B2} \leftarrow$ *split_Attributes*(A_B);
      $S_1 \leftarrow A_{G1} \cup A_{B1}$; $S_2 \leftarrow A_{G2} \cup A_{B2}$;
      $S'_1 \leftarrow$ *random_Subspace*(S_1);
      $S'_2 \leftarrow$ *random_Subspace*(S_2);
      $T_1^i \leftarrow$ *build_Tree* (D_E^i, S'_1);
      $T_2^i \leftarrow$ *build_Tree* (D_E^i, S'_2);
      $C\_RF \leftarrow C\_RF \cup T_1^i \cup T_2^i$;
    end
  end
  return C_RF ;
end

4 Experimental Results

We conduct the experimentation on seven (7) well known data sets that are publicly available from the UCI Machine Learning Repository (*UCI Machine Learning Repository* n.d.). The data sets used in the experimentation are listed in TableTable 5. For example, the Chess data set has zero numerical and 36 categorical attributes, 3196 records and the domain size of the class attribute is 2.

Table 5: Description of the data sets

Data Set Name	Non-class Attributes Num.	Cat.	No. of Records	No. of Classes
Breast Cancer	33	00	194	2
Chess	00	36	3196	2
Ionosphere	34	00	351	2
Libras Movement	90	00	324	15
Lung Cancer	56	00	27	2
Soybean	00	35	47	4
Statlog Vehicle	18	00	846	4

4.1 Experimental Setup

Some data sets contain missing values in them. We first remove the records with missing value/s before any experimentation. We also remove any identifier attributes such as *Transaction_ID* from the data sets. For every decision forest building algorithm we consistently use the following setting. We generate 100 trees since it is considered to be large enough to ensure convergence of the ensemble effect (Geurts, Ernst & Wehenkel 2006). Majority voting (Polikar 2006) is used for classification. In majority voting, the class value predicted by the majority number of trees of an ensemble is considered to be the final prediction of the ensemble. The minimum number of records in a leaf is 2. All trees are fully grown and are not pruned to avoid the pruning effect on the experimentation.

The experimentation is conducted by a machine with Intel(R) 3.4 GHz processor and 4GB Main Memory (RAM) running under 64-bit Windows 8 Operating System. All the results reported in this paper are obtained using 10-fold-cross-validation (10-CV) (Islam & Giggins 2011) for every data set. All the prediction accuracies (Ensemble Accuracy and Average Individual Accuracy) reported in this paper are in percentage. The best results are emphasized through **bold-face**. In the rest of this paper, we call Regular Random Forest (Breiman 2001) as RRF, Stratified Random Forest (Ye et al. 2014) as SRF and the proposed Complement Random Forest as CRF.

4.2 Results

We now present some experimental results to justify the usefulness of the basic components of the proposed CRF. For this extensive experimentation we select five data sets shown in Table 6. The results also demonstrate the effectiveness of CRF compared to two high quality existing techniques called RRF and SRF.

A recent extension technique (Adnan et al. 2014) (let us call it *RET*) suggests that the extension of the attribute space by adding the best $|A|/2$ attributes (according to Gain Ratio) with $|A|$ original attributes results in increased accuracy for a single decision tree. We argue that the Average Individual Accuracy (AIA) of the trees obtained by RRF can also improve if RRF is applied on an extended space data set.

Therefore, in Table 6 we present the average individual accuracy of the trees obtained by RRF, RRF on RET (i.e. RRF on the data sets extended by RET), and SRF. The AIA of the trees clearly increases when RRF is applied on the RET extended data sets. In fact the AIA of the trees obtained from RRF on RET (75.44%) is better than the AIA of the RRF (69.95%) and SRF (71.60%) trees.

However, Table 7 shows that the ensemble accuracy of RRF decreases when RRF is applied on the RET extended data sets. In fact the ensemble accuracy of RRF on RET (84.59%) is worse than RRF (87.13%) or SRF (87.33%). We next explore the reasons for the drop of the ensemble accuracy of RRF on RET although the AIA of the trees increases.

Following our discussion in Section 2 we realize that the prediction/classification accuracy of an ensemble depends on the individual accuracy of the trees and the diversity among the trees (Polikar 2006),(Ho 1998). Since the individual accuracy of the trees obtained by RRF on RET is high (see Table 6) while the ensemble accuracy is low (see Table 7) we explore the diversity among the trees in Table 8.

Table 6: Average Individual Accuracy

Data Set Name	RRF	RRF on *RET*	SRF
Breast Cancer	68.1646	**70.6163**	68.9749
Chess	68.3558	71.2596	**75.4006**
Ionosphere	87.8490	**89.1183**	87.3205
Lung Cancer	56.8335	**75.8612**	57.6390
Soybean	68.5637	**70.3251**	68.6773
Average	69.9533	**75.4361**	71.6025

Table 7: Ensemble Accuracy

Data Set Name	RRF	RRF on *RET*	SRF
Breast Cancer	78.8780	**80.9850**	80.5480
Chess	**95.0570**	72.0050	94.9670
Ionosphere	**93.7310**	92.8650	93.1610
Lung Cancer	68.8890	**83.8890**	68.8890
Soybean	**99.0910**	93.1820	**99.0910**
Average	87.1292	84.5852	**87.3312**

Following an approach taken in the literature (Amasyali & Ersoy 2014) we compute the average diversity among the trees by first computing the Kappa (K) value of a single tree T_i with the ensemble of the trees except the single tree T_i. Kappa estimates the diversity between two trees T_i and T_j. The combined prediction of the ensemble (computed through majority voting) can be seen as another single tree T_j. Then Kappa can be computed between T_i and the virtual T_j as shown in Equation 1, where $Pr(a)$ is the probability of the observed agreement between two classifiers T_i and T_j, and $Pr(e)$ is the probability of the random agreement between T_i and T_j.

$$K = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} \quad (1)$$

Once Kappa of every single tree T_i is computed we then compute the Average Individual Kappa (AIK) for an ensemble of trees as shown in Table 8. The higher the Kappa value, the lower the diversity. When RRF is applied on RET the diversity among the trees decreases (see Table 8) as Kappa increases from 0.40 to 0.52. We therefore realize that the decrease of diversity causes the decrease of ensemble accuracy as shown in Table 7.

Step 2, Step 3 and Step 4 (see Algorithm 1) of CRF aim to increase the diversity among the trees and the average individual accuracy of the trees as discussed in Section 3. In order to examine whether the steps really increase the diversity and individual

Table 8: Average Individual Kappa

Data Set Name	RRF	RRF on *RET*	SRF
Breast Cancer	**0.0944**	0.2632	0.1414
Chess	**0.4863**	0.5548	0.5838
Ionosphere	0.7823	0.8341	**0.7688**
Lung Cancer	**0.0650**	0.3740	0.0750
Soybean	**0.571**4	0.5914	0.5732
Average	**0.3999**	0.5235	0.4285

Table 11: Comparison of Ensemble Accuracies among RRF, SRF, CRF, CRF-Boosting, SysFor, MDMT and C4.5

Data Set Name	RRF	SRF	CRF	SysFor	MDMT	CS4	C4.5
Breast Cancer	78.8780	80.5480	**80.9840**	77.8260	70.5480	72.7460	70.5480
Chess	95.0570	94.9670	95.1820	**96.0010**	95.9700	95.9700	95.9700
Ionosphere	93.7310	93.1610	93.7400	92.8730	88.8890	92.0160	92.0160
Libras Movement	76.1100	78.8880	**83.6120**	73.0560	68.0560	76.1090	49.4430
Lung Cancer	68.8890	68.8890	87.7780	73.8890	88.8890	68.8890	
Soybean	99.0910	99.0910	**100.0000**	81.5910	93.1820	91.5910	72.2730
Statlog Vehicle	74.1430	73.8970	**76.5230**	71.9060	67.5250	72.8500	65.9610
Average	83.6999	84.2059	**88.2599**	83.1631	79.7227	84.3101	73.5857

Table 9: Performance of CRF' on RET

Data Set Name	EA	AIA	Kappa
Breast Cancer	80.0220	70.2911	0.2287
Chess	78.4540	72.2406	0.5103
Ionosphere	92.8660	89.1062	0.8286
Lung Cancer	83.8890	76.8501	0.3760
Soybean	98.1820	71.6705	0.6053
Average	86.6826	76.0317	0.5098

accuracy resulting in an increased ensemble accuracy we now implement Step 2, Step 3 and Step 4 of Algorithm 1 and call it CRF'. We then apply CRF' on the RET extended data sets. Table 9 presents the ensemble accuracy, average individual accuracy and Kappa for the CRF' on RET. From Table 9 and Table 6 we can see that average individual accuracy of trees obtained from CRF' on RET (76.03%) is higher than RRF on RET (75.44%). Moreover, the average individual Kappa of the trees obtained from CRF' on RET (0.51) is lower than RRF on RET (0.52) meaning that the CRF' on RET trees are more diverse than the RRF on RET trees (see Table 9 and Table 8). Finally from Table 9 and Table 7 we get the ensemble accuracy of the CRF' on RET trees (86.68%) is higher than the RRF on RET trees (84.59%). Therefore, these results justify the steps of Algorithm 1 suggesting that the steps increase the diversity among the trees, average individual accuracy of the trees and ensemble accuracy of the ensemble.

We argue that the ensemble accuracy of RRF on RET (84.59%) is worse than RRF (87.13%) as shown in Table 7) due to the decrease of the diversity among the trees obtained from RRF on RET compared to the trees obtained from RRF as shown in Table 8. The diversity of the trees obtained from RRF on RET decreases from the diversity of RRF due to the attribute space extension according to the RET (Adnan et al. 2014). As discussed in Section 2, the attribute space extension (Adnan et al. 2014) causes the repetition of the same original attributes (see Table 1 and Table 2) again and again in the extended space resulting in a decrease in diversity among the trees. Again the ensemble accuracy of CRF' on RET is worse than RRF (and also SRF) due to the same reason.

Therefore, our proposed attribute space extension technique (as presented in Step 1 of Section 3) aims to increase diversity among the trees. We now implement the complete CRF (i.e. all steps of Algorithm 1) and apply it on all seven (7) data sets as shown in Table 5.

CRF uses the bagging approach to generate bootstrap sample data sets (see Step 2 of Algorithm 1). We realize that CRF could also use a boosting approach (Bernard, Adam & Heutte 2012) to create sample data sets. Therefore, for a comparison purpose we also implement a modified CRF where in Step 2 we use a boosting approach (Bernard et al. 2012) to create sample data sets. We call this modified CRF as CRF-Boosting.

Table 10: Comparison of Ensemble Accuracies between CRF and CRF-Boosting

Data Set Name	CRF	CRF-Boosting
Breast Cancer	**80.9840**	79.9320
Chess	**95.1820**	95.1790
Ionosphere	93.7400	**94.0250**
Libras Movement	**83.6120**	81.3880
Lung Cancer	87.7780	**88.8890**
Soybean	**100.0000**	**100.0000**
Statlog Vehicle	**76.5230**	73.3100
Average	**88.2599**	86.8176

The boosting approach (Bernard et al. 2012) used in CRF-Boosting computes the weight of each original record in order to prepare the next sample data set. If the weight of a record is high then the chance of the record being included in the next sample data set is also high. The weight of a record is computed using all the trees built so far.

Let us assume that we have five sample data sets and thereby five trees. We now need to compute the weights of the records to prepare the sixth sample data set. Suppose one tree predicts the class value of a record R_i correctly and the remaining four trees predict it incorrectly then the weight of R_i is $W_i = 1 - \frac{1}{5} = 0.8$. Similarly if four trees predict it correctly then $W_i = 1 - \frac{4}{5} = 0.2$.

Table 10 shows that in five out of seven data sets CRF achieves a higher ensemble accuracy than CRF-Boosting. The average ensemble accuracy (over all data sets) of CRF (88.26%) is also higher than CRF-Boosting (86.82%). A possible reason can be the fact that the boosting approach is likely to create low quality sample data sets (especially towards the end) where majority of the records are difficult to classify. As a result the trees obtained from these low quality sample data sets are also supposed to have low individual accuracy. The use of these trees through the majority voting in predicting future records is likely to cause low ensemble accuracy.

Therefore, in this study we only compare CRF (not CRF-Boosting) with several decision forests algorithms. For a comprehensive analysis, we implement several recent forest building algorithms namely SRF (Ye et al. 2014), RRF (Breiman 2001), SysFor (Islam & Giggins 2011), MDMT (Hu, Li, Wang,

Daggard & Shi 2006), CS4 (Li & Liu 2003), and a well known decision tree algorithm called C4.5 (Quinlan 1993),(Quinlan 1996). The accuracy of C4.5 (Quinlan 1993),(Quinlan 1996) can be seen as a benchmark. Table 11 shows that the average (over all data sets) ensemble accuracy of CRF (88.26%) is considerably higher than RRF (83.70%), SRF (84.21%), SysFor (83.16%), MDMT (79.72%), CS4 (84.31%) and the benchmark algorithm C4.5 (73.59%). It is interesting to see that even the average ensemble accuracy of CRF-Boosting (86.82%) is higher than all other existing techniques.

As CRF is considered as a subsequent improvement of SRF and RRF, we now compare the tree structures of RRF, SRF and CRF in Table 12 and Table 13 where we present the average number of leaf nodes and average depth of the trees, respectively. Table 12 suggests that CRF has less number of leaf nodes (14) than RRF (17.58) and SRF (18.08). This indicates that the number of rules generated from CRF is less than RRF and SRF. Also, the depth of trees generated by CRF (5.26) is less than that of RRF (6.00) and SRF (6.16) (see Table 13). This implies that the rules generated from CRF are more concise, and thus more preferable (Geng & Hamilton 2006).

Table 12: Average Number of Leaf Nodes

Data Set Name	RRF	SRF	CRF
Breast Cancer	13.59	13.32	**13.11**
Chess	20.81	24.38	**9.46**
Ionosphere	12.13	12.30	**11.01**
Libras Movement	50.81	50.46	**42.90**
Lung Cancer	3.63	3.52	**3.31**
Soybean	4.52	4.48	**4.21**
Statlog Vehicle	75.10	**73.66**	74.38
Average	17.58	18.08	**14.00**

Table 13: Average Depth of Decision Tree

Data Set Name	RRF	SRF	CRF
Breast Cancer	6.67	**6.54**	6.66
Chess	7.58	8.71	**3.42**
Ionosphere	**6.39**	6.48	6.42
Libras Movement	11.01	**10.98**	11.33
Lung Cancer	2.48	2.41	**2.18**
Soybean	1.85	1.84	**1.52**
Statlog Vehicle	14.37	**14.30**	14.40
Average	6.00	6.16	**5.26**

5 Conclusion

We present a novel decision forest algorithm that aims to produce decision trees with high average individual accuracies and diversity. We experimentally analyse the importance of individual accuracy and diversity. The usefulness of various components of the proposed technique is analyzed and evaluated logically and experimentally. The proposed technique achieves better average ensemble accuracy (over seven publicly available data sets) than six existing techniques. It also achieves higher ensemble accuracy than all existing techniques in five out of seven data sets. The average ensemble accuracy of the proposed technique (88.26%) is significantly higher than the accuracy of a well known single tree algorithm called C4.5 (73.59%). The trees produced by the proposed technique are also compact in the sense that they have lower number of logic rules and shallower depth compared to some existing techniques.

There are many applications (such as medical diagnosis) where even a slight improvement in accuracy is highly desirable. They are happy to accept a high time complexity to build a classifier. Unlike many real time applications, they do not need to build classifiers frequently. However, it is always useful to reduce the time complexity. While we understand that all data sets (such as many bioinformatic data sets) are not always huge, we often get huge data sets these days. Therefore, our future research plan is to explore the usefulness of parallel processing to reduce the time complexity of the proposed technique, and thereby evaluate and increase its scalability.

References

Adnan, M. N. & Islam, M. Z. (2014), A comprehensive method for attribute space extension for random forest, *in* 'Proceedings of 17th International Conference on Computer and Information Technology'.

Adnan, M. N., Islam, M. Z. & Kwan, P. (2014), Extended space decision tree, *in* 'Machine Learning and Cybernetics, Communications in Computer and Information Science', Vol. 481, pp. 219–230.

Amasyali, M. F. & Ersoy, O. K. (2014), 'Classifier ensembles with the extended space forest', *IEEE Transactions on Knowledge and Data Engineering* **16**, 145–153.

Bernard, S., Adam, S. & Heutte, L. (2012), 'Dynamic random forests', *Pattern Recognition Letters* **33**, 1580–1586.

Bishop, C. M. (2008), *Pattern Recognition and Machine Learning*, Springer-Verlag New York Inc., NY, U.S.A.

Breiman, L. (1996), 'Bagging predictors', *Machine Learning* **24**, 123–140.

Breiman, L. (2001), 'Random forests', *Machine Learning* **45**, 5–32.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1985), *Classification and Regression Trees*, Wadsworth International Group, CA, U.S.A.

Burges, C. J. C. (1998), 'A tutorial on support vector machines for pattern recognition', *Data Mining and Knowledge Discovery* **2**, 121–167.

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002), 'Smote: Synthetic minority over-sampling technique', *Journal of Artificial Intelligence Research* **16**, 321–357.

Geng, L. & Hamilton, H. J. (2006), 'Interestingness measures for data mining: A survey', *ACM Computing Surveys* **38**, 1–32.

Geurts, P., Ernst, D. & Wehenkel, L. (2006), 'Extremely randomized trees', *Machine Learning* **63**, 3–42.

Han, J. & Kamber, M. (2006), *Data Mining Concepts and Techniques*, Morgan Kaufmann Publishers.

Ho, T. K. (1998), 'The random subspace method for constructing decision forests', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**, 832–844.

Hu, H., Li, J., Wang, H., Daggard, G. & Shi, M. (2006), A maximally diversified multiple decision tree algorithm for microarray data classification, *in* 'Proceedings of the Workshop on Intelligent Systems for Bioinformatics (WISB)', Vol. 73, pp. 35–38.

Islam, M. Z. & Giggins, H. (2011), Knowledge discovery through sysfor - a systematically developed forest of multiple decision trees, *in* 'Proceedings of the 9th Australian Data Mining Conference'.

Jain, A. K. & Mao, J. (1996), 'Artificial neural network: A tutorial', *Computer* **29**(3), 31–44.

Li, J. & Liu, H. (2003), Ensembles of cascading trees, *in* 'Proceedings of the third IEEE International Conference on Data Mining', pp. 585 – 588.

Mitchell, T. M. (1997), *Machine Learning*, McGraw-Hill, NY, U.S.A.

Polikar, R. (2006), 'Ensemble based systems in decision making', *IEEE Circuits and Systems Magazine* **6**, 21–45.

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, U.S.A.

Quinlan, J. R. (1996), 'Improved use of continuous attributes in c4.5', *Journal of Artificial Intelligence Research* **4**, 77–90.

Rodriguez, J. J., Kuncheva, L. I. & Alonso, C. J. (2006), 'Rotation forest: A new classifier ensemble method', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**, 1619–1630.

Tan, P. N., Steinbach, M. & Kumar, V. (2006), *Introduction to Data Mining*, Pearson Education.

UCI Machine Learning Repository (n.d.), http://archive.ics.uci.edu/ml/datasets.html. Last Accessed: 07/05/2015.

Ye, Y., Wu, Q., Huang, J. Z., Ng, M. K. & Li, X. (2014), 'Stratified sampling of feature subspace selection in random forests for high dimensional data', *Pattern Recognition* **46**, 769–787.

Zhang, G. P. (2000), 'Neural networks for classification: A survey', *IEEE Transactions on Systems, Man, and Cybernetics* **30**, 451–462.

Zhang, G., Patuwo, B. E. & Hu, M. Y. (1998), 'Forecasting with artificial neural networks: The state of the art', *International Journal of Forecasting* **14**, 35–62.

A Differentially Private Decision Forest

Sam Fletcher **Md Zahidul Islam**

School of Computing and Mathematics,
Centre for Applied Machine Learning (CAML)
Charles Sturt University,
Bathurst, New South Wales 2795, Australia,
Email: safletcher@csu.edu.au
Email: zislam@csu.edu.au

Abstract

With the ubiquity of data collection in today's society, protecting each individual's privacy is a growing concern. Differential Privacy provides an enforceable definition of privacy that allows data owners to promise each individual that their presence in the dataset will be almost undetectable. Data Mining techniques are often used to discover knowledge in data, however these techniques are not differentially privacy by default. In this paper, we propose a differentially private decision forest algorithm that takes advantage of a novel theorem for the local sensitivity of the Gini Index. The Gini Index plays an important role in building a decision forest, and the sensitivity of it's equation dictates how much noise needs to be added to make the forest be differentially private. We prove that the Gini Index can have a substantially lower sensitivity than that used in previous work, leading to superior empirical results. We compare the prediction accuracy of our decision forest to not only previous work, but also to the popular Random Forest algorithm to demonstrate how close our differentially private algorithm can come to a completely non-private forest.

Keywords: Differential Privacy, Decision Forest, Data Mining, Machine Learning, Privacy.

1 Introduction

Data collection and analysis plays an ever-growing role at in all facets of society, whether it be economic, medical, political, militaristic, academic or anything in between. For some of these areas, people's privacy is a concern that needs addressing. This most often occurs when data is collected about individuals, with some of the data being "sensitive" – that is, data that an individual would not like becoming public knowledge. Regardless of individuals' motivations for a desire for privacy, it is considered a basic human right by many, including the U.N. (UN General Assembly 1948), and is codified in many country's laws. It is therefore vital that when a company, government agency, or any other party is performing data collection and analysis, they have methods for guaranteeing the privacy of those individuals whose data is being used. In many situations, individuals may simply refuse to offer their data for collection if they do not have a privacy guarantee.

Differential privacy (Dwork 2006, Dwork et al. 2006, Dwork 2007, 2008, 2011, Dwork & Roth 2014, McSherry & Talwar 2007, McSherry 2009) is one such method for guaranteeing individuals' privacy. It does so by making a promise to each individual who supplies information to a dataset: "Any information that could be discovered about you with your data in the dataset could also, with high probability, be discovered *without* your data in the dataset". In other words, the output of any query Q performed on dataset D will be indistinguishable from the output of the same query Q performed on dataset D', where D' differs from D by at most one record (the record of any individual). The privacy guarantees made by differential privacy are far greater than those made by other popular privacy-preservation methods, such as k-anonymity (Sweeney 2002, LeFevre et al. 2005) or other generalization or noise addition techniques (Fung et al. 2010, Fletcher & Islam 2015). We define differential privacy in full in Section 2.1.

If the output of Q is to be restricted in some way to enforce differential privacy, the next question is "How do we enforce differential privacy while still outputting useful knowledge?". It is this question that we will answer, by querying the dataset in a way that allows us to produce a high quality, differentially private Decision Forest. A Decision Forest is the term used for a collection of different Decision Trees, which are a common type of classifier used in Data Mining. Decision Trees work by iteratively selecting attributes in the dataset that can most accurately classify a "class attribute"[1]. When an attribute is selected, the records in the dataset are split up according to what value they have for the chosen attribute[2]. For each of these partitions, the process is then repeated until a termination condition is met. Common termination conditions include a maximum number of times a partition will be split, a minimum number of records remaining in a partition, or when a partition can classify the class attribute with 100% accuracy. Our proposed Decision Forest will be based off CART (Breiman et al. 1984), and is explained in more detail in Section 2.2.

1.1 Problem Statement

Dataset D is a two-dimensional matrix of rows and columns, where each row (i.e. record) $r \in D$ de-

[1] The class attribute is the attribute that the user wishes to accurately predict the value of for future records, where the value is not known.

[2] The process is slightly different for continuous attributes that lack discrete values, but we will be focusing on discrete (i.e. categorical) attributes in this paper. Note that methods exist to "discretize" continuous attributes (Kotsiantis & Kanellopoulos 2006).

scribes a single individual, and each column is an attribute a in the set of attributes A. Each r possesses one discrete value $v \in a; \forall a \in A$. We symbolize that record r has value v for attribute a by writing $r_a = v$. The subset $D_{a_v} \subseteq D$ is the subset in which $r_a = v; \forall r \in D_{a_v}$. A subset with multiple value requirements is denoted similarly, with each requirement separated by a comma (see Figure 1).

Each r also has a class value c, from the class attribute C. The aim of a decision forest is to correct predict r_C (the class value c of record r) for records $r \in B : B \cap D = \emptyset$, where B and D are drawn from the same population.

A user is given limited access to D, in which they are allowed to query D in an ϵ-differentially private way. For any given query Q, the value of ϵ can be equal to or less than the amount provided to the user by the data owner. We define this amount as the total privacy budget β, and will be dividing β into smaller parts for each query Q.

Our aim is to build τ decision trees ($0 < \tau < \infty$) by only submitting ϵ-differentially private queries Q to D, and without exceeding our total budget β. The decision trees should be of acceptably high quality so that meaningful knowledge can be discovered.

1.2 Our Contributions

Our contributions can be summarized as the following:

- We present a differentially-private decision forest algorithm (referred to as DPDF).
- We output all the rules in the forest, including their confidence and support.
  - We also output the confidence and support of every subset of the rules (i.e. increasingly more general versions of the final rules, by removing the antecedents one at a time).
- We propose and prove a theorem for the worst-case local sensitivity[3] of the Gini Index.
- We demonstrate that the prediction accuracy of our DPDF is close to the accuracy of a standard Random Forest classifier (Breiman 2001), even for low ϵ values.

We also provide a URL to our publicly available code for DPDF.[4] In Section 2 we provide some necessary background on differential privacy, the CART algorithm and the prediction accuracy measure. In Section 3 we propose our differentially private decision forest algorithm and discuss the important components of it. In Section 4 we discuss another technique that is similar to ours, DiffPID3 (Friedman & Schuster 2010), and how it differs from our technique. In Section 5 we empirically compare our technique to DiffPID3, using Random Forest (Breiman 2001) as a benchmark of prediction accuracy on various datasets. In Section 6 we conclude the paper.

2 Background

2.1 Differential Privacy

While differential privacy has many applications beyond those used in this paper (McSherry & Talwar 2007), and can be phrased more generally to encompass those applications, we phrase the below definitions in a way that is more specific to our scenario, with dataset D.

Definition 1 (Differential Privacy (Dwork 2006))**.** *A query $Q : Q(D) \rightarrow Y$ satisfies ϵ-differential privacy if for all datasets D and D' differing by at most one record,*

$$Pr(Q(D) = y \in Y) \leq e^{\epsilon} \times Pr(Q(D') = y \in Y) \ . \quad (1)$$

This definition allows a data collector to make a strong promise to each individual in D: that for any query Q, the output observed is $\frac{1}{\exp(\epsilon)}$ as likely to occur even if they had not been in D. It does not promise that a malicious user cannot find out any information about them, but it does promise that any information they can find, they could have found without the individual even being in D. For example, there might exist a strong pattern that the malicious user knows (by using secondary information outside D) an individual matches, and that pattern would exist in D with or without the individual.

In order for Definition 1 to be possible for query Q to achieve, there must be a randomized component in Q, preventing any output y from being 100% likely. Two mechanisms are commonly used to inject randomness into queries: the Laplace Mechanism and the Exponential Mechanism. Before we define these mechanisms, we first need to define the "sensitivity" of Q:

Definition 2 (Sensitivity (Dwork et al. 2006))**.** *A query Q has sensitivity $\Delta(Q)$, where:*

$$\Delta(Q) = \max_{K,K'} |Q(K) - Q(K')| \quad (2)$$

and K and K' are any datasets that differ by at most one record.

Using Definition 2, we now define:

Definition 3 (The Laplace Mechanism (Dwork et al. 2006, Dwork & Roth 2014))**.** *A query Q satisfies ϵ-differential privacy if it outputs $y + Lap(\frac{\Delta(Q)}{\epsilon})$, where $y \in Y : Q(D) \rightarrow Y$ and $Lap(x)$ is an i.i.d. random variable drawn from the Laplace distribution with mean 0 and scale x (i.e. variance $2x^2$).*

Note that the Laplace Mechanism requires $Y \rightarrow \mathbb{R}$. For non-real outputs, we can use the Exponential Mechanism:

Definition 4 (The Exponential Mechanism (McSherry & Talwar 2007))**.** *Using a utility function $u(Q, y) : u \rightarrow \mathbb{R}$ where u has a higher value for more preferable outputs $y \in Y$, a query Q satisfies ϵ-differential privacy if it outputs y with probability proportional to $\exp(\frac{\epsilon u(Q,y)}{2\Delta(u)})$. That is,*

$$Pr(Q(D) = y) \propto \exp\left(\frac{\epsilon \times u(Q, y)}{2\Delta(u)}\right) \ . \quad (3)$$

We will later take advantage of two more theorems that have been proven about differential privacy:

Definition 5 (The Composition Theorem (McSherry & Talwar 2007))**.** *The application of queries Q_i, each satisfying ϵ_i-differential privacy, satisfies $\sum_i \epsilon_i$-differential privacy.*

Definition 6 (The Parallel Composition Theorem (McSherry 2009))**.** *Let D_i be a disjoint subset of dataset D. Let $Q_i(D_i)$ satisfy ϵ-differential privacy; then $\sum_i Q_i(D_i)$ also satisfies ϵ-differential privacy.*

[3] Sensitivity is an important component of Differential Privacy, described later.

[4] Our code can be found at `http://csusap.csu.edu.au/~zislam/` or you can email us.

2.2 CART

The structure of our decision trees will follow a similar structure to CART (Breiman et al. 1984). CART uses a recursive process in which attributes A (describing the records in D) are used to classify class attribute C. For any given attribute $a \in A$, the records in D are split into disjoint subsets D_{a_v} defined by which value $v \in a$ they have. We define each D_{a_v}'s ability to correctly classify a record's class value $c \in C$ using the Gini Index (Breiman et al. 1984):

$$G_{D,C}(a) = -\sum_{v \in a} \frac{|D_{a_v}|}{|D|} \left(1 - \sum_{c \in C} \left(\frac{|D_{a_v,c}|}{|D_{a_v}|}\right)^2\right) \quad . \tag{4}$$

The Gini Index is a measure of how often a randomly chosen record $r \in D$ would be incorrectly predicted to have class value $c : c \neq r_C$ if the predicted class value was randomly drawn from the distribution of class labels in D.

The attribute a with maximum $G_{D,C}(a)$ is chosen to split D into disjoint subsets $D_{a_v}; \forall v \in a$, and the process is repeated with each D_{a_v} being considered as it's own dataset D. The recursive process terminates when one of the following conditions are met:

- All records in D have the same class value c.
- All attributes have been used previously in the current recursion chain (an attribute can only be used once, since records in D_{a_v} cannot be further split by a).
- $|D_{a_v}|$ is below a user-defined minimum size. This is often done to prevent over-fitting to the training data.
- The depth of D_{a_v} in the tree has reached a user-defined maximum (i.e. the current recursion chain has repeated the maximum number of times). This is often done to limit computational complexity and limit the complexity (length) of the decision rules (the number of attributes required to predict the class attribute).

The output of this recursive algorithm is a decision tree T. An example of a decision tree can be seen in Figure 1. A tree T can be considered as a graph, and in this context (and no longer in the context of the recursive algorithm) the subsets D_i are often called "nodes", where i represents all the attributes in A (and the values of those attributes) that were used to define the subset D_i. The first node (i.e. subset) D is known as the "root node" (at $d = 1$ in Figure 1), and the final nodes D_i are known as "leaf nodes" (at $d > 1$ in Figure 1 if no more nodes exist lower in the chain).

2.3 Prediction Accuracy

The success of a decision tree T made with CART is usually measured with the prediction accuracy measure, and we use this measure in our paper. Prediction accuracy works by taking each record $r \in B : r \notin D; \forall r \in B$ (i.e. records not used in the tree-building process[5]) and following the logic of T into the disjoint subset D_{a_v} that matches the value r_{a_v}, starting from the first attribute used to split D (called the "root node") until the final split at the end of the recursion chain (called the "leaf node"). Note that it is only possible for r to match one $root \rightarrow leaf$ recursion chain in T. We then predict that $r_C = c$, where c is the most common class value in the leaf that r matched. After repeating this for all $r \in B$, the percentage of cases where our prediction was correct is called the "prediction accuracy" of T.

[5] B is often called the "testing data", as opposed to the "training data" D used to "train" the tree about the patterns in D.

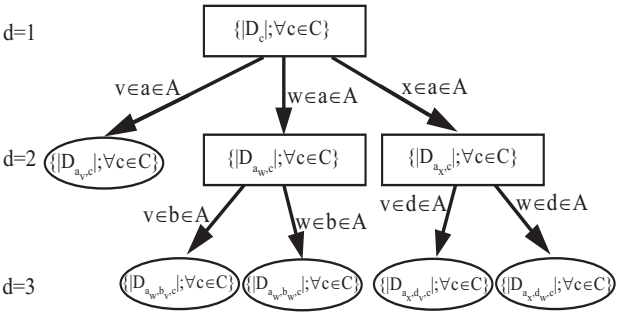


Figure 1: An example of a decision tree. Each arrow lists the value of an attribute that the records are being filtered according to. D is a dataset, with D_i being a subset of the dataset meeting the criteria i; C is the set of class values; A is the set of attributes, with each attribute having a set of values; d is the depth of the tree.

To calculate prediction accuracy for a forest F, the predicted class value for a record r from each tree $T \in F$ is tallied, and an algorithm is used to select the most appropriate class value to predict overall. This is known as "voting", and various voting algorithms have been developed over the years (Breiman 2001, Islam & Giggins 2011).

3 A Differentially Private Decision Forest

In order to make a decision tree (and from there a decision forest), we need to consider what information we require about D. In order to make the decision tree differentially private, we then need to query D in a way that returns to us all the information we need, without distorting the answers too much.

We break down the tree-building process into the following queries:

$$P_1(D_i) = \{|D_{i,c}|; \forall c \in C\} \tag{5}$$

and

$$P_2(D_i) = a' \in A : G_{D_i,C}(a') = \max_{a \in A}(G_{D_i,C}(a)) \quad , \tag{6}$$

where $D_i \subseteq D$, and i represents all the attributes in A (and the values of those attributes) that define what records are in D_i. These two queries are all that is required to recursively build the tree described in Section 2.2.

To make these queries differentially private, we add uncertainty to the output of each query in the following way:

$$Q_1(D_i) = \{|D_{i,c}| + Lap(\frac{1}{\epsilon}); \forall c \in C\} \tag{7}$$

and

$$Q_2(D_i) = a' \in A : Pr(G_{D_i,C}(a') = \max_{a \in A}(G_{D_i,C}(a))) \propto \exp\left(\frac{\epsilon \times G_{D_i,C}(a)}{2\Delta(G)}\right) \quad . \tag{8}$$

In words, $Q_1(D_i)$ outputs a histogram of the frequencies of each class value c in D_i, with Laplace noise added to each class count using Definition 3. The scale of the Laplace distribution is equal to $\frac{1}{\epsilon}$ for several reasons: when outputting a count of records, the sensitivity of the count is always $\Delta = 1$ (Dwork et al. 2006); and when outputting a histogram, the sensitivity remains equal to one because each bin of the histogram is disjoint – adding or removing one record can only affect one bin in the histogram (see Definition 6).

In words, $Q_2(D_i)$ outputs the attribute with the best Gini Index result with a certain probability, as described in Definition 4. The probability of any attribute $a \in A$ being outputted is dependent on the Gini Index of a, the sensitivity of the Gini Index, and ϵ. Since $\Delta(G)$ and ϵ are unchanging for all a, we can see that an attribute with a good Gini Index is exponentially more likely to be outputted than an attribute with a poor Gini Index. High $\Delta(G)$ and low ϵ both reduce this likelihood; we discuss how we reduce $\Delta(G)$ to a much lower value than that suggested by Definition 2 in Section 3.1.

The next question is how to divide the total privacy budget β amongst $Q_1(D_i)$ and $Q_2(D_i)$ for all i. Recall that a D_i exists for every $v \in a$, for every a used to split the previous D_i in the recursive algorithm described in Section 2.2, until each recursion chain has met one of the termination conditions listed in Section 2.2. Figure 1 provides an illustration of this. As per the Composition Theorem (Definition 5), we know that the ϵ we use for each query will be summed together, until it totals β and our connection to the dataset is severed by the data owner. We also know that for each depth level d, each D_i is disjoint, and $\sum_i |D_i| = |D|$ for each d (i.e. all records will belong to one and only one D_i for each d). We can therefore treat each D_i on the same depth level in a similar way to how we treat bins in a histogram, and that adding or removing one record can only affect one D_i. This allows us to apply the Parallel Composition Theorem (Definition 6) and ask Q_1 and Q_2 in a way that returns the output for each D_i, and only subtract ϵ from β once for Q_1 and once for Q_2.

Given the above, each query we use is given the following ϵ budget:

$$\epsilon = \frac{\beta}{2\delta - 1} \tag{9}$$

where $0 < \delta < \infty$ is the maximum depth we allow d to reach. We multiply δ by two because we are asking two queries per depth level: Q_1 and Q_2. We subtract one because upon reaching the end of a recursion chain (a leaf in the tree), we only need to ask Q_1 (to get the final distribution of class values).

We also expand our algorithm to be capable of producing multiple trees. Decision forests are known to often produce higher prediction accuracy than a single decision tree (Breiman 2001, Islam & Giggins 2011). The reason for this depends on the decision forest algorithm used, but is essentially because each tree will select different attributes a at different points in the tree, leading to different rules that might have better accuracy. For our algorithm, we guarantee that each tree will be different by requiring the first attribute chosen (the root) to be different for each tree. The noisy output of Q_2 also provides potential for different attributes to be chosen at other points in the trees.

When calculating the prediction accuracy of forest F in this paper, we use a simple voting technique of taking the weighted average of the predicted class values of record r, where the weight of each tree T's prediction is defined by the confidence[6] of the most common class value in the leaf that record r fits into. This is repeated for each record r in the testing data B. Any other voting technique can easily be used though, as long as it can be calculated using only the distribution of class values in D_i (unless some of budget β remains for more queries).

After defining the number of trees τ to build, the first time each tree (after the first tree) asks Q_2, the attributes chosen as roots of the previous trees are removed. There will therefore be τ different attributes used as the root attributes of the τ trees. The trees are completely independent beyond that.

We adjust the budget ϵ given to each query to account for τ:

$$\epsilon = \frac{\beta}{\tau(2\delta - 1)} \tag{10}$$

where $1 < \tau < \infty$.

We provide the full algorithm for DPDF in Algorithm 1.

3.1 The Local Sensitivity of the Gini Index

The sensitivity seen in Definition 2 is sometimes referred to as the "global sensitivity" of query Q, due to it making no assumptions about the data and simply returning the worst possible difference between any two datasets K and K' that differ by one record. By using the output of query $Q_1(D_i)$, we learn information about D_i that allows us to greatly reduce the sensitivity of $Q_2(D_i)$. We provide a theorem for the local sensitivity of $G_{D,C}(a)$ (and therefore of $Q_2(D_i)$), and its proof, below:

Theorem 1 (Local Sensitivity of the Gini Index)**.** *The sensitivity of the Gini Index $\Delta(G_{D,C}(a))$ when applied to data with known size $|D|$ is*

$$\Delta(G_{D,C}(a)) = 1 - \left(\frac{|D|}{|D|+1}\right)^2 - \left(\frac{1}{|D|+1}\right)^2 \quad . \tag{11}$$

It is independent of a and C, and therefore we can abbreviate $\Delta(G_{D,C}(a))$ to $\Delta(G_D)$.

Proof. From Equation 2 we see that in order to maximize $\Delta(G_{D,C}(a))$ we must produce a maximum and a minimum $G_{D,C}(a)$ such that both outputs are possible by only changing one record in D. Using Equation 4, we reduce the problem to

$$\max |\sum_{c \in C} \left(\frac{|D_{a_v,c}|}{|D_{a_v}|}\right)^2 - \sum_{c \in C} \left(\frac{|D'_{a_v,c}|}{|D'_{a_v}|}\right)^2 | \tag{12}$$

and will then extrapolate to all $v \in a$.

For D, we can write $\sum_{c \in C} \left(\frac{|D_{a_v,c}|}{|D_{a_v}|}\right)^2$ in a more general way:

$$\sum_i \left(\frac{x_i}{y}\right)^2 \tag{13}$$

where $\sum_i x_i = y$. For D', if we assume that we are considering $v : r_a = v$, where r is in D' but

[6] "Confidence" is the percentage of records in D_i that have the most common class value.

not in D, and that $x_1 = c : r_C = c$ we can write $\sum_{c \in C} \left(\frac{|D'_{a_v,c}|}{|D'_{a_v}|}\right)^2$ as

$$\left(\frac{x_1+1}{y+1}\right)^2 + \sum_{i=2}\left(\frac{x_i}{y+1}\right)^2 . \quad (14)$$

That is, the class count x_1 and the total y were increased by one because r was added to D'. It is possible that $x_1 = 0$ for D, which we write as

$$\left(\frac{1}{y+1}\right)^2 + \sum_{i=2}\left(\frac{x_i}{y+1}\right)^2 . \quad (15)$$

Remembering that $\sum_i x_i = y$, we can separate the numerators and denominators of Equations 13, 14 and 15 and see that

$$\begin{aligned}(y+1)^2 - y^2 &> \left((x_1+1)^2 + \sum_{i=2}(x_i)^2\right) - \sum_{i=1}(x_i)^2 \\ &> \left(1 + \sum_{i=2}(x_i)^2\right) - \sum_{i=1}(x_i)^2 \quad (16)\end{aligned}$$

which means the denominator is guaranteed to increase by more than the numerator (and thus result in a smaller number) when adding r to D, and Equation 15 will always be smaller than Equation 14. Thus we maximize Equation 12 by using Equations 13 (for D) and 15 (for D').

By taking advantage of the fact that

$$\sum_{i=1}^{2}\left(\frac{x_i}{y}\right)^2 > \sum_{i=1}^{>2}\left(\frac{x_i}{y}\right)^2 \quad (17)$$

and

$$\left(\frac{y}{y}\right)^2 > \left(\frac{x_1}{y}\right)^2 + \left(\frac{x_2}{y}\right)^2 \quad (18)$$

where $\sum_i x_i = y$, we can see that the worst-case scenario (i.e. where Equation 12 is maximized) is when $|D_{a_v,c}| = |D_{a_v}|$ (i.e. all records in D_{a_v} have the same class value c) and $D' = D \cup r$ where $r_C \neq c$. That is,

$$\max\left(\sum_{c \in C}\left(\frac{|D_{a_v,c}|}{|D_{a_v}|}\right)^2\right) = \left(\frac{|D_{a_v}|}{|D_{a_v}|}\right)^2 \quad (19)$$

and

$$\begin{aligned}\min&\left(\sum_{c \in C}\left(\frac{|D'_{a_v,c}|}{|D'_{a_v}|}\right)^2\right) \\ &= \left(\frac{|D_{a_v}|}{|D_{a_v}|+1}\right)^2 + \left(\frac{1}{|D_{a_v}|+1}\right)^2 , \quad (20)\end{aligned}$$

meaning that Equation 12 is equal to

$$\left|\left(\frac{|D_{a_v}|}{|D_{a_v}|}\right)^2 - \left(\left(\frac{|D_{a_v}|}{|D_{a_v}|+1}\right)^2 + \left(\frac{1}{|D_{a_v}|+1}\right)^2\right)\right| . \quad (21)$$

If Equation 21 is the maximum difference for $v \in a$, then it is also the maximum difference $\forall v \in a$, meaning the weighted average performed in the Gini Index (Equation 4) can be simplified:

$$\begin{aligned}-\sum_{v \in a}\frac{|D_{a_v}|}{|D|}&\left(1 - \sum_{c \in C}\left(\frac{|D_{a_v,c}|}{|D_{a_v}|}\right)^2\right) \\ &= -\left(1 - \sum_{c \in C}\left(\frac{|D_c|}{|D|}\right)^2\right) . \quad (22)\end{aligned}$$

From Equations 19 and 20, we know that Equation 12 (and therefore Equation 22) is optimal when all records in D have the same class value. Therefore when considering Equation 22 for D, we get

$$-\left(1 - \left(\frac{|D|}{|D|}\right)^2\right) = -(1-1) = 0 , \quad (23)$$

and when considering Equation 22 for D', we get

$$-\left(1 - \left(\frac{|D|}{|D|+1}\right)^2 - \left(\frac{1}{|D|+1}\right)^2\right) . \quad (24)$$

Combining the solutions for D and D' (Equations 23 and 24), we arrive at

$$\begin{aligned}\Delta(G_{D,C}(a)) &= \max_{D,D'}|G_{D,C}(a) - G_{D',C}(a)| \\ &= |0 - \left(1 - \left(\frac{|D|}{|D|+1}\right)^2 - \left(\frac{1}{|D|+1}\right)^2\right)| , \quad (25)\end{aligned}$$

noting that the above proof holds for the alternate case where one record is removed from a dataset by considering $|D| = |D'| - 1$.

□

Using Theorem 1 we can calculate the global sensitivity of the Gini Index, where the size of D is not known (recalling that the sensitivity is when $\Delta(G)$ is maximal):

$$\begin{aligned}\Delta(G) &= \max_{0<|D|<\infty}(1 - \left(\frac{|D|}{|D|+1}\right)^2 - \left(\frac{1}{|D|+1}\right)^2) \\ &= 1 - \left(\frac{1}{1+1}\right)^2 - \left(\frac{1}{1+1}\right)^2 \\ &= 0.5 .\end{aligned}$$

If $|D|$ is known, even a modest size of $|D| = 100$ heavily reduces the sensitivity of the Gini Index:

$$\begin{aligned}\Delta(G_D) &= 1 - \left(\frac{100}{100+1}\right)^2 - \left(\frac{1}{100+1}\right)^2 \\ &= 0.0196 .\end{aligned}$$

This drastically reduces the amount of noise added to the outputs of queries using the Exponential Mechanism (Definition 4) where the Gini Index (Equation 4) is the utility function u.

For DPDF, we can calculate $|D|$ using the sum of the class value frequencies we learned with query $Q_1(D)$:

$$|D| = \sum_{f \in Q_1(D)}|f| \quad (26)$$

Using this logic, we define a minimum number of records in D before the recursion chain is terminated (as explained in Section 2.2). For our experiments we set the minimum size of D to 100, thus making the upper limit of the sensitivity of Q_2 equal to $\Delta(G_D) = 0.0196$.

3.2 Pruning the Tree

Aside from calculating $|D_i|$, there is another advantage to using query $Q_1(D_i)$ at every D_i node, and not just in the leaf nodes where we need the class value distribution for predicting $r_C; \forall r \in B$ (where B is the testing data). By knowing the class distribution in every D_i node in every $root \rightarrow leaf$ recursion chain, it allows us to compare the leaf nodes to their parent nodes (the node above them in the chain) and assess their quality. By "quality", we mean "ability to correctly classify records r in B". If a parent node has higher quality than it's average child node (assuming that all the child nodes are leaf nodes), we perform what is known as "pruning".

Pruning is a component of many decision tree (and forest) algorithms, where leaf nodes are removed if they do not increase the prediction capabilities of the tree (Breiman et al. 1984, Quinlan 1993). The longer the $root \rightarrow leaf$ chains are, the more complicated they are, and the more they divide the records into smaller subsets, potentially over-fitting the tree to the training data D at the expense of prediction accuracy on the testing data B. If a leaf node is not actively helping the tree, it is more beneficial to remove the leaf. Some pruning techniques achieve pruning by using a validation set $B' : B' \cap B = \emptyset \; and \; B' \cap D = \emptyset$, such as CART's minimal cost complexity pruning (Breiman et al. 1984) and reduced error pruning (Quinlan 1993). However these techniques reduce the size of the training dataset D, which would increase the amount of relative noise added by Q_1 and Q_2. Since this is something we want to avoid, we instead perform pruning using the information we have already gathered during the tree-building process, specifically with Q_1.

Since all values $v \in a$ need to be handled, instead of removing individual bad leaf nodes, we measure the average quality of all leaf nodes $\forall v \in a$ and compare that to their parent node (which they all have in common). If

$$G(Q_1(D_i)) \geq \sum_{v \in a} \frac{|D_{i,a_v}|}{|D_i|} G(Q_1(D_{i,a_v})) \qquad (27)$$

where $D_{i,a_v} \forall v \in a$ are leaf nodes, we remove all of D_i's leaf nodes, causing D_i to become a leaf node instead. In the above equation, a is the attribute used to split D_i. Also recall that $Q_1(D_i)$ is the distribution of class values in D_i, which is the only information required to calculate the Gini Index G.

The reason that Equation 27 can possibly be true is that our tree building algorithm always splits a node D_i until a termination condition is met, even if all possible attributes to choose from have lower Gini Index results than D_i. Even if some attributes are better, the noisy output of $Q_2(D_i)$ might cause a poor attribute to be chosen. By allowing these situations to potentially occur, we help DPDF avoid getting stuck in a local optima – even if a child node y has worse Gini Index than it's parent node x, the child node z of y could still beat x's Gini Index! Including this pruning step in our algorithm then checks if either of these situations occurs, and retracts the tree to the global optima within the space explored by T.

3.3 Outputting the Rules and Subrules

Not only does possessing the output of $Q_1(D_i); \forall i$ allow us to perform pruning, but it also allows us to have many more rules than would be possible if we only had the class distribution of the leaf nodes. By a "rule", we mean the attributes chosen along a $root \rightarrow leaf$ chain, leading to the prediction of a certain class value c with a certain confidence. Depending on what sort of knowledge the user is searching for, these rules can be extremely valuable, allowing the user to see humanly-readable patterns in the data. The alternative is that tree T (or forest F) merely acts as a "black box" classifier, where records $r \in B$ are inputted and a predicted class value is outputted, with no information on *why*.

Not only does DPDF output the $root \rightarrow leaf$ rules, but also all $root \rightarrow node$ subsets of the $root \rightarrow leaf$ rules. This is only possible because we know what the predicted class value is for every node D_i, and the confidence of that prediction. Given that the user has a very strict privacy budget β with which to learn about dataset D, the more information we can gain from our queries – and the more we can recycle that information for multiple purposes – the better.

4 Related Work

Attempts at differentially private decision trees have been made in the past, most notably the Differentially Private ID3 algorithm (DiffPID3) by Friedman & Schuster (2010).

DiffPID3 uses a similar algorithm to our method, with some very important exceptions:

- Instead of getting the distribution of class values in each D_i, Q_1 just returns a noisy count of $|D_i|$.
  - This places limitations on DiffPID3's ability to prune the tree (Friedman & Schuster 2010).
  - It also does not allow for the first termination condition we use at each D_i subset (i.e. node): "All records in D_i have the same class value c", listed in Section 2.2. This increases the computation time of the algorithm by a small amount, both because it causes the algorithm to spend time making redundant nodes, and also because it increases the amount of pruning that needs to be done.
- They provide an extension to their algorithm to handle continuous attributes, however only at heavy cost to the budget β. Their results suggest a more feasible solution will need to be developed to handle continuous attributes with realistic β values.
- They do not discuss extending their algorithm beyond a single tree, while our algorithm builds τ distinct trees.
- They divide the privacy budget less efficiently, with $\epsilon = \frac{\beta}{2\delta}$, which makes a non-trivial difference at common values of δ.
- Most importantly of all, they use the global sensitivity of the Gini Index ($\hat{\Delta}(G) = 0.5$), adding a huge amount of unnecessary noise to their tree and reducing the prediction accuracy of the tree, as seen in Section 5.

We demonstrate in Section 5 that our improvements over their algorithm have a substantial positive effect on the prediction accuracy of the classifier.

Algorithm 1 The proposed Differentially Private Decision Forest (DPDF)

procedure DPDF(D, C, A, β, τ, δ)
  $\epsilon = \frac{\beta}{\tau(2\delta-1)}$
  $A_{root} = \{\}$ ▷ A_{root} will be used as a global variable
  $F = \{\}$
  for $t = 1, \ldots, \tau$ **do**
    $T =$ BUILDTREE(D, C, A, ϵ, δ, 1, True) ▷ The forest is composed of trees
    $T =$ PRUNETREE(T)
    $F = F \cup T$
  end for
  return F
end procedure

procedure BUILDTREE(D, C, A, ϵ, δ, d, root)
  T = {} ▷ The start of a tree, or a subtree
  $N_D^C = \{|D_c| + Lap(\frac{1}{\epsilon}); \forall c \in C \in D\}$ ▷ i.e. $Q_1(D)$, the noisy frequency of each class value in D
  $|D| = \sum_{f_c \in N_D^C} f_c$
  if $d \leq \delta$ and $|D| \geq 100$ and $\frac{f_c}{|D|} < 1; \forall f_c \in N_D^C$ and $|A| > 0$ **then**
    $\Delta(G_D) = 1 - (\frac{|D|}{|D|+1})^2 - (\frac{1}{|D|+1})^2$ ▷ The worst-case local sensitivity
    $a =$ SPLITTINGATTRIBUTE(D, C, A, ϵ, $\Delta(G_D)$, root)
    $A = A - \{a\}$ ▷ We cannot split on an attribute twice in a $root \rightarrow leaf$ chain
    for all $v \in a$ **do** ▷ Recall that A is public knowledge
      $D_{a_v} = \{r : r_a = v, \forall r \in D\}$ ▷ Note that D_{a_v} must remain on the server to preserve privacy
      $T = T \cup$ BUILDTREE(D_{a_v}, C, A, ϵ, δ, $d+1$, False) ▷ Attach a subtree to the tree
    end for
  end if
  return $\{T, N_D^C\}$ ▷ The tree is composed of nodes, each composed of a subtree and a Class Histogram
end procedure

procedure SPLITTINGATTRIBUTE(D, C, A, ϵ, $\Delta(G_D)$, root)
  if root **then**
    $A = A - A_{root}$ ▷ Two trees cannot have the same root attribute
  end if
  a' = Using the Exponential Mechanism, return $a' \in A$:

$$Pr(G_{D_i,C}(a') = \max_{a \in A}(G_{D_i,C}(a))) \propto \exp\left(\frac{\epsilon \times G_{D_i,C}(a)}{2\Delta(G)}\right)$$ ▷ i.e. $Q_2(D)$

  if root **then**
    $A_{root} = A_{root} \cup \{a'\}$ ▷ Recall that A_{root} is global
  end if
  return a'
end procedure

procedure PRUNETREE(T)
  for all Parent Nodes $P \in T$ **do** ▷ A Parent Node is any node with Child Nodes
    Child Nodes H^P = {All Child Nodes of P}
    if All Nodes $i \in H^P$ are Leaf Nodes **then** ▷ A Leaf Node is a Node with 0 Child Nodes
      $|H_i^P|$ = The sum of all Class Counts in H_i^P ▷ Each Node in H^P has a N_D^C
      $|P| = \sum_{i \in H_P} |H_i^P|$
      if $\sum_{i \in H^P} \frac{|H_i^P|}{|P|} \times G(H_i^P) < G(P)$ **then** ▷ We only need N_D^C to calculate the Gini Index
        Remove H^P from P ▷ P is now a leaf node
        PRUNETREE(T with updated P)
      end if
    end if
  end for
  return T
end procedure

5 Experiments and Results

Using six datasets from the UCI Machine Learning Repository (Bache & Lichman 2013), we compare the prediction accuracy of our proposed algorithm, DPDF, to DiffPID3 (Friedman & Schuster 2010). We implement the main version of DiffPID3 (Friedman & Schuster 2010) (that uses the Exponential Mechanism), and avoid datasets with continuous attributes so that DiffID3 is not disadvantaged by its expensive usage of the privacy budget β for continuous attributes. As a benchmark to demonstrate the potential the chosen datasets have for classification, we provide the prediction accuracy results of the popular Random Forest algorithm developed by Breiman (2001). We build the Random Forest using all of the default parameter settings in sci-kit learn 0.15.2 (Pedregosa et al. 2011). All reported prediction accuracies for all algorithms are the average prediction accuracy results of performing 10 iterations of stratified 10-fold cross-validation. This involves randomly dividing each dataset into 10 equal partitions in a way that keeps the class distribution in each partition as close to the whole dataset as possible. Nine of the partitions are then combined to make D and are used to build the classifier (whether it is DPDF, DiffPID3 or Random Forest). The final partition is used as the testing data B. This is repeated with all 10 combinations of nine partitions, so that each partition is used as B once. Ten iterations of this cross-validation process are performed, with the partitions being randomly generated each time. This means that 100 prediction accuracy results are produced, and the average of these are what we report in all our figures.

We test two parameter settings for our DPDF: where $\tau = 1$ and where $\tau = 4$. For all experiments with DPDF or DiffPID3, $\delta = 5$. We use 5 values of β for our experiments: 0.1, 0.25, 0.5, 1.0 and 2.0. Note that no privacy preservation of any kind is applied to Random Forest.

The datasets used are all publicly available and have the following names in the UCI Machine Learning Repository: "Car Evaluation" (Figure 2), "Chess (King-Rook vs. King)" (Figure 3), "Connect4" (Figure 4), "Mushroom" (Figure 5), "Nursery" (Figure 6), and "Tic-Tac-Toe Endgame" (Figure 7). The number of records in each dataset ranges from 958 to 67557; the number of attributes ranges from six to 42; the number of class values ranges from two to 18.

For most datasets, our algorithm halves the difference between DiffPID3's prediction accuracy and Random Forest's prediction accuracy. The biggest improvement over DiffID3 is seen with the Nursery dataset, where DPDF with $\tau = 1$ beats DiffPID3 by almost 25 percentage points, and comes within 7 percent percentage points of Random Forest. With the Tic-Tac-Toe and Connect4 datasets, we can see some overlap between DiffPID3 and DPDF when $\tau = 1$, however in both cases, DPDF always beats DiffPID3 when $\tau = 4$, indicating the benefit of having multiple trees. Interestingly, sometimes $\tau = 1$ beats $\tau = 4$; this may be due to the lower ϵ value available to each query Q when $\tau = 4$, leading to more noisy outputs. In Table 1, we demonstrate the difference between the size of ϵ for DiffPID3 and DPDF when $\tau = 1$ and $\tau = 4$, for each of the five β values we are testing.

As expected, the prediction accuracy of DiffPID3 and DPDF (for both $\tau = 1$ and $\tau = 4$) generally increases as the budget β increases. In a few situations this does not happen, notably for the Connect4 dataset. However in this dataset, Random Forest produces a worse prediction accuracy than all the differentially private algorithms at all β values, suggesting that the noisy outputs of DiffPID3 and DPDF helped the tree-building process. This may be due to the trees in Random Forest getting stuck in local optima, or simply that decision trees are not a good choice for data mining the Connect4 dataset. Prediction Accuracy behaving in this way when applying privacy-preservation techniques has been explored by previous work (Fletcher & Islam 2014).

Overall, it appears that for most datasets (except Tic-Tac-Toe, perhaps because it is by far the smallest dataset and thus has the noisiest outputs), DPDF produces a decision forest of acceptable quality for most data mining needs. This is true even when β is very low; as low as $\beta = 0.1$, where each query Q has $\epsilon = 0.003$ when $\tau = 4$. The user also has the complete list of rules and sub-rules from F, including the class distribution of each rule. This means the user can easily remove any rules or sub-rules with low confidence, leaving them with a shorter list of high quality rules.

Technique	Privacy Budget β				
	0.1	**0.25**	**0.5**	**1.0**	**2.0**
DiffPID3	0.010	0.025	0.050	0.100	0.200
DPDF, $\tau = 1$	0.011	0.028	0.056	0.111	0.222
DPDF, $\tau = 4$	0.003	0.007	0.014	0.028	0.056

Table 1: The size of ϵ per query, depending on the technique used and the total privacy budget β. In the example shown, the depth of the trees is $\delta = 5$.

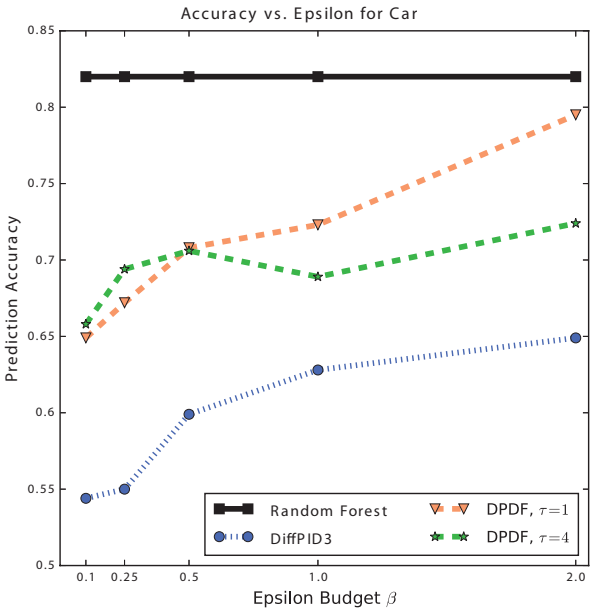


Figure 2: A comparison of our technique (DPDF) to DiffPID3 using the Car dataset, with Random Forest included as a benchmark. We test two parameter settings for DPDF: $\tau = 1$ and $\tau = 4$.

6 Conclusion

The success of DPDF at even very low β values provides the user with some valuable options – instead of using their entire allocated privacy budget β on a single run of DPDF, they can instead use only a fraction of it. The rest of the budget could be spent on anything the user wishes. This might include some preliminary queries on D in order to tune the δ and τ parameters, which would be an interesting direction

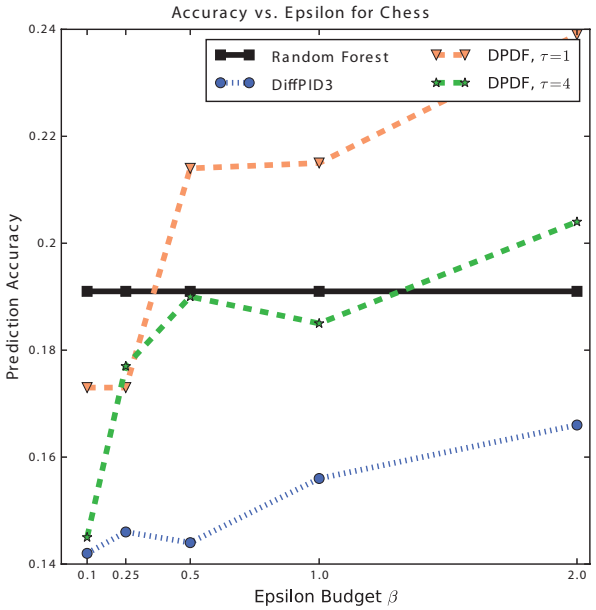


Figure 3: A comparison of our technique (DPDF) to DiffPID3 using the Chess dataset, with Random Forest included as a benchmark. We test two parameter settings for DPDF: $\tau = 1$ and $\tau = 4$. Note that there are 18 class values, so randomly guessing would give a prediction accuracy of 5.55%; hence the low Prediction Accuracy.

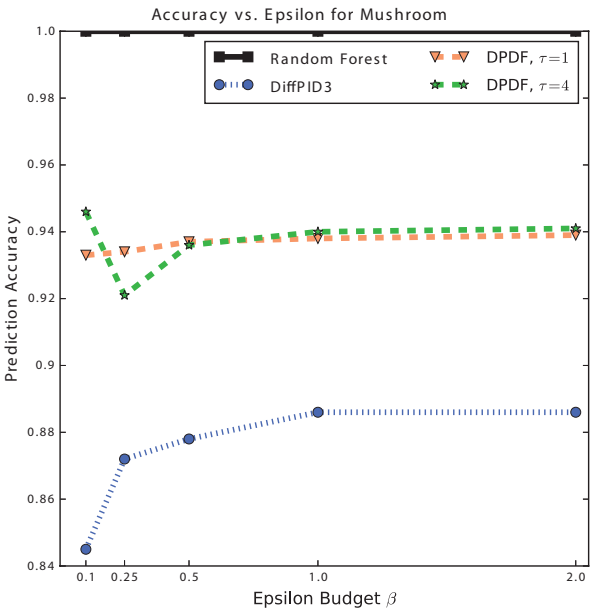


Figure 5: A comparison of our technique (DPDF) to DiffPID3 using the Mushroom dataset, with Random Forest included as a benchmark. We test two parameter settings for DPDF: $\tau = 1$ and $\tau = 4$.

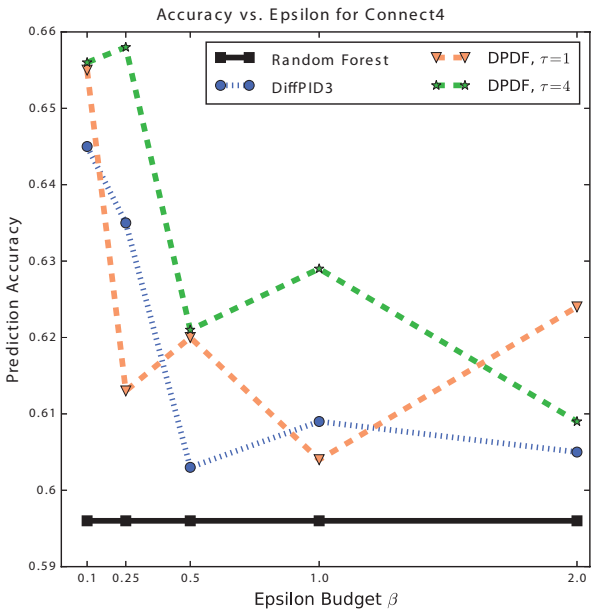


Figure 4: A comparison of our technique (DPDF) to DiffPID3 using the Connect4 dataset, with Random Forest included as a benchmark. We test two parameter settings for DPDF: $\tau = 1$ and $\tau = 4$.

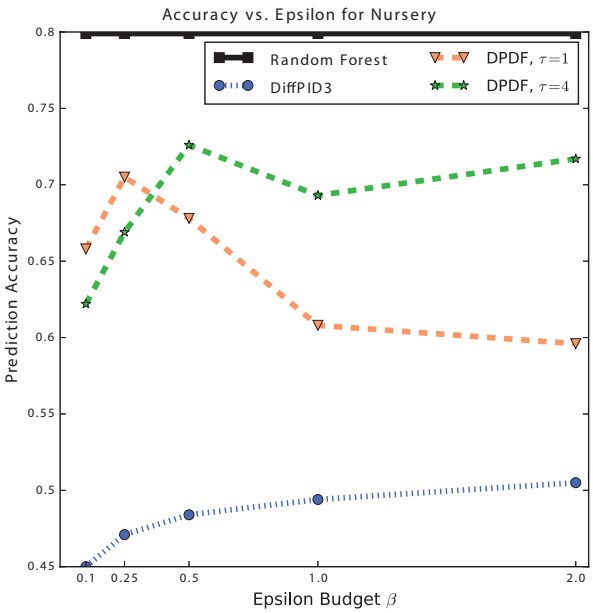


Figure 6: A comparison of our technique (DPDF) to DiffPID3 using the Nursery dataset, with Random Forest included as a benchmark. We test two parameter settings for DPDF: $\tau = 1$ and $\tau = 4$.

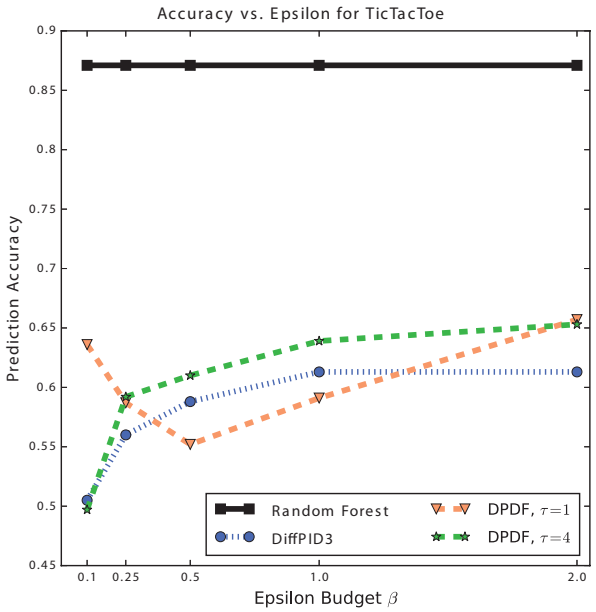


Figure 7: A comparison of our technique (DPDF) to DiffPID3 using the Tic-Tac-Toe dataset, with Random Forest included as a benchmark. We test two parameter settings for DPDF: $\tau = 1$ and $\tau = 4$.

for future research. It could include multiple runs of DPDF, each with different parameters. It could include completely different data mining algorithms, such as clustering (as long as it is differentially private). The strong mathematical properties of differential privacy allow the data owner to guarantee the individuals in the dataset that their presence in the dataset is almost completely undetectable, no matter how a user decides to divide their β budget. Our novel theorem on the local sensitivity of the Gini Index will be useful to any future classification algorithms that wish to perform differentially private data mining. We provide the code required to implement DPDF, so that data miners and fellow researchers may take advantage of it.

References

Bache, K. & Lichman, M. (2013), 'UCI Machine Learning Repository'.
URL: *http://archive.ics.uci.edu/ml/*

Breiman, L. (2001), 'Random forests', *Machine learning* pp. 1–35.

Breiman, L., Friedman, J., Stone, C. & Olshen, R. (1984), *Classification and regression trees*, Chapman & Hall/CRC.

Dwork, C. (2006), Differential Privacy, *in* 'Automata, languages and programming', Vol. 4052, Springer Berlin Heidelberg, Venice, Italy, pp. 1–12.

Dwork, C. (2007), 'An Ad Omnia Approach to Defining and Achieving Private Data Analysis'.

Dwork, C. (2008), 'Differential Privacy: A survey of results', *Theory and Applications of Models of Computation* pp. 1–19.

Dwork, C. (2011), 'A firm foundation for private data analysis', *Communications of the ACM* **54**(1), 86–95.

Dwork, C., McSherry, F., Nissim, K. & Smith, A. (2006), 'Calibrating noise to sensitivity in private data analysis', *Theory of Cryptography* pp. 265–284.

Dwork, C. & Roth, A. (2014), *The Algorithmic Foundations of Differential Privacy*, Vol. 9, Now Publishers.

Fletcher, S. & Islam, M. Z. (2014), Quality evaluation of an anonymized dataset, *in* '22nd International Conference on Pattern Recognition', IEEE, Stockholm, Sweden, pp. 3594–3599.

Fletcher, S. & Islam, M. Z. (2015), 'An Anonymization Technique using Intersected Decision Trees', *Journal of King Saud University - Computer and Information Sciences* **27**(3).

Friedman, A. & Schuster, A. (2010), Data Mining with Differential Privacy, *in* '16th SIGKDD Conference on Knowledge Discovery and Data Mining', ACM, Washington, DC, USA, pp. 493–502.

Fung, B., Wang, K., Chen, R. & Yu, P. (2010), 'Privacy-preserving data publishing: A survey of recent developments', *ACM Computing Surveys (CSUR)* **42**(4), 14.

Islam, M. Z. & Giggins, H. (2011), Knowledge discovery through SysFor: a systematically developed forest of multiple decision trees, *in* 'Ninth Australasian Data Mining Conference-Volume 121', Australian Computer Society, Inc., Ballarat, Australia, pp. 195–204.

Kotsiantis, S. & Kanellopoulos, D. (2006), 'Discretization techniques: A recent survey', *GESTS International Transactions on Computer Science and Engineering* **32**(1), 47–58.

LeFevre, K., DeWitt, D. & Ramakrishnan, R. (2005), Incognito: Efficient full-domain k-anonymity, *in* 'Proceedings of the 2005 ACM SIGMOD international conference on Management of data', ACM, pp. 49–60.

McSherry, F. (2009), Privacy integrated queries: an extensible platform for privacy-preserving data analysis, *in* 'Proceedings of the 35th SIGMOD international conference on Management of data', ACM, Providence, USA, pp. 19–30.

McSherry, F. & Talwar, K. (2007), 'Mechanism Design via Differential Privacy', *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)* pp. 94–103.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine Learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

Quinlan, J. R. (1993), *C4.5: programs for machine learning*, 1st edn, Morgan kaufmann.

Sweeney, L. (2002), 'Achieving k-anonymity privacy protection using generalization and suppression', *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **10**(5), 571–588.

UN General Assembly (1948), 'Universal Declaration of Human Rights'.

Mining Productive Emerging Patterns and Their Application in Trend Prediction

Vincent Mwintieru Nofong

School of Information Technology and Mathematical Science,
University of South Australia, GPO Box 2471, Adelaide, SA 5001
Email: vincent.nofong@mymail.unisa.edu.au

Abstract

Emerging pattern mining is an important data mining task for various decision making. However, it often presents a large number of emerging patterns most of which are not useful as their emergence are due to random occurrence of items. Such emerging patterns would most often be detrimental in decision making where inherent relationships between the items of emerging patterns are relevant. Additionally, most studies on emerging pattern mining focus on mining interesting categories of emerging patterns for classification and seldom discuss their application in trend prediction. To enable mine the set of emerging patterns with inherent item relations for decision making such as trend prediction, we employ a correlation test on the items of emerging patterns and introduce the productive emerging patterns as the set of emerging patterns with inherent item relations. We subsequently propose and develop PEPs, an efficient framework for mining our proposed productive emerging patterns. We further discuss and show the possible application of emerging patterns in trend prediction. Our experimental results shows PEPs is efficient, and the productive emerging pattern set which is smaller than the set of all emerging patterns, shows potentials in trend prediction.

Keywords: Frequent Patterns, Emerging Patterns, Productiveness Measure, Trend Prediction.

1 Introduction

Emerging Patterns (EPs), the set of patterns whose frequencies increase from one dataset to another, are vital in various decision making. In static datasets such as those with classes (male vs. female, cured vs. not cured), emerging patterns can reveal useful and hidden contrast patterns between datasets to support decision making such as classifier construction (Dong and Li 1999, Li et al. 2001), disease likelihood prediction (Li et al. 2003), discovering patterns in gene expression data (Li and Wong 2001), and so on. In sequential datasets, emerging patterns are useful in decision making such as, studying and understanding customers' behaviour (Tsai and Shieh 2009), predicting future purchases (Nofong et al. 2014) and so on.

Though emerging pattern mining is an important data mining task, it is a difficult task as the downward closure property in frequent pattern mining is not applicable in emerging pattern mining (Cheng et al. 2010, Dong and Li 1999, Poezevara et al. 2011). Over the past years however, various studies have been proposed for efficient mining of emerging patterns (Dong and Li 2005, Li et al. 2003, Li and Wong 2001) and interesting emerging patterns (Fan and Ramamohanarao 2003, 2006, 2002, Li et al. 2001, Terlecki and Walczak 2007, Soulet et al. 2004). Though these works have been useful in mining emerging patterns for various decision making, they are faced with the following challenges:

- They often present a too large or a too small number of emerging patterns for decision making. Reporting a large number of emerging patterns makes it difficult to identify the set of useful ones as some might be: i.) redundant, or ii.) emerging due to random occurrence of items. Such redundant emerging patterns, or those due to random occurrence of items, would most often be detrimental in decision making where non-redundancy or inherent relationships between items of an emerging pattern are vital. On the other hand, reporting a small number of emerging patterns may result in missing some useful emerging patterns that are needed in decision making.
- They often focus on mining interesting sets of emerging patterns for classification and seldom discuss their application in trend prediction. Though emerging patterns can reveal useful emerging trends in time-stamped datasets for trend prediction, this useful application of emerging patterns is unexplored as it is hardly mentioned in existing works on emerging pattern mining.
- Though some categories of emerging patterns such as *jumping* emerging patterns (Fan and Ramamohanarao 2006, Terlecki and Walczak 2007) and *essential* emerging patterns (Fan and Ramamohanarao 2002) are very useful in classifier formation, they will not be ideal in trend prediction. This is because, per their definitions, such emerging patterns in time-stamped datasets will more likely be spikes or noise, and not emerging trends.
- Though the emerging patterns reported in (Fan and Ramamohanarao 2003) can be applicable in trend prediction, some useful emerging patterns needed in decision making might be missed. For instance, on a Twitter dataset, (Fan and Ramamohanarao 2003) misses some interesting and useful emerging hashtags such as,

"#tcot[1]#romneyryan2012", "#tcot#Obama", and "#news#Syria". The emergence of these hashtags though formed by items with inherent relationships (correlated items) and reflective of true emerging trends are missed in (Fan and Ramamohanarao 2003) because their rates of emergence are lower than those of their emerging subsets.

Motivated by the importance of emerging patterns in decision making and the aforementioned challenges in their discovery, we address these challenges as follows. We initially employ a correlation test on emerging patterns and introduce the productive emerging patterns as the set of emerging patterns with inherent item relationships. Subsequently, we propose and develop PEPs, an efficient framework for mining the set of productive emerging patterns, and show their possible application in trend prediction.

We make the following contributions to the discovery of emerging patterns.

- We propose and introduce the productive emerging pattern set as the set of emerging patterns with inherent item relationships.
- We propose and develop PEPs, an efficient productive emerging pattern mining framework.
- We show a possible application of emerging patterns in trend prediction.

In addition to these contributions, it is also worth noting that our proposed productive emerging pattern set achieves a major size reduction in the number of reported emerging patterns.

2 Related Works

The concept to emerging pattern mining was introduced by Dong and Li in (Dong and Li 1999) where they proposed an emerging pattern detection technique for static datasets with classes. They referred to an emerging pattern as an itemset whose support increases significantly from one dataset to another. More specifically, they defined an emerging pattern as an itemset whose growth rate is greater than a given threshold. Emerging pattern mining has since been researched on in works such as (Fan and Ramamohanarao 2003, 2006, 2002, Garcia-Borroto et al. 2014, Li et al. 2003, 2001, Terlecki and Walczak 2007, Soulet et al. 2004).

Over the past years however, some researchers argued that the emerging pattern definition proposed in (Dong and Li 1999) often generates too many emerging patterns making it difficult identifying the set of interesting and useful ones for decision making. Various constraints and techniques were thus proposed to enable mine interesting categories of emerging patterns. Such works include, but not limited to: jumping EPs (Fan and Ramamohanarao 2006, Li et al. 2001, Terlecki and Walczak 2007), essential EPs (Fan and Ramamohanarao 2002), and interesting EPs (Fan and Ramamohanarao 2003, Soulet et al. 2004).

Though the above mentioned works have been useful in mining emerging patterns for various decision making, they are faced with several challenges summarized as follows. Firstly, they often report a too large or a too small emerging pattern sets for decision making. Secondly, they focus on mining interesting sets of emerging patterns for classification and seldom discuss their application in trend prediction. Thirdly, most categories of emerging patterns mined in works such as (Fan and Ramamohanarao 2006, 2002, Terlecki and Walczak 2007) though very good in classifier formation, are not applicable in trend prediction. Additionally, it is worth noting that though emerging patterns mined in works such as (Fan and Ramamohanarao 2003) are applicable in trend prediction, it often misses some useful emerging patterns needed in decision making.

Inspired by the importance of emerging patterns, the aforementioned challenges in their discovery, and their possible application in trend prediction, we focus on how to mine the set of emerging patterns with inherent item relationships and their possible application in trend prediction.

[1]The hashtag #tcot, "Top Conservatives On Twitter" provides a way for conservatives and Republicans to locate and follow the tweets of their like-minded brethren.

3 Preliminaries

The problem of frequent pattern mining and its associated notation can be given as follows. Let $\mathrm{I} = \langle i_1, i_2, ..., i_n \rangle$ be a set of literals, called items. Then, a transaction is a nonempty set of items. A pattern S is a set of transactions satisfying some conditions of measures like frequency. A pattern is of length-k if it has k items, for example, $S = \{a, b, c\}$ is a length-3 pattern.

Given a database of n transactions, $\mathbf{D} =< T_1, T_2, T_3, \ldots, T_n >$, where each T_m in $\mathbf{D}$ is identified by m called TID, the *cover* of a pattern S in $\mathbf{D}$, $cov_{\mathbf{D}}(S)$, is the set of *TIDs* of transactions that contain S. That is,

$$cov_{\mathbf{D}}(S) = \{m : T_m \in \mathbf{D} \wedge S \subseteq T_m\} \tag{1}$$

The *support* of a pattern S in $\mathbf{D}$, $sup_{\mathbf{D}}(S)$, is defined as,

$$sup_{\mathbf{D}}(S) = \frac{|cov_{\mathbf{D}}(S)|}{|\mathbf{D}|} \tag{2}$$

where $|cov_{\mathbf{D}}(S)|$ is called the *support count* of S in $\mathbf{D}$.

Frequent pattern mining is the process of discovering all patterns in a database, $\mathbf{D}$, whose frequencies are larger than or equal to a user specified minimum support (η). A pattern S in $\mathbf{D}$ is said to be productive in $\mathbf{D}$ if (Webb 2010): for all S_1, S_2 (such that, $S_1 \subset S,\ S_2 \subset S,\ S_1 \cup S_2 = S,\ S_1 \cap S_2 = \emptyset$), $sup_{\mathbf{D}}(S) > sup_{\mathbf{D}}(S_1) sup_{\mathbf{D}}(S_2)$.

3.1 Emerging Patterns

Given two datasets, $\mathbf{D_i}$ and $\mathbf{D_{i+1}}$, the growth rate of a pattern S, $GR(S)$, from $\mathbf{D_i}$ to $\mathbf{D_{i+1}}$ is defined as (Dong and Li 1999):

$$GR(S) = \frac{sup_{D_{i+1}}(S)}{sup_{D_i}(S)} \tag{3}$$

Based on the growth rate, Dong and Li in (Dong and Li 1999) introduced the concept of emerging pattern mining. Formally, they defined an emerging pattern as follows.

Definition 1 *(Dong and Li 1999) Given $\rho > 1$ as the growth-rate threshold, a pattern S is said to be a* ρ-emerging *(ρ-*EP *or simply* EP*) from $\mathbf{D_i}$ to $\mathbf{D_{i+1}}$ if $GR(S) \geq \rho$.*

For any two datasets, Definition 1 will report all patterns whose growth rates are greater than or equal to the specified growth rate threshold, ρ.

Though Definition 1 has been accepted and used in mining emerging patterns, it has the following challenges. Firstly, a large number of emerging patterns are often reported and this makes it difficult to comprehend and identify the set of useful ones for decision making. Secondly, the emergence threshold ρ largely determines the number of discovered emerging patterns. If ρ is set low, a large set of emerging patterns will be discovered, most of which might be trivial. However, if ρ is set high, some useful emerging patterns needed in decision making will be missed.

Over the past years, some researchers argued that finding all EPs above a minimum growth rate constraint as proposed in (Dong and Li 1999) often generates too many emerging patterns to be analysed. Soulet et. al. in (Soulet et al. 2004) thus proposed a condensed representation approach for mining emerging patterns based on closed patterns. Fan and Ramamohanarao in (Fan and Ramamohanarao 2003), whose work is quite similar to ours however proposed a way of selecting the set of *interesting* emerging patterns. They define an interesting emerging pattern as follows.

Definition 2 *(Fan and Ramamohanarao 2003) Given $\rho > 1$ as the growth rate threshold, a pattern S is an interesting emerging pattern from $\mathbf{D_i}$ to $\mathbf{D_{i+1}}$ if:*

1. *S is frequent in both $\mathbf{D_i}$ and $\mathbf{D_{i+1}}$,*
2. *$GR(S) \geq \rho$,*
3. *$\forall Y \subset S, GR(Y) < GR(S)$, and,*
4. *$|S| = 1$, or $|S| > 1$ and $\forall Y \subset S$ such that $|Y|=|S|$-1, then, $\chi^2[sup_{\mathbf{D_i}}(S), sup_{\mathbf{D_{i+1}}}(S), sup_{\mathbf{D_i}}(Y), sup_{\mathbf{D_{i+1}}}(Y)] \geq \eta$.*

In Definition 2, the authors aimed at identifying the set of emerging patterns that:

- Cover both datasets - Condition 1.
- Have sharp discriminating powers - Condition 2.
- Are not subsumed by their emerging subsets - Condition 3.
- Have significantly different supports from their immediate subsets to ensure the items of an emerging pattern are correlated - Condition 4.

Though Definition 2 will report a set of emerging patterns as interesting, some useful emerging patterns which capture and reflect vital contrasts or emerging trends will be missed for the following reasons:

1. When ρ is set high in Condition 2: Similar as in Definition 1, when ρ is set high, some useful emerging patterns with inherent item relationships whose rates of emergence are lower than ρ will be missed.
2. The subsumption rule in Condition 3: Some useful emerging patterns whose rates of emergence are lower than their emerging subsets will be missed due to the subsumption rule in Condition 3. For instance, in a Twitter dataset[2] for the month of November 2012, when we set $\varepsilon = 0.04\%$, $\rho = 1.0$ and $\eta = 0.0$, some emerging hashtags which have inherent item relationships (correlated items), such as; #tcot#romneyryan2012, #tcot#Obama, and #news#Syria, reflecting important emerging trends from 1^{st} to 2^{nd}, and from 2^{nd} to 3^{rd} of November, were missed by Definition 2. These emerging hashtags were missed as their growth rates are less than those of their emerging subsets, #tcot, #romneyryan2012, #Obama and #news respectively. However, the emergence of these subsets do not indicate the emergence of their supersets. That is, though the emerging hashtag #tcot could be easily associated with #romneyryan2012 in #tcot#romneyryan2012, so cannot be said of the emergence of #Obama in #tcot#Obama. Similarly, the emergence of #news does not in anyway imply that news about Syria, #news#Syria, is also emerging.
3. When $\eta \gg 0.0$ in Condition 4: Though this is aimed at finding emerging patterns with correlated items, some useful emerging patterns with correlated items will be missed when $\eta \gg 0.0$ in Condition 4. This is because some emerging patterns with correlated items might not have significantly different supports from their immediate subsets. Such emerging patterns which could be useful in decision making will thus be missed when $\eta \gg 0.0$ in Condition 4.

[2]Obtained from CNetS (`http://carl.cs.indiana.edu/data/`).

4 Problem Statement and Definitions

With Definitions 1 and 2, though the set of emerging patterns and interesting emerging patterns can be identified, as mentioned in Sections 1, 2 and 3, some of these reported emerging patterns may be emerging due to random occurrence of items or some emerging patterns with inherent item relationships needed in decision making will be missed. To avoid these situations, we begin by defining an emerging pattern as follows.

Definition 3 *Given ε as the minimum support, a pattern S is an emerging pattern from $\mathbf{D_i}$ to $\mathbf{D_{i+1}}$ if it is frequent in both $\mathbf{D_i}$ and $\mathbf{D_{i+1}}$ and $GR(S) > 1.0$.*

For any two datasets, Definition 3 will detect and report all frequent patterns whose growth rates are greater than 1.0. Definition 3 requiring emerging patterns to have a growth rate greater than 1.0 eliminates the situation in Definitions 1 and 2 where ρ largely controls the number of reported emerging patterns. Also, the minimum support threshold like in Definition 2, ensures only frequent patterns that are emerging are reported. However, given two datasets and the same minimum support with $\rho = 1.0$ and $\eta = 0.0$, though Definition 2 will report a smaller set of emerging patterns, it will miss some vital emerging patterns having inherent item relationships. Definition 3 will however report all such emerging patterns.

Though Definition 3 will not miss emerging patterns with inherent item relationships, some reported emerging patterns in Definition 3 might be emerging due to random occurrence of items. In decision making where inherent relationships among items of an emerging pattern are vital, emerging patterns that are emerging due to random occurrence of items could be detrimental. This is because such emerging patterns which do not encode inherent item relationships will more likely be spikes or noise, and not emerging trends.

To enable detect and report only emerging patterns with inherent item relationships for decision making, we test for positive correlations among all items of an emerging pattern. We employ a productiveness measure proposed in (Webb 2010) for this

test and refer to emerging patterns with inherent item relationships as productive emerging patterns. Formally we define a productive emerging pattern as follows.

Definition 4 *An emerging pattern, S, from $\mathbf{D_i}$ to $\mathbf{D_{i+1}}$, is a productive EP if, $\forall S_1, S_2$ (such that, $S_1 \subset S \ \wedge S_2 \subset S \ \wedge S_1 \cup S_2 = S \ \wedge S_1 \cap S_2 = \phi$) then $sup_{D_i}(S) > sup_{D_i}(S_1)sup_{D_i}(S_2)$ and $sup_{D_{i+1}}(S) > sup_{D_{i+1}}(S_1)sup_{D_{i+1}}(S_2)$.*

Definition 4 implies an emerging pattern is productive if and only if every subset that can be formed from it have inherent item relationships in both $\mathbf{D_i}$ and $\mathbf{D_{i+1}}$. This productiveness measure for every subset is to ensure all items of an emerging pattern encode inherent relationships and not due to random occurrences. This measure in Definition 4 covers the case where an emerging pattern has more than two subsets of items that are independent of one another (Webb 2010). Since the supersets of a non-productive pattern will always contain the non-productive pattern, we use this productiveness measure as one of our main pruning strategies in PEPs to avoid reporting emerging patterns with non-productive subsets. In the rest of our work, we represent the set of productive emerging patterns from $\mathbf{D_i}$ to $\mathbf{D_{i+1}}$ as pE_i^{i+1}.

With Definition 4, our emerging pattern mining problem can now be defined as the process of mining all productive emerging patterns from dataset, $\mathbf{D_i}$ to $\mathbf{D_{i+1}}$, given a minimum support ε, and how they can be employed in trend prediction.

5 Productive Emerging Pattern Mining and Their Application in Trend Prediction

In this section, we firstly discuss and introduce PEPs, our Productive Emerging Pattern mining framework. We follow up with a discussion on how the detected productive emerging patterns can be applied in trend prediction.

5.1 Productive Emerging Pattern Mining

To efficiently mine the set of productive emerging patterns, we propose PEPs, an efficient productive emerging pattern mining framework shown in Algorithm 1. PEPs employs the Apriori-like candidate

Algorithm 1: PEPs($D_i, D_{i+1}, \varepsilon$)

Input: D_i, D_{i+1}, minimum support ε
Output: Productive EP set, pE_i^{i+1}

1 Create set $pE_i^{i+1} = \emptyset$
2 ScanData(D_i, ε) to return F_i
3 ScanData(D_{i+1}, ε) to return F_{i+1}
4 Create set L
5 **for *each*** *item* $a_y \in F_i$ **do**
6 **if** $a_y \in F_{i+1}$ **then**
7 Let $(a_y, cov_{D_i}(a_y)) = F_i(a_y)$
8 Let $(a_y, cov_{D_{i+1}}(a_y)) = F_{i+1}(a_y)$
9 Add $(a_y, cov_{D_i}(a_y), cov_{D_{i+1}}(a_y))$ to L
10 Sort L in item descending order
11 MineEPs(L, ε)
12 **return** pE_i^{i+1}

generation technique in (Agrawal and Srikant 1995). However, for any two datasets, PEPs stores the TIDs of each frequent length-1 item in both datasets to avoid repeated scanning of the datasets and for quick implementation.

PEPs employs three major steps in the productive emerging pattern mining process: i.) finding the length-1 frequent items in the two datasets, ii.) identifying the common length-1 frequent items, and iii.) mining the productive emerging patterns from the common length-1 frequent items. We discuss each step in the following sections.

5.1.1 Finding Frequent Length-1 Items

For any two datasets, $\mathbf{D_i}$ and $\mathbf{D_{i+1}}$, this step finds the set of frequent length-1 items in both datasets with regards to the minimum support using Algorithm 2 (in Lines 2 and 3 of Algorithm 1) as follows.

Algorithm 2: ScanData(D_n, ε)

Input: Dataset, D_n, minimum support, ε
Output: Frequent length-1 items, F_n

1 Create HashMap h_n
2 Create set F_n
3 **for *each*** *transaction* $T \in D_n$ **do**
4 **for *each*** *length-1 item* $a_y \in T$ **do**
5 **if** $a_y \notin h_n$ **then**
6 Create set $cov_{D_n}(a_y) = \{TID\}$
7 Add $(a_y, cov_{D_n}(a_y))$ to h_n
8 **else**
9 Let $(a_y, cov_{D_n}(a_y)) = h_n(a_y)$
10 $cov_{D_n}(a_y) = cov_{D_n}(a_y) + TID$
11 Update h_n with $(a_y, cov_{D_n}(a_y))$
12 **for *each*** *item* $a_y \in h_n$ **do**
13 Let $(a_y, cov_{D_n}(a_y)) = h_n(a_y)$
14 **if** $sup_{D_n}(a_y) \geq \varepsilon$ **then**
15 Add $(a_y, cov_{D_n}(a_y))$ to F_n
16 **return** F_n

For any dataset D_n, as shown in Lines 1 and 2 of Algorithm 2, a hashmap h_n, and the set F_n respectively, are created. From Lines 3 to 11 of Algorithm 2, for each item a_y in each transaction T of D_n, if a_y is not contained in h_n, its coverset $cov_{D_n}(a_y)$ is created and the TID of T added to $cov_{D_n}(a_y)$ in Line 6. The tuple $(a_y, cov_{D_n}(a_y))$ is then added to h_n in Line 7 of Algorithm 2. Else, if a_y is already contained in h_n, $(a_y, cov_{D_n}(a_y))$ is obtained from h_n in Line 9 as $h_n(a_y)$ and the TID of T added to $cov_{D_n}(a_y)$ in Line 10. h_n is then updated with $(a_y, cov_{D_n}(a_y))$ in Line 11.

After all items and their coversets in D_n are added to h_n, the set of frequent length-1 items in D_n are obtained from h_n from Lines 12 to 15 as follows. For each item a_y in h_n, $(a_y, cov_{D_n}(a_y))$ is obtained from h_n in Line 13 as $h_n(a_y)$. As shown in Line 14, if a_y is frequent (that is, $sup_{D_n}(a_y) \geq \varepsilon$), the tuple $(a_y, cov_{D_n}(a_y))$ is added to F_n in Line 15 of Algorithm 2. The set F_n, which contains all frequent length-1 items in D_n and their coversets is then returned in Line 16. For the two datasets, $\mathbf{D_i}$ and $\mathbf{D_{i+1}}$, ScanData(D_i, ε) and ScanData(D_{i+1}, ε) in Lines 2 and 3 of Algorithm 1 will return F_i and F_{i+1} respectively. Figure 1 illustrates the outcome of this process, that is, F_1 and F_2 from toy datasets $\mathbf{D_1}$ and $\mathbf{D_2}$ at $\varepsilon = 0.1$. The set of common frequent length-1 items in $\mathbf{D_i}$ and $\mathbf{D_{i+1}}$ are then obtained from F_i and F_{i+1}.

D_1

TID	Transaction
1	{a, e, f}
2	{c, e, f}
3	{b, c, d, e, f}
4	{c, d, f}
5	{b, e, f}
6	{a, b}
7	{b, c, d, e}
8	{c, d, e, f}
9	{b}

D_2

TID	Transaction
1	{b, e}
2	{b, c, d, e}
3	{b, c, d, f}
4	{b, c, d}
5	{b, e}
6	{b, c}
7	{c, e, f}
8	{b, c, d}
9	{c, d, e}
10	{e, f}

F_1

Pattern	cov_{D_1}
{e}	{1,2,3,5,7,8}
{b}	{3,5,6,7,9}
{f}	{1,2,3,4,5,8}
{d}	{3,4,7,8}
{a}	{1,6}
{c}	{2,3,4,7,8}

F_2

Pattern	cov_{D_2}
{f}	{3,7,10}
{d}	{2,3,4,8,9}
{c}	{2,3,4,6,7,8,9}
{e}	{1,2,5,7,9,10}
{b}	{1,2,3,4,5,6,8}

Figure 1: Set of Frequent Length-1 Items, F_1 and F_2 in $\mathbf{D_1}$ and $\mathbf{D_2}$ at $\varepsilon = 0.1$

5.1.2 Identifying Common Length-1 Frequent Items

This step (from Lines 4 to 10 of Algorithm 1) finds the set of common length-1 frequent items in $\mathbf{D_i}$ and $\mathbf{D_{i+1}}$ as follows. As shown in Line 4 of Algorithm 1, the set L to store the common length-1 frequent items and their coversets in $\mathbf{D_i}$ and $\mathbf{D_{i+1}}$ is created. From Lines 5 to 10 of Algorithm 1, the common frequent length-1 items are identified as follows. For each frequent length-1 item, a_y in F_i, if a_y is also in F_{i+1} (that is, frequent in $\mathbf{D_{i+1}}$), a_y and its coversets, $cov_{D_i}(a_y)$ and $cov_{D_{i+1}}(a_y)$, are obtained in Lines 7 and 8 as $F_i(a_y)$ and $F_{i+1}(a_y)$ respectively. The tuple $(a_y, cov_{D_i}(a_y), cov_{D_{i+1}}(a_y))$ is then added to L in Line 9. The set L is then sorted in item descending order in Line 10. The set of productive emerging patterns are then mined from L in Line 11 of Algorithm 1 by calling MineEPs(L, ε). For our running example, Figure 2 illustrates the sorted L obtained from F_1 and F_2 in Figure 1.

L

Pattern	cov_{D_1}	cov_{D_2}
{b}	{3,5,6,7,9}	{1,2,3,4,5,6,8}
{c}	{2,3,4,7,8}	{2,3,4,6,7,8,9}
{d}	{3,4,7,8}	{2,3,4,8,9}
{e}	{1,2,3,5,7,8}	{1,2,5,7,9,10}
{f}	{1,2,3,4,5,8}	{3,7,10}

Figure 2: Sorted L, The Set of Common Frequent Length-1 Items from F_1 and F_2 in Figure 1

5.1.3 Mining Productive Emerging Patterns

This step mines all productive emerging patterns from L by calling MineEPs(L, ε) (Algorithm 3) in Line 11 of Algorithm 1. Algorithm 3 mines the set of productive emerging patterns from L as follows. In Line 3 of Algorithm 3, if there are no items in L, that is $|L| = 0$, the productive emerging pattern mining terminates and the set pE_i^{i+1} returned in Line 4. Else while $|L| > 0$, the productive emerging patterns are mined from L in the nested for-loop (from Lines 6 to 23 of Algorithm 3) as follows.

In the first for-loop within L (from index $k = 0$ to $|L| - 1$), the tuple $(a_k, cov_{D_i}(a_k), cov_{D_{i+1}}(a_k))$ at the k^{th}-index is obtained in Line 8 as $L[k]$. If a_k is a length-1 item, $GR(a_k)$ is evaluated in Line 10. The tuple $(a_k, GR(a_k))$ is added to pE_i^{i+1} in Line 12 if a_k is emerging, that is, $GR(a_k) > 1.0$. While still at the k^{th}-index, the second for-loop within L (from index $l = (k + 1)$ to $|L| - 1$) starts in Line 13 as follows. Each tuple $(a_l, cov_{D_i}(a_l), cov_{D_{i+1}}(a_l))$ in the l^{th}-index is obtained in Line 14 as $L[l]$. In Line 15, if a_k and a_l have common length-$(|a_k| - 1)$ prefixes, that is, $P_{a_k}[0, |a_k|\text{-}1] = P_{a_l}[0, |a_l|\text{-}1]$, a candidate frequent pattern, S, is created in Line 16 as $S = (a_k \cup a_l, cov_{D_i}(a_k) \cap cov_{D_i}(a_l), cov_{D_{i+1}}(a_k) \cap cov_{D_{i+1}}(a_l))$.

If S is frequent and productive in both D_i and D_{i+1}, it is added to TempL in Line 18. This ensures only frequent and productive patterns are kept as they both follow the anti-monotone property. $GR(S)$ is evaluated in Line 19 and S added to pE_i^{i+1} in Line 21 if S is emerging, that is, $GR(S) > 1.0$. For each k^{th}-index in the first for-loop, the second for-loop repeats till all indexes in L are iterated in the second for-loop. When both nested for-loops are complete, L is recreated in Line 22 from TempL and the content of TempL cleared in Line 23. The size of L is checked and the nested looping repeats until $|L| = 0$.

Stage I

L during first nested looping

Pattern	cov_{D_1}	cov_{D_2}
{b}	{3,5,6,7,9}	{1,2,3,4,5,6,8}
{c}	{2,3,4,7,8}	{2,3,4,6,7,8,9}
{d}	{3,4,7,8}	{2,3,4,8,9}
{e}	{1,2,3,5,7,8}	{1,2,5,7,9,10}
{f}	{1,2,3,4,5,8}	{3,7,10}

EPs detected during first nested looping

Productive EPs
{b}
{c}
{d}
{c, d}

Stage II

L during second nested Looping

Pattern	cov_{D_1}	cov_{D_2}
{c, d}	{3,4,7,8}	{2,3,4,8,9}
{e, f}	{1,2,3,5,8}	{7,10}

No EPs detected during second nested looping

Stage III

L = {ϕ} after second nested looping. Productive EP mining process terminates as $|L| = 0$

Figure 3: Productive Emerging Pattern Mining from L (see Figure 2) at $\varepsilon = 0.1$

We illustrate the productive emerging pattern mining process in Figure 3 on L (see Figure 2) obtained from the toy transactional databases in Figure

Algorithm 3: MineEPs(L, ε)

Input: Set L, minimum support, ε.
Output: Productive EPs set, pE_i^{i+1}

```
1  Let P_{c_n}[0, b] be the the length-b prefix of c_n
2  Create set TempL = ∅
3  if |L| = 0 then
4      return pE_i^{i+1}
5  else
6      while |L| > 0 do
7          for k = 0 to |L| − 1 do
8              Let (a_k, cov_{D_i}(a_k), cov_{D_{i+1}}(a_k)) = L[k]
9              if |a_k| = 1 then
10                 Evaluate GR(a_k)
11                 if GR(a_k) > 1.0 then
12                     Add (a_k, GR(a_k)) to pE_i^{i+1}
13             for l = k + 1 to |L| − 1 do
14                 Let (a_l, cov_{D_i}(a_l), cov_{D_{i+1}}(a_l)) = L[l]
15                 if P_{a_k}[0, |a_k|-1] = P_{a_l}[0, |a_l|-1] then
16                     Create S = (a_k ∪ a_l, cov_{D_i}(a_k) ∩ cov_{D_i}(a_l), cov_{D_{i+1}}(a_k) ∩ cov_{D_{i+1}}(a_l))
17                 if S is frequent and productive in both D_i and D_{i+1} then
18                     Add S to TempL
19                     Evaluate GR(S)
20                     if GR(S) > 1.0 then
21                         Add (S, GR(S)) to pE_i^{i+1}
22         L =TempL
23         TempL.clear()
```

1 at $\varepsilon = 0.1$. As seen in Figure 3, three stages (I, II, and III) are involved in mining the productive emerging patterns from L. We discuss the processes at each stage as follows.

1. **Stage I:** This stage shows L before the first nested looping and the detected productive emerging patterns during the first nested looping. During this first nested looping within L, length-1 frequent patterns $\{b\}, \{c\}$ and $\{d\}$ are added to pE_i^{i+1} in Line 12 of Algorithm 3 since they are all emerging. Productive frequent pattern $\{c, d\}$ is also added to pE_i^{i+1} in Line 21 as it is emerging. Though patterns $\{b, c\}, \{b, d\}, \{b, e\}, \{b, f\}, \{c, e\}, \{c, f\}, \{d, e\}$ and $\{d, f\}$ generated in Line 16 during the first nested looping are frequent, they are all pruned in Line 17 for the following reasons:
   - $\{b, e\}, \{b, f\}$ and $\{d, e\}$ are non-productive in both $\mathbf{D_1}$ and $\mathbf{D_2}$.
   - $\{b, c\}$ and $\{b, d\}$ are non-productive in only $\mathbf{D_1}$.
   - $\{c, e\}, \{c, f\}$ and $\{d, f\}$ are non-productive in only $\mathbf{D_2}$.
2. **Stage II:** This stage shows L recreated from TempL after the complete first nested looping. The second nested looping repeats on L in Stage II. No productive emerging patterns are detected in this stage as no candidate length-3 pattern can be formed from $\{c, d\}$ and $\{e, f\}$ since they do not have same common prefixes.
3. **Stage III:** This stage shows L recreated from TempL after the complete second nested looping. L in this stage has no items since no length-3 patterns were generated in Stage II. The productive emerging pattern mining process terminates in this stage since $|L| = 0$.

Patterns $\{b\}, \{c\}, \{d\}$ and $\{c, d\}$ are thus reported in Line 12 of Algorithm 1 as the set of productive emerging patterns detected from $\mathbf{D_1}$ and $\mathbf{D_2}$ at $\varepsilon = 0.1$.

5.2 Employing Detected Productive Emerging Patterns in Trend Prediction

Though most categories of emerging patterns are mined for classification purposes, in this section, we investigate the possible application of our detected emerging patterns in trend prediction.

Since the supports of an emerging pattern with time can be likened to a stochastic process, we cannot directly employ linear regression in modelling and predicting the emergence of an emerging pattern. As a preliminary step towards trend prediction with emerging patterns, we employ intuition in predicting the continuous emergence and future supports of S.

To predict trends based on emerging patterns, for any consecutive datasets $D_i, D_{i+1} \ldots D_n$ with time, for instance, consecutive; daily, monthly or yearly customer purchases. We take any three consecutive datasets, for example, D_i, D_{i+1} and D_{i+2} where,

1. D_i and D_{i+1} are used as the training set. For a given minimum support (ε), we mine the set of productive emerging patterns from D_i to D_{i+1} and productive "decaying patterns" (DPs) from D_i to D_{i+1}. Our decaying patterns (DPs), from D_i to D_{i+1}, are often referred to as emerging patterns, from D_{i+1} to D_i in previous works mining emerging patterns for classification purposes.

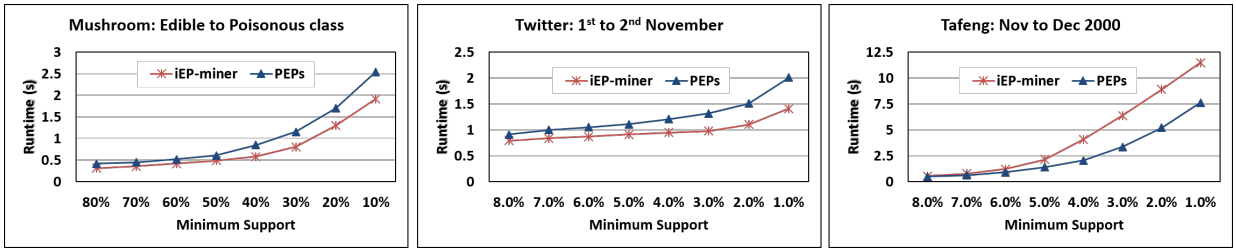


Figure 4: Emerging Pattern Detection Runtime ($\rho = 1.0, \eta = 3.841$)

Table 1: Emerging Patterns in Trend Prediction

Twitter Dataset: Trend Prediction given $\rho = 1.0, \eta = 3.841$ at $\varepsilon = 0.01$						
Days in Nov	Approach	Total EPs	Total DPs	Precision	Recall	F1-mesure
$1^{st}, 2^{nd}$ and 3^{rd}	iEP-miner	18	10	64.29	40.91	50.00
	PEPs	29	10	61.54	54.55	**57.83**
$2^{nd}, 3^{rd}$ and 4^{th}	iEP-miner	17	12	86.21	54.35	66.67
	PEPs	25	19	90.91	86.96	**88.89**
$3^{rd}, 4^{th}$ and 5^{th}	iEP-miner	29	7	33.33	26.67	**29.63**
	PEPs	39	7	26.09	26.67	26.37
$4^{th}, 5^{th}$ and 6^{th}	iEP-miner	9	21	73.33	56.41	63.77
	PEPs	9	36	82.22	94.87	**88.10**
$5^{th}, 6^{th}$ and 7^{th}	iEP-miner	4	26	96.67	72.50	82.86
	PEPs	4	35	97.44	95.00	**96.20**
$6^{th}, 7^{th}$ and 8^{th}	iEP-miner	21	10	87.10	65.85	75.00
	PEPs	30	10	90.00	87.80	**88.89**
$7^{th}, 8^{th}$ and 9^{th}	iEP-miner	27	1	39.29	22.00	28.21
	PEPs	45	1	34.78	32.00	**33.33**
$8^{th}, 9^{th}$ and 10^{th}	iEP-miner	10	21	87.10	56.25	68.35
	PEPs	15	35	92.00	95.83	**93.88**
Tafeng Retail Dataset: Trend Prediction given $\rho = 1.0, \eta = 3.841$ at $\varepsilon = 0.01$						
Months	Approach	Total EPs	Total DPs	Precision	Recall	F1-mesure
Nov, Dec and Jan	iEP-miner	55	79	80.60	48.43	60.50
	PEPs	76	151	82.82	84.30	**83.56**
Dec, Jan and Feb	iEP-miner	82	52	51.49	35.94	42.33
	PEPs	144	79	45.74	53.13	**49.16**

2. D_{i+1} and D_{i+2} are used as our test set. For the same given minimum support (ε), we mine the set of productive emerging patterns from D_{i+1} to D_{i+2} and productive decaying patterns from D_{i+1} to D_{i+2}.
3. For a detected productive emerging pattern, S_1 from the training set, we predict its presence in the test set as a productive emerging pattern, that is, $sup_{D_{i+2}}(S_1) > sup_{D_{i+1}}(S_1)$.
4. For a detected productive decaying pattern, S_2 from the training set, we predict its presence in the test set as a productive decaying, that is, $sup_{D_{i+2}}(S_2) \leq sup_{D_{i+1}}(S_2)$, or being infrequent in D_{i+2}, that is, $sup_{D_{i+2}}(S_2) < \varepsilon$.

6 Empirical Analysis

For our experimental analysis, the following implementations are compared.

1. PEPs: This is an implementation of our proposed productive emerging pattern detection framework. For any two given datasets, PEPs detects and reports all frequent and productive emerging and decaying patterns.
2. iEP-miner: This is our implementation of the method proposed in (Fan and Ramamohanarao 2003). For any two given datasets, iEP-miner detects and reports all interesting emerging and decaying patterns.

We compared the performance of PEPs and iEP-miner on: (i.) runtime and (ii.) trend prediction effectiveness with detected EPs. All methods are implemented in Java and experiments carried on a 64-bit Windows 7 PC (Intel Core i5, CPU 2.50GHz, 4GB Memory). The following datasets were used in our experimental analysis:

1. Mushroom datasets: We obtained this dataset from `http://cgi.csc.liv.ac.uk/~frans/KDD/Software/LUCS-KDD-DN/DataSets`.
2. Twitter Dataset: This dataset consists of daily hashtags of tweets for the month of November 2012. We obtained this data from CNetS (`http://carl.cs.indiana.edu/data/`).
3. Tafeng Retail Dataset: This dataset, obtained from AIIA Lab (`http://aiia.iis.sinica.edu.tw`) comprises of four months of customer transactions from TaFeng Warehouse. That is

customers transactions for the months of November and December 2000, and that of January and February 2001.

6.1 Runtime

Figure 4 shows the runtime of PEPs and iEP-miner. Though PEPs reports higher number of emerging patterns, its performance is comparable to that of iEP-miner which detects fewer number of emerging patterns. As shown in Figure 4, iEP-miner slightly outperforms PEPs at low minimum supports in the mushroom and Twitter dataset. This is because at low minimum supports, more emerging patterns which do not satisfy Conditions 3 and 4 of Definition 2 are pruned. Most of these pruned emerging patterns in iEP-miner are however productive, hence the slight out-performance. However, as can be seen in Figure 4 on the Tafeng retail dataset, PEPs slightly outperforms iEP-miner in the emerging pattern discovery process.

6.2 Decision Making

Table 1 shows a preliminary application of emerging patterns in trend prediction based on our intuition prediction approach described in Section 5.2. We employed the $F1$-measure as the overall goodness measure and evaluate our precision and recall as:

$$Prec = \frac{\#EPs + \#DPs\ correctly\ predicted}{\#EPs + \#DPs\ in\ category} \quad (4)$$

$$Recall = \frac{\#EPs + \#DPs\ correctly\ predicted}{\#EPs + \#DPs\ in\ test\ set} \quad (5)$$

As can be seen in Table 1, productive emerging patterns turn out as the best set for trend prediction as they have higher F1-scores compared to same predictions based on interesting emerging patterns (proposed by (Fan and Ramamohanarao 2003)).

7 Conclusions and Future Works

Productive emerging patterns are emerging patterns whose emergence from one dataset to another are due to inherent item relationships and not due to random occurrence of items. Non-productive emerging patterns, the set of emerging patterns whose emergence are due to random occurrence of items will be detrimental in decision making where inherent relationships between items of emerging patterns are relevant. We make use of a correlation test and introduce the productive emerging pattern set as the set of emerging patterns whose emergence are due to inherent item relationships. We develop PEPs, a productive emerging pattern mining framework and show a potential application of emerging patterns in trend prediction. Our experimental results show that PEPs is efficient, and the productive emerging pattern set which achieves a size reduction in the number of reported emerging patterns shows potential in trend prediction. Our future works are in two areas: i.) trend prediction, which will involve forming a more technical trend prediction model based on our detected productive emerging patterns, and, ii.) classification, where we tend to investigate on the effectiveness of our productive emerging patterns are in classifier formation.

References

Agrawal, R. and Srikant,R. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, 1995.

Cheng, M. W. K., Choi, B. K. K., and Cheung, W. K. W. Hiding emerging patterns with local recoding generalization. In Zaki, M. J., Yu, J. X., Ravindran, B., and Pudi, V., editors, *Advances in Knowledge Discovery and Data Mining*, volume 6118 of *LNCS*, pages 158–170. Springer Berlin Heidelberg, 2010.

Dong, G. and Li, J. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 43–52, New York, USA, 1999. ACM.

Dong, G. and Li, J. Mining border descriptions of emerging patterns from dataset pairs. *Knowledge and Information Systems*, 8(2):178–202, 2005.

Fan, H. and Ramamohanarao, K. An efficient single-scan algorithm for mining essential jumping emerging patterns for classification. In Chen, M. S., Yu, P. S., and Liu, B., editors, *Advances in Knowledge Discovery and Data Mining*, volume 2336 of *LNCS*, pages 456–462. Springer Berlin Heidelberg, 2002.

Fan, H. and Ramamohanarao, K. Efficiently mining interesting emerging patterns. In Dong, G., Tang, C., and Wang, W., editors, *Advances in Web-Age Information Management*, volume 2762 of *LNCS*, pages 189–201. Springer Berlin Heidelberg, 2003.

Fan, H. and Ramamohanarao, K. Fast discovery and the generalization of strong jumping emerging patterns for building compact and accurate classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 18(6):721–737, 2006.

Garcìa-Borroto, M., Martìnez-Trinidad, J.F., and Carrasco-Ochoa, J. A. A survey of emerging patterns for supervised classification. *Artificial Intelligence Review*, 42(4):705–721, 2014.

Li, J. and Wong, L. Emerging patterns and gene expression data. *Genome Informatics*, 12:3–13, 2001.

Li, J., Dong, G., and Ramamohanarao, K. Making use of the most expressive jumping emerging patterns for classification. *Knowledge and Information Systems*, 3(2):1–29, May 2001.

Li, J., Liu, H., Downing, J. R., Yeoh, A. E. J., and Wong, L. Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (all) patients. *Bioinformatics*, 19(1):71–78, 2003.

Li, J., Dong, G., Ramamohanarao, K., and Wong, L. Deeps: A new instance-based lazy discovery and classification system. *Machine Learning*, 54(2):99–124, 2004.

Nofong, V. M., Liu, J. and Li, J. A study on the applications of emerging sequential patterns. In Wang, H. and Sharaf, M. A., editors, *Databases Theory and Applications*, volume 8506 of *LNCS*, pages 62–73. Springer International Publishing, 2014.

Poezevara, G., Cuissart, B., and Crèmilleux, B. Extracting and summarizing the frequent emerging graph patterns from a dataset of graphs. *Journal of Intelligent Information Systems*, 37(3):333–353, 2011.

Soulet, A., Crèmilleux, B. and Rioult, F. Condensed representation of emerging patterns. In Dai, H., Srikant, R., and Zhang, C., editors, *Advances in Knowledge Discovery and Data Mining*, volume 3056 of *LNCS*, pages 127–132. Springer Berlin Heidelberg, 2004.

Terlecki, P. and Walczak, K. Jumping emerging patterns with negation in transaction databases classification and discovery. *Information Sciences*, 177 (24):5675 – 5690, 2007.

Tsai, C.Y. and Shieh, Y. C. A change detection method for sequential patterns. *Decision Support Systems*, 46(2):501 – 511, 2009. ISSN 0167-9236.

Webb. G. I. Self-sufficient itemsets: An approach to screening potentially interesting associations between items. *ACM Transactions on Knowledge Discovery from Data*, 4(1):3:1–3:20, January 2010.

Aspect-Based Opinion Mining from Product Reviews Using Conditional Random Fields

Amani K. Samha, Yuefeng Li, Jinglan Zhang
Science and Engineering Faculty, Queensland University of technology
Brisbane 4000, Queensland, Australia
asamha@gmail.com, {y2.li, Jinglan.zhang} @qut.edu.au

Abstract

Product reviews are the foremost source of information for customers and manufacturers to help them make appropriate purchasing and production decisions. Natural language data is typically very sparse; the most common words are those that do not carry a lot of semantic content, and occurrences of any particular content-bearing word are rare, while co-occurrences of these words are rarer. Mining product aspects, along with corresponding opinions, is essential for Aspect-Based Opinion Mining (ABOM) as a result of the e-commerce revolution. Therefore, the need for automatic mining of reviews has reached a peak. In this work, we deal with ABOM as sequence labelling problem and propose a supervised extraction method to identify product aspects and corresponding opinions. We use Conditional Random Fields (CRFs) to solve the extraction problem and propose a feature function to enhance accuracy. The proposed method is evaluated using two different datasets. We also evaluate the effectiveness of feature function and the optimisation through multiple experiments.

Keywords: Opinion Mining, Customer reviews, Product reviews, Conditional random fields, Feature Function.

1 Introduction

The growth of world-wide web platforms such as social media, forums, blogs and product reviews has led people to post their opinions and benefit from others' past experiences. User-generated reviews have become an exciting reference in most fields, such as business, education and e-commerce, as they contains opinionated information about services and products (Moghaddam, Jamali and Ester 2011). Analysing such information enhances the decision-making process when selling, buying and providing services. In the business world, for example, reviews help to improve the way that services or products are offered and eliminate customer dissatisfaction. Obtaining such information will guarantee that feedback is delivered to the manufacturer or service provider. For potential customers, it creates awareness from others' past experiences and thus enhances the decision-making process. The ability to post reviews is provided by many e-commerce websites, such as Amazon, Yahoo Shopping and eBay, among others, and allows customers to post their opinions freely. While this seems straightforward, the process becomes complicated when there are large numbers of reviews. Therefore, the enormous number of online opinionated customer reviews creates the need for systems to gather important information, analyse it, and extract useful knowledge to ensure that end-users can benefit with minimal effort. Opinion mining is classified into three branches: the document level, which aims to provide an overall opinion; the sentence level, which produces opinions based on the sentence; and the feature level, which examines each feature in the review. This is known as ABOM (Himmat and Salim 2014; Liu 2012; Liu and Zhang 2012).

ABOM, which is the base case study of this work, involves several tasks. First, it aims to efficiently identify and extract product entities, which include the actual product, its components, functionality, attributes and the aspects of the product (Ding, Liu and Zhang 2009). The next task is to find the corresponding opinions for each entity extracted from relevant reviews. Opinions are also known as 'sentiments', which are the adjectives that are given by users to describe the product. A number of researchers have attempted to solve the opinion mining problem using different approaches via supervised, unsupervised and semi-supervised learning. These include rule-based methods (Guo et al. 2009; Hu and Liu 2004a, 2004b; Liu, Hu and Cheng 2005; Moghaddam and Ester 2010), statistical methods (Guo et al. 2009; Wang, Lu and Zhai 2010; Choi and Cardie 2010; Titov and McDonald 2008) and lexicon approaches(Zhao and Li 2009; Noy 2004; Zhang et al. 2011; Taboada et al. 2011; Wogenstein et al. 2013).

In this paper, we study the problem of ABOM as a sequence labelling problem, and propose a computational technique to model ABOM of product reviews. Recent research has shown that the sequence labelling approaches based on conditional relations enhance the accuracy and performance of unstructured prediction problems. There are some proposed models for sequence labelling tasks, such as CRF (Lafferty, McCallum and Pereira 2001), Hidden Markov Models (HMM) (Eddy 1996) and Max-Margin Markov Networks (Roller 2004), among others. These models have shown enormous improvement and considerable success in certain practical tasks, such as natural language processing, pattern recognition and information extraction. We employ a supervised learning approach using the CRF model to identify and extract aspects, as well as extract and map

opinions as a sequence labelling problem. CRF is a class of statistical modelling methods often applied in pattern recognition and machine learning that is used for structured prediction. This is particularly important for opinion mining of product reviews. We propose techniques for selecting the best features for the proposed CRF model and optimising its accuracy.

The goal of our work includes identifying product entities and mapping them to the corresponding opinions along with their orientation as a subjective ABOM pattern, which is represented as the form of a single word or multi-word expressions. CRF is used to encode known relationships between reviewers' opinions and construct consistent interpretations of the reviews. With this approach, CRF predicts the sequence of labels for a given input sequence. Here, the reviews were considered as input sequences and POS tags and opinion tags were used as output labels. The Center for SprogTeknologi (CST) online tagger was used for performing POS tagging and opinion tagging was done manually. To tune and evaluate the CRF model, we trained and tested the model with an annotated dataset, obtained from Hu and Liu (2004a) and Marcińczuk and Janicki (2012). The essentials tasks of POS tagging and opinion tagging are described below.

Unlike other systems that consider a single feature of the entity, in which previous work considers nouns and/or noun phrases to be product aspects, our method attempts to find the best combination of features that makes the word eligible to be a product aspect. These features include, but are not limited to tokenisation, part of speech tagging, chunking, word distance features and position features. Our experimental results confirm the effectiveness and accuracy of the proposed solution. The major contributions of this paper include:

1. Hand annotation of the dataset.
2. Proposing statistical frameworks to automatically find the ABOM pattern by considering linguistics to expand the list of words are that are likely to be product aspects, then mapping the relationships to corresponding opinions, without considering domain knowledge and based only on strict matches.
3. Extracting all possible aspects and opinions and improving the accuracy of aspect and opinion extractions by proposing a technique to select the best feature functions considering three inputs to the CRF model (Labels | Words, POS tagging, Chunking) = $(T| W, P , C)$.
4. Identifying and mapping the relationships and boundaries between product aspects and opinions by combining basic linguistic features and n-grams, where all the comparison were made based on strict matches only.

The rest of the paper is organized as follows: *section 2* explains the related work; *section 3* stated the problem of ABOM, *section 4* describes the design, train and test of the CRF model. *Section 5* is the feature function of the CRF model. *Section 6* is the experiment and error analysis. Finally, *section 7* includes discussion and future work.

2 Related work

As mentioned above, and according to Pang and Lee (2008), the opinion mining task can be classified as follows, based on the extraction task: word/phrase level, sentence level (Wiebe et al. 2004; Moghaddam and Ester 2011; Hu and Liu 2004a) or document level (Turney 2002; Pang, Lee and Vaithyanathan 2002; Yu et al. 2008; Lim et al. 2010; Liu, Hu and Cheng 2005). Many studies on opinion mining have been conducted at the document level, which aims to find the orientation of the review rather the precise likes and dislikes reported. Turney (2002) used point-wise mutual information to calculate the average semantic orientation of the extracted phrases to determine the polarity of the whole document. (Hatzivassiloglou 2000) proposed a statistical supervised method that works by combining dynamic adjectives, semantic oriented adjectives and gradable adjectives as a simple subjective classifier. (Pang and Lee 2002) studied the effectiveness of sentiment classification using machine learning techniques with movie review data. As a result of the general orientation of the whole review, the mining process missed the detail of what likes and dislikes the review contained. To address this problem, more research was conducted at the sentence and phrase levels. The concept of mining aspects and corresponding opinions was first addressed by Hu and Liu (2004) using information extraction techniques and based on aspect frequency. These approaches were useful when associating aspect extraction with the fact that aspects are most commonly nouns. However, such models highlight the limitations of not extracting infrequent aspects and also by the fact that some extracted nouns are not aspects. Proprdue and Et (2005) improved the Hu and Liu's system by developing a system to remove frequent nouns that are not aspects, such that it achieved high precision but low recall; however, this failed to solve the problem of infrequent aspect extraction.

In general, ABOM (Samha, Li and Zhang 2014) comes under phrase-level opinion mining, and aims to produce a detailed sentiment analysis at the aspect level. Vivekanandan and Aravindan (2014) categorized the ABOM approaches into three groups: first, the frequency-based approaches, which are based on frequent aspects of products. These assume that the frequent aspects are more important than non-frequent aspects (Hu and Liu 2004a; Baccianella, Esuli and Sebastiani 2009; Zhuang et al. 2006). Second, the relational-based approaches map relations between aspects and opinions and assume that the closest are more likely to be accurate (Zhuang et al. 2006; Hu and Liu 2004a, 2004b). Finally, the model-based approaches aim to overcome the limitations of the other approaches. Some of the commonly used supervised learning techniques are HMM (Abbasi Moghaddam 2013) and CRF (Qi and Chen 2010; Huang et al. 2012; Jakob and Gurevych 2010; Xu et al. 2010). In this paper we have used CRF and attempted to overcome some of the limitations of other models.

Jin, Ho and Srihari (2009a, 2009b) have considered opinion mining as a sequence labelling problem built under HMM (lexicon-based) using linguistic features. HMM models assume that each feature is generated independently and ignore the underlying relationships

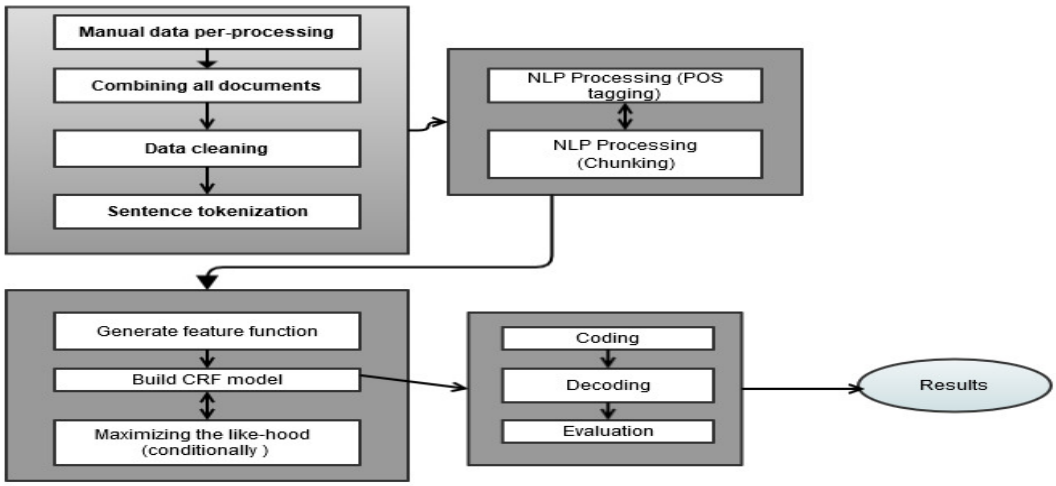


Figure 1: Architecture of the proposed model

between the actual words and labels, as well as the overlapping features (Qi and Chen 2010). CRF overcomes these limitations because it is a discriminative model that models the overlapping dependent features (Peng and McCallum 2006). Choi et al. (2005) view sentiment analysis as a hybrid task information extraction problem that combines CRF as a sequence tagging task and AutoSlog (Riloff 1996) to learn the extraction patterns. Even though their system employs extraction learning with CRF, it showed a recall of 54% with exact match. CRF has been implemented in different languages, linguistic features (Xu et al. 2010) where the strict match performance was around 50%. Here, we have developed a CRF model that can address this problem and extract frequent and infrequent product aspects, along with their corresponding orientations.

3 Aspect-Based Opinion Mining

3.1 Problem Statement

Let $D = \{d_1, d_2, \ldots, d_n\}$ be a set of opinionated documents, where each d_i consists of a set of reviews $R = \{r_1, r_2, \ldots, r_n\}$. Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of sentences, where each s_i consists of words $W = \{w_1, w_2, \ldots, w_n\}$, the corresponding part of speech tags s $P = \{p_1, p_2, \ldots, p_n\}$, and the corresponding chunking phrases $C = \{c_1, c_2, \ldots, c_n\}$.

3.2 Problem definition

Given a sequence of words, $W = \{w_1, w_2, \ldots, w_n\}$, with the corresponding part of speech tags $P = \{p_1, p_2, \ldots, p_n\}$ and the corresponding chunking phrases for each word $C = \{c_1, c_2, \ldots, c_n\}$. The ABOM task can be defined as a sequence labelling problem. We employ CRF to find the most likely sequence of labels $T = \{t_1, t_2, \ldots, t_n\}$

3.3 The big picture

Figure 1 illustrated the architecture of the whole model and Figure 2 shows an overview picture of the whole model. We started with labelling of the dataset using the tags listed in Table 2. However, data first needs to be pre-processed and cleaned. So all abnormal characters are removed from the text using regular expressions. Then we used the OpenNLP (Baldridge 2005) to detect and split sentences.

After manually labelling the cleaned data, we prepare the dataset for build a CRF model. We use OpenNLP to do the POS tagging and chunking for all words to satisfy the input equation (T|W, POS, Chunking). After that, we train the CRF model with the feature function. Finally, we tested the CRF model and generate results.

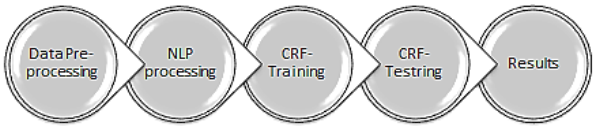


Figure 2: System big picture

3.4 Data Set preparation

3.4.1 Entity Definition

The focus is to define and extract product entities and corresponding opinions then label the training dataset using tags. According to Banitaan et al. (2010) and Glance, Hurst and Tomokiyo (2004) there are different categories of entities (Table 1). However, the broad overview categorises them into four entity groups that represent different types of words in the review text. These four categories are components, functions, features and opinions. As an example, (Table 1) includes an example of entity categories related to the word *'camera'* (Glance, Hurst and Tomokiyo 2004). Some entities may not fit in any categories. Therefore, we can form a fifth category, called *'other'*, and leave it open for any suggested categories that not belong to any of these four entity category.

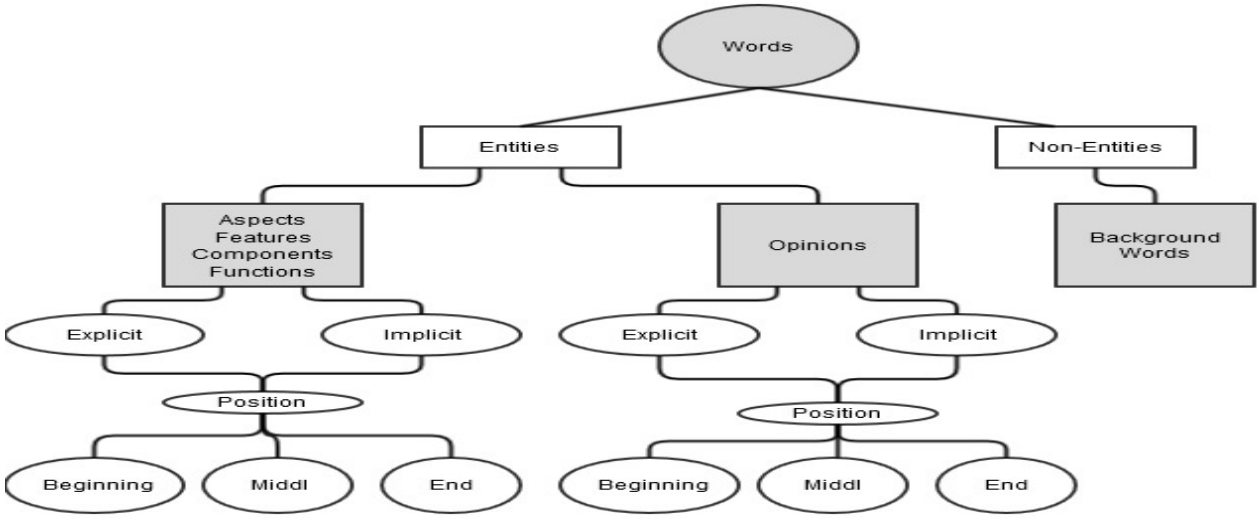


Figure 3: Dataset tagging process

Entity	Description
Components	Physical objects of a camera, including the camera itself, the LCD, viewfinder and battery
Functions	Capabilities provided by a camera, including movie playback, zoom and autofocus
Features	Properties of components or functions, such as colour, speed, size, weight, and clarity
Opinions	Ideas and thoughts expressed by reviewers on the product, its features, components or functions
Other	Other possible entities defined by the domain

Table 1: Entity categories

3.4.2 Pre-processing

Pre-processing is a necessary step, since the dataset is raw and must be prepared for training and then for testing. At this stage, all abnormal characters and HTML tags, such as *<b>, [ ], "",* are removed. Next, all sentences are combined into one single document, and then sentences were detected using OpneNLP tools (Baldridge 2005).

3.4.3 Dataset tagging process

According to the entity definition, this experiment defined five types of tags, where the tags are based on entities, defined in Figure 3, divided into two main categories. The first category is *'Features'*, and includes the product itself, its components, functions, features, attributes and the aspects of the product. Each category is based on its meaning, both explicit and implicit.

Then we used the most positional and represented labels following the Beginning-Middle-End (BME) labelling schema: *B-Target*, identifying the beginning of feature/opinion target; *M-Target*, identifying the middle position of the word, where it may have more than one middle tag. Finally is the *E-Target*, which represents the end position of the word in the sentence.

Tag	Labels	Examples
Background words	(B)	I(B) bought(B) this(B)
Explicit aspect or feature	(Feature_B) (Feature_M) (Feature_E)	to(Feature_B) use(Feature_E)
Implicit aspect or feature	(Feature_B_Imp) (Feature_M_Imp) (Feature_E_Imp)	affordable (Feature_B_Imp)
Positive and negative explicit opinions	(Opinion_B_P/N_Exp) (Opinion_M_P/N_Exp) (Opinion_E_P/N_Exp)	Inexpensive (Opinion_E_P_Exp)
Positive and negative implicit opinions	(Opinion_B_P/N_Imp) (Opinion_M_P/N_Imp) (Opinion_E_P/N_Imp)	real(Opinion_B_P_Imp) buy(Opinion_E_P_Imp)

Table 2: Tags

4 CRF model Design, Train and Test

Product features are mostly nouns or noun phrases; whereas opinions are adjectives or adjectival phrases that are most likely appear closer to the nouns. Natural language is usually a sequence of words that form sentences as a meaningful sequence based on grammatical rules. Therefore, the sequence is a sentence and a word is a primary element of it. There are enormous elements that we can assign to each individual word, such as parts of speech, chunking and more. Therefore, the problem of ABOM can be formulated as a sequence-labelling task. The solution to the sequence-labelling problem is based on natural language processing techniques, where we aim to assign a single label to each element in a sequence. First-order CRF (Lafferty, McCallum and Pereira 2001; McDonald and Pereira 2005; Sutton and McCallum 2006) considers the dependencies between at most three adjacent labels.

CRF was proposed by (Lafferty, McCallum and Pereira 2001). It is a probabilistic method for extracting and labelling sequential data that encode dependencies between different entities of a sequence, and typically outperforms other supervised learning algorithms, such as

support vector machine learning. It has demonstrated high performance in information extraction, particularly in entity recognition (Klinger and Friedrich 2009). CRFs are resolved according to undirected graphical models over sets of random variables. It is formally defines as follows: Let $G = (V, E)$, a considering undirected graph, let$Y = (Y_v)$, $v \in V$ where each node $\in V$ is corresponding to each of the random variables that $\in Y$, and (X, Y) is a CRF illustrated, in (Figure 4). X is a set of variables *'input'* over the observation sequence to be labelled and Y is a set of random variables *'output'* over the corresponding labelling to be predicted. In this paper, the CRF model works as an extraction model that computes the probability of $Y = (T)$, which represents the probability of the sequence of hidden labels to the sequence of input $X = (W, P, C)$, which represents the observed labels, that aims to find the most probable label sequence $Y's$, given an observation sequence in the problem of sequence label modelling. Therefore, we are looking to represent a distribution over a large number of random variables using only local functions requiring only a small number of variables.

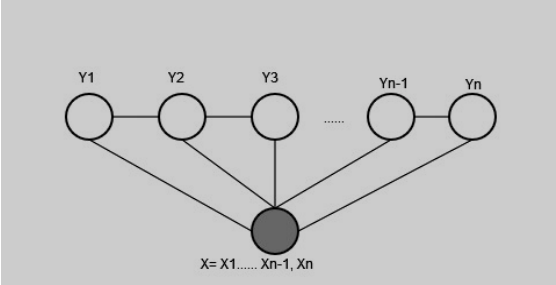


Figure 4: Linear CRF graphical structure

CRF defined as:

$$P(Y \mid X) = \frac{1}{Z(X)} \prod_{j=1}^{n} \psi_j (X, Y)$$

With a normalisation factor:
$Z(x) = \sum y \prod i\epsilon N \phi (y_i, x_i)$ and a feature suction is defined as: f_k as $\phi_i (y_i, x_i) = \exp(\sum_k \lambda_k f_k(y_i, x_i))$. In prediction, we output the most probable label that maximize the likehood of $\hat{Y} = \arg_y MAX P(Y|X)$.

The main task of this paper, is viewing ABOM as a sequential tagging problem which use a set of statistical and natural language features to train the liner-chine CRF. The relation between aspects and opinions are mapped by understanding the syntactic based on observations.

5 Feature Selection

Feature selection is a very important step in any information extraction system. Thus, modelling the perfect subset of features is significant to increase the performance of the ABOM model. In this paper, the extracted features are used to map the relationships between observation labels and hidden labels. Since the product aspects/features are mostly likely to be nouns or/and noun phrases and opinions are most likely to be adjectives or/and adjectival phrases, selecting features is based on natural language processing techniques and a probabilistic language model. These features are divided into two categories:

5.1 Basic features

Basic features are linguistic features. These features are extracted using the OpenNlp toolkit (Baldridge 2005) as follows:

- Token feature f1: This represents the string of the current token in which every word of the text is a token w_i. Tokenisation worked well in Zhang and Liu (2014) and Jakob and Gurevych (2010). In this paper, the token is the value of the actual word of the sentence. It values each token in the sentence by the natural word and the position of the word in the sentence indexed by relative position to the word.
- Part-of-speech tagging feature $f2$ and chunking features f3: are two syntactic features (Marcińczuk and Janicki 2012) that examine the phrase level in depth, considering the token and its surrounding words. f2 is used to classify each $w_n \in W$ into one of a set of tags, such as verbs, nouns or adjectives, while $f3$ is used to classify each $w_n \in W$ to the applicable chunk based on phrases. f2 and f3 are used to map the relationship between product aspects and opinions. Here, we used Part of Speech Tagger from the Open NLP toolkit (Baldridge 2005).
- Chunking feature f3 : text chunking is used to recognise the relatively simple syntactic structure of sentences. POS tagging shows the product aspects at a word level only; however, some product aspects are noun phrases, which are more likely to be nearest to the opinion words (Tjong Kim Sang and Buchholz 2000). For chunking we used the Chunker tools from the OpneNLP toolkit (Baldridge 2005) that was trained on conll2000 (Tjong Kim Sang and Buchholz 2000) shared task data.
- Sentence segmentation feature f4: this feature is used to segment each review into sentences. This feature helps to find the boundaries of the opinionated sentences.

5.2 Advanced features

Advanced features are the basic features mixed with certain statistical features to form rules, as follows:

N-grams features $f5$: since POS tagging and chunking map the synaptic structure of the sentence in a simple way, n-gram was added as a feature, as it performs well in sentiment classification (Pak and Paroubek 2010; Dave, Lawrence and Pennock 2003; Pang, Lee and Vaithyanathan 2002). From this point, we experimented with the best settings usage of unigrams, bigrams, and trigrams and combined them with the basic features, as shown in (Table3) .

- Context features $f6$ considers the token feature$f1$ to obtain contextual information, where the tokens near the target token may indicate its type and to which category it

Item sequences	Attributes	Description
1	w[t-2], w[t-1], w[t], w[t+1], w[t+2],	(5 features of trigram words)
2	w[t-1]|w[t], w[t]|w[t+1],	(2 features of bigram words)
3	pos[t-2], pos[t-1], pos [t], pos [t+1], pos [t+2],	(5 features of trigram POS tagging)
4	pos[t-2]|pos[t-1], pos[t-1]|pos[t], pos[t]|pos[t+1], pos[t+1]|pos[t+2],	(4 features of POS tagging relations (2-order))
5	pos[t-2]|pos[t-1]|pos[t], pos[t-1]|pos[t]|pos[t+1], pos[t]|pos[t+1]|pos[t+2]	(3 features of trigram POS tagging relations (3-order))
6	chunk[t-2], chunk [t-1], chunk [t], chunk [t+1], chunk [t+2],	(5 features of trigram chunk tags)
7	chunk [t-2]|chunk [t-1], chunk [t-1]|chunk [t], chunk [t]|chunk [t+1], chunk [t+1]|chunk [t+2],	(4 features of chunk tagging relations (2-order))
8	chunk [t-2]|chunk [t-1]|chunk [t], chunk[t-1]| chunk [t]|chunk [t+1], chunk [t]|chunk [t+1]| chunk [t+2]	(3 features of trigram chunk tagging relations (3-order))

Table 3: CRF advanced features

belongs. This works using f2 and f3 features as added features to the neighbouring words of different n-grams, where we study the surrounding words in combination with other features, such as n-grams, POS and chunking. Therefore, we formed rules based on observations using f5, as shown in Table 3.

- Position of the word featuref_7: we used tags applicable for the word's position in the sentence, for instance, *_B* 'beginning of sentence', *_M* 'middle of the sentence' and *_E* is 'end of sentence'.

The combination of both feature sets increased the accuracy of the CRF model. Some definitions are necessary to clarify the reading of the features:

- W is the word features: include word at position t-2, t-1, t, t+1, t+2: trigram of words.
- w[t-1]|w[t]: associations between words features: represents the concurrency of bigram of words.
- Corresponding with POS: part of speech tag and chunk: Chunker Tag.

6 Experimental Framework

CRFsuite (Okazaki 2007), a fast implementation of CRF (Lafferty, McCallum and Pereira 2001), was used to train our model. In the training phase, CRFsuite predicted some wrong labels; for instance, the product aspects that we were interested in might be a single word or consist of multi-word strings; however, we needed some scripts to help with the dataset. Therefore, we wrote few Python scripts that aim to align the CRFsuite output tags with the original input labelled file. We then evaluated the work by calculating precision, recall and F-score measures on the actual word, post-tagging and chunking recognition rather than individual words.

Additionally, we divided the actual and predicted aspects into four categories: correct (self-explanatory), missed (actual chunks not identified by the model), wrong label (word sequences that were correctly extracted but wrongly classified), and false positives (self-explanatory) to obtain a more detailed picture.

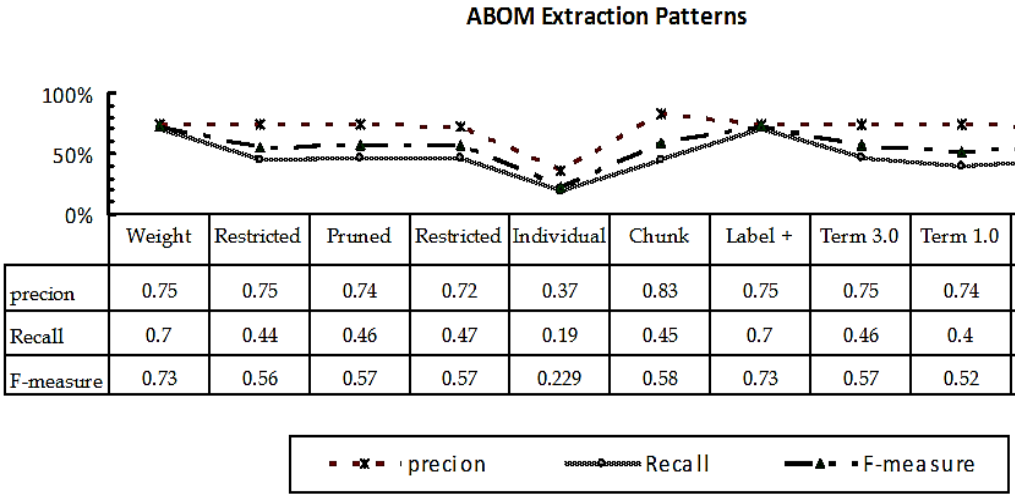
Performance	Individual label assignment	Chunk recognition	Label + POS+ chunk
Precision	0.37	0.83	0.75
Recall	0.19	0.45	0.7
F-measure	0.229	0.58	0.73
Correctly identified chunks	-	0.45	0.50
Missed chunks	-	0.53	0.49
Incorrectly labelled chunks	-	0.01	0.017
False positives	-	0.08	0.212

Table 4: Model label performance

We examined the model's performance by measuring the accuracy at every level of the experiment. We began by measuring the performance of the model using the labels alone, which we consider as the baseline, as shown in (Table 4), where it shows poor performance in general. Due to the limited matched examples in the dataset, rare tags did not occur often enough to generalise from them, especially for context-dependent features. However, the word itself is a suitable predictor of the label. On the other hand, if the model is too heavily trained on words, then it will not be able to make good predictions for words that it has never seen a common occurrence when dealing with natural language data.

We then added the chunking tags to the labels, which improved precision and recall. The performance was improved by using the actual word, POS tags and chunking. From this point, the actual word, the label, the POS tag and the chunking tag were used in model experiments along with several different feature sets.

CRFsuite allows the possibility of providing scaling values for each feature; this value is multiplied by the learned weight when predicting the value of a label, making it possible to adjust the importance of the feature to some degree. We modified the feature extraction script to allow scaling values to weight the value of the word-based features or part-of-speech based features more



	Weight	Restricted	Pruned	Restricted	Individual	Chunk	Label +	Term 3.0	Term 1.0	Term 3.0	Term 2.0
precion	0.75	0.75	0.74	0.72	0.37	0.83	0.75	0.75	0.74	0.73	0.71
Recall	0.7	0.44	0.46	0.47	0.19	0.45	0.7	0.46	0.4	0.45	0.44
F-measure	0.73	0.56	0.57	0.57	0.229	0.58	0.73	0.57	0.52	0.56	0.55

heavily. (Table 5) contains the results of preliminary tests on an 80/20 split of the manually tagged data using several combinations of scale values.

Weights	**Term 3.0 POS 1.0**	**Term 1.0 POS 3.0**	**Term 3.0 POS 2.0**	**Term 2.0 POS 3.0**
Individual tags				
Precision	0.36	0.35	0.35	0.36
Recall	0.20	0.19	0.20	0.20
F-score	0.24	0.23	0.24	0.24
Extraction task				
Precision	0.75	0.74	0.73	0.71
Recall	0.46	0.40	0.45	0.44
F-score	0.57	0.52	0.56	0.55
Correct	0.46	0.40	0.45	0.44
Missed	0.52	0.57	0.52	0.53
Label error	0.00	0.01	0.01	0.02
False positive	0.20	0.20	0.21	0.21

Table 5: 80/20 data with different scales

Performance	**Restricted tags**	**Pruned sentences**	**Restricted tags, pruned sentences**
Individual tags			
Precision	0.44	0.36	0.43
Recall	0.26	0.21	0.27
F-Score	0.30	0.24	0.32
Extraction task			
Precision	0.75	0.74	0.72
Recall	0.44	0.46	0.47
F-Score	0.56	0.57	0.57
Correct	0.44	0.46	0.47
Missed	0.54	0.51	0.51
Label error	0.00	0.02	0.01
False positive	0.19	0.20	0.23

Table 6: Extracted tags results

Because there were so few examples of some of the labels, we tried consolidating some of them. Specifically, we combined the *_M* and *_E* tags into one group *(_M)*. The *_B* tag marks the beginning of an aspect, but we can detect the end when we reach a *B* tag, or a subsequent *_B* tag. This yielded a more marked improvement in performance on individual tag performance, but had little effect on performance in the extraction task. Since the number of background tags is so much larger than the number of aspect tags, we also removed sentences that did not contain opinions and trained the model on the more limited dataset (see Table 6).

10 -fold cross validation of the dataset		
Precision	0.75	0.75
Recall	0.49	0.51
F-score	0.59	0.61
Correctly identified chunks	0.49	0.51
Missed chunks	0.48	0.46
Incorrectly labelled chunks	0.01	0.01
False positives	0.21	0.20

Table 7 10-cross validation

Combining the methods yielded the following results on the 80/20 split (scaling value for term features = 3.0, reduced tag set, pruned sentences).

Tags	**B**	**Feature**	**Opinion**
Weight	**-0.25**	**0.25**	**0.25**
P	0.91	0.72	0.74
R	0.96	0.66	0.71
F	0.94	0.7	0.73
Tags	**B**	**Feature**	**Opinion**
Weight	**0**	**0.5**	**0.5**
P	0.91	0.73	0.75
R	0.97	0.67	0.7
F	0.94	0.7	0.73

Table 8: Extraction task after weighting

The item accuracy for training the CRF of 10-fold cross validation ranged between 83% and 87%.

At this stage, we noticed low recall in extracting aspects and opinion; therefore we tried to balance the labels by giving less weight to the dominant tag and high weight to the features and opinions, where the item accuracy was 2,674/2,812 (0.95%), and the instance accuracy was 111/184 (0.60%).

6.1 Experiment set up and results

In this experiment, we used two datasets of product reviews, in order to train and test the system with different data. One was collected by Qi and Chen (2010) from Yahoo Shopping of different cameras. The other

dataset was collected by Hu and Liu (2004a, 2004b) from Amazon for nine different products. Five random cameras were chosen from both datasets, which gave 1,025 full reviews, consisting of 2,500 opinionated sentences. Then dataset was then tagged using the tag sets described in (Table 2).

For each review, each sentence was hand-labelled, which accumulated of 35,877 terms, with the distribution of labels illustrated in (Figure 3). Words belonging to any product aspects and opinions had *_B, _M*, or *_E*, infixes according to whether they are the first word in a phrase representing the aspect, a word in the middle of the phrase, or the last word in the phrase (some of these tags were combined as described in the methods section). Any word that did not belong to these categories received a background tag, *B*. The distribution of labels is shown in (Table 9).

Most terms were unambiguously associated with a particular label; the average overlap in the sets of terms in all pairs of labels was *3%* and *82%* of the terms received a unique label.

We extracted the following features for each word: the word itself; the part-of-speech of the word; nearby words and part-of-speech tags in a window of configurable size, indexed by relative position to the word. The n-grams of words and parts-of-speech of the length of the window size containing the word; and a *'beginning of sentence'* or *'end of sentence'* tag where applicable. Part-of-speech tagging was done with the default Maxent tagger of the nltk library (trained on the Penn treebank corpus).

6.2 Error Analysis

The error analysis indicated some detected mistakes that we face during experiment. Most of the errors were due to the nature of the data, since it does not following a constant sentence structure, in which case the proposed CRF model would not detect the pattern easily.

Label	No of labels
B	28,541
Feature_B	2,526
Feature_M	359
Feature_E	881
Feature_B_Imp	192
Feature_M_Imp	42
Feature_E_Imp	33
Opinion_B_N_Exp	439
Opinion_M_N_Exp	163
Opinion_E_N_Exp	248
Opinion_B_P_Exp	1,549
Opinion_M_P_Exp	179
Opinion_E_P_Exp	525
Opinion_B_N_Imp	25
Opinion_M_N_Imp	15
Opinion_E_N_Imp	18
Opinion_B_P_Imp	55
Opinion_M_P_Imp	45
Opinion_E_P_Imp	42

Table 9: Distribution of labels

Unsurprisingly, precision was high. Many of the correctly identified aspects occurred many times in the training set (for example, *'camera'* was extracted as a feature in 29 of 34 appearances and *'great'* in 18 of 20). Almost 50 aspects were correctly extracted despite occurring only once.

Most of the items that were missed occurred only once or twice. The highest single number of misses was five out of 34 instances of *'camera'*. The next highest were all four occurrences of *'sensor'* (feature), and three of five occurrences of *'photos'* (feature).

These four chunks were correctly extracted, but assigned the wrong label, such as *user (Opinion_P_Exp / Feature_Imp)* and *quality (Opinion_P_Exp / Feature)*. There seems to be a trend of mis-characterising the polarity of opinions, and perhaps mistaking opinions for some features. The false positives are the most interesting errors, these are some examples: *product (None / Feature_Imp) (1), rechargeable battery (None / Feature) (1), LCD (None / Feature) (1*). Most, if not all, are entirely reasonable extracts. Some of the extractions, especially the features, are clearly mis-tagged in the original dataset: *'large view screen'*, *'rechargeable battery'*, *'wide angle coverage'*, *'viewfinder'*, etc. Camera models, such as *'Kodak camera'* and *'Canon XS'* were also correctly identified.

7 Discussion and Conclusion

In this paper, we have analysed the ABOM problem. We propose a CRF-based method to extract all possible aspects and corresponding opinions in reviews and integrate basic linguistic features with statistical features and combined features. As a result, the model achieves high performance.

We were able to achieve high performance when applying CRFs to opinion mining by the selected feature functions. However, when attempting to improve the performance, this seemed to be determined by the limitations of the dataset rather than the defects of the technique. We had 2,500 sentences, and only 60% of them expressed explicit opinions and features.

We considered using a bootstrapping process to augment our data; however, the performance was not as we expected. We wrote a bootstrapping script that used votes from several models to output sentences where all models agreed, but the danger is that agreement might not be a good indication of correctness in this case. This script would nonetheless be useful in easing the process of manually annotating data, as it would be easier to correct tags than to assign them from scratch. Incorrectly, tagged data is also a problem, particularly when there is a limited opinionated dataset that are manually tagged. The impact of a mistake is much greater since it less likely to be overshadowed by correct instances when there are not many of the latter. Nonetheless, it is clear from the comparison with the baseline use of word frequency that the ability of CRF to exploit context results is definitely helpful. Further work might include adding features based on semantics, as well as improving the quality of the training data by adding more opinionated data. In future work, domain knowledge will be added to the identification process and then integrated with the use of current features to enable more effective features.

8 Acknowledgment

The authors would like to thank Mahnoosh Kholghi for her opinions and recommendations.

9 References

Abbasi Moghaddam, S. (2013): *Aspect-based opinion mining in online reviews*. Applied Sciences, School of Computing Science.

Baccianella, S., Esuli, A. and Sebastiani, F. (2009): Multi-facet rating of product reviews. *Advances in Information Retrieval*, 461–472, Springer.

Baldridge, J. (2005): The opennlp project. http://opennlp.apache.org/index.html. Accessed 2 February 2012.

Banitaan, S., Salem, S., Jin, W. and Aljarah, I. (2010): A formal study of classification techniques on entity discovery and their application to opinion mining. edited, 29–36, ACM.

Choi, Y. and Cardie, C. (2010): Hierarchical sequential learning for extracting opinions and their attributes. *Proc. ACL 2010 Conference Short Papers*, 269–274, ACL.

Choi, Y., Cardie, C., Rilof, E. and Patwardhan, S. (2005): Identifying sources of opinions with conditional random fields and extraction patterns. *Proc. Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Vancouver, Canada, ACL, doi: 10.3115/1220575.1220620.

Dave, K., Lawrence, S. and Pennock, D.M. (2003): Mining the peanut gallery: Opinion extraction and semantic classification of product reviews, 519–528, ACM.

Ding, X., Liu, B. and Zhang, L. (2009): Entity discovery and assignment for opinion mining applications. *Proc. 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, ACM, doi: 10.1145/1557019.1557141.

Eddy, S.R. (1996): Hidden Markov models. *Current Opinion in Structural Biology* **6**(3):361–365.

Glance, N., Hurst, M. and Tomokiyo, T. (2004): Blogpulse: Automated trend discovery for weblogs. *WWW 2004 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*.

Guo, H., Zhu, H., Guo, Z., Zhang, X.X. and Su, Z. (2009): Product feature categorization with multilevel latent semantic association. *Proc. 18th ACM Conference on Information and Knowledge Management*, 1087–1096, ACM.

Himmat, M. and Salim, N. (2014): Survey on product review sentiment classification and analysis challenges. *Proc. First International Conference on Advanced Data and Information Engineering (DaEng-2013),* 213–222. Herawan, T., Mat Deris, M. and Abawajy, J. (eds). Springer, Singapore.

Hu, M. and Liu, B. (2004a): Mining and summarizing customer reviews. *Proc. 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 168–177, ACM.

Hu, M. and Liu, B. (2004b): Mining opinion features in customer reviews. *AAAI*, 755–760.

Huang, S., Liu, X., Peng, X. and Niu, Z. (2012): Fine-grained product features extraction and categorization in reviews opinion mining. *IEEE 12th International Conference on Data Mining Workshops,* 680–686, IEEE.

Jakob, N. and Gurevych, I. (2010): Extracting opinion targets in a single- and cross-domain setting with conditional random fields. *Proc. 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA, USA, ACL.

Jin, W., Ho, H.H. and Srihari, R,K. (2009a): A novel lexicalized HMM-based learning framework for web opinion mining. *Proc. 26th Annual International Conference on Machine Learning*, 465–472, Citeseer.

Jin, W., Ho, H.H. and Srihari, R.K. (2009b): OpinionMiner: A novel machine learning system for web opinion mining and extraction. *Proc. 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, 1195–1204, ACM.

Klinger, R. and Friedrich, C.M. (2009): Feature subset selection in conditional random fields for named entity recognition. *RANLP*, 185–191.

Lafferty, J., McCallum, A. and Pereira, F.C.N. (2001): Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lim, E.P., Nguyen, V.A., Jindal, N., Liu, B. and Lauw, H.W. (2010): Detecting product review spammers using rating behaviors. *Proc. 19th ACM International Conference on Information and Knowledge Management*, 939–948, ACM.

Liu, B. (2012): Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies* **5**(1):1–167.

Liu, B. and Zhang, L. (2012): A survey of opinion mining and sentiment analysis. *Mining Text Data*, 415–463.

Liu, B., Hu, M. and Cheng, J. (2005): Opinion observer: Analyzing and comparing opinions on the web. *Proc. 14th International Conference on the World Wide Web*, 342–351, ACM.

Marcińczuk, M. and Janicki, M. (2012): Optimizing CRF-based model for proper name recognition in Polish texts. *Computational Linguistics and Intelligent Text Processing*, 258–269: Springer.

McDonald, R. and Pereira, F. (2005): Identifying gene and protein mentions in text using conditional random fields. *BMC Bioinformatics* **6**(Suppl 1): S6.

Moghaddam, S. and Ester, M. (2010): Opinion digger: An unsupervised opinion miner from unstructured product reviews. *Proc.19th ACM International Conference on Information and Knowledge Management*, 1825–1828, ACM.

Moghaddam, S. and Ester, M. (2011): AQA: Aspect-based opinion question answering. *IEEE 11th International Conference on Data Mining Workshops*, 89–96, IEEE.

Moghaddam, S., Jamali, M. and Ester, M. (2011): Review recommendation: Personalized prediction of the quality of online reviews. *Proc 20th ACM International Conference on Information and Knowledge Management*, 2249–2252, ACM.

Noy, N.F. (2004): Semantic integration: A survey of ontology-based approaches. *SIGMOD Record* **33**(4):65–70. doi: 10.1145/1041410.1041421.

Okazaki, N. (2007): crfsuite. http://www.chokkan.org/software/crfsuite.

Pak, A. and Paroubek, P. (2010): Twitter as a corpus for sentiment analysis and opinion mining. *LREC*, 1320–1326.

Pang, B. and Lee, L. (2008): Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval* **2**(1–2):1–135.

Pang, B., Lee, L. and Vaithyanathan, S. (2002): Thumbs up? Sentiment classification using machine learning techniques.' *Proc. ACL-02 Conference on Empirical Methods in Natural Language Processing, Volume 10*, 79–86, ACL.

Peng, F. and McCallum, A. (2006): Information extraction from research papers using conditional random fields. *Information Processing & Management* **42**(4):963–979.

Qi, L. and Chen, L. (2010): A linear-chain CRF-based learning approach for web opinion mining. *Web Information Systems Engineering–WISE 2010*, 128–141, Springer.

Riloff, E. (1996): Automatically generating extraction patterns from untagged text. *Proc. National Conference on Artificial Intelligence*, 1044–1049.

Roller, B., Taskar, C. and Guestrin, D. (2004): Max-margin Markov networks. *Advances in Neural Information Processing Systems* **16**: 25.

Samha, A.K., Li, Y. and Zhang, J. 2014. Aspect-based opinion extraction from customer reviews. *arXiv preprint arXiv:1404.1982*.

Sutton, C. and McCallum, A. (2006): An introduction to conditional random fields for relational learning, vol. 2: *Introduction to statistical relational learning*. MIT Press.

Taboada, M., Brooke J., Tofiloski, M., Voll, K. and Stede, M. (2011): Lexicon-based methods for sentiment analysis. *Computational Linguistics* **37**(2): 267–307.

Titov, I. and McDonald, R. (2008): Modeling online reviews with multi-grain topic models. *Proc. International Conference on the World Wide Web*, 111–120, ACM.

Tjong, K.S., Erik, F. and Buchholz, S. (2000): Introduction to the CoNLL-2000 shared task: Chunking. *Proc. 2nd Workshop on Learning Language in Logic and 4th Conference on Computational Natural Language Learning,* Volume 7, 127–132, ACL.

Turney, P.D. (2002): Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews, 417–424, ACL

Vivekanandan, K. and Aravindan, J.S. (2014): Aspect-based opinion mining: A survey. *International Journal of Computer Applications* **106**:

Wang, H., Lu, Y. and Zhai, C. (2010): Latent aspect rating analysis on review text data: A rating regression approach. *Proc. 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 783–792, ACM.

Wiebe, J., Wilson, T., Bruce, R., Bell, M. and Martin, M. (2004): Learning subjective language. *Computational Linguistics* **30**(3):277–308.

Wogenstein, F., Drescher, J., Reinel, D., Rill, S. and Scheidt, J. (2013): Evaluation of an algorithm for aspect-based opinion mining using a lexicon-based approach. *Proc. 2nd International Workshop on Issues of Sentiment Discovery and Opinion Mining*, 5, ACM.

Xu, B., Zhao, T.J., Zheng, D.Q. and Wang, S.Y. (2010): Product features mining based on conditional random fields model. *International Conference on Machine Learning and Cybernetics (ICMLC)*, 3353–3357: IEEE.

Yu, L., Ma, J., Tsuchiya, S. and Ren, F. (2008): Opinion mining: A study on semantic orientation analysis for online documents. *7th World Congress on Intelligent Control and Automation,* 4548–4552, IEEE.

Zhang, L. and Liu, B. (2014): Aspect and entity extraction for opinion mining, *Data Mining and Knowledge Discovery for Big Data,* 1–40. Chu, W.W. (ed), Springer, Berlin Heidelberg.

Zhang, L., Ghosh, R., Dekhil, M., Hsu, M. and Liu, B. (2011): Combining lexicon-based and learning-based methods for Twitter sentiment analysis. *HP Laboratories Technical Report HPL-2011,* 89.

Zhao, L. and Li, C. (2009): Ontology-based opinion mining for movie reviews. *Knowledge Science, Engineering and Management*, 204–214.

Zhuang, L., Jing, F., Zhu, X.Y. and Zhang, L. (2006): Movie review mining and summarization. *Proc. 15th ACM International Conference on Information and Knowledge Management*, 43–50.

FSMEC: A Feature Selection Method based on the Minimum Spanning Tree and Evolutionary Computation

Amer Abu Zaher, Regina Berretta, Ahmed Shamsul Arefin, Pablo Moscato
The Priority Research Centre for Bioinformatics, Biomarker Discovery and Information-based Medicine
Information-based Medicine Program, Hunter Medical Research Institute
School of Electrical Engineering and Computer Science, The University of Newcastle, Australia

Amer.AbuZaher@uon.edu.au, {Regina.Berretta, Ahmed.Arefin, Pablo.Moscato}@newcastle.edu.au.

Abstract

In feature selection we aim at reducing the dimensionality of a dataset by excluding characteristics that do not compromise, and potentially enhance, the classification of a set of samples. We present a new type of supervised and multivariate feature selection approach that works by constructing proximity graphs in such a way that the number of edges connecting samples from different classes is minimised. We present this general idea using the Minimum Spanning Tree as a proximity graph and an Evolutionary Algorithm approach is used to search for a feature subset. We compare the performance of our algorithm against other feature selection methods, *(alpha,beta)-k*-Feature Set, and a ranking-based feature selection method, based on the use of CM1-scores. We employ two publicly available real-world datasets (one with training and test variants). The classification accuracies have been evaluated using a total of 49 methods from an open source data mining and machine learning package WEKA.

Keywords: Feature selection, evolutionary algorithm, proximity graph, minimum spanning tree.

1 Introduction

Feature selection is an essential task in data mining, and it is particularly relevant in bioinformatics where, in many cases, the number of features greatly exceed the number of samples (e.g., see (Dash & Liu, 1997; Guyon & Elisseeff, 2003; Liu & Yu, 2005; Saeys, Inza, & Larrañaga, 2007; Yang et al., 2012)). The core idea is to select the best subset of features that may potentially describe the whole dataset without losing important information, which may be useful for discrimination in classes of interest. Feature selection methods are generally classified as either: filter, wrapper or hybrid by the nature of its approach. The *filter* approach is simple, fast, and generally computational efficient however, does not consider the potential benefits given by learning algorithms, which may influence the selection of features that act synergistically (Dash & Liu, 1997; Kohavi & John, 1997; Yu & Liu, 2003). The filter approach consists of two types: univariate and multivariate. *Univariate* methods start by individually ranking the features and by assigning a score to each of the available features according to a pre-defined criterion (Dash & Liu, 1997). In this manner, generally the best-scored features are selected, while the others are discarded. However, this process does not consider the mutual information between features. When the top-scoring features are highly correlated in a given dataset it may be necessary to also consider other features for discriminating all pairs of samples. On the other hand, *multivariate* methods rank a group of features instead of individual features and decide which combination of a subgroup of features is the best. The *wrapper* approach combines, in a feedback loop, the selection process into search, learning and evaluation phases, employing a classifier to evaluate the selected subset of features. Therefore, it is computationally more expensive. At the same time, it is more accurate than other methods, even though it may overfit the training data which is an issue of concern for small datasets (Dash & Liu, 1997; Kohavi & John, 1997). The *hybrid* approach includes characteristics of both the filter and wrapper approaches, allowing for a feedback loop between the feature selection process and the learning algorithm.

The proposed method (FSMEC) is a supervised filtering multivariate approach. To understand its rationale we first note that we are given an $m \times n$ matrix of values in which each of the m rows represents a feature and each of the n columns a sample. In addition, for each of the n samples we have an associated class label. If k rows are selected, we can then calculate (using the resulting $k \times n$ submatrix) an $n \times n$ distance matrix between the samples. A coefficient in this matrix represents a distance between a pair of samples only considering the subset of selected features. Using this $n \times n$ matrix as input, a Minimum Spanning Tree (MST) can be found. The number of edges in the resulting MST that are connecting samples from different classes as well as the total number of selected features can then be used as a quality measure of the subset of features. This contribution explores some variants of interest involving these quality measures.

Evolutionary Computation (EC) is a technique (Back, Fogel, & Michalewicz, 1997; Fogel, 2006), that has been successfully used in a variety of fields including feature selection (ElAlami, 2009; Wu, Tang, Hor, & Wu, 2011). We propose its use for searching a subset of features that produces a MST with the best fitness value. We investigate in this contribution as set of fitness functions and details will be given later in the paper. The proposed FSMEC method is tested on two datasets and the results are compared with those of two other feature selection techniques. One method is based on the use of CM1 scores (Marsden, Budden, Craig, & Moscato, 2013) and

the other one based on the *(α,β)-k*-Feature Set problem (Berretta, Costa, & Moscato, 2008). To evaluate the performance of each feature selection algorithm, we used several classifiers available in the widely used WEKA software package (Witten, Frank, & Hall, 2011).

The structure of this paper is as follows. Section 2 explains how we use the MST to evaluate the quality of a subset of features. In Section 3 the proposed evolutionary algorithm is described in details. Section 4 presents the results. Finally, Section 5 concludes this paper.

2 The MST and Feature Selection

The input of the problem is a matrix H_{mxn} and an array C_n, where m denotes the number of features, n denotes the number of samples, h_{ij} holds the value of feature i in sample j and c_j holds the class of the sample j. Next, giving k selected features, a distance matrix D_{nxn} is calculated, where d_{ij} is the distance between samples i and j. Note that each subset of features will, in general, lead to the computation of a different distance matrix.

A complete, undirected and weighted graph $G(V,E,W)$ is used to represent the matrix H_{kxn} (considering only the k features), where the nodes in V represent each sample ($|V|= n$), E is a set of edges reflecting the relationship between samples, and W is a set of edge weights with $w_{ij} = d_{ij}$. Given the graph G described above, a sub-graph of G named *spanning tree* is one that links all the nodes together with ($n-1$) edges. There are several possible spanning trees for a graph G. The MST is a spanning tree with the lowest total sum of weights of its ($n-1$) edges (Graham & Hell, 1985), which we will denote is as $G_{MST}(V,E_{MST},W_{MST})$. The search for the MST is a combinatorial optimization problem and there are different efficient algorithms to solve this problem. Two well-known algorithms are Kruskal's (Kruskal, 1956) and Prim's (Prim, 1957). In this paper, Prim's algorithm is adopted due to the density of the graph and the minimal usage of memory (Graham & Hell, 1985). The quality of the subset of k features is evaluated based on its size, the number of edges connecting nodes (samples) from different classes (denoted by e) in the built MST, and a score calculated for the constructed MST (described later). An Evolutionary Algorithm, described in the next section, is applied for searching the best subset of features. The merit of different fitness functions is also evaluated.

3 FSMEC-Feature Selection Based on MST using Evolutionary Computation

Evolutionary Computation EC is used as a search strategy for finding a subset of k features. The evolutionary algorithm (EA) used in our approach is presented in Algorithm 1.

First, a population of p individuals is created by *initialisePop()*. An individual represents a solution to the problem, which in our case is a subset of features. It is represented by an array S of size m, where $s_i=1$ if feature i is selected and 0 if otherwise.

At each generation, the algorithm applies the following operators: parent selection (*selectParents (parent1, parent2)*), crossover (*crossover (parent1, parent2)*), and mutation (*mutation(offspring)*). We apply an update procedure *updatePop(P)* that replaces the worst individual with a new one if the latter has a better fitness than the former.

Algorithm 1: Pseudo-code of the Evolutionary Algorithm implemented for the FSMEC.

```
n: number of samples; m: number of features
p: number of individuals in the population.
P: population of individuals and their fitness values.
1  P ← initialisePop()
2  REPEAT
3      selectParents(parent1,parent2)
4      offspring ← crossover(parent1,parent2)
5      offspring ← mutation(offspring)
6      fitness(offspring)
7      population ← updatePop(P)
8  UNTIL max_numberOfGenerations or
numberOfGenerationWithoutImprovement
```

3.1 Population Initialisation

The population is structured as a list P of objects with each object containing an individual and its fitness value. The *initialisePop()* initialises the population of 40 individuals ordered by their fitness values. At each generation, a new individual is generated and may replace the worst individual in the population according to *updatePop(P)*. Since the population is ordered, a binary search procedure is used to locate the new individual in the population. It is worth mentioning that the time complexity of this process, $O(log(|S|))$, is significantly better than re-sorting the population for each new generation which would cost $O(|S|log(|S|))$. Two population initialisation strategies have been implemented in the EA:

Population initialisation strategy 1 - *5Bins*. In this strategy we divide P into five equal bins. Each bin b, $1 \le b \le 5$, for each individual, (b*10)% of features are randomly selected. For example, for b=1, 10% of features are randomly selected in each individual, for b=2, 20% of features are selected, and so forth.

Population initialisation strategy 2 (*5Bins-CM1*). It is similar to strategy 1 (5Bins), but uses the CM1 score to influence the probability of a feature to be chosen. Initially, we compute the CM1 score (described in Section 4) for each feature and normalise the CM1 values (CM1_norm). Similarly to strategy 1 (5bins), P is divided in 5 bins. In the first bin, 10% of features are chosen, but features with CM1_norm greater than 0.5 will have 90% chance to be chosen, while the features with CM1_norm score less than 0.5 will have 10% of probability to be chosen. In the second bin, 20% of features are chosen, but features with CM1_norm greater than 0.5 will have 80% chance to be chosen, while the features with CM1_norm score less than 0.5 will have 20% chance to be chosen. For the bins 3, 4 and 5, an equivalent strategy is employed.

3.2 Optimisation criteria - Fitness functions

In this work, we are considering only two classes. We have four fitness functions to deal with them:

Mine. Minimising the number of edges, *e*, connecting samples from different classes in the MST. (Min *e*)

MineMink. Minimising the summation of the normalised *e* (i.e. the number of inter-class edges in the MST connecting samples from different classes divide by the total number of edges in the MST) and normalised *k* (the number of selected features).

$$Min \frac{e}{n-1} + \frac{k}{m}$$

For the next two we need to define a score for the built MST (*MSTscore*). Consider a MST as a graph $G_{MST}(V,E_{MST},W_{MST})$. The set of nodes V has a bipartition in V_A and V_B; V_A are the set of samples that belong to class A and V_B are the set of samples that belong to class B. Using the weights of the edges that connect nodes from the same class, we can define a score for the MST given a partition of its set of vertices as follows:

$$MSTscore(A,B) = \frac{\frac{1}{|V_A|}\sum_{\substack{i\in V_A \\ j\in V_A}} w_{ij} - \frac{1}{|V_B|}\sum_{\substack{i\in V_B \\ j\in V_B}} w_{ij}}{1 + max_{\substack{i\in V_B \\ j\in V_B}}\{w_{ij}\} - min_{\substack{i\in V_B \\ j\in V_B}}\{w_{ij}\}}$$

Note that in general *MSTscore(A,B)* is different than *MSTscore(B,A)*. We can then define the other two functions:

MineMaxMSTscore: In this case the objective is to minimize the ratio between the number of edges, *e*, connecting samples from different classes in the MST and the score of the *MSTscore(A,B)*

$$Min \frac{e}{MSTscore(A,B)}$$

MineMinkMaxMSTscore: In this case the objective is to minimize the summation of normalised *e* and normalised *k*; divided by the *MSTscore(A,B)*

$$Min \frac{\frac{e}{n-1} + \frac{k}{m}}{MSTscore(A,B)}$$

3.3 Breeding

We refer to the process in which two "parent" solutions are selected and a new solution is created. It consists of four operations: *selecting* parent solutions from the population; creating a new offspring solution from the selected parents using *crossover* and *mutation* operations and (in case the new offspring has a better fitness function than the worst individual in the population) *replacing* the worst individual in the population with the new offspring. Each operation is described below.

Selection: selects two solutions as parents: the solution with the best fitness value in the population and another selected uniformly at random.

Crossover: a *uniform* crossover with a crossover rate of 40% is used.

Mutation: two mutation strategies have been applied in the EA and both are *1-flip mutation* with 5% mutation rate. In strategy 1 – **Mutation1** – 5% of each of the individual solutions are randomly mutated. In the second mutation strategy – **Mutation-CM1** – normalised scores (as used in the 5Bins-CM1 population initialisation) are used in the following way. If the feature is selected and the CM1_norm is less than 0.5, then the feature is discarded. If the feature is not selected and the CM1_norm is greater than 0.5, then it is selected.

Replacement. Finally, the mutated offspring is tested in the replacement strategy. If its fitness is better than the worst individual, then it is included in the population and the worst individual is discarded. A binary search is performed to locate the position of the offspring in the population. Two stopping criteria have been used in the FSMEC algorithm: the EA will stop after a predetermined *maximum number of generations* (10000) or the best individual is *unchanged for a fix number of generations* (1000).

4 Computational Results

For the purpose of evaluating the performance of our FSMEC algorithm, two datasets are used to carry out the computational experiments, which are described next. Our algorithm was coded in Python 2.7 and executed under Unix operating system in a machine with Dual Xeon 2.67 GHz, 8 cores and 32 GB RAM.

4.1 Datasets

The properties of the two datasets used are summarised in (Table 1) and described next.

Dataset Name		features	samples	Classes	
Shakespearean era plays and poems (Craig & Whipp, 2010)		220	256	Plays	202
				Poems	54
Alzheimer's disease (Ray et al., 2007)	Training	120	83	AD	43
				NDC	40
	Test	120	81	AD	42
				NDC	39

Table 1. Datasets used for evaluating the FSMEC algorithm.

Shakespearean era plays and poems dataset. This dataset contains 256 works of the Shakespearean era and they belong to two classes: plays (202) and poems (54) as samples and 220 *functional words* as features. The "frequency of use" of these 220 words have been extracted from a cohort of 66907 words previously analysed by Arefin et al. in (Arefin, Vimieiro, Riveros, Craig, & Moscato, 2014). The goal is then to identify 'a subset of functional words' that is able to group the texts into the two classes; plays and poems.

Alzheimer's Disease dataset. It consists of two sub-datasets: the training and test datasets from Ray et al. (Ray et al., 2007). The *training* dataset contains the relative abundances 120 proteins (z-scores, which will be used as features) measured on 83 people who have been classified into two classes: 43 Alzheimer's Disease (AD) samples and 40 Non Demented Control (NDC) samples. The *test* dataset contains 120 proteins and 81 patients classified into two classes - 42 AD samples and 39 NDC samples. The test dataset also contains 11 samples labelled as 'Other Dementia' OD samples, which were excluded from the analysis.

Different methods have been introduced to find an optimum molecular test for an earlier diagnosis of

Alzheimer's disease (Berretta et al., 2008; Ravetti & Moscato, 2008; Ray et al., 2007). They define the signatures that are able to distinguish between NDC and AD samples and hence predict the AD samples that already have a Mild Cognitive Impairment. In the same manner, the FSMEC is used to find a subset from the 120 proteins that can separate the AD patients from NDC ones.

4.2 Evolutionary Algorithm Analysis

The initial tests were conducted to evaluate and setup an initial configuration for the EA. All the results were analysed using the Wilcoxon test. We tested different crossovers operators, mutation rates, population sizes and stop criteria. Our preliminary tests indicate that the best results are achieved when we apply the uniform crossover (crossover rate 0.4), a 1-flip mutation (mutation rate 0.05), a population size of 40, and when the number of generations without improvements is equal to 1000.

EA	Population Strategy	Mutation Strategy
EA1	5Bins	Mutation1
EA2	5Bins-CM1	Mutation-CM1

Table 2. The configuration of the two Evolutionary Algorithms tested (according to population initialisation and mutation strategies).

Fitness function	EA	*k*	*e*	*Fit*	*Time*	*Gen*
Mine	EA1	64	2	**2.0**	333	1599
	EA2	101	3.2	3.2	303	1686
MineMink	EA1	2	96	**0.01**	320	1702
	EA2	5	40	0.02	422	1616
MineMaxMSTscore	EA1	112	2.8	**9.5**	1511	5847
	EA2	59.6	4	13.7	2363	9143
MineMinkMaxMSTscore	EA1	107	3	**0.036**	1541	5110
	EA2	60.8	3.6	0.052	2586	10000

Table 3. Average number of selected features *k*, inter-class edges *e*, fitness *Fit*, running *time*, and generations *gen* obtained by EA1 and EA2 using the Shakespeare era plays and poems dataset for each optimisation criteria.

Fitness function	EA	*k*	*e*	*Fit*	*Time*	*Gen*
Mine	EA1	43	5.9	5.9	18.3	1683
	EA2	53	5.8	**5.8**	19.5	1950
MineMink	EA1	53	5	**0.068**	25.6	2114
	EA2	35	7	0.086	24.2	1647
MineMaxMSTscore	EA1	55.3	7.6	44.3	103.1	5934
	EA2	31.4	6.5	**35.0**	115.3	7592
MineMinkMaxMSTscore	EA1	52.2	7.1	**0.98**	76.2	4385
	EA2	17.6	9.6	1.20	92.9	6587

Table 4: Average number of selected features *k*, inter-class edges *e*, fitness *Fit*, running *time*, and generation *gen* obtained by EA1 and EA2 using the Alzheimer disease training dataset for each optimisation criteria.

In the next set of experiments we tested the two different population initialisation strategies (*5Bins* and *5Bins-CM1*) and the two mutation strategies (*Mutation1* and *Mutation-CM1*) as depicted in Table 2. Tables 3 and 4 show the results obtained from the Shakespearean era plays and poems and Alzheimer's disease training datasets, respectively, for each optimisation criteria. Each row in these tables is the average of 10 executions. These tables highlight the best EA for each optimisation criteria. For the Shakespearean era plays and poems dataset (Table 3), the EA1 achieved the best average fitness score (*Fit*) for the four optimisation criteria. For the Alzheimer's disease training dataset, the EA1 was the best for two optimisation criteria (*MineMink* and *MineMinkMaxMSTscore*) and EA2 performed better for the other two optimisation criteria. EA1 was superior for Shakespearean era plays and poems dataset, and for the Alzheimer's disease training dataset there was a tie between EA1 and EA2.

4.3 Classification Performance

The next computational test aimed to evaluate the practical use of the set of features obtained by our FSMEC for a learning algorithm.

Type	Classifier	Type	Classifier
Bayes	BayesNet	Meta	RandomSub_Space
Bayes	NaiveBayes	Meta	RotationForest
Bayes	NaiveBayesUpdatable	Meta	ThresholdSelector
Function	Logistic	Mesc	HyperPipes
Function	SimpleLogistic	Mesc	VFI
Function	RBFNetwork	Rules	ConjunctiveRule
Function	SMO	Rules	DecisionTable
Function	SPegasos	Rules	Jrib
Function	VotedPerceptron	Rules	NNge
Lazy	IB1	Rules	OneR
Lazy	Kstar	Rules	Part
Lazy	LWL	Rules	Ridor
Meta	AdaBoost	Tree	ADTree
Meta	AttributeSelectedClassifier	Tree	BFTree
Meta	Bagging	Tree	FT
Meta	ClassificationViaRegression	Tree	LADTree
Meta	Dagging	Tree	LMT
Meta	Decorate	Tree	DecisionStump
Meta	END	Tree	J48
Meta	FilteredClassifier	Tree	J48graft
Meta	LogitBoost	Tree	RepTree
Meta	MultiBoostAB	Tree	NBTree
Meta	MultiClassClassifier	Tree	Random_Forest
Meta	OrdinalClass	Tree	RandomTree
Meta	RandomCommittee		

Table 5: List of classifiers associated with their types as categorised in WEKA (Witten et al., 2011) version 3.6.4.

For each row in the tables 3 and 4, the subset of features (out of 10 solutions) that has the best fitness value (*Fit*) was selected to be evaluated by a learning algorithm. We have used 49 machine learning algorithms from the well-known WEKA software package (Witten et al., 2011). Table 5 lists all of the classifiers considered, along with their respective types as categorised in WEKA (version 3.6.4). In each case, the average specificity, sensitivity,

classification accuracy, and the Matthews' correlation coefficient (MCC) have been calculated using 10-fold cross validation, which means that the original dataset is randomly divided into 10 equal subsets: 9 are used as training sets, and the remaining subsets are used as test sets. Evaluation is repeated 10 times, such that each subset is utilised exactly once for this purpose (Witten et al., 2011). The results are shown in Tables 6 and 7. Each row in these tables shows the evolutionary algorithm applied (EA1 or EA2), the size of the subset of features (k), the number of inter-class edges (e), and the fitness value (*Fit*) for the specific optimisation criterion. It also shows the average classification accuracy and the average MCC of 49 classifiers.

Optimisation Criteria	EA	*k*	*e*	*Fit*	ACC	MCC
Mine	EA1	84	1	1	**0.959**	**0.876**
	EA2	89	1	1	0.952	0.855
MineMink	EA1	94	1	0.018	**0.957**	**0.869**
	EA2	33	4	0.016	0.955	0.866
MineMaxMSTscore	EA1	117	2	6.7	0.954	0.861
	EA2	50	3	6.3	**0.959**	**0.876**
MineMinkMaxMSTscore	EA1	110	2	0.027	**0.960**	**0.880**
	EA2	52	1	0.028	0.956	0.868

Table 6: Average accuracy and Matthews' correlation coefficient (MCC) using the best subset of features from FSMEC for the Shakespeare era plays and poems dataset. Each row shows number of features (*k*), the number of inter-class edges (*e*), the fitness value (*Fit*), the average accuracy and MCC.

Optimisation Criteria	EA	*k*	*E*	*Fit*	ACC	MCC
Mine	EA1	56	5	5	0.859	0.720
	EA2	41	5	5	**0.861**	**0.725**
MineMink	EA1	4	57	0.045	0.860	0.721
	EA2	6	35	0.073	**0.867**	**0.737**
MineMaxMSTscore	EA1	55	6	35.4	0.862	0.725
	EA2	27	5	29.8	**0.875**	**0.752**
MineMinkMaxMSTscore	EA1	42	7	0.888	0.873	0.748
	EA2	15	8	0.795	**0.880**	**0.762**

Table 7. Average accuracy and Matthews' correlation coefficient (MCC) using the best subset of features from FSMEC for the Alzheimer's disease dataset. Each row shows number of features (*k*), the number of inter-class edges (*e*), the fitness value (*Fit*), the average accuracy ACC and MCC.

Table 6 illustrates the results for the Shakespearean era plays and poems dataset and Table 7 for the Alzheimer's disease training dataset. The FSMEC has been applied on the Alzheimer disease *training* dataset to find subsets of features, which in turn have been used to build a classification model over the Alzheimer disease *test* dataset. For each dataset and optimisation criterion, the best classification accuracy and MCC results are highlighted. In the previous experiment, EA1 performed better for the Shakespearean era plays and poems dataset (see Table 3). When we evaluate the classification performance, the results were similar, showing that EA1 obtained better results for 3 out of 4 optimisations criteria (see Table 6). In the case of the Alzheimer's disease dataset (see Table 7) the results showed a better performance of EA2. After analysing the different optimisation criteria, we found that the *MineMinkMaxMSTscore* obtained slightly better classification performance for both datasets, independently of the EA applied.

4.4 Benchmark Techniques

To examine the performance of the proposed method (FSMEC), two feature selection methods ((α,β)-*k*-Feature set and CM1 score) are used as benchmark techniques. The *(α,β)-k-Feature Set* is a supervised, multivariate filter method based on combinatorial optimization first proposed by Cotta et al. (Cotta, Sloper, & Moscato, 2004) and then used by many other applications (Berretta et al., 2008; Berretta, Mendes, & Moscato, 2005, 2007; de Paula, Ravetti, Berretta, & Moscato, 2011; Hourani, Berretta, Mendes, & Moscato, 2008; Ravetti, Rosso, Berretta, & Moscato, 2010). The task is to identify k features such that at least α features of these k can explain the dichotomy between samples that belong to different classes.In addition, those k features should satisfy that at least β features must explain the similarities between samples from the same class. A mathematical programming software called CPLEX has been used to obtain solutions using integer programming techniques (Berretta et al., 2008). For further details about the (α,β)-*k*-Feature Set problem we refer to (Berretta et al., 2008).

In contrast, the *CM1* score (Marsden et al., 2013) is a supervised, univariate filter method. It works by individually ranking the features according to their expression values in order to identify features presenting differentiation between samples from a target class and samples from the outclass. Consider V_A and V_B are a partition of samples (V) in the dataset of interest (i.e. $V_A \cup V_B = V$ and $V_A \cap V_A = \emptyset$), such that V_A is a set of all samples that belong to one class and V_B is a set of all samples that are not labelled as class A. . The CM1 score for a feature k can then be defined as

$$CM1_k = \frac{\frac{1}{|A|}\sum_{i \in A} h_{ki} - \frac{1}{|B|}\sum_{i \in B} h_{ki}}{1 + \max_{i \in V_B}\{h_{ki}\} - \min_{i \in V_B}\{h_{ki}\}}$$

where h_{kxi}, as described before, holds the value of the feature k in the sample i.

Table 8 summarises the size of the features' sets obtained by the benchmark techniques and the FSMEC. For CM1, we are following the same approach of [13], we select the 20 highest and 20 lowest CM1 markers for both words in the Shakespeare era plays and poems dataset and features in the Alzheimer's disease dataset.

Table 9 summarises the average MCC obtained by the benchmark techniques and the FSMEC. It shows the average MCC results achieved using all (*ALL*) the features *(ALL),* the benchmark methods ((α,β)-*k*-Feature set and CM1) and the results obtained by the four FSMEC's optimisation criteria.

	Dataset							
	Shakespeare				Alzheimer's			
All	220				120			
(α,β)-*k*-FEATURE SET	140				10			
CM1	40				40			
	EA1		EA2		EA1		EA2	
	k	*e*	*k*	*e*	*k*	*e*	*K*	*e*
Mine	84	1	89	1	56	5	41	5
MineMink	94	1	33	4	57	4	35	6
MineMaxMSTscore	117	2	50	3	55	6	27	5
MineMinkMaxMSTscore	110	2	52	1	42	7	15	8

Table 8. The size of resulting features' subsets (*k*) obtained by the benchmark techniques and the number of inter-class edges *e* and the value of *k* for the FSMEC for each optimisation criteria and considering the EA1 and EA2.

	Dataset							
	Shakespeare				Alzheimer's			
	ACC		MCC		ACC		MCC	
All	0.949		0.849		0.848		0.698	
(α,β)-*k*-FEATURE SET	0.952		0.856		**0.891**		**0.784**	
CM1	0.941		0.824		0.860		0.722	
	EA1		EA2		EA1		EA2	
	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
Mine	0.959	0.876	0.952	0.855	0.859	0.720	0.861	0.725
MineMink	0.957	0.869	0.955	0.866	0.86	0.721	0.867	0.737
MineMaxMSTscore	0.954	0.861	0.959	0.876	0.862	0.725	0.875	0.752
MineMinkMaxMSTscore	**0.960**	**0.880**	0.956	0.868	0.873	0.748	0.880	0.762

Table 9. The best average MCC and ACC (accuracy) results achieved from the FSMEC' four optimisation criteria and the two benchmark methods for the dataset under study. The (All) means all features *m* without applying a feature selection.

It also shows that most of FSMEC's optimisation criteria demonstrated their superiority in terms of the MCC and the accuracy over the other methods in case of the Shakespearean era plays and poems dataset. If we compare our four FSMEC's optimisation criteria, the *MineMinkMaxMSTscore* achieved the best MCC results with 0.880 and the best accuracy with 96.0% compared with *Mine*, *MineMink* and *MineMaxMSTscore*. Notably, the value of *e* is 2 in case of the best *MineMinkMaxMSTscore* while the value of *e* is 1 in case of both *Min e* and the *MineMink* with both providing very close MCC values.

In case of the Alzheimer's disease training dataset, the (α,β)-*k*-Feature Set provided the best average MCC result (0.784), however our proposed method is highly competitive with *MineMinkMaxMSTscore* attaining a MCC of 0.762. Additionally, the FSMEC obtained better results than CM1 and using all features (*ALL*).

Notably, the *MineMinkMaxMSTscore* in case of both the Shakespeare era plays and poems dataset and Alzheimer's disease dataset using EA1 or EA2 always obtained better classification performance compared with other optimisation criteria. However, there is no clear winner between EA1 and EA2.

Next, we selected the five best performing WEKA classifiers in each experiment from Tables 8 and 9. These results are organised in Tables 10-13. Tables 10 and 11 show the results achieved for the Shakespeare era plays and poems dataset using EA1 (Table 10) and EA2 (Table 11). Tables 12 and 13 show the results for the Alzheimer's disease dataset using EA1 and EA2, respectively. Each table shows the number of features (*k*), ACC and MCC achieved by each classifier. We also show the average and median of the results for each method in the last two rows. Note that the list of classifier methods in each table is the union of the five best performing methods in each experiment. In each row of each table we highlighted the best result(s).

Analysing the results in Tables 10 and 11 we note that the *MineMinkMaxMSTscore* optimisation criterion continues to lead the results by achieving 14 best results (6 using EA1 and 8 using EA2). It also achieves the best average and median results. The *MineMaxMSTscore* optimisation criterion also obtained good results, achieving 8 best results (3 for EA1 and 5 for EA2). Figure 1 shows the MST that has been constructed from the selected features and using the *MineMinkMaxMSTscore* optimisation criterion for Shakespeare era plays and poems dataset, using EA2, with 52 features. In the case of the Alzheimer's disease dataset (see Table 12 and 13), the *MineMinkMaxMSTscore* optimisation criterion also achieves excellent results. When we compare using median, the best results are achieved by *MineMinkMaxMSTscore* and (α,β)-*k*-Feature Set method.

In the next section, we show the effect of the different optimisation criteria in the classification performance.

4.5 Effect of the number of fitness functions on the MCC

An empirical study has been made in order to investigate the effect of the four optimisation criteria on the classification performance. We run EA2 five times for each optimisation criterion using Alzheimer's disease training and the Shakespearean era plays and poems datasets. Each 10 generations of the EA2, we saved the best individual of the population in a set. Then, for each solution in this set, we used the same 49 classification algorithms and calculated the average of MCC values.

The results of this experiment are reported in Figures 2 and 3. Figure 2 illustrates the results for the Shakespearean era plays and poems dataset while Figure 3 for Alzheimer's disease dataset, with the performance evaluated in terms of the MCC. The horizontal line (*x*-axis) shows the fitness value for each solution during EA2 execution every 10 generations, while the average MCC values for each of the corresponding solution are shown in the vertical line (*y*-axis).

The majority of the results indicate that minimising any one of the fitness functions under study generally results in maximising the average MCC.

5 Conclusion

This work presents a new method based on the Minimum Spanning Tree to find a solution to the feature selection problem. Accordingly, four fitness functions (based on this criterion) have been tested: *Mine*, *MineMink*, *MineMaxMSTscore*, and *MineMinkMaxMSTscore*. An evolutionary algorithm (EA) has been used to address the combinatorial optimisation problem. Two sorts of experiments have been made on two real life datasets in order to select the best performing EAs (parameters, operators, and fitness functions). The first test is used to tune the population size, maximum number of generations, mutation rate, and crossover operator. The results of this experiment have been analysed by the Wilcoxon test and accordingly the best performing operators and parameters were selected. In the second test, two EAs have been implemented to investigate whether the CM1 score can improve the solutions evolved by the EA (i.e. EA2) or not (i.e. EA1). First, the results were varied (i.e. EA1 performed better for the Shakespearean era plays and poems dataset while the EA2 attained better results for the Alzheimer's disease).

Next, we used 49 machine learning algorithms from the WEKA software package to evaluate the set of features obtained by our FSMEC. Moreover, we selected the best five performing classifiers for each of the methods to better analyse the results. The results show that the proposed method can be successfully used to reduce the number of features and increase the classification performance. In other words, the FSMEC has produced improved classification performance, when compared against all the features before applying our method. We have also compared our method with two state of the art feature selection methods on two real world datasets.

In case of the Shakespeare era plays and poems dataset the FSMEC' four optimisation criteria were managed to outperform the others in terms of MCC using 49 Weka classifiers and even though using the best five performing classifiers. Our method did not attain the best MCC results in case of the Alzheimer's disease using 49 Weka classifiers but using the best five performing classifiers our Mine*M*in*k*Max*MSTvalue* achieved the best results for both datasets. Finally, an investigation has been made to evaluate the effect of fitness function on the MCC values. Most of the results in this investigation drew an upward trend-line to increase the MCC values when the fitness value minimises.

In the future, two-way improvement will be considered. In the first direction, we will continue improving the EA by employing Memetic Algorithms (Moscato et al., 1989). In addition, we will test our method using different proximity graphs.

6 References

Arefin, A. S., Vimieiro, R., Riveros, C., Craig, H., & Moscato, P. (2014). An Information Theoretic Clustering Approach for Unveiling Authorship Affinities in Shakespearean Era Plays and Poems. *PLoS ONE, 9*(10), e111445. doi: 10.1371/journal.pone.0111445

Back, T., Fogel, D. B., & Michalewicz, Z. (1997). *Handbook of evolutionary computation*: IOP Publishing Ltd.

Berretta, R., Costa, W., & Moscato, P. (2008). Combinatorial optimization models for finding genetic signatures from gene expression datasets *Bioinformatics* (pp. 363-377): Springer.

Berretta, R., Mendes, A., & Moscato, P. (2005). *Integer programming models and algorithms for molecular classification of cancer from microarray data*. Paper presented at the Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38.

Berretta, R., Mendes, A., & Moscato, P. (2007). Selection of discriminative genes in microarray experiments using mathematical programming. *Journal of Research and Practice in Information Technology, 39*(4), 287-299.

Cotta, C., Sloper, C., & Moscato, P. (2004). Evolutionary search of thresholds for robust feature set selection: application to the analysis of microarray data *Applications of Evolutionary Computing* (pp. 21-30): Springer.

Craig, H., & Whipp, R. (2010). Old spellings, new methods: automated procedures for indeterminate linguistic data. *Literary and Linguistic Computing, 25*(1), 37-52.

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent data analysis, 1*(3), 131-156.

de Paula, M. R., Ravetti, M. G., Berretta, R., & Moscato, P. (2011). Differences in abundances of cell-signalling proteins in blood reveal novel biomarkers for early detection of clinical Alzheimer's disease. *PloS one, 6*(3), e17481.

ElAlami, M. E. (2009). A filter model for feature subset selection based on genetic algorithm. *Knowledge-Based Systems, 22*(5), 356-362.

Fogel, D. B. (2006). *Evolutionary computation: toward a new philosophy of machine intelligence* (Vol. 1): John Wiley & Sons.

Graham, R. L., & Hell, P. (1985). On the history of the minimum spanning tree problem. *Annals of the History of Computing, 7*(1), 43-57.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research, 3*, 1157-1182.

Hourani, M. a., Berretta, R., Mendes, A., & Moscato, P. (2008). Genetic signatures for a rodent model of Parkinson's disease using combinatorial optimization methods *Bioinformatics* (pp. 379-392): Springer.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence, 97*(1–2), 273-324. doi: http://dx.doi.org/10.1016/S0004-3702(97)00043-X

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem.

Proceedings of the American Mathematical society, 7(1), 48-50.

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on, 17*(4), 491-502.

Marsden, J., Budden, D., Craig, H., & Moscato, P. (2013). Language Individuation and Marker Words: Shakespeare and His Maxwell's Demon. *PloS one, 8*(6), e66813.

Moscato et al., P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report, 826*, 1989.

Prim, R. C. (1957). Shortest connection networks and some generalizations. *Bell system technical journal, 36*(6), 1389-1401.

Ravetti, M. G., & Moscato, P. (2008). Identification of a 5-protein biomarker molecular signature for predicting Alzheimer's disease. *PloS one, 3*(9), e3111.

Ravetti, M. G., Rosso, O. A., Berretta, R., & Moscato, P. (2010). Uncovering molecular biomarkers that correlate cognitive decline with the changes of hippocampus' gene expression profiles in Alzheimer's disease. *PloS one, 5*(4), e10153.

Ray, S., Britschgi, M., Herbert, C., Takeda-Uchimura, Y., Boxer, A., Blennow, K., . . . Karydas, A. (2007). Classification and prediction of clinical Alzheimer's diagnosis based on plasma signaling proteins. *Nature medicine, 13*(11), 1359-1362.

Saeys, Y., Inza, I., & Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *bioinformatics, 23*(19), 2507-2517.

Witten, I. H., Frank, E., & Hall, M. A. (2011). Data Mining: Practical Machine Learning Tools and Techniques.

Wu, Y.-L., Tang, C.-Y., Hor, M.-K., & Wu, P.-F. (2011). Feature selection using genetic algorithm and cluster validation. *Expert Systems with Applications, 38*(3), 2727-2732.

Yang, S., Yuan, L., Lai, Y.-C., Shen, X., Wonka, P., & Ye, J. (2012). *Feature grouping and selection over an undirected graph.* Paper presented at the Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining.

Yu, L., & Liu, H. (2003). *Feature selection for high-dimensional data: A fast correlation-based filter solution.* Paper presented at the ICML.

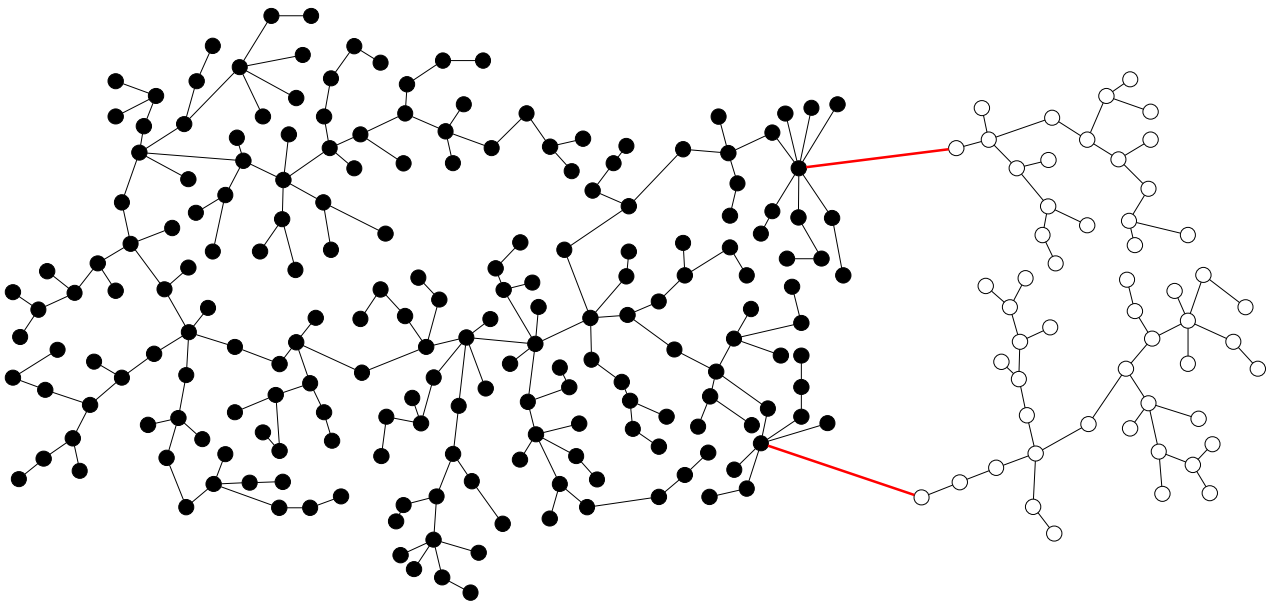


Figure 1. The MST constructed from our *MineMinkMaxMSTscore* optimisation criterion for the Shakespearean dataset using EA2. The size of resulted features is 52. The nodes represent the works, which are classified into plays (202 nodes in black) and poems (54 nodes in white). The red edges show the inter-class edges.

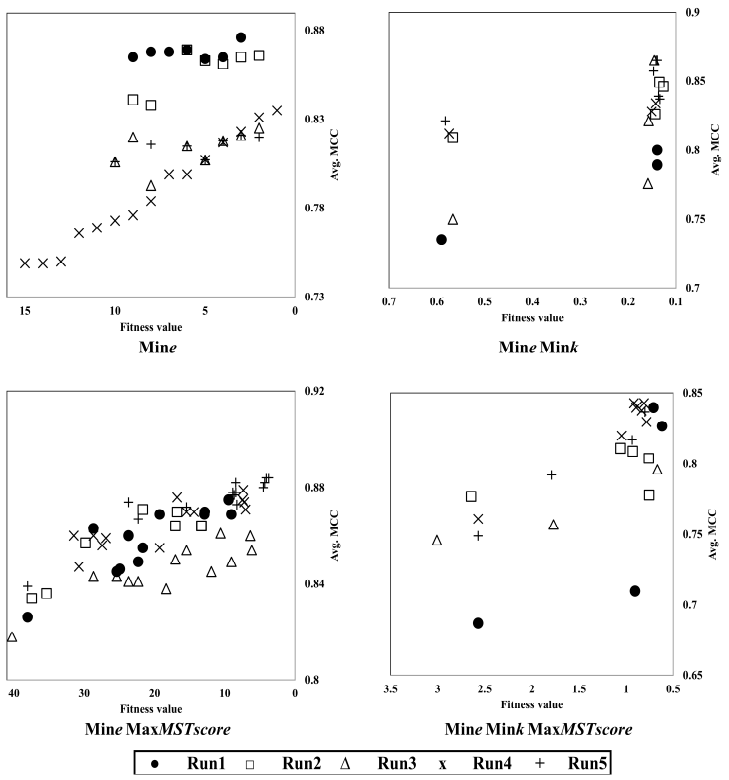


Figure 2. Effect of the Fitness functions (in x-axis): *Mine*, *MineMink*, *MineMaxMSTscore*, and *MineMinkMaxMSTscore* optimisation criteria on the average MCC value (in y-axis) in case of the Shakespearean era plays and poems dataset.

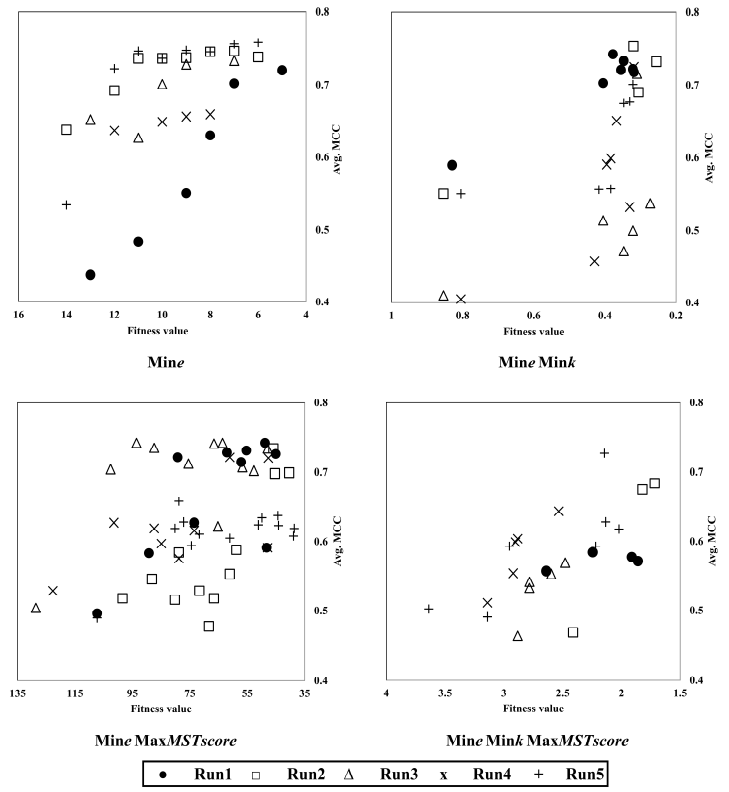


Figure 3. Effect of the Fitness functions (in x-axis): *Mine*, *MineMink*, *MineMaxMSTscore*, and *MineMinkMaxMSTscore* optimisation criteria on the average MCC value (in y-axis) in case of the Alzheimer's disease dataset.

	ALL		**(α,β)-*k*-FEATURE SET**		**CM1**		***Mine***		***MineMink***		***MineMax MSTscore***		***MineMinkMax MSTscore***	
EA							EA1		EA1		EA1		EA1	
k	220		140		40		84		44		117		110	
Classifier Name	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
Bayes_Net	0.977	0.934	0.934	0.977	0.957	0.883	0.973	0.922	**0.984**	**0.955**	0.969	0.915	**0.984**	**0.955**
Simple_Logistic	0.984	0.955	0.905	0.969	0.957	0.872	**1.000**	**1.000**	0.965	0.892	0.984	0.953	0.988	0.965
RBF_Network	0.977	0.934	**0.945**	**0.980**	0.969	0.910	0.977	0.932	0.977	0.934	0.973	0.924	0.980	0.945
SMO	**0.996**	**0.988**	0.977	0.992	0.980	0.943	0.977	0.930	0.984	0.953	0.992	0.977	0.992	0.976
S_Pegasos	0.965	0.892	0.905	0.969	0.953	0.856	0.977	0.930	0.988	0.965	0.969	0.905	**0.996**	**0.988**
IBK	0.945	0.832	0.856	0.953	0.973	0.917	0.973	0.917	**0.977**	**0.929**	0.957	0.868	0.961	0.880
LWL	0.930	0.792	0.815	0.938	0.926	0.779	0.930	0.795	0.922	0.762	0.937	0.818	**0.941**	**0.828**
Dagging	0.977	0.929	**0.942**	**0.980**	0.957	0.869	0.980	0.941	0.969	0.905	0.980	0.941	0.977	0.929
Decorate	0.961	0.883	0.954	0.984	0.965	0.895	0.973	0.918	**0.996**	**0.988**	0.984	0.954	0.977	0.930
Random_Committee	0.980	0.945	0.918	0.973	0.937	0.805	0.969	0.906	**0.988**	**0.965**	0.980	0.941	0.977	0.930
OneR	0.910	0.722	0.749	0.918	0.910	0.722	0.918	0.749	0.926	0.771	0.926	0.771	**0.937**	**0.815**
FT	0.977	0.930	0.918	0.973	0.961	0.881	**0.988**	**0.965**	0.965	0.892	0.973	0.917	**0.988**	**0.965**
LMT	0.984	0.955	0.905	0.969	0.957	0.872	**1.000**	**1.000**	0.965	0.892	0.984	0.953	0.988	0.965
NBTree	0.988	0.965	**0.977**	**0.992**	0.961	0.884	0.984	0.954	0.980	0.941	0.961	0.883	0.969	0.908
Avearge	0.968	0.904	0.907	0.969	0.955	0.863	0.973	0.919	0.970	0.910	0.969	0.909	**0.975**	**0.927**
Median	**0.977**	**0.930**	0.918	0.973	0.957	0.872	**0.977**	**0.930**	**0.977**	0.929	0.973	0.924	**0.977**	**0.930**

Table 10. Performance results for the top five WEKA models for *ALL* (all features), CM1, (α,β)-*k*-FEATURE SET, *Mine*, *MineMink*, and *MineMaxkMSTscore* and *MineMinkMaxMSTscore* in terms of accuracy and MCC achieved for Shakespeare era plays and poems dataset, using EA1.

	ALL		**(α,β)-*k*-FEATURE SET**		**CM1**		***Mine***		***MineMink***		***MineMax MSTscore***		***MineMinkMax MSTscore***	
EA							EA2		EA2		EA2		EA2	
k	220		140		40		89		53		50		52	
Classifier Name	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
Bayes_Net	**0.977**	**0.934**	0.977	0.934	0.957	0.883	0.973	0.924	0.973	0.920	**0.977**	**0.934**	0.973	0.924
Naive_Bayes	0.977	0.932	0.977	0.932	0.945	0.860	0.977	0.932	0.969	0.908	**0.988**	**0.966**	0.977	0.932
Simple_Logistic	0.984	0.955	0.969	0.905	0.957	0.872	0.957	0.869	0.953	0.859	0.977	0.929	**0.992**	**0.977**
RBF_Network	0.977	0.934	0.980	0.945	0.969	0.910	0.980	0.945	0.973	0.920	**0.988**	**0.966**	0.973	0.922
S_Pegasos	0.965	0.892	0.969	0.905	0.953	0.856	0.973	0.917	0.941	0.821	0.984	0.953	**0.996**	**0.988**
IBK	0.945	0.832	0.953	0.856	0.973	0.917	0.957	0.869	0.965	0.892	**0.980**	**0.941**	0.969	0.905
Dagging	0.977	0.929	0.980	0.942	0.957	0.869	0.980	0.941	0.953	0.855	0.977	0.929	**0.984**	**0.953**
Decorate	0.961	0.883	**0.984**	**0.954**	0.965	0.895	0.977	0.931	0.977	0.930	0.984	0.953	0.984	0.954
Logit_Boost	0.969	0.905	0.961	0.883	0.953	0.859	0.969	0.910	0.973	0.917	0.969	0.905	**0.980**	**0.941**
Hyper_Pipes	0.953	0.877	0.969	0.915	0.953	0.870	0.957	0.886	0.969	0.908	0.973	0.920	**0.977**	**0.934**
FT	0.977	0.930	0.973	0.918	0.961	0.881	0.949	0.845	0.957	0.874	0.984	0.953	**0.992**	**0.977**
LMT	0.984	0.955	0.969	0.905	0.957	0.872	0.957	0.869	0.953	0.859	0.977	0.929	**0.992**	**0.977**
NBTree	0.988	0.965	**0.992**	**0.977**	0.961	0.884	0.977	0.932	0.973	0.917	0.969	0.905	0.953	0.861
Random_Forest	0.973	0.918	0.973	0.917	0.957	0.869	**0.977**	**0.932**	0.957	0.868	0.973	0.917	0.973	0.918
Avearge	0.972	0.917	0.973	0.921	0.958	0.878	0.969	0.907	0.963	0.889	0.979	0.936	**0.980**	**0.940**
Median	0.977	0.930	0.973	0.918	0.957	0.872	0.973	0.921	0.967	0.900	0.977	0.932	**0.979**	**0.938**

Table 11. Performance results for the top five WEKA models for ALL (all features), CM1, (α,β)-*k*-FEATURE SET, *Mine*, *MineMink*, and *MineMaxMSTscore* and *MineMinkMaxMSTscore* in terms of accuracy and MCC achieved for Shakespeare era plays and poems dataset, using EA2.

	ALL		(α,β)-k-FEATURE SET		CM1		*Mine*		*MineMink*		*MineMax MSTscore*		*MineMinkMax MSTscore*	
EA							EA1		EA1		EA1		EA1	
k	120		10		40		56		23		55		42	
Classifier Name	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
Logistic	0.840	0.679	0.901	0.803	0.827	0.658	0.827	0.654	0.827	0.655	**0.914**	**0.827**	0.802	0.604
Simple_Logistic	0.889	0.779	0.926	0.852	0.914	0.829	0.889	0.779	0.889	0.779	0.914	0.827	**0.951**	**0.901**
SMO	0.864	0.728	0.951	0.902	0.926	0.852	0.889	0.778	0.926	0.852	0.951	0.902	**0.963**	**0.926**
S_Pegasos	0.852	0.708	**0.914**	**0.830**	0.852	0.703	0.827	0.655	0.901	0.804	**0.914**	**0.827**	0.840	0.685
Classification_Via_Regression	0.926	0.852	0.889	0.778	0.877	0.753	0.914	0.829	**0.938**	**0.877**	0.864	0.728	**0.938**	**0.878**
Dagging	0.877	0.760	**0.951**	**0.901**	0.914	0.829	0.840	0.685	0.827	0.658	0.864	0.731	0.864	0.741
Decorate	0.877	0.757	0.889	0.780	0.889	0.778	**0.951**	**0.902**	0.938	0.877	0.901	0.802	**0.951**	**0.902**
Logit_Boost	0.852	0.703	0.889	0.778	0.877	0.753	**0.926**	**0.852**	0.889	0.778	0.901	0.802	0.901	0.804
Multi_Class_Classifier	0.840	0.679	0.901	0.803	0.827	0.658	0.827	0.654	0.827	0.655	**0.914**	**0.827**	0.802	0.604
Rotation_Forest	0.926	0.853	**0.951**	**0.902**	0.914	0.827	0.938	0.878	0.926	0.855	0.926	0.852	0.914	0.830
NNge	0.877	0.753	0.926	0.852	0.914	0.827	**0.951**	**0.902**	0.926	0.853	0.889	0.780	0.926	0.853
LMT	0.889	0.779	0.926	0.852	0.914	0.829	0.889	0.779	0.889	0.779	0.914	0.827	**0.951**	0.901
Random_Forest	0.753	0.505	0.889	0.779	**0.914**	**0.827**	0.815	0.630	0.840	0.685	0.827	0.663	0.877	0.762
AVERAGE	0.866	0.733	**0.916**	**0.832**	0.889	0.779	0.883	0.767	0.888	0.777	0.899	0.800	0.898	0.799
Median	0.877	0.753	**0.914**	**0.830**	**0.914**	**0.827**	0.889	0.779	0.889	0.779	**0.914**	0.827	**0.914**	**0.830**

Table 12. Performance results for the top five WEKA models for ALL (all features), CM1, (α,β)-*k*-FEATURE SET, *Mine*, *MineMink*, and *MineMaxMSTscore* and *MineMinkMaxMSTscore* in terms of accuracy ACC and MCC achieved for Alzheimer's disease dataset, using EA1.

	ALL		(α,β)-k-FEATURE SET		CM1		*Mine*		*MineMink*		*MineMax MSTscore*		*MineMink MaxMSTscore*	
EA							EA2		EA2		EA2		EA2	
k	120		10		40		41		31		27		15	
Classifier Name	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
Bayes_Net	0.852	0.708	0.889	0.780	0.889	0.780	0.852	0.708	0.877	0.757	**0.901**	**0.802**	**0.901**	**0.804**
Naive_Bayes	0.827	0.654	**0.926**	**0.852**	0.877	0.754	0.889	0.778	0.864	0.728	0.877	0.753	0.901	0.802
Simple_Logistic	0.889	0.779	0.926	0.852	0.914	0.829	**0.938**	**0.877**	**0.938**	**0.878**	0.901	0.805	0.877	0.753
SMO	0.864	0.728	**0.951**	**0.902**	0.926	0.852	0.901	0.802	0.914	0.827	**0.951**	**0.902**	0.926	0.855
IBK	0.914	0.827	0.901	0.802	0.827	0.654	0.877	0.756	0.889	0.779	0.914	0.829	**0.951**	**0.905**
LWL	0.889	0.783	0.889	0.783	0.889	0.783	0.864	0.729	**0.914**	**0.833**	0.889	0.788	0.901	0.810
Classification_Via_Regression	0.926	0.852	0.889	0.778	0.877	0.753	0.889	0.778	**0.938**	**0.877**	0.877	0.753	0.877	0.753
Dagging	0.877	0.760	**0.951**	**0.901**	0.914	0.829	0.901	0.807	0.901	0.807	0.901	0.802	0.926	0.852
Decorate	0.877	0.757	0.889	0.780	0.889	0.778	0.926	0.852	0.926	0.852	**0.938**	**0.877**	0.914	0.827
Filtered_Classifier	0.827	0.654	0.827	0.654	**0.840**	**0.679**	0.815	0.632	0.778	0.556	0.815	0.63	0.815	0.630
Logit_Boost	0.852	0.703	0.889	0.778	0.877	0.753	**0.901**	**0.802**	0.877	0.754	0.877	0.753	**0.901**	**0.802**
Random_Committee	0.84	0.687	0.914	0.833	0.889	0.783	0.889	0.780	0.889	0.791	0.864	0.728	**0.938**	**0.877**
Random_Sub_Space	**0.914**	**0.827**	0.877	0.753	0.901	0.802	0.901	0.805	0.840	0.678	**0.914**	**0.827**	0.901	0.803
Rotation_Forest	0.926	0.853	**0.951**	**0.902**	0.914	0.827	0.901	0.802	0.926	0.852	0.938	0.877	0.926	0.852
Decision_Table	0.815	0.630	0.778	0.559	0.802	0.609	**0.827**	**0.654**	0.802	0.604	0.753	0.506	0.815	0.630
NNge	0.877	0.753	**0.926**	**0.852**	0.914	0.827	0.889	0.778	0.914	0.827	**0.926**	**0.852**	**0.926**	**0.852**
ADTree	0.864	0.728	0.852	0.703	0.864	0.728	0.864	0.728	0.852	0.708	**0.877**	**0.753**	**0.877**	**0.753**
BFTree	0.915	0.830	0.915	0.830	0.902	0.804	0.915	0.855	0.864	0.728	0.927	0.854	**0.940**	**0.879**
J48graft	0.864	0.729	**0.901**	0.802	0.864	0.729	0.84	0.678	0.864	0.728	0.889	0.779	**0.901**	**0.803**
AVERAGE	0.875	0.751	0.889	0.779	0.881	0.762	0.881	0.765	0.875	0.751	0.884	0.769	**0.897**	**0.793**
Median	0.877	0.753	0.889	0.780	0.889	0.778	0.889	0.780	0.877	0.754	0.889	0.779	**0.901**	**0.803**

Table 13. Performance results for the top five WEKA models for ALL (all features), CM1, (α,β)-*k*-FEATURE SET, *Mine*, *MineMink*, and *MineMaxMSTscore* and *MineMinkMaxMSTscore* in terms of accuracy ACC and MCC achieved for Alzheimer's disease dataset, using EA2.

Genetic Programming for Extracting Edge Features Using Two Blocks

Wenlong Fu[1] **Mengjie Zhang**[2] **Mark Johnston**[3]

[1] School of Mathematics, Statistics and Operations Research
Victoria University of Wellington, PO Box 600, Wellington, New Zealand
Email: wenlong.fu@msor.vuw.ac.nz

[2] School of Engineering and Computer Science
Victoria University of Wellington, PO Box 600, Wellington, New Zealand
Email: mengjie.zhang@vuw.ac.nz

[3] Institute of Science and the Environment
University of Worcester, Henwick Grove, Worcester WR2 6AJ, United Kingdom
Email: m.johnston@worc.ac.uk

Abstract

In low-level edge detection, single pixels have been popularly used to extract edge features. However, the extracted edge features might not have good ability to effectively mark edge points on images with noise or/and textures. Single pixels can be extracted based on a local window. To automatically search pixels to extract edge features using Genetic Programming, search operators based on single pixels and single blocks of pixels have been proposed. Single blocks of pixels can be used to improve detection performance on natural images, but the computational cost is high. In this paper, to reduce the computational cost of using blocks of pixels, a new search operator based on two blocks of pixels is proposed. The experiment results show that the proposed search operator can effectively reduce computational cost on evolved edge detectors, remaining good detection performance. *Keywords:* Genetic Programming, Edge Detection, Feature Extraction

1 Introduction

Edge detection is a subjective task, and has been investigated more than three decades (Kunt 1982, Papari & Petkov 2011). In general, there are three stages in edge detection: pre-processing, feature extraction and post-processing. Feature extraction is an import stage. Techniques for pre-processing, such as filtering, and post-processing, such as thinning edges, can be routinely cooperated with feature extraction approaches (Martin et al. 2004, Moreno et al. 2009*a*).

In low-level edge detection, pixels from local windows are extracted to construct edge features for discriminating pixels as edge points or non-edge points. Window size is a trade-off between detection accuracy and localisation (Basu 2002). To avoid using windows, search operations need to be based on full individual images. Since edge features are implicit, there are no generic approaches to extracting edge features from images. The intensity value of a single pixel is often not sufficient to discriminate that pixel as an edge point or a non-edge point. To automatically search for neighbouring pixels for constructing edge features, Genetic Programming (GP) has been applied to edge detection (Harris & Buxton 1996, Poli 1996, Fu et al. 2011*b*).

However, it is difficult to suppress textures using single pixels (Papari & Petkov 2011, Ganesan & Bhattacharyya 1997). From the results in (Harris & Buxton 1996, Poli 1996, Fu et al. 2011*b*), detected results are also affected by noise and some textures. From human observation, boundary information requires a reasonable area surrounding the boundary to be recognised, so features extracted from small areas (based on blocks) are useful to detect edges via filtering noise and textures (Martin et al. 2004, Papari & Petkov 2011). To suppress noise and textures, single blocks of pixels have been proposed to extract edge features by GP (Fu et al. 2012*b*). From the results, it is effective to use single blocks of pixels to improve detection accuracy. However, it is found that the computational cost of the evolved edge detectors for detecting images is obviously increased, comparing to the evolved edge detector based on single pixels only. It is worth further investigating how to reduce computational cost and remain detection accuracy when GP utilises search operators based on blocks of pixels to extract edge features.

1.1 Goals

The goal of this paper is to reduce the computational cost of GP using blocks of pixels to evolve edge detec-

tors to extract edge features while remaining detection accuracy. Little prior domain edge knowledge, only using training images and their ground truth, is provided for automatic edge feature extraction. In the existing work (Fu et al. 2012*b*), single blocks of pixels were combined to construct GP edge detectors. There are a large number of candidate combinations on the proposed blocks. However, there are also many combinations which are not useful for extracting edge information. To reduce the number of candidates and remove some useless combinations, a new search operator based on two blocks (a simple combination of two single blocks) is proposed. Specifically, the following research objectives will be investigated.

- Whether constructing search operators using two blocks of pixels is better than using a single block of pixels to evolve low-level edge detectors, in terms of detection accuracy.
- Whether constructing search operators using two blocks of pixels is faster than using a single block of pixels to evolve low-level edge detectors.

Different from the existing work (Fu et al. 2012*b*), we further investigate the searching (pixels) behaviour from GP. We utilise different combinations of search operators to evolve edge detectors.

1.2 Organisation

In the remainder of this paper, Section 2 briefly describes edge detection background and presents existing work in edge detection using GP with discussions. Section 3 introduces the GP system using blocks of pixels. Section 4 describes the experiment design. After the experiment results with discussions are presented in Section 5, conclusions are drawn and potential future investigation is suggested in Section 6.

2 Background

Relevant background on edge detection is briefly described in this section. Then, we give a brief survey on GP for edge detection.

2.1 Edge Detection

Edge features are used to discriminate pixels as edge points or not. In general, in the pre-processing stage, techniques are designed to filter noise and suppress textures; and in the post-processing stage, techniques are introduced to thin edges, remove standalone edge points, and link broken edges (Papari & Petkov 2011). The stages of pre-processing and post-processing are commonly employed to different edge detectors (Lopez-Molina et al. 2013). Therefore, edge features extracted by edge detectors usually determine the detectors' performance (Moreno et al. 2009*a*). In (Martin et al. 2004), different boundary detectors are trained by combining the same set of basic features, and those learnt detectors have similar results.

From the way of extracting edge features, we can extract edge features from low-level, mid-level and high-level. Generally, image context or object knowledge can be used to extract mid-level or/and high-level edge features. For instance, the Gestalt Laws (Papari & Petkov 2008) and the objects' similarity (Bai et al. 2008) has been used for edge feature extraction. Also, high-level features can be combined from low-level features. Low-level features are directly extracted from raw pixels in a local area, and they do not consider the image content.

In the early stage of the edge detection development, the edge feature extraction approaches mainly focused on low-level edge feature extraction (Ganesan & Bhattacharyya 1997, Kunt 1982). "Edge" is considered discontinuities on one-dimensional or two-dimensional signals (Basu 2002). Methods using differentiation techniques are popular to obtain edge responses on discontinuities. Edge detectors have utilised the gradients as edge responses, such as the Prewitt and Sobel edge detectors (Ganesan & Bhattacharyya 1997). To filter noise, methods via combining pre-processing (Gaussian filtering) and differentiation are employed to extract edge features, such as the Canny edge detector (Canny 1986). From retinal receptive fields using Gaussian functions, Gaussian-based edge detectors have been widely developed (Basu 2002). A common computational framework, using single raw pixels in a small local window to extract edge features, was suggested for edge detection (Ganesan & Bhattacharyya 1997).

Window size is a trade-off between localisation and noise rejection. A small window can be used to find details of edges, but the detected results are easily affected by noise and textures (Papari & Petkov 2011, Basu 2002, Bertero et al. 1988). From (Bertero et al. 1988), it is shown that the computation of the derivatives of a digital image is an ill-posed problem because there are no unique solutions for derivatives. When using a window to calculate edge responses, a big window has problems for edge localisation, a small window has problems of rejecting noise.

To reject noise and suppress textures, blocks of pixels are used to extract edge features. The difference of two blocks of pixels are used to indicate edge responses, such as dissimilarity indication from statistical test techniques (Lim & Jang 2002). In (Lim & Jang 2002), different two-sample tests are employed to extract edge features. Besides directly using the intensities of pixels, the intermediate results of blocks of pixels are employed to suppress noise and textures, such as the surround suppression technique (Grigorescu et al. 2004). In the surround suppression technique, gradients and the outputs of an edge detector using Difference of Gaussians (DoG (Marr & Hildreth 1980)) are combined together for edge detection.

2.2 Related Work for Edge Detection Using GP

This subsection briefly surveys the related work for edge detection using GP. From the view of employing knowledge in GP for edge detection, GP approaches for edge detection can be categorised as low-level extraction methods (little edge knowledge) and combination methods with some edge knowledge.

2.2.1 Low-level Extraction

When only little edge knowledge is used in designed tasks of GP for evolving edge detectors, it is expected that the outputs of the evolved edge detectors using pixels have similar outputs from human design (marked edges or existing designed edge detectors). In the training stage, generally, the desired outputs from human design are used. To extract edge features, search operators using single pixels (shifting functions) have been employed for evolving detectors based on full images (Poli 1996, Fu et al. 2011*b*, 2012*c*,*b*). Terminals including bits of the pixels in a 4×4 window were used to evolve edge detectors which were used to design digital circuits for edge detection (Golonek et al. 2006). Also, edge responses are generated based on existing knowledge, and then they are approximated by GP. In (Harris & Buxton 1996), one-dimensional step signals and the relevant edge responds were designed. GP was utilised to evolve formulae to fit these responses. These formulae were utilised to design one-dimensional edge detectors. Responses from existing edge detectors are also approximated by GP, such as the outputs of the Sobel and Canny edge detectors (Ebner 1997, Hollingworth et al. 1999, Harding & Banzhaf 2008).

The advantage of GP evolving edge detectors with little knowledge is that the training stage is independent of the background of edge detection. Different from human design, new programs can be found by GP. Based on desired outputs, GP is flexible to evolve different edge detectors from specific tasks. However, the search space for these techniques is too large, how to efficiently and effectively search pixels to construct edge detectors is a remaining issue.

2.2.2 Extraction using Some Edge Knowledge

In order to efficiently construct edge detectors, some simple edge knowledge has been used. From analysing edges and non-edges in a local area, Zhang and Rockett (Zhang & Rockett 2005) summarised edge patterns and non-edge patterns with 13×13 windows, and then edge detectors using pixels from a 13×13 window as terminals were evolved. Some image operators also have been used to construct edge detectors. Morphological operators erosion and dilation as the terminal set were employed to evolve morphological edge detectors (Quintana et al. 2006, Wang & Tan 2010). Gaussian operators and other image operators were used to evolve a high-level feature by GP, and this edge feature was combined with other edge features as a trained logistic regression classifier to detect object boundaries (Kadar et al. 2009). Additionally, three basic features were employed to combined composite features by GP (Fu et al. 2012*a*, 2013*b*,*a*).

The advantage of using edge knowledge to evolve new edge detectors is that the performance of these evolved edge detectors is not too low. However, these methods are dependent on the used knowledge. When only training images and their ground truth are provided, how to effectively and efficiently search pixels to extract edge features still needs to be investigated.

3 GP System Based on Full Images

This section describes the new GP system modified from the existing work (Fu et al. 2012*b*).

3.1 Sets of Terminals and Functions

This GP system uses a full image as a terminal. In general, constants are helpful for constructing GP programs in many applications (Koza et al. 1999, Poli et al. 2008). Besides the full image I, the terminal set contains random constants. The range of random constants rnd is from -10 to 10 based on initial experiments.

For the function set, the common arithmetic operators include the addition ($+$), subtraction ($-$), multiplication ($*$), division ($\div$), absolute (abs), square ($square$) and square root ($sqrt$). All functions work on each element of a matrix, such as each pixel of the input image I. The $+, -, *$, abs, $square$ have their usual meanings. The square root function $sqrt$ is protected, which produces a result of 0 for negative inputs. Division $\div$ is also protected, producing a result of 1 for a 0 divisor.

In existing work, there are two proposed search operators as functions: one search operator $s_{n,m}$ (Fu et al. 2011*a*) based on a single pixel and the other search operator $block_{t,l,w,d}$ (Fu et al. 2012*b*) based on a single block of pixels.

3.1.1 Function $s_{n,m}$

Function $s_{n,m}$ shifts its argument (a single two-dimensional matrix input) by n columns and m rows. If n is positive, a right shifting operation performs on the input, otherwise a left shifting operation performs on its argument. If m is positive, the two-dimensional input shifts down, otherwise shifts up. Its argument can come from image I or an intermediate result of a subtree, $sub(I)$, constructed by the GP system.

Note that if the two-dimensional input is rnd, rnd is considered as a two-dimensional matrix with its elements being equal to a single random constant. The shifting operation performs on the bits of rnd, and its value is multiplied by 2^{n+m} so that the GP system can generate a large range of different constants. Here n and m are randomly selected from $\{-2, -1, 0, 1, 2\}$. When a new shifting function $s_{n,m}$ is generated and

11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55
11	11	20	30	40	55	55

(a) before calling $s_{-1,0}$

11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55
11	20	30	40	55	55	55

(b) after calling $s_{-1,0}$

Figure 1: Example two-dimensional matrix and its result after calling $s_{-1,0}$.

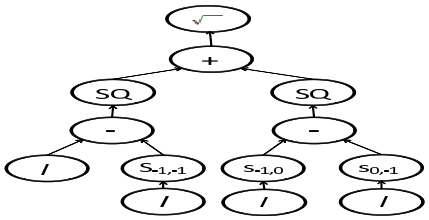


Figure 2: The 2×2 Roberts detector constructed with GP. Nodes "$\sqrt{\ }$" and "SQ" are functions *sqrt* and *square*, respectively.

its argument is image I, each pixel or one of its neighbours in a 5×5 window has equal probability to be selected.

Figure 1 (a) shows an example of a small two-dimensional matrix (for a ramp edge), and (b) is its result after calling a shifting function $s_{-1,0}$. Note that the last column of the shifted result is filled by the nearest element in the matrix. Via using shifting functions to implicitly search neighbours, neighbours of each discriminated pixel can be combined for constructing edge features with common operators.

It is possible for this GP system to generate some existing edge detectors. For instance, the 2×2 window Roberts detector (see Equation (3)) (Ganesan & Bhattacharyya 1997) is represented by the GP edge detector $GE_{Roberts}$, which is given by Equation (6), where $\circledast$ is a convolution operator. In order to employ the neighbours of each discriminated pixel (including each discriminated pixel itself) used in the Roberts detector, functions $s_{1,1}$, $s_{1,0}$ and $s_{0,1}$ are used to select the pixels around each discriminated pixel. Figure 2 presents a tree representation of the 2×2 window Roberts filter based on full image I.

$$R_{Roberts,x} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \circledast I \quad (1)$$

$$R_{Roberts,y} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \circledast I \quad (2)$$

$$R_{Roberts} = \sqrt{R^2_{Roberts,x} + R^2_{Roberts,y}} \quad (3)$$

$$GE_{Roberts,x} = I - s_{-1,-1}(I) \quad (4)$$

$$GE_{Roberts,y} = s_{-1,0}(I) - s_{0,-1}(I) \quad (5)$$

$$GE_{Roberts} = \sqrt{GE^2_{Roberts,x} + GE^2_{Roberts,y}} \quad (6)$$

3.1.2 Function $block_{t,l,w,d}$

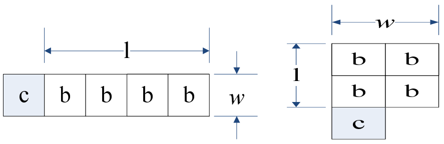
In general, it is a difficult task for GP to evolve a good edge detector using only single pixels to remove influence of noise and textures. The edge detector employs the intensity level of each single pixel, and it is sensitive to the pixel intensities. For instance, a pixel, which is not an edge point and is not affected by noise or texture, is easily marked as a non-edge point by an edge detector, such as the Sobel edge detector (Ganesan & Bhattacharyya 1997). After adding noise to the pixel, the edge detector might detect it as an edge point. In order to suppress noise and textures, information for an edge from a local area needs to be used. In general, a local area for indicating a pixel as an edge point includes a set of its neighbours.

If only $s_{n,m}$ is used to filter noise and textures, filters as subtrees need to be constructed, and then a program is constructed by these subtrees. However, the sizes of GP trees including filters are normally very large. If a large size tree is allowed in the GP system, the search space (tree size) will be exponentially increased, which leads to an increase in the computational cost. For example, the size of a full binary tree from depth $dp - 1$ to dp is increased by $O(2^{dp-1})$. Additionally, some existing filters cannot be constructed by the simple GP system, such as a median filter (Bovik et al. 1987).

To suppress textures and reject noise, existing work employs a set of pixels. The dissimilarity of two blocks of pixels was indicated by statistical approaches (Lim & Jang 2002). A surround suppression technique utilised blocks of pixels' intermediate results (gradients) to remove some texture responses (Grigorescu et al. 2004). To simulate this idea, new approaches to using a set of pixels or their intermediate results need to be developed so that the GP system has some ability to search blocks of pixels to reject noise and suppress textures.

Search operator $block_{t,l,w,d}$ is designed to find blocks of pixels to extract edge features so that the GP system can effectively construct edge detectors with some ability to filter noise and textures. The search operator $block_{t,l,w,d}$ includes approaches (t) to transforming a block of pixels to a single variable (such as the mean of intensities of all pixels in the block), the block size parameters l (the length of the block) and w (the width of the block), and the directional position d (where the block is located around a discriminated pixel). The argument of the search operator is a two-dimensional matrix, which is image I or an intermediate result from a subtree $sub(I)$. Figures 3 (a) and (b) describe two examples for the blocks of pixels specified by the search operator. Here, a block of pixels ("bbbb") is specified relative to the discriminated pixel ("c"). In Figure 3 (a), the parameters for the block are $l = 4$, $w = 1$ and $d =$"right"; and in Figure 3 (b), the parameters for the block are $l = 2$, $w = 2$ and $d =$"up".

The parameter t is used to transform a set of pixels to a single variable. The purpose of using a block of



(a) Right (b) Up

Figure 3: Two example blocks of pixels specified by search operator $block_{t,l,w,d}$.

pixels is to indicate some special characteristics from the set. Mean and standard deviation are commonly used to summarise a group of data. Therefore, mean ($t = 0$) and standard deviation ($t = 1$) are employed in the search operator $block_{t,l,w,d}$.

The search operator $block_{t,l,w,d}$ is used to find a set of pixels around a discriminated pixel, and it has no ability to move to a new position. From the comparison between $s_{n,m}$ and $block_{t,l,w,d}$, the latter only searches a limited area for constructing edge detectors. Without $s_{n,m}$, the pixels only found by the search operator $block_{t,l,w,d}$ are dependent on its parameters l, w and d. However, $s_{n,m}$ can search pixels which are far away from the relevant discriminated pixel, and the ability to find pixels is dependent on not only its parameters n and m, but also the maximum depth of a GP tree.

In contrast to the existing approaches to utilising blocks of pixels, the search operator $block_{t,l,w,d}$ provides four distinguished characteristics. Firstly, an unfixed size block (used to find a set of pixels) is different from the common way using a fixed block (Dollar et al. 2006, Martin et al. 2004, Papari & Petkov 2011). It is possible that GP can find a suitable size block to construct edge detectors, which decreases the computational cost but does not affect the ability to detect edges. Secondly, flexible positions to find a block of pixels are used after calling the search operator $s_{n,m}$. In a ramp edge or a stair edge, the closest neighbours might affect the detection results because of its discontinuity. Using a combination of $s_{n,m}$ and $block_{t,l,w,d}$, such as $block_{t,l,w,d}(s_{n,m}(I))$, possibly avoids this influence. Thirdly, different directional positions are used together to construct edge detectors so that the directional calculation is avoided. Lastly, a flexible input in $block_{t,l,w,d}$ comes from original image I or an intermediate result after some processing on image I (subtree). After calling a subtree $sub(I)$ for filtering noise or textures, $block_{t,l,w,d}(sub(I))$ might help improve detection performance.

Figure 4 shows an example of searching blocks of pixels for discriminating the centre pixel (blue) in four different directions. Here, the neighbours with the same colour are in the same block. There are two major areas with intensities 11 and 55 respectively. Three columns with intensities 20, 30 and 40 are the critical region between both major areas. From existing methods, a comparison between the right block and the left block or between the right block and the bottom block is easy to find the ramp edge. For instance, the mean of pixel intensities from different blocks can be compared. The difference of the means of pixel intensities in the left block and the right block is $\frac{11*4+20*2}{6} - \frac{40+55*3}{4} = -37.25$. Here, out of the right area is considered as the area with intensity 55. To discriminate the right neighbour (at the sixth column), all blocks move right by a pixel, and the right block has four "55" (three in the figure and one being out of the right area). The differences of the means for the right neighbours (the sixth, seventh and eighth columns) are -34.67 ($\frac{11*2+20*2+30*2}{6} - \frac{55*4}{4}$), -25.00, and -13.33, respectively. The differences of the means between the left block and the up block for the pixel and its relative right neighbours are -21.00, -27.17, -25.00 and -13.33, respectively. Since the means from the left block and the up block are close, the relative response on the ramp edge is thick.

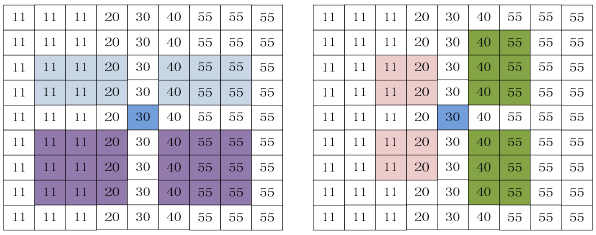
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55
11	11	11	20	30	40	55	55	55

Figure 4: Example of searching neighbours by $block_{t,l,w,d}$.

In Figure 4, if only using $s_{n,m}$, a high threshold 14 might be used to detect the boundary between the seventh and sixth columns based on the difference of two columns. The difference of the intensities between the seventh and sixth columns is 15 ($55 - 40$). However, if the intensities of the pixels at the last three columns are changed to 50 and the intensities of the pixels at the first three columns are changed to 10, the responses from the third columns to seventh columns are the same ($30 - 20 = 40 - 30 = 50 - 40 = 10$).

3.2 New Search Function

While $block_{t,l,w,d}$ can be used to search good edge features as discussed previously, the search space is huge. To reduce the search space of using blocks of pixels, two blocks with the same size are combined by common operators, and a new search operator is developed for combining two blocks of pixels. In edge detection, edge points are often located at boundaries between "different" areas. The detection mainly focuses on finding differences. Subtraction $-$ and division $\div$ are usually used for indicating differences from numerics. Therefore, we employ $-$ and $\div$ in the new search operator tb_{j,t,l,w,d_1,d_2} ($j =$'$-$' or $j =$'$\div$') using two blocks of pixels. Equations (7) and (8) give the definition of the search operator tb_{j,t,l,w,d_1,d_2} for $j =$'$-$' and $j =$'$\div$', respectively. Here, ε is a small positive constant, d_1 and d_2 are different directions,



(a) horizontal direction (b) vertical direction

Figure 5: Examples of using two blocks of pixels.

and $tb_{\div,t,l,w,d_1,d_2}$ requires that $block_{t,l,w,d_2}$ must be larger than or equal to 0.

$$tb_{-,t,l,w,d_1,d_2} = block_{t,l,w,d_1} - block_{t,l,w,d_2} \quad (7)$$

$$tb_{\div,t,l,w,d_1,d_2} = \frac{block_{t,l,w,d_1}}{block_{t,l,w,d_2} + \varepsilon} \quad (8)$$

Figures 5 (a) and (b) show combinations of two blocks of pixels used for tb_{j,t,l,w,d_1,d_2}. To discriminate the centre pixel, four combinations of two blocks are shown in Figure 5. Here, the same colour blocks are used in the same search operator tb_{j,t,l,w,d_1,d_2}. In this study, the combinations of d_1 and d_2 are limited to the four different combinations: left up and right up, left bottom and right bottom, left up and left bottom, and right up and right bottom.

Using tb_{j,t,l,w,d_1,d_2} is approximately considered as a small subset of using $block_{t,l,w,d}$. The aim of proposing tb_{j,t,l,w,d_1,d_2} is efficiently reducing the space for searching blocks of pixels while remaining detection accuracy. Note that tb_{j,t,l,w,d_1,d_2} is not randomly selected from combinations of $block_{t,l,w,d}$. There are two elements of prior knowledge used in tb_{j,t,l,w,d_1,d_2}, namely the comparison using difference and the approach (t) frequently used in the evolved edge detectors when $block_{t,l,w,d}$ is used. Additionally, when a random constant rnd is used as an argument in $block_{t,l,w,d}$ or tb_{j,t,l,w,d_1,d_2}, the return value is the same as rnd.

3.3 Fitness Function

The class label for edge detection only has "edge point" or "non-edge point" here, and the main class is "edge point". For the output of a program, 0 is employed as the threshold for discriminating a pixel as an edge point (larger than 0) or a non-edge point (less than or equal to 0), and all images use the pixel intensities. Since only low-level edge detectors are evolved in this study and searching operators are employed to find features to construct GP detectors (namely this study only focuses on the way to extract feature for edge detection), the output is directly evaluated without post-processing, following (Moreno et al. 2009*b*).

The F-measure technique (Martin et al. 2004) has been widely used for evaluating edge detectors' performance. The F-measure technique is a combination of recall and precision with a parameter factor α (from 0 to 1). Here, recall is the number of correctly predicted edge points as a proportion of the total number of pixels on the true edges, and precision is the number of correctly predicted edge points as a proportion of the total number of predicted edge points. The fitness function using F-measure technique is given in Equation (9). To balance recall and precision, α is set to 0.5 in this paper.

(a) image 23080 (b) GT of image 23080

(c) sampled result (d) GT of the sampling

Figure 6: Example training image 23080, its ground truth (GT) and its sampled result of size 125 × 125 pixels.

$$F = \frac{\text{recall} * \text{precision}}{\alpha * \text{recall} + (1-\alpha) * \text{precision}} \quad (9)$$

4 Experiment Design

This section describes an image dataset and the settings for the GP system.

4.1 Image Dataset

To evolve subjective edge detectors, the Berkeley Segmentation Dataset (BSD) (Martin et al. 2004) including natural images with ground truth provided are chosen. To sample training images as the training data, a small training dataset is employed. This image data set contains 20 BSD training images. The 20 images are images 42078, 106020, 68077, 23080, 216053, 61060, 41004, 113044, 134008, 161062, 163014, 189011, 207056, 236017, 249061, 253036, 271031, 299091,311081, and 385028. For each image, a subimage of size 125 × 125 pixels are randomly extracted. Figure 6 shows training image 23080 and its randomly sampled subimage of size 125 × 125 pixels. The test dataset is the 100 BSD full sized (481 × 321) test images.

4.2 Sets of Terminals and Functions

To investigate the influence of the function set, three settings (search operators) are used different combinations of the three search operators for the GP system. The first setting is from the existing work (Fu

et al. 2012*b*), namely including search functions $s_{n,m}$ and $block_{t,l,w,d}$. We use $Set_{s,b}$ to indicate the GP system including $s_{n,m}$ and $block_{t,l,w,d}$. The second setting uses the proposed search function tb_{j,t,l,w,d_1,d_2} and function $s_{n,m}$. we use $Set_{s,tb}$ to indicate the GP system including $s_{n,m}$ and tb_{j,t,l,w,d_1,d_2}. Set_s is used to indicate the GP system only using $s_{n,m}$. Note that the other functions and terminals are used with these three settings.

4.3 Parameter Settings

The parameter values for $block_{t,l,w,d}$ are: $t = 0$ for the mean or $t = 1$ for the standard deviation, l is from 3 to 7, w is from 1 to 7, and d is one of left, right, up or down direction. From the initial experiment results, standard deviation has higher occurrences than mean in the evolved edge detectors, so only standard deviation is used in tb_{j,t,l,w,d_1,d_2}. The parameter values for GP are: population size 800; maximum depth (of a program) 10; maximum generation 200; and probabilities for mutation 0.15, crossover 0.80 and elitism (replication) 0.05. These values are chosen based on common settings and initial experiments. Each GP experiment for each setting is repeated for 30 independent runs.

5 Results and Discussions

In order to compare this GP system with the existing methods using a moving window, a common system (Zhang & Rockett 2005) uses all pixels in a 5×5 window as terminals, and employs all functions in the proposed GP system, except for the three search operators. Note that n and m in $s_{n,m}$ are from -2 to 2. Using a single $s_{n,m}$ is equal to selecting a pixel from a 5×5 window. Therefore, the 5×5 window is chosen in the common system. The common system detects edge points based on raw pixels, not full images. In the common system, fixed neighbours are given as terminals. Like the Sobel edge detector, the common system moves the window pixel by pixel to detect edge points. $Set_{5\times5}$ is used to indicate the evolved edge detectors using the 5×5 window to detect pixels one by one. Note that this paper mainly compares the proposed GP system with the GP system in (Fu et al. 2012*b*).

5.1 Overall Results

Table 1 gives the test performance of the means and standard deviations of F, the maximum F of the evolved edge detectors, and the means and standard deviations of the test time on the 100 BSD test images. Here the test time on each image is on a single machine with CPU 3.1 GHz based on an implementation in C++. From Table 1, the highest mean of F comes from the edge detectors evolved by GP using $Set_{s,b}$ (the search operators $s_{n,m}$ and $block_{t,l,w,d}$). The common method using the 5×5 moving window ($Set_{5\times5}$) has the lowest mean of F. From the comparison of the best evolved edge detector (maximum F) from each setting, $Set_{s,b}$ has the best edge detector with $F = 0.3170$ in Table 1, which is increased by 21% ($\frac{0.3170-0.2619}{0.2619}$), compared with the best edge detector from $Set_{5\times5}$. The F values of the best edge detectors from GP using search operators with a single block of pixels are higher than 0.3. However, the F values of the best evolved edge detectors using single pixels are lower than 0.3.

For the computational cost, the evolved edge detectors using single pixels (less than 0.05 second) are much faster than the evolved edge detectors including the search operators with blocks of pixels (their average being longer than 0.1 second). Compared with the evolved edge detectors from $Set_{s,b}$, the evolved edge detectors from $Set_{s,tb}$ (using the search operator tb_{j,t,l,w,d_1,d_2}) obviously reduce the computational cost of detecting images. However, the average time of these edge detectors from $Set_{s,tb}$ is longer than the average time of the edge detectors from Set_s. In $Set_{s,tb}$, 18 of the 30 evolved edge detectors take obviously less than 0.1 second to detect a BSD image. Therefore, the search operator using two blocks of pixels effectively reduces the computational cost, compared to those using a single block of pixels. Note that a GP edge detector executes from the bottom up, which leads to redundancies existing in the calculations, such as the same subtrees. This is a potential reason that the computational cost in the evolved edge detectors using blocks of pixels is heavy.

The standard deviations in these evolved edge detectors using blocks of pixels (on detecting time) are large. If using $block_{t,l,w,d}$, the detecting time for evolved edge detectors is varied. From the detecting times in Table 1, it is found that combining tb_{j,t,l,w,d_1,d_2} with $s_{n,m}$ can improve the stability of the test times when blocks of pixels are used.

Note that the standard deviations of F in Table 1 are small, although the test times are varied. Since using $block_{t,l,w,d}$ is not stable for the detecting cost, all evolved edge detectors using $block_{t,l,w,d}$ cannot be considered as heavy computational cost detectors. In these experiments, some evolved edge detectors using $block_{t,l,w,d}$ take less than 0.1 second to detect a BSD test image. For instance, one evolved edge detector from $Set_{s,b}$ with $F = 0.2894$ only takes 0.04 seconds for detecting a BSD image.

5.2 Statistical Comparisons

Table 2 gives p-values using two-sample t-tests for each pair of settings. Here, the first group comes from the relevant setting in the first column and the second group comes from the relevant setting in the first row; $\downarrow$ indicates that the first group (in the first column) is significantly worse than second group (in the first row); and $\uparrow$ indicates that the first group is significantly better than the second group when using the significance level 0.05. Note that each setting has 30 independent runs. From the table, the evolved edge

Table 1: Test performance (mean ± standard deviation) of the GP edge detectors on the 100 BSD test images. Note that the time is for testing one BSD image.

	F	maximum (F)	time (seconds)
$Set_{5\times5}$	0.2563 ± 0.0046	0.2619	0.0403 ± 0.0140
Set_s	0.2632 ± 0.0098	0.2807	0.0213 ± 0.0093
$Set_{s,b}$	0.2953 ± 0.0161	0.3170	0.2971 ± 0.1939
$Set_{s,tb}$	0.2895 ± 0.0103	0.3063	0.1031 ± 0.0566

Table 2: Statistical p-values (two-samples t-tests) among constructed GP edge detectors from $Set_{5\times5}$, Set_s, $Set_{s,b}$, $Set_{s,tb}$ on the 100 BSD test images.

	Set_s	$Set_{s,b}$	$Set_{s,tb}$
$Set_{5\times5}$	$0.0011 \downarrow$	$0.0000 \downarrow$	$0.0000 \downarrow$
Set_s		$0.0000 \downarrow$	$0.0000 \downarrow$
$Set_{s,b}$			0.1106

detectors from Set_s are significantly better than the evolved edge detectors from $Set_{5\times5}$. It seems that this GP system using the search operator $s_{n,m}$ is better than the common approach to using the 5×5 moving window. From the p-values of comparing Set_s with the other settings (using blocks of pixels) respectively, the evolved edge detectors using blocks of pixels are significantly better than the evolved edge detectors using single pixels only. There is non-significant difference between $Set_{s,tb}$ and $Set_{s,b}$. From Table 2, the evolved edge detectors using blocks of pixels are significantly better the evolved edge detectors only using single pixels (Set_s and $Set_{5\times5}$). Based on the comparison between $Set_{s,b}$ and $Set_{s,tb}$, replacing $block_{t,l,w,d}$ with tb_{j,t,l,w,d_1,d_2}, GP evolves the edge detectors which remains detection performance, in terms of F.

From Tables 1 and 2, it is suggested that the search operator tb_{j,t,l,w,d_1,d_2} can improve efficiency for constructing edge detectors, while keeping the accuracy at the same time.

5.3 Visual Results

Figure 7 shows an example image detected by the best edge detectors using single pixels and blocks of pixels ($Set_{s,b}$, $Set_{5\times5}$, $Set_{s,b}$ and $Set_{s,tb}$). From the edge detectors using single pixels only, namely from $Set_{s,b}$ and $Set_{5\times5}$, the detected results are affected by noise and textures in image 385039. When search operators based on blocks of pixels are used, the detected results from $Set_{s,b}$ and $Set_{s,tb}$ are not strongly affected by textures. Comparing the edge detector from $Set_{s,tb}$ to the edge detector from $Set_{s,b}$, we find that there are no obvious difference. It seems that all of the GP evolved edge detectors using blocks of pixels suppress textures and filter noise. Compared with the edge detectors using single pixels only, the edge detectors using blocks of pixels do not decrease obviously in finding true edge points, but obviously remove lots of falsely predicted edge points, in terms of the detected visual results. The edge evolved by GP with $Set_{s,tb}$ are thicker than those with $Set_{s,b}$, but this can be easily thinned by a simple post-processing technique.

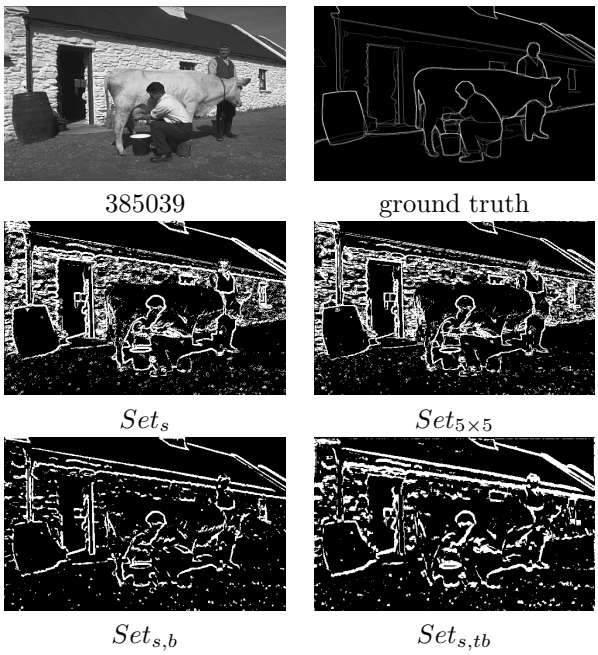


Figure 7: Example test images detected by the best edge detectors from Set_s $Set_{5\times5}$, $Set_{s,b}$ and $Set_{s,tb}$, respectively.

Note that when an offset is not allowed between a predicted edge point to a true edge point, recall and precision usually are not large. The difference of F between Set_s and $Set_{s,b}$ is small, but the visual detected results from them are obviously different.

6 Conclusions

The goal of this paper was to reduce the computational cost of GP using blocks of pixels to extract edge features while remaining detection accuracy. A search operator based on two blocks of pixels was proposed to reduce the number of potential combinations of the existing search operator based on a single block of pixels. From the experiment results, the GP system using the proposed new search operator based on two blocks of pixels can be used to reduce the computational cost on the evolved edge detectors while

remaining detection accuracy. It seems that reducing the search space on blocks of pixels is efficient to improve the performance of the evolved detectors, in terms of the computational cost.

For future work, the investigation on building blocks for edge detection will be conducted so that we can understand how GP works on construction of edge detectors. The computational complexity of using varying blocks in GP will be investigated. Also, different performance evaluation criteria will be employed to compare GP evolved edge detectors with existing edge detectors.

References

Bai, X., Yang, X. & Latecki, L. J. (2008), 'Detection and recognition of contour parts based on shape similarity', *Pattern Recognition* **41**(7), 2189–2199.

Basu, M. (2002), 'Gaussian-based edge-detection methods: a survey', *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **32**(3), 252–260.

Bertero, M., Poggio, T. & Torre, V. (1988), 'Ill-posed problems in early vision', *Proceedings of the IEEE* **76**(8), 869–889.

Bovik, A., Huang, T. S. & Munson, D.C., J. (1987), 'The effect of median filtering on edge estimation and detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-9**(2), 181–194.

Canny, J. (1986), 'A computational approach to edge detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **8**(6), 679–698.

Dollar, P., Tu, Z. & Belongie, S. (2006), Supervised learning of edges and object boundaries, *in* 'Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition', Vol. 2, pp. 1964–1971.

Ebner, M. (1997), On the edge detectors for robot vision using genetic programming, *in* 'Proceedings of Horst-Michael Groβ, Workshop SOAVE 97 - Selbstorganisation von Adaptivem Verhalten', pp. 127–134.

Fu, W., Johnston, M. & Zhang, M. (2011*a*), Genetic programming for edge detection: a global approach, *in* 'Proceedings of the 2011 IEEE Congress on Evolutionary Computation', pp. 254–261.

Fu, W., Johnston, M. & Zhang, M. (2011*b*), Genetic programming for edge detection based on accuracy of each training image, *in* 'Proceedings of the 24th Australasian Joint Conference on Artificial Intelligence', pp. 301–310.

Fu, W., Johnston, M. & Zhang, M. (2012*a*), Automatic construction of invariant features using genetic programming for edge detection, *in* 'Proceedings of the Australasian Joint Conference on Artificial Intelligence', pp. 144–155.

Fu, W., Johnston, M. & Zhang, M. (2012*b*), Genetic programming for edge detection using blocks to extract features, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference', pp. 855–862.

Fu, W., Johnston, M. & Zhang, M. (2012*c*), Genetic programming for edge detection via balancing individual training images, *in* 'Proceedings of the IEEE Congress on Evolutionary Computation', pp. 2597–2604.

Fu, W., Johnston, M. & Zhang, M. (2013*a*), Genetic programming for edge detection using multivariate density, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference', pp. 917–924.

Fu, W., Johnston, M. & Zhang, M. (2013*b*), Triangular-distribution-based feature construction using genetic programming for edge detection, *in* 'Proceedings of the IEEE Congress on Evolutionary Computation', pp. 1732–1739.

Ganesan, L. & Bhattacharyya, P. (1997), 'Edge detection in untextured and textured images: a common computational framework', *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* **27**(5), 823–834.

Golonek, T., Grzechca, D. & Rutkowski, J. (2006), Application of genetic programming to edge detector design, *in* 'Proceedings of the International Symposium on Circuits and Systems', pp. 4683–4686.

Grigorescu, C., Petkov, N. & Westenberg, M. A. (2004), 'Contour and boundary detection improved by surround suppression of texture edges', *Image and Vision Computing* **22**(8), 609–622.

Harding, S. & Banzhaf, W. (2008), 'Genetic programming on GPUs for image processing', *International Journal of High Performance Systems Architecture* **1**(4), 231–240.

Harris, C. & Buxton, B. (1996), Evolving edge detectors with genetic programming, *in* 'Proceedings of the First Annual Conference on Genetic Programming', pp. 309–314.

Hollingworth, G., Smith, S. & Tyrrell, A. (1999), Design of highly parallel edge detection nodes using evolutionary techniques, *in* 'Proceedings of the Seventh Euromicro Workshop on Parallel and Distributed Processing', pp. 35–42.

Kadar, I., Ben-Shahar, O. & Sipper, M. (2009), Evolution of a local boundary detector for natural images via genetic programming and texture cues, *in* 'Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation', pp. 1887–1888.

Koza, J., Bennett III, F. H., Andre, D. & Keane, M. A. (1999), *Genetic Programming III: Darwinian Invention and Problem Solving*, Morgan Kaufmann.

Kunt, M. (1982), Edge detection: a tutorial review, *in* 'Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing', Vol. 7, pp. 1172–1175.

Lim, D. H. & Jang, S. J. (2002), 'Comparison of two-sample tests for edge detection in noisy images', *Journal of the Royal Statistical Society. Series D (The Statistician)* **51**(1), 21–30.

Lopez-Molina, C., De Baets, B. & Bustince, H. (2013), 'Quantitative error measures for edge detection', *Pattern Recognition* **46**(4), 1125–1139.

Marr, D. & Hildreth, E. (1980), 'Theory of edge detection', *Proceedings of the Royal Society of London, Series B, Biological Sciences* **207**(1167), 187–217.

Martin, D., Fowlkes, C. & Malik, J. (2004), 'Learning to detect natural image boundaries using local brightness, color, and texture cues', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**(5), 530–549.

Moreno, R., Puig, D., Julia, C. & Garcia, M. (2009*a*), A new methodology for evaluation of edge detectors, *in* 'Proceedings of the 16th IEEE International Conference on Image Processing (ICIP)', pp. 2157–2160.

Moreno, R., Puig, D., Julia, C. & Garcia, M. (2009*b*), A new methodology for evaluation of edge detectors, *in* 'Proceedings of the 16th IEEE International Conference on Image Processing', pp. 2157–2160.

Papari, G. & Petkov, N. (2008), 'Adaptive pseudo dilation for gestalt edge grouping and contour detection', *IEEE Transactions on Image Processing* **17**(10), 1950–1962.

Papari, G. & Petkov, N. (2011), 'Edge and line oriented contour detection: state of the art', *Image and Vision Computing* **29**, 79–103.

Poli, R. (1996), Genetic programming for image analysis, *in* 'Proceedings of the First Annual Conference on Genetic Programming', pp. 363–368.

Poli, R., Langdon, W. B. & McPhee, N. F. (2008), *A Field Guide to Genetic Programming*, Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`. With contributions by J. R. Koza.

Quintana, M. I., Poli, R. & Claridge, E. (2006), 'Morphological algorithm design for binary images using genetic programming', *Genetic Programming and Evolvable Machines* **7**, 81–102.

Wang, J. & Tan, Y. (2010), A novel genetic programming based morphological image analysis algorithm, *in* 'Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation', pp. 979–980.

Zhang, Y. & Rockett, P. I. (2005), Evolving optimal feature extraction using multi-objective genetic programming: a methodology and preliminary study on edge detection, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference', pp. 795–802.

Particle Swarm Optimisation for Feature Selection: A Size-Controlled Approach

Tony Butler-Yeoman, Bing Xue, and Mengjie Zhang

School of Engineering and Computer Science, Victoria University of Wellington
PO Box 600, Wellington 6140, New Zealand
Email: {butlertony, Bing.Xue, Mengjie.Zhang}@ecs.vuw.ac.nz

Abstract

Feature selection is a preprocessing step in classification tasks, which can reduce the dimensionality of a dataset and improve the classification accuracy and efficiency. However, many current feature selection algorithms select an unnecessarily large feature subsets, particularly on datasets with high dimensionality. This paper proposes a new particle swarm optimisation (PSO) based feature selection approach, where a new method is proposed to find the possible smallest size that potentially good feature subsets can have to guide the PSO algorithm to search for smaller feature subsets. The proposed algorithm is examined and compared with original PSO based feature selection and two typical feature selection method on twelve benchmark datasets of varying difficulty. The experimental results show that the proposed algorithm successfully further reduces the dimensionality of the dataset over original PSO and one of the conventional method, and maintains or even increases the classification performance in most cases. The proposed algorithm selects more features than the other conventional method, but achieves better classification performance in most cases, which shows that the proposed algorithm can balance the classification performance and the number of features in most cases. Furthermore, the proposed algorithm also shows better efficiency and consistency performance in terms of selecting consistent features across different stochastic runs.
Keywords: Classification, Feature Selection, Particle Swarm Optimisation

1 Introduction

In many classification tasks, the datasets often include a large number of features, so as to represent the target concept as completely as possible. However, as many features are redundant or irrelevant, this results in noise in the dataset that reduces the performance of many classification algorithms (Dash and Liu, 1997). Furthermore, the large number of features contribute to the "curse of dimensionality" (Dash and Liu, 1997; Guyon and Elisseeff, 2003), which is one of the major obstacles in classification. Feature selection is the process of choosing a subset of the relevant features from a large number of original features. The chosen feature subset should be small and accurately describe the target concept. As a preprocessing step, feature selection is a practical and well-known solution to the problems of high-dimensionality data, resulting in a fast and high-performing classification process.

Based on the evaluation criteria, feature selection algorithms are generally classified into two categories: *filter* approaches and *wrapper* approaches (Dash and Liu, 1997; Guyon and Elisseeff, 2003). Their main difference is that wrapper approaches include a classification/learning algorithm in the feature subset evaluation step. The classification algorithm is used to evaluate the goodness (i.e. the classification performance) of the selected features. A filter feature selection process is independent of any classification algorithm. Filter algorithms are often computationally less expensive and more general than wrapper algorithms. However, filters ignore the performance of the selected features on a classification algorithm while wrappers evaluate the feature subsets based on the classification performance, which usually results in better performance achieved by wrappers than filters for a particular classification algorithm (Dash and Liu, 1997; Mitra et al., 2002; Liu and Zhao, 2009; Liu et al., 2010).

Feature selection is a challenging task since the space of possible feature subsets is the power set of the features, hence there are 2^n possible feature subsets for a dataset with n features. If all features were completely independent, an efficient greedy algorithm could search this space fast by identifying and removing irrelevant features, leaving only the most useful features. However, since features are often interacting with each other, which leads to that individually relevant features may become redundant and individually weakly relevant features may become highly relevant when combined with other features (Guyon and Elisseeff, 2003). Therefore, a powerful global search algorithm that can consider all features at the same time is needed to find the optimal feature subset(s). Although different types of search techniques have been applied to features selection (Guyon and Elisseeff, 2003; Liu et al., 2010), existing approaches still suffer from the problem of being stagnation in local optima. Evolutionary computation (EC) techniques are well-known for their promising search ability, and have been applied to feature selection tasks with some success, such as genetic algorithms (GAs) (Zhu et al., 2007; Lin et al., 2014), genetic programming (Muni et al., 2006; Neshatian and Zhang, 2012; Espejo et al., 2010; Purohit et al., 2010), differential evolution (DE) (Bharathi P T, 2014*a*,*b*), and particle swarm optimisation (PSO) (Boubezoul and Paris, 2012; Xue et al., 2013*b*; Vieira et al., 2013). Compared with GAs and GP, PSO is easier to implement, has fewer parameters, computationally less expensive, and can converge more quickly (Engelbrecht, 2007). However,

the search of the original PSO algorithm often selects a relatively large number of features, which may include redundant features. Further reduction on the number of features and analysis on the consistency of features selected across different stochastic runs are still needed.

1.1 Goals

The goal of this paper is to develop a new PSO based wrapper feature selection approach that can select a significantly smaller number of features and maintain the classification performance over using the original feature set and the features selected by standard PSO. To achieve this goal, a new method is proposed to find a target size which roughly indicates the smallest size of feature subsets that can achieve the highest or close to the highest classification performance. The target size is used to guide the search of PSO in addition to the original PSO search mechanism that focuses mainly on the finding the highest classification performance. Specifically, we will investigate:

- whether incorporating the target size method into PSO for feature selection can further reduce the number of features selected over standard PSO,
- whether the classification performance of PSO for feature selection can be maintained or even improved in the new approach,
- whether the proposed approach can outperform conventional feature selection methods, and
- whether the proposed approach can more consistently select key informative features than standard PSO.

2 Background

2.1 Particle Swarm Optimisation

Particle Swarm Optimisation is an evolutionary computation technique inspired by social behaviour proposed by Kennedy and Eberhart (Kennedy and Eberhart, 1995; Shi and Eberhart, 1998). PSO maintains a population of *particles*, called a *swarm*, each of which encodes a candidate solution in the search space. PSO initialises each particle in the swarm to a random position in the space, and iterates the position of each particle based on the experience of the particle and its neighbours. The position of particle i is represented by a vector, $x_i = (x_{i,1}, \ldots, x_{i,n})$, where n is the dimension of the search space. The velocity is represented by a similar vector $v_i = (v_{i,1}, \ldots, v_{i,n})$, where each component of the vector is limited to a predefined range $[-v_{max}, v_{max}]$. The best previous position (according to some fitness function) of particle i is recorded as the personal best, $pbest_i = (p_{i,1} \ldots p_{i,n})$, and the best position found by the population as a whole is recorded as the global best, $gbest = (g_1 \ldots g_n)$. At each iteration of the algorithm, PSO updates the velocity and position of each particle according to the following equations:

$$x_{i,d}^{t+1} = x_{i,d}^{t} + v_{i,d}^{t+1} \tag{1}$$

$$\begin{aligned} v_{i,d}^{t+1} = {} & w \cdot v_{i,d}^{t} + c_1 \cdot r_{1,i} \cdot (p_{i,d} - x_{i,d}^{t}) \\ & + c_2 \cdot r_{2,i} \cdot (g_d - x_{i,d}^{t}) \end{aligned} \tag{2}$$

Here $0 < d \leq n$ denotes the component of the position or velocity vector, and t represents the t-th iteration of the algorithm. w is a predefined constant for the inertia weight, and c_1 and c_2 are predefined acceleration constants. Each $r_{1,i}$ and $r_{2,i}$ are random values uniformly distributed over $[0, 1]$. This description of PSO is applicable to real-valued search spaces. However, feature selection, along with many other problems, occurs in a discrete search space and requires a modified algorithm. Binary PSO (Kennedy and Eberhart, 1997) is such an algorithm. In binary PSO, the values of the components of all position vectors (x_i, $pbest_i$, and $gbest_i$) are restricted to 0 or 1. Equation (2) is still used to update the velocity, each component of which now indicates the probability of the corresponding component in the position vector being 1. A sigmoid function $s(v_{i,d})$ is used to transform the components of the velocity into a unit range. Binary PSO updates the position of each particle according to the following equation:

$$x_{i,d} = \begin{cases} 1, & rand() < s(v_{i,d}) \\ 0, & otherwise \end{cases} \tag{3}$$

where

$$s(x) = \frac{1}{1 + e^{-x}}$$

2.2 Related Work

The following sections survey a relevant selection of recent and prominent work on feature selection.

2.2.1 Traditional Algorithms

FOCUS (Almuallim and Dietterich, 1994) and Relief (Kira and Rendell, 1992) are two founding feature selection algorithms. Each takes a filter approach and so a learning system is not used in evaluation. FOCUS exhaustively searches the space of feature subsets to find a small subset that accurately describes the target concept. This is highly effective, but of course exhaustive search is infeasible on large feature spaces. Relief does not perform exhaustive search, instead it scores each feature individually based on its relevance to the class label and selects a set of the best scoring features. This is much more efficient, but does not take into account feature interaction (Kononenko, 1994); this could lead to, for example, ignoring features that are only useful in combination, or selecting two highly redundant features.

Sequential Forward Selection (SFS) (Whitney, 1971) and Sequential Backward Selection (SBS) (Marill and Green, 1963) are two more widely-known algorithms, both using a wrapper approach. These perform a greedy search in the feature space, with SFS (SBS) starting with an empty (full) feature set and iteratively selecting features to add (remove). Because this is a greedy search, both algorithms are susceptible into local optima. To overcome this and other issues, the 'plus-l-take-away-r' method (Stearns, 1976) and sequential floating algorithms (SFFS and SBFS) (Pudil et al., 1994) were proposed, which are variants on SFS and SBS.

2.2.2 Feature Selection with PSO

Evolutionary computation techniques, including GAs (Lin et al., 2014) and GP (Neshatian and Zhang, 2012), have been broadly applied to feature selection problems. For brevity, this section will focus on using PSO for feature selection (Cervante et al., 2012; Xue

et al., 2014; Lane et al., 2013, 2014; Xue et al., 2015; Nguyen et al., 2014; Tran et al., 2014).

Both continuous PSO and binary PSO have been used for both filter and wrapper, single objective and multi-objective feature selection. A number of new PSO algorithms have been proposed to improve performance on feature selection problems, including initialisation strategies, representation, fitness functions, and the search mechanisms. Xue et al. (Xue et al., 2013*a*) developed a new initialisation strategy to mimic the typical forward and backward feature selection methods in the PSO search process, which showed that good initialisation significantly increased the performance of PSO for feature selection. There are only a few works on developing new representations in PSO for feature selection. The typical representation has been slightly modified to simultaneously perform feature selection and parameter optimsation of a classification algorithm, mostly optimising the parameters in the kernel functions of SVMs (Lin et al., 2008; Huang and Dun, 2008; Vieira et al., 2013; Boubezoul and Paris, 2012). The length of the new representation is equal to the total number of features and parameters. The representation was encoded in three different ways, being continuous encoding (Lin et al., 2008), binary encoding (Vieira et al., 2013), and a mixture of binary and continuous encoding (Huang and Dun, 2008; Boubezoul and Paris, 2012). Since PSO was originally proposed for continuous optimisation, continuous encoding performed better than the other two encoding schemes. Lane et al. (Lane et al., 2013) proposed the use of PSO and statistical clustering (which groups similar features into the same cluster) for feature selection, where a new representation was proposed to incorporate statistical feature clustering information during the search process of PSO. In the new representation, features from the same cluster were arranged together and only a single feature was selected from each cluster. The proposed algorithm was shown to be able to significantly reduce the number of features.

Learning from neighbours' experience, i.e. social interaction through *gbest*, and learning from each individual's own experience through *pbest*, are the key ideas in PSO. Chuang et al. (Chuang et al., 2008) developed a *gbest* resetting mechanism by including zero features in order to guide the swarm to search for small feature subsets. Xue et al. (Xue et al., 2013*a*) considered the number of features when updating *pbest* and *gbest* during the search process of PSO, which could further reduce the number of features over the traditional updating *pbest* and *gbest* mechanism without deteriorating the classification performance.

The fitness function plays an important role in PSO for feature selection. Many existing works used only the classification performance as the fitness function (Liu et al., 2011; Zhang and Hu, 2005; Tang et al., 2005; Yong et al., 2015), which led to relatively large feature subsets. However, most of the fitness functions used different ways to combine both the classification performance and the number of features into a single fitness function (Huang and Dun, 2008; Fdhila et al., 2011; Ramadan and Abdel Kader, 2007). However, it is difficult to determine in advance the optimal balance between them without *a priori* knowledge. Most of the algorithms select a relatively large number of features.

Algorithm 1 Size-Controlled PSO Feature Selection

```
1: divide Dataset into a training set and a test set
2: randomly initialise position and velocity of each particle
3: while the stopping criterion is not met do
4:     evaluate the fitness (classification performance) of each particle on the training set
5:     for each particle p do
6:         update the pbest of p
7:         update the gbest of p
8:     end for
9:     for each particle p do
10:        update the velocity of p according to Equation (4)
11:        update the position of p according to Equation (1)
12:    end for
13: end while
14: Test process:
15: calculate the classification accuracy of the features selected by gbest on the test set
16: return the position of gbest (the selected feature subset)
17: return the training and test classification accuracies
```

3 Proposed Approach

3.1 Size-Controlled PSO for Feature Selection

In PSO for feature selection, PSO focuses on searching for the best classification performance according to the fitness function. However, since the search space is huge in most cases, besides the classification performance, PSO needs further guidance to search towards a feature subset with not only high classification accuracy, but also a small size. Therefore, we propose a size-controlled PSO for feature selection, where the search of PSO is also guided by a *target* size T. The influence of T is produced through the velocity updating equation, which is shown by Equation (4).

$$\begin{aligned} v_{i,d}^{t+1} &= w \cdot v_{i,d}^{t} \\ &\quad + c_1 \cdot r_{1,i} \cdot (p_{i,d} - x_{i,d}^{t}) \\ &\quad + c_2 \cdot r_{2,i} \cdot (g_d - x_{i,d}^{t}) \\ &\quad + c_3 \cdot r_{3,i} \cdot s(T - |p|) \end{aligned} \tag{4}$$

where,

$$s(x) = \frac{1}{1 + e^{-x}}$$

where $|p|$ shows the number of features selected by particle i, c_3 is a third acceleration constant, and each $r_{3,i}$ is a random value uniformly distributed in $[0, 1]$.

Algorithm 1 shows the pseudo-code for the size-controlled PSO. The position of each particle is still updated using Equation (1). To implement this size-controlled PSO, the key issue is how to determine the target size T.

Algorithm 2 Sequential Random Target Size Optimisation

1: classification accuracy indicates how good a feature subset is;
2: $Accuracy''$ indicates how good an integer n is;
3: d: the dimensionality of the data;
4: initialise $range \leftarrow [1, d]$.
5: Start:
6: **for** iteration i from 1 to maximum iterations **do**
7: randomly sample a set of N integers and their values are within $range$, *i.e. randomly sample a set of candidate sizes*;
8: **for** each candidate integer n from set N **do**
9: randomly sample s different feature subsets and each subset contains n features;
10: evaluate the classification accuracies of the s subsets;
11: find the highest classification accuracy and assign it as the $Accuracy''$ of the candidate size n;
12: **end for**
13: $topAcc \leftarrow$ the best $Accuracy''$ so far regardless of size;
14: find all possible candidate sizes with $Accuracy'' \in [topAcc - toleranceAcc]$, and assign the smallest candidate size to $bestSize$;
15: $range \leftarrow [1, \frac{2*R*d}{i}]$;
16: centre $range$ on $bestSize$;
17: **if** $bestSize$ is not changed **then**
18: exit;
19: **end if**
20: **end for**
21: return $bestSize$ as target size T.

3.2 Sequential Random Target Size Optimisation

In this section, we propose a sequential random target size optimisation (SRTSO) method to find the target size T that the size-controlled PSO requires.

The Pseudocode of SRTSO can found in Algorithm 2. SRTSO aims to find a good size ($bestSize$) with the expectation that the feature subset(s) containing $bestSize$ features can achieve the optimal or near optimisal classification performance. Meanwhile, $bestSize$ is the smallest size that can achieve such good classification performance. In SRTSO, there are two evaluation criteria: (1) the classification accuracy shows how good a feature subset is; (2) $Accuracy''$ shows how good the size n is, and since SRTSO generates multiple feature subsets that contain n features, the best classification accuracy among them is used as the $Accuracy''$ value.

SRTSO follows an iteratively searching process. In each iteration, a set of N integers (candidate sizes) are randomly generated, where all integers are within $range$, Line 7. $range$ is initialised as $[1, d]$ and dynamically changes during the search as $range = [1, \frac{2Rd}{i}]$, where R is a constant value, d is the dimensionality and i means the ith iteration of SRTSO. $\frac{2Rd}{i}$ can ensure that the candidate size that SRTSO searches for becomes smaller along with the search process, i.e. the increase of i. Furthermore, $range$ is also determined/adjusted by the classification performance in each iteration through centering it around a value called $bestSize$. $bestSize$ is calculated from Line 8 to 14. $bestSize$ shows the smallest sizes which have $Accuracy'' \in [topAcc - toleranceAcc]$, where $topAcc$ is the highest classification SRTSO has found so far and $toleranceAcc$ is a very small percentage. Using the constraint of $Accuracy'' \in [topAcc - toleranceAcc]$ to determine the $bestSize$ value is to ensure that the $bestSize$ is a small value and also always has a highest $Accuracy''$.

When the maximum number of iteration has been reached or $bestSize$ is the same in two iterations, SRTSO is terminated. SRTSO returns $bestSize$ as the target size T, which guides PSO to search the regions where the size of feature subsets is T.

By applying the SRTSO method to determine T in Equation 4 in PSO, a new approach named SCTSOFS is proposed to solve feature selection problems.

Table 1: Datasets

Dataset	Number of features	Number of instances	Number of classes
WDBC	30	569	2
Ionosphere	34	351	2
Splice	61	3190	4
Hill Valley	100	606	2
Gas 6	128	1694	3
Musk 1	166	476	2
Semeion	256	1593	2
Arrhythmia	278	452	2
Madelon	500	2600	2
Isolet 5	617	1599	26
Multiple Features	649	2000	10
Amazon	10000	1599	50

4 Experimental Design

Twelve datasets (in Table 1) were chosen from the UCI machine learning repository (Bache and Lichman, 2013) as benchmark problems to evaluate the performance of SCTSOFS. These were chosen to represent a range of features, instances, and classes to represent different types of tasks. For each dataset, the instances are randomly divided into $^2/_3$ for the training set and $^1/_3$ for the testing set such that class distribution is approximately maintained.

To perform fitness evaluations in the feature selection process, a classification algorithm is needed to calculate the classification accuracy. There are many options for the algorithm, such as K-nearest neighbour (KNN), Decision Trees, Support Vector Machines, and Naive Bayes. KNN was chosen with $k = 1$ (1NN) due to its simplicity and wide use in existing papers. Each evaluation uses 10-fold cross validation on the training set (Guyon and Elisseeff, 2003).

The performance of SCTSOFS is compared with that of all features, standard PSO for feature selection and two conventional feature selection algorithms: SFS and SBS. The parameters in PSO and SCTSOFS follows common settings suggested in (Clerc and Kennedy, 2002): inertia weight $w = 0.7298$, acceleration constants $c_1 = c_2 = 1.49618$, maximum velocity $v_{max} = 6$. PSO uses a population size of 30 and the maximum iterations of 50 and SCTSOFS uses a population size of 10 and the maximum iterations of 30, and its third acceleration constant $c_3 = c_1$. For SCTSO, the maximum iteration is 2, $N = 30$, $s = 10$, $R = 0.3$, and $tolerance = 0.01$. By using such settings, it ensures that PSO and SCTSOFS have the same number of evaluations for fair comparison purposes. PSO and SCTSOFS are performed for 30 independent runs on each dataset. SFS and SBS are

Table 2: Accuracy (%) and size results for All features, standard PSO, and SCTSOFS

Dataset	Algorithm	Training Set Acc.			Test Set Acc.		
		Mean size	Mean (± stdev)	Best	Mean (± stdev)	Signif-icance	Best
WDBC	All	30.0	93.9	93.9	96.3	−	96.3
	PSO	14.4	96.8 ± 0.2	97.4	96.8 ± 1.1	=	98.4
	SCTSOFS	9.8	96.2 ± 1.1	97.1	96.8 ± 1.1		98.9
Ionosphere	All	34.0	88.0	88.0	84.5	−	84.5
	PSO	12.8	94.0 ± 0.6	95.7	87.3 ± 2.0	=	90.6
	SCTSOFS	7.9	93.7 ± 0.9	95.7	86.6 ± 3.2		93.1
Splice	All	60.0	72.3	72.3	69.6	−	69.6
	PSO	23.9	78.9 ± 0.8	80.7	75.4 ± 1.8	−	79.0
	SCTSOFS	14.0	80.5 ± 2.4	85.0	78.0 ± 3.3		84.0
Hill Valley	All	100.0	57.2	57.2	56.5	+	56.5
	PSO	48.0	61.3 ± 0.4	62.1	55.2 ± 0.8	=	56.9
	SCTSOFS	25.7	60.2 ± 1.4	62.5	54.8 ± 1.7		57.9
Gas 6	All	128.0	100.0	100.0	99.8	+	99.8
	PSO	35.8	100.0 ± 0.0	100.0	99.8 ± 0.2	=	100.0
	SCTSOFS	3.4	100.0 ± 0.0	100.0	99.7 ± 0.2		100.0
Musk 1	All	166.0	84.2	84.2	74.7	−	74.7
	PSO	79.6	92.8 ± 0.8	94.6	79.4 ± 2.7	=	83.0
	SCTSOFS	48.6	90.3 ± 2.4	92.4	79.3 ± 3.1		85.5
Semeion	All	265.0	97.1	97.1	96.4	+	96.4
	PSO	136.2	98.2 ± 0.1	98.4	95.9 ± 0.6	+	97.2
	SCTSOFS	89.0	96.9 ± 1.1	98.0	95.3 ± 0.6		96.2
Arrhythmia	All	278.0	53.8	53.8	55.0	=	55.0
	PSO	135.5	64.4 ± 0.7	66.1	56.4 ± 2.2	=	60.3
	SCTSOFS	108.9	62.2 ± 2.8	67.1	55.8 ± 3.5		63.6
Madelon	All	500.0	54.0	54.0	54.3	−	54.3
	PSO	244.7	60.0 ± 0.6	60.9	55.6 ± 1.6	=	59.4
	SCTSOFS	150.2	59.1 ± 1.7	65.9	55.9 ± 2.3		60.3
Isolet 5	All	617.0	80.6	80.6	71.2	−	71.2
	PSO	304.2	84.1 ± 0.3	85.0	74.3 ± 1.0	+	76.9
	SCTSOFS	238.3	82.6 ± 1.1	83.9	72.7 ± 1.4		76.0
Multiple Features	All	649.0	97.8	97.8	96.9	+	96.9
	PSO	326.2	98.5 ± 0.1	98.7	97.1 ± 0.4	+	98.0
	SCTSOFS	168.7	97.4 ± 0.8	98.1	96.2 ± 0.9		97.7
Amazon	All	10000.0	13.8	13.8	11.4	−	11.4
	PSO	4919.1	18.1 ± 0.4	19.1	12.1 ± 0.9	−	14.0
	SCTSOFS	480.6	19.2 ± 0.8	20.7	13.3 ± 1.5		16.8

deterministic methods, which produce one single solution (feature subset) on each dataset and their settings follow common settings in Weka (Witten and Frank, 2005). A statistical significance test, Wilcoxon test with the significance level as 0.05, is performed on the test accuracies of different algorithms.

5 Results and Discussions

5.1 Comparisons with All Features

Table 2 summarises the classification accuracy and the feature subset size of "All" features, PSO and SCTSOFS, where "mean ± stdev" shows the average and the standard deviation of the accuracies from the 30 independent runs. In terms of the training set, the classification performance of SCTSOFS is better than using the full set of features on ten out of the twelve datasets. The comparison on test set accuracy between SCTSOFS and using a full set of features is generally positive. SCTSOFS performs significantly better on seven of the twelve datasets, worse on four, and equally on one. This indicates that feature selection using SCTSOFS is mostly beneficial to classification accuracy, but this may depend on the dataset.

In terms of the feature subset size, SCTSOFS substantially reduce the dimensionality to at least one third of the original number of features. For example, the number of features is reduced to around 150 on average from the original 500 on the Madelon dataset, but still increase the classification accuracy.

5.2 Comparisons with Standard PSO

According to Table 2, it can be seen that in terms of the subset size, SCTSOFS significantly outperforms the standard PSO feature selection algorithm. SCTSOFS selects a subset with slightly more than half the number of features that PSO selects. The best improvement of SCTSOFS over standard PSO is on the Gas 6 and Amazon datasets, i.e. selecting an order of magnitude fewer features. The results show that SCTSOFS has largely accomplished its goal of reducing feature subset size.

With regards to the accuracy on the test set, SCTSOFS is similar or significantly better than PSO on nine out of the twelve datasets, but significantly worse on three datasets. This indicates an overall trend of having similar test set performance. The results on the training set suggest there is no noticeable difference in the level of overfitting between the two algorithms. However, overfitting is suggested for *both* algorithms on some datasets, in particular Musk 1.

5.3 Comparison with Traditional methods

Table 3 shows the classification accuracy and subset size of SFS and SBS on each dataset, where the empty

cells for the three large datasets mean that the algorithm (SFS or SBS) cannot produce a solution by running for a week. There is a symbol of +, −, or = for each algorithm on each dataset, representing the result of a significance test between the classification accuracy of the corresponding algorithm and SCTSOFS on the test set; a + indicates that the corresponding method is statistically significantly better, a − indicates SCTSOFS is better, and an = indicates they are not statistically distinguishable. Statistical significance tests for the size of the feature subset are not given, as the difference is clear.

According to Table 3, SCTSOFS very clearly outperforms SBS in terms of subset size, with SBS selecting the majority of features on most datasets. SBS has statistically equal or lower test set accuracy on ten of the twelve datasets. On these datasets, SCTSOFS is a superior algorithm in all respects.

In contrast, the comparison between SFS and SCTSOFS is mixed. SFS selects very small subsets on all datasets. On the test set, the accuracy of SCTSOFS is statistically equal to or better than SFS on eight of the twelve datasets. The possible reason is that although SCTSOFS uses a size control method, but the main focus is still to optimise the classification performance. Therefore, SCTSOFS outperforms SFS in terms of the classification performance in most cases.

5.4 Computational Cost

Table 4 summarises the running time of each algorithm and the empty cells for the three large datasets mean that the algorithm (SFS or SBS) cannot produce a solution by running for a week. It can be observed that SCTSOFS is faster than standard PSO on all the twelve datasets, although SCTSOFS has extra calculation in SCTSO to find the target size T. This is likely due to the evaluated subsets being, on average, much smaller in SCTSOFS than PSO. The speed difference ranges from near-identical to less than half the running time.

Comparing SCTSOFS with the two traditional methods, SFS runs more quickly on seven of the twelve datasets and SBS also takes orders of magnitude longer to run on almost all datasets, except for the smallest dataset. SBS on the three large datasets (Isolet 5, Multiple Features, and Amazon) and SFS on the two large datasets (Multiple Features and Amazon) cannot even produce a solution within a week. The main reason is that the number of evaluations in SCTSOFS is a fixed number, and the number of evaluations in SFS and SBS increases very quickly along with the number of features in the datasets.

5.5 Analysis of Selected Features

This section compares the features selected by the standard PSO algorithm and SCTSOFS, with the aim of comparing the *consistency* of the two algorithms. Here, consistency measures the similarity of the selected features across multiple runs. This is an important performance metric for a stochastic feature selection algorithm, as it indicates the ability to consistently identify high-quality features in a classification task. This can be analysed based on the frequency of each feature in a dataset being selected is analysed. A uniform frequency among all or most features indicates that the feature selection algorithm is highly inconsistent, indiscriminately selecting features. Similarly, if some features are very commonly selected and others are very rarely selected, the algorithm is highly consistent and selective in its outputs.

To show the consistency, the number of times that each feature is selected by PSO or SCTSOFS in the 30 independent runs are collected, and scaled to [0, 1] as frequency values. 0 means that the feature is not selected at all through the 30 runs and 1 means it is selected across all the 30 runs.

Table 3: Accuracy (%) and size results for SFS and SBS

Method	Size	Train Acc.	Sig.	Test Acc.	Sig.	Size	Train Acc.	Sig.	Test Acc.	Sig.
	WDBC					Ionosphere				
SFS	7.0	93.9	−	96.3	−	5.0	93.2	−	86.4	=
SBS	21.0	96.3	=	98.4	+	26.0	90.1	−	82.0	−
	Splice					Hill Valley				
SFS	6.0	88.9	+	88.2	+	1.0	56.3	−	49.5	−
SBS	50.0	75.1	−	69.8	−	90.0	59.0	−	55.2	=
	Gas 6					Musk 1				
SFS	2.0	100.0	+	99.3	−	14.0	93.7	+	79.3	=
SBS	3.0	100.0	+	99.3	−	84.0	93.4	+	80.4	=
	Semeion					Arrhythmia				
SFS						10.0	65.8	+	59.0	+
SBS	206.0	98.3	+	95.9	+	123.0	64.5	+	54.3	−
	Madelon					Isolet 5				
SFS	11.0	88.3	+	86.4	+	40.0	90.1	+	80.4	+
SBS	488.0	57.2	−	53.8	−					
	Multiple Features					Amazon				
SFS	14.0	98.9	+	95.9	=					
SBS										

Table 4: Running time (milliseconds) of algorithms

Algorithm	WDBC	Ionosphere	Splice
All	40	45	895
SFS	7854	3299	177342
SBS	13434	8564	2356288
PSO	15694 ± 1642	7007 ± 672	1072188 ± 88625
SCTSOFS	14877 ± 3070	5504 ± 703	686714 ± 209926
Algorithm	**Hill Valley**	**Gas 6**	**Musk 1**
All	62	199	57
SFS	11842	37932	97880
SBS	211320	2305076	1515172
PSO	43607 ± 2651	162846 ± 22318	43556 ± 5375
SCTSOFS	40780 ± 8940	109839 ± 18292	30494 ± 5738
Algorithm	**Semeion**	**Arrhythmia**	**Madelon**
All	709	81	7378
SFS	23204	70472	2519672
SBS	64670783	6230209	183583434
PSO	972074 ± 78612	76910 ± 3713	9640148 ± 650128
SCTSOFS	703928 ± 142299	60132 ± 13983	6454478 ± 2519219
Algorithm	**Isolet 5**	**Multiple Features**	**Amazon**
All	1482	3862	43292
SFS	8844293	4223189	
SBS			
PSO	1832157 ± 28898	5418443 ± 471782	54771372 ± 1488128
SCTSOFS	1502773 ± 290200	2933193 ± 459691	25809611 ± 5421677

Table 5: Q-measure of PSO and SCTSOFS

Algorithm	WDBC	Ionosphere	Splice
PSO	0.447	0.468	0.485
SCTSOFS	0.417	0.634	0.590
Algorithm	Hill Valley	Gas 6	Musk 1
PSO	0.345	0.241	0.282
SCTSOFS	0.435	1.109	0.308
Algorithm	Semeion	Arrhythmia	Madelon
PSO	0.211	0.210	0.185
SCTSOFS	0.245	0.214	0.209
Algorithm	Isolet 5	Multiple Features	Amazon
PSO	0.198	0.171	0.153
SCTSOFS	0.204	0.245	0.671

5.5.1 Statistics-based Analysis

The frequency values of the features being selected can be treated as distributions. The concept of consistency is characterised by the probability distribution of features being highly non-uniform, and so a comparison to the discrete uniform distribution of n elements, where n is the dimensionality of the data, gives a metric for an algorithm's consistency. Supposing $f_1 \dots f_n$ are the frequency values for the n feature over a number of experiments, the Q-value of those frequencies is defined as follows:

$$Q = \sum_{i=1}^{n} \left| \frac{f_i}{\sum_{i=1}^{n} f_i} - \frac{1}{n} \right|$$

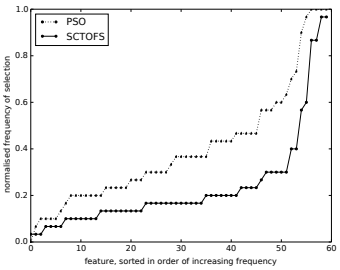
This, in essence, normalises the frequencies to form a proper probability distribution and then sums the absolute values of the difference between each feature and the worst-case values of the uniform distribution. A high Q-value indicates that the algorithm shows high consistency, and a low Q-value indicates inconsistency.

Table 5 gives the Q-value for the benchmark datasets. The results are near-universally favourable towards SCTSOFS, showing a Q-value greater than or equal to that of the standard PSO algorithm in all cases except for the WDBC dataset. This indicates that the new method SCTSOFS is more consistent in terms of finding relevant features.

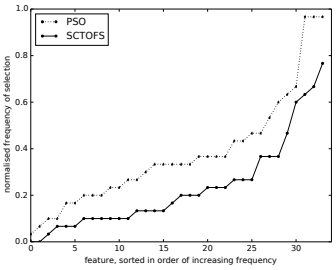
5.5.2 Graph-based Analysis

To better show the results, Figure 1 takes the Splice and Ionosphere datasets as examples to plot the scaled values. Other datasets follow a similar pattern and they are not presented here due to the page limit.

In Figure 1, the scaled values are sorted in an ascending order. Following the horizontal axis, from the left to right, the scaled values are increasing, which shows the lowest frequency to the highest frequency of features being selected. The deeper the curve is the more consistent the algorithm is, because how deepness of the curve shows how well the algorithm distinguish different features. Both plots in Figure 1 show SCTSOFS positively in comparison to standard PSO, particularly, Splice shows over two-thirds of the features being selected no more than 20% of the time, in contrast with standard PSO. It is also shown that SCTSOFS as a more bowed curve, indicating a higher consistency. These results do, however, corroborate with the above statistics-based analysis. In addition, the area under the curve for SCTSOFS is significantly smaller than that of the standard PSO algorithm in both cases. This is expected, as the area represents the average size of the selected feature subsets.



(a) Splice



(b) Ionosphere

Figure 1: Sorted frequencies of features being selected.

6 Conclusions and Future Work

The goal of this paper was to develop a new PSO approach to feature selection with the expectation of significantly reducing the number of features and achieving the similar or even better classification than using all features and standard PSO. The goal was achieved by developing a sequential random target size optimisation method to find a rough target size (number of features), which was used to guide the PSO search towards feature subsets with high classification performance and small number of features.

The proposed algorithm, SCTSOFS, was compared with the standard PSO for feature selection and two typical traditional methods, SFS and SBS. The results show that in most cases, SCTSOFS substantially reduced the dimensionality of the dataset over the original feature set, the feature sets selected by standard PSO, and SBS, and the classification performance was maintained or even improved. Although SFS selected a smaller number of features in many cases, the classification performance of SCTSOFS was generally better, which suggested that although SCTSOFS seriously considered the number of features during feature selection process, but it did not sacrifice the classification performance. Furthermore, SCTSOFS used a shorter running time than PSO in all cases, a longer time than SFS and SBS on small datasets, but a much shorter time than SFS and SBS on large datasets. Since PSO and SCTSOFS are stochastic methods, we also analyse their consistency in terms of selecting features in different runs, the analysis showed that SCTSOFS selected more consistent features across different independent runs, which can provide a better suggestions to real-world users to find informative or key features in complex problems.

Several questions have been raised in the process of this research, such as the consistency analysis for stochastic feature selection algorithms. The analy-

sis presented in this paper can provide straightforward illustrations, but is relatively simple and lack of theoretical prove. Further developments in this area would yield deeper insights into the operation, effectiveness, and applicability of these types of algorithms.

References

Almuallim, H. and Dietterich, T. G. (1994), 'Learning boolean concepts in the presence of many irrelevant features', *Artificial Intelligence* **69**, 279–305.

Bache, K. and Lichman, M. (2013), 'Uci machine learning repository'.
URL: *http://archive.ics.uci.edu/ml*

Bharathi P T, P. S. (2014*a*), 'Differential evolution and genetic algorithm based feature subset selection for recognition of river ice type', *Journal of Theoretical and Applied Information Technology* **7**(1), 254–262.

Bharathi P T, P. S. (2014*b*), 'Optimal feature subset selection using differential evolution and extreme learning machine', *International Journal of Science and Research (IJSR)* **3**, 1898–1905.

Boubezoul, A. and Paris, S. (2012), 'Application of global optimization methods to model and feature selection', *Pattern Recognition* **45**(10), 3676 – 3686.

Cervante, L., Xue, B., Shang, L. and Zhang, M. (2012), A dimension reduction approach to classification based on particle swarm optimisation and rough set theory, *in* '25nd Australasian Joint Conference on Artificial Intelligence', Vol. 7691 of *Lecture Notes in Computer Science*, Springer, pp. 313–325.

Chuang, L. Y., Chang, H. W., Tu, C. J. and Yang, C. H. (2008), 'Improved binary PSO for feature selection using gene expression data', *Computational Biology and Chemistry* **32**(29), 29– 38.

Clerc, M. and Kennedy, J. (2002), 'The particle swarm– explosion, stability, and convergence in a multidimensional complex space', *IEEE Transactions on Evolutionary Computation* **6**(1), 58–73.

Dash, M. and Liu, H. (1997), 'Feature selection for classification', *Intelligent Data Analysis* **1**(4), 131–156.

Engelbrecht, A. P. (2007), *Computational intelligence: an introduction (2. ed.)*, Wiley.

Espejo, P., Ventura, S. and Herrera, F. (2010), 'A survey on the application of genetic programming to classification', *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **40**(2), 121–144.

Fdhila, R., Hamdani, T. and Alimi, A. (2011), Distributed MOPSO with a new population subdivision technique for the feature selection, *in* 'International Symposium on Computational Intelligence and Intelligent Informatics (ISCIII'11)', pp. 81 –86.

Guyon, I. and Elisseeff, A. (2003), 'An introduction to variable and feature selection', *The Journal of Machine Learning Research* **3**, 1157–1182.

Huang, C. L. and Dun, J. F. (2008), 'A distributed PSO-SVM hybrid system with feature selection and parameter optimization', *Application on Soft Computing* **8**, 1381–1391.

Kennedy, J. and Eberhart, R. (1995), Particle swarm optimization, *in* 'IEEE International Conference on Neural Networks', Vol. 4, pp. 1942–1948.

Kennedy, J. and Eberhart, R. (1997), A discrete binary version of the particle swarm algorithm, *in* 'IEEE International Conference on Systems, Man, and Cybernetics', Vol. 5, pp. 4104–4108.

Kira, K. and Rendell, L. A. (1992), 'A practical approach to feature selection', *Assorted Conferences and Workshops* pp. 249–256.

Kononenko, I. (1994), 'Estimating attributes: Analysis and extensions of relief', *Lecture Notes in Computer Science* **784**, 171.

Lane, M., Xue, B., Liu, I. and Zhang, M. (2013), Particle swarm optimisation and statistical clustering for feature selection, *in* 'AI 2013: Advances in Artificial Intelligence', Vol. 8272 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 214–220.

Lane, M., Xue, B., Liu, I. and Zhang, M. (2014), Gaussian based particle swarm optimisation and statistical clustering for feature selection, *in* 'Evolutionary Computation in Combinatorial Optimisation', Vol. 8600 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 133–144.

Lin, F., Liang, D., Yeh, C.-C. and Huang, J.-C. (2014), 'Novel feature selection methods to financial distress prediction', *Expert Systems with Applications* **41**(5), 2472–2483.

Lin, S. W., Ying, K. C., Chen, S. C. and Lee, Z. J. (2008), 'Particle swarm optimization for parameter determination and feature selection of support vector machines', *Expert Systems with Applications* **35**(4), 1817–1824.

Liu, H., Motoda, H., Setiono, R. and Zhao, Z. (2010), Feature selection: An ever evolving frontier in data mining, *in* 'FSDM', Vol. 10 of *JMLR Proceedings*, JMLR.org, pp. 4–13.

Liu, H. and Zhao, Z. (2009), Manipulating data and dimension reduction methods: Feature selection, *in* 'Encyclopedia of Complexity and Systems Science', Springer, pp. 5348–5359.

Liu, Y., Wang, G., Chen, H. and Dong, H. (2011), 'An improved particle swarm optimization for feature selection', *Journal of Bionic Engineering* **8**(2), 191–200.

Marill, T. and Green, D. (1963), 'On the effectiveness of receptors in recognition systems', *IEEE Transactions on Information Theory* **9**(1), 11–17.

Mitra, P., Murthy, C. and Pal, S. (2002), 'Unsupervised feature selection using feature similarity', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(3), 301–312.

Muni, D., Pal, N. and Das, J. (2006), 'Genetic programming for simultaneous feature selection and classifier design', *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* **36**(1), 106–117.

Neshatian, K. and Zhang, M. (2012), Improving relevance measures using genetic programming, *in* 'European Conference on Genetic Programming (EuroGP 2012)', Vol. 7244 of *Lecture Notes in Computer Science*, Springer, pp. 97–108.

Nguyen, H., Xue, B., Liu, I. and Zhang, M. (2014), PSO and statistical clustering for feature selection: A new representation, *in* 'Simulated Evolution and Learning', Vol. 8886 of *Lecture Notes in Computer Science*, pp. 569–581.

Pudil, P., Novovicova, J. and Kittler, J. V. (1994), 'Floating search methods in feature selection', *Pattern Recognition Letters* **15**(11), 1119–1125.

Purohit, A., Chaudhari, N. and Tiwari, A. (2010), Construction of classifier with feature selection based on genetic programming, *in* 'IEEE Congress on Evolutionary Computation (CEC'10)', pp. 1–5.

Ramadan, R. M. and Abdel Kader, R. F. (2007), 'Face recognition using particle swarm optimization-based selected features', *International Journal of Signal Processing, Image Processing and Pattern Recognition* **2**(2), 51–65.

Shi, Y. and Eberhart, R. (1998), A modified particle swarm optimizer, *in* 'IEEE International Conference on Evolutionary Computation (CEC'98)', pp. 69–73.

Stearns, S. (1976), On selecting features for pattern classifier, *in* 'Proceedings of the 3rd International Conference on Pattern Recognition', IEEE Press, Coronado, Calif, USA, pp. 71–75.

Tang, E. K., Suganthan, P. and Yao, X. (2005), Feature selection for microarray data using least squares SVM and particle swarm optimization, *in* 'IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'05)', pp. 1–8.

Tran, B., Xue, B. and Zhang, M. (2014), Overview of particle swarm optimisation for feature selection in classification, *in* 'Simulated Evolution and Learning', Vol. 8886 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 605–617.

Vieira, S. M., Mendonça, L. F., Farinha, G. J. and Sousa, J. M. (2013), 'Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients', *Applied Soft Computing* **13**(5), 3494–3504.

Whitney, A. (1971), 'A direct method of nonparametric measurement selection', *IEEE Transactions on Computers* **C-20**(9), 1100–1103.

Witten, I. H. and Frank, E. (2005), *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*, Morgan Kaufmann.

Xue, B., Cervante, L., Shang, L., Browne, W. N. and Zhang, M. (2014), 'Binary PSO and rough set theory for feature selection: A multi-objective filter based approach', *International Journal of Computational Intelligence and Applications* **13**(02), 1450009.

Xue, B., Zhang, M. and Browne, W. (2013*a*), Novel initialisation and updating mechanisms in PSO for feature selection in classification, *in* 'Applications of Evolutionary Computation', Vol. 7835 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 428–438.

Xue, B., Zhang, M. and Browne, W. N. (2013*b*), 'Particle swarm optimization for feature selection in classification: A multi-objective approach', *IEEE Transactions on Cybernetics* **43**(6), 1656–1671.

Xue, B., Zhang, M. and Browne, W. N. (2015), 'A comprehensive comparison on evolutionary feature selection approaches to classification', *International Journal of Computational Intelligence and Applications* **14**(02), 1550008.

Yong, Z., Dunwei, G., Ying, H. and Wanqiu, Z. (2015), 'Feature selection algorithm based on bare bones particle swarm optimization', *Neurocomputing* **148**, 150–157.

Zhang, C. and Hu, H. (2005), Using PSO algorithm to evolve an optimum input subset for a SVM in time series forecasting, *in* 'IEEE International Conference on Systems, Man and Cybernetics (SMC'05)', Vol. 4, pp. 3793–3796.

Zhu, Z., Ong, Y.-S. and Dash, M. (2007), 'Markov blanket-embedded genetic algorithm for gene selection', *Pattern Recognition* **40**(11), 3236–3248.

Improving Bridge Deterioration Modelling Using Rainfall Data from the Bureau of Meteorology

Qing Huang **Kok-Leong Ong** **Damminda Alahakoon**

Business Analytics, La Trobe Business Schoool, La Trobe University
Plenty Road, Bundoora, Victoria 3083
Email: 18531979@students.latrobe.edu.au, {kok-leong.ong, d.alahakoon}@latrobe.edu.au

Abstract

Failure in bridges carry serious consequences so their appropriate maintenance is paramount. Often, authorities are faced with limited funding and available contractors who are able to carry out the maintenance checks and works. Therefore, a predictive model that can forecast the future state of a bridge component will enable the authority to prioritise and deploy resources to where it is most needed. The challenge faced in this paper is the requirement from the Victorian road authorities to develop an effective predictive model. Prior attempts have been made by using different techniques to construct an alternate predictive model but with limited results. The problem lie in the data itself. With data manually recorded by different contractors, it is noisy and erroneous. Attempts to data cleaning has led to little improvement in the overall model performance. Finally we turned to data augmentation to increase the proportion of reliable data. In our quest to do so, we ended up pulling rainfall data from the BoM to augment the data provided by VicRoads. We consider rainfall data as a candidate for augmentation because literature in civil engineering has correlated bridge component deterioration to the presence of water moisture. Since high rainfall contributes to increased deterioration, leveraging the rainfall information should lead to improved predictive performance. Initial experiments on the predictive performance of the baseline and "high rainfall" models suggest the viability of this approach.

Keywords: Markov chains, bridge deterioration modelling, service life, rate of deterioration, data augmentation

1 Introduction

In many countries, bridge failures are increasing due to ageing of its components. This issue is even more acute in countries (e.g., Australia) where the population is also growing quickly thus, putting on additional stress to the infrastructure. With limited public funds to maintain a wide network of bridges, most authorities such as VicRoads in Victoria would like to deploy a Bridge Management System (BMS) to help optimise maintenance plans for thousands of bridges under its portfolio. A key component of such a system is the Bridge Deterioration model, which is a predictive model that forecasts bridge conditions at a future date so as to determine maintenance priority.

Our research work was carried out in cooperation with VicRoads in the state of Victoria, Australia. VicRoads currently carries a database of more than 180,000 bridge inspection records for over 7,000 bridges state wide. VicRoads has hoped to use the inspection data to build a predictive model that will help optimise their maintenance work schedule so that the limited public funds can be effectively deployed to bridges most in need of maintenance at the lowest cost to each job. The research brief in our case is to analyse their inspection data with the goal of developing a more accurate predictive model so as to improve the effectiveness of their system.

There has been considerable research on various modelling techniques in the area of bridge deterioration including, Markov Chain models (Ranjith et al. 2011, Wellalage et al. 2014*a*), envelopment models (Wakchaure & Jha 2011, Ozbek et al. 2010), and ANN (artificial neural network) models (Sobanjo n.d., Huang 2010). Regardless of the techniques used, the objective is to further improve the overall predictive performance of the system. Among these techniques, Markov Chain-based models are most widely-used even though it was also well-documented in the literature recently about their limitations. Many prior works have been done to overcome these limitations and the next Section provides the elaboration on these works.

With the state of the literature suggesting that the available techniques are limited in delivering the desired predictive performance, we decided that an alternate algorithm to construct the model may not be the best "plan of attack" and so, we looked at improving the predictive performance by working on the data. Attempts to enhace the data quality led to insignificant improvements. We soon realised that this may not be a good approach too because the data is obtained through the process of recording the inspection outcomes manually. To make this worse, the inspections are done by different contractors and hence, there is a high level of noise, variation and error in the data that limits the impact of typical data cleaning techniques.

We had to undertake a different approach to the problem. Working with the civil engineers, we learned about the factors that caused a bridge component to deteriorate. And according to (Huang et al. 2010, Zhao & Chen 2002), the most significant deterioration factors include the age of a bridge, bridge material used, traffic volume, and also the amount of rainfall a bridge was exposed to. Our assumption from this background knowledge helps formulate the hypothesis that "if factors contributing to deterioration of a bridge component" could be included in the de-

Regions	# Records	Breakdown of attributes
All Area	189,247	*Structure Information*, e.g., length, width, year constructed, featured crossed, etc.
Eastern	25,793	
Western	23,705	
Northern	28,692	*Identification information*, e.g., structure ID, road name, road number, region, etc.
North East	37,565	
South West	24,839	
Metropolitan South East	23,171	*Inspection information*, e.g., inspection date, condition ratings (PC1, PC2, PC3, PC4) and inspection comments
Metropolitan North West	25,482	

Table 1: Summary of the bridge data set from VicRoads across major regions. Note that the southern region data set is missing because the south of Melbourne is the Port Philip Bay.

velopment of a predictive model, then there will be a potential chance to increase predictive performance". More importantly, this led us to consider adding quality data to the otherwise noisy data set. This means augmenting features that would reduce the proportion of noisy data with the idea that this would help lift the predictive model over the original baseline.

Among the various factors that contribute to bridge deterioration, we found that historical rainfall data is publicly available from the Australian Bureau of Meteorology (BoM). In this paper, we report our attempt to verify the above hypothesis by augmenting the original data set with the rainfall data from the BoM. Our experiments on the baseline model (built without consideration to rainfall) and the new "high rainfall" model (built by focusing on the high rainfall data sets) suggest the viability of this approach.

To discuss the above, the rest of this paper is organised as follows. Section 2 first reviewed the current works and the outcomes of these attempts. Section 3 then presents the reader with an overview of the bridge inspection records and also introduce the rainfall data from the BoM. A discussion of how we augment the data is also mentioned before we discuss the modelling and validation in Section 4. This is where we see promising difference between the baseline bridge deterioration model and the new model built by considering the high rainfall data. Finally in Section 5, we conclude by outlining the next steps following the work in this paper.

2 Literature review

Research on bridge deterioration modelling started a decade ago and has since seen many various modelling techniques being developed. These techniques can be classified into three main categories: deterministic models, stochastic models and others.

Deterministic models identify a direct relationship between condition ratings (a number from 1 to 4 reflecting the degree of deterioration of a bridge or its components) and the factors affecting bridge deterioration. Normally this is done using a regression model. For example, (Thompson et al. 2012) described an Average Time to Failure model to determine the average life expectancy of a structure or a component. In another example, (Madanat & Ibrahim 1995) attempted to build a common linear model to describe the deterioration rate over time, in which the result was linear. These models took into consideration neither the uncertainties around bridge deterioration nor the existence of unobserved explanatory variables. Thus, stochastic models were developed and its used quickly became popular.

Stochastic models are more capable of capturing the probabilistic nature of the bridge's deterioration process. One major category of stochastic models is based on the Markov theory. For example, (Jiang & Sinha 1989) applied Markov Chains to predict the bridge service life. The transition probabilities from one state to another was solved by non-linear programming and was used to evaluate the life expectancy of a bridge. Another example of the Markovian model being used was reported in the work by (Cesare et al. 1992), where historical data of 850 bridges in the State of New York was used to predict the future condition of the bridges. The results were used to determine the repair policies then.

As stochastic models became popular, its weaknesses were also revealed. In (Thompson & Johnson 2005) for example, the authors concluded from their Markov modelling (on historical data of bridge maintenance records in California) that the quality and quantity of available data would affect the validity of the model. Most of the historical data set contained a large proportion of good condition records, from which very few state transitions can be observed. This was because many records were produced after maintenance. However the actual maintenance records were always absent so that such data cannot be identified and adjusted. In such a situation, the estimated transition probabilities did not reliably reflect the actual bridge deterioration features.

Furthermore, (Aboura et al. n.d.) pointed out that Markov Chains carry the underlying assumption that the transition probabilities are independent of time. This however contradicts the fact that the transition probabilities of a bridge condition to the next state would change as time passes by, i.e., the longer the bridge is in a current state, the higher the chance of it moving to the next successive worse state. To minimise this shortcoming, a number of improvements were made. For example, (Ng & Moses 1998) used the semi-Markov process to incorporate a time factor into traditional Markov models. To consider the age effect, (Sobanjo & Thompson 2011) proposed to incorporate the Weibull model into the Markov Chain while (Maovi & Hajdin 2014) approached this problem with an EM algorithm to improve the reliability of the estimated transition probabilities.

In addition to Markov-based techniques, researchers have recently turned to alternate techniques in an attempt to further improve predictive performance. This include the use of Artificial Neural Network (ANN). One example is seen in (Tokdemir et al. n.d.), who used ANN (and genetic algorithms) to predict the deterioration of highway bridges. A shortcoming of the ANN (and genetic algorithms in general) is that they are computationally intensive. To reduce the computational time, a hybrid optimisation was then proposed by (Callow et al. 2013). One of the strengths of the ANN technique is its ability to deal

Station #	Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Annual	Mean
76077	1891	29.8	0.0	33.8	55.3	47.7	25.8	1.6	30.1	4.1	-	-	-	-	23.60
76077	1892	12.2	5.4	0.8	8.6	26.2	28.7	34.1	22.1	38.6	70.4	45.3	6.6	299.0	24.92
76077	1893	0.0	0.0	9.9	28.6	52.4	35.0	19.9	27.6	51.8	17.2	38.9	1.5	282.8	23.57

Table 2: A snapshot of the monthly rainfall data from the Bureau of Meteorology (BoM) website.

with some of the data issues that these types of problems have. In (Lee et al. 2008, Bu et al. 2012) for example, the Backward Prediction Model was used to generate additional data to augment the otherwise limited historical inspection records.

Despite the various techniques and approaches, Markov-chains remained popular. As (Ng & Moses 1998, Sobanjo & Thompson 2011, Maovi & Hajdin 2014) suggested, the performance of Markov models could be improved when more reliable methods are adopted to estimate the transition probabilities. Therefore, a number of advanced techniques have been used for transition probability estimation, such as Markov Chain Monte Carlo (MCMC) simulation as seen in (Karunarathna et al. 2013, Wellalage et al. 2015). Both works implemented MCMC methods to obtain transition probabilities that best described the transition features of the historical data set. While these efforts focused on the model itself, another factor, namely input data, is also of great significance and should be considered when trying to improve the model performance. As (Huang et al. 2014) pointed out, the right data is equally crucial.

This research thus makes an attempt to move towards the "right data" by augmentation of the existing data set with publicly available information. This creates a richer and also higher quality data input that we then used to establish if an improvement could be made in terms of predictive performance. If so, this method is clearly more scalable than focusing on the specific model as data would be the input to all. The success reported in this method thus scales across different approaches to building the model; not just the Markov Chains used in the experiments of this paper.

3 Overview of data sources

There are four categories of data sets from two sources used for the experiments, which are bridge inspection data and bridge location data from VicRoads, and weather station data and rainfall data from BoM.

Bridge inspection data This is a collection of historical inspection records about the condition states of Victorian bridges across regions. The information of interest is the ratings given to each bridge structure component at end of each inspection. VicRoads adopted a rating scale of 1 to 4, with '1' suggesting that the bridge structure component is in good condition and 4 to mean that the component is damaged or has failed. As shown in Table 1, there are seven regions with the data from 'All Area' containing the data from all the seven regions. Each data set contains 32 attributes detailing information about the bridge structure, identification information, condition ratings of a structure component, and comments from inspectors.

Bridge location data VicRoads also has data about the geographical information of a bridge structure in the form of 'Latitude' and 'Longitude', i.e. the coordinates of a structure. We used the bridge's location to identify the nearest weather station of a structure so as to obtain the corresponding rainfall information.

Weather station data The weather station data was obtained from BoM website. There are two attributes of interest: *station number* and *coordinates*. *Station number* was a unique ID number assigned by BoM to a weather station. We used this information to match up with the historical rainfall data on the BoM website. The coordinates of the weather stations were used together with the bridge location information to identify the bridges that are near to a weather station.

Rainfall data BoM provides rainfall data at the daily, monthly and annual levels. As the bridge inspection data was recorded with respect to each inspection year, annual rainfall amount (in millimetres) was selected for augmentation. Once the nearest weather station is determined for a structure, its corresponding inspection record was extended with an annual rainfall amount calculated from the rainfall data recorded by that weather station.

As (Huang et al. 2014) discussed, there are a few data issues regarding the bridge data that should be dealt with before applying them to the model. In this research, following data issues were considered as well as pre-processing steps were conducted.

In taking advice from (Huang et al. 2014) to ensure that the data is of high quality prior to model development, we undertook data cleaning within the constraints available. This includes the following issues.

- Some of the records have "inspection dates" that were before the registered "construction dates". Others have missing "construction dates". Such records cannot be used for Markov Chain modelling since we need to determine the age of a bridge as part of the modelling. These records were eliminated from the data set. Fortunately, these offending records only accounted for 0.03% of the entire data set.

- Most of the condition ratings are recorded as '1', suggesting that maintenance was conducted throughout the inspections. For the inspection records available for analytics, VicRoads does not have the maintenance records making it difficult to identify which inspection record was made after a maintenance. For example, an inspection on a given day was recorded with a condition rating of '2' but the following inspection record saw that condition rating being revised to a '1'. Without access to the maintenance records, it is unclear whether the change from '2' to '1' was a result of a maintenance or a difference in judgement by the inspectors. In this paper,

Data set	Features selected	Comments
Bridge inspection data	• Structure ID • Construction Date • Inspection Date • Component Number • Condition Rating	• Augmented with monthly rainfall data • Calculate parameters for Markov Chain • Input for model training
Bridge location data	• Structure ID • Coordinates (Latitude, Longitude)	• Determine nearest weather station for a structure
Weather station data	• Station number • Coordinates	• Determine nearest weather station for a structure
Rainfall data	• Station Number • Rainfall Amount	• Augment bridge inspection data with a mean annual rainfall amount

Table 3: Summary of features selected and the rationale for selecting a feature in the data set.

we assumed that the component was maintained whenever an improvement of condition was found between two consecutive records. The component age was then recalculated based on this assumption.

- Table 2 shows a snapshot of how the rainfall data looks like on the BoM. As we can see in the snapshot, the amount of rainfall is not recorded every month because there are times where the weather station did not operate as expected, or that the BoM isn't sure of the reliability of the data. In such cases, the rainfall for that month is missing. To extrapolate the annual rainfall data which we need, we impute the missing values with the statistical long term mean obtained from the BoM website. Once all values are incorporated, the annual rainfall is then calculated for our purpose.

- Lastly, we need to select the best features for building our model. According to (Guyon & Elisseeff 2003, Kira & Rendell n.d.), feature selection is an essential process in model construction especially when there are many irrelevant features in the data set. As we fuse the four data sources for model training, we should select only relevant features. This involves first creating the 'joined' data set by mapping the bridge location data and the weather station to identify the bridges' proximity to weather station. Once that's done, we can pull the relevant annual data for each bridge structure. We then obtain a subset of features as the basis for building our model. The selected features are given in Table 3.

For benchmarking purposes, we went with the popular Markov Chains approach, building both the baseline and the "high rainfall" models. The output of running the Markov Chains is a Transition Probability Matrix derived from the bridge age and condition ratings. Figure 1 shows how the data set is used to build the two models for comparison. The full data set was split into a number of sub-sets based on component number in order to examine the effects of the rainfall from component-level. Each subset, which can be named as 'all rainfall group', was then classified into 'high rainfall' group if the rainfall amount is no less than 600mm and 'low rainfall' group otherwise. The value '600mm' is a statistical long-term average annual rainfall provided by BoM website. Different rainfall groups were used to train the same Markov Chain model respectively, from which the results were used for prediction on the same testing data set in order to compare the prediction accuracy. The 90/10 split is the typical model validation ratio with 10 folds applied to each evaluation as described in Section 4.

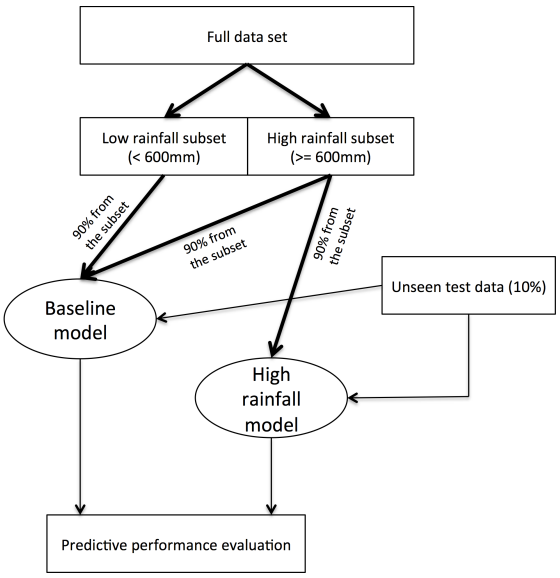


Figure 1: Building two Markov Chain models with the first using the data set from VicRoads and the second, using the data set from VicRoads that has been augmented with rainfall data. The predictive performance of these two models are then compared.

Component Number	Component Type
24C	Abutment made of cast-in-Situ concrete
51S	Bridge railing/Barriers made of steel
52O	Bridge approaches made of other materials
54O	Waterway made of other materials
55S	Bridge approach barriers made of steel

Table 4: A description of the various components. Where "other materials" are mentioned, this means materials other than steel, precast concrete, case-in-Situ concrete or timber.

4 Modelling and Validation

Markov Chain-based model was used for data training and its performance was evaluated by the prediction accuracy of bridge future condition on testing data set. The parameters of the Markov model, namely the transition probabilities, were obtained using Metropolis-Hastings Algorithm (MHA) implemented in Matlab. MHA is one of the most popular methods of Markov Chain Monte Carlo (MCMC) simulation, which is powerful to simulate multivariate distributions [28]. A number of research, such as (Karunarathna et al. 2013, Tran 2007, Wellalage et al. 2014*b*), have used MHA to calibrate the Markov Chains to obtain transition probabilities.

A Markov model describes a system that transits from state i to state j at a single time interval with

Region	Western									
Component Number	24C		51S		52O		54O		55S	
	Baseline	High rainfall	Baseline	High rainfall	Baseline	High rainfall	Baseline	High rainfall	Baseline	High rainfall
Fold 1	0.899	0.078	1.151	1.126	0.839	0.836	0.757	0.867	1.250	1.334
Fold 2	0.741	0.142	1.230	1.227	0.753	0.722	0.566	0.690	1.561	1.563
Fold 3	0.695	0.725	0.959	0.862	0.909	0.904	0.655	0.740	1.307	1.204
Fold 4	0.608	0.117	1.083	1.115	1.106	1.112	0.885	0.891	1.685	1.603
Fold 5	0.211	0.170	1.408	1.338	0.770	0.757	0.771	0.765	1.432	1.428
Fold 6	0.389	0.381	0.937	0.976	0.791	0.788	0.581	0.640	1.400	1.285
Fold 7	0.195	0.143	0.969	0.897	0.972	0.990	0.517	0.542	1.217	1.154
Fold 8	0.140	0.140	1.175	1.060	0.931	0.902	1.119	1.121	1.661	1.577
Fold 9	0.302	0.314	1.153	1.134	0.595	0.565	0.604	0.627	1.309	1.258
Fold 10	0.739	0.773	1.253	1.191	0.974	1.483	0.694	0.708	1.277	1.223

Table 5: Experimental results for the Western data set over 10 folds. The numbers are the average RMSE values for the 10% test data in the given fold tested on both the baseline and the "high rainfall" model. A smaller RMSE value indicates a lower error rate between the actual "overall condition rating" (OCR) and the predicted OCR. Between the Baseline and "high rainfall", the model that has a lower RMSE value indicates better performance. As an example, in Fold 1, on Component 24C, the baseline model produces a RMSE value of 0.899 while the "high rainfall" model produces a 0.078 RMSE value. This means that the "high rainfall" model has produced a better predictive performance for this particular run.

Component	24C		51S		52O		54O		55S	
	Baseline	High rainfall	Baseline	High rainfall	Baseline	High rainfall	Baseline	High rainfall	Baseline	High rainfall
All area	0.230	0.232	1.110	1.094	0.718	0.705	0.587	0.590	1.405	1.381
Eastern	0.231	0.231	1.040	1.037	0.697	0.696	0.598	0.604	1.405	1.402
Western	0.492	0.298	1.132	1.093	0.864	0.906	0.715	0.759	1.410	1.363
MNW	0.173	0.170	0.906	0.903	0.582	0.582	0.469	0.469	1.352	1.349
SW	0.182	0.182	1.040	1.055	0.678	0.635	0.555	0.551	1.405	1.396
NE	0.203	0.227	1.209	1.202	0.800	0.797	0.713	0.715	1.405	1.405
N	0.059	0.179	1.289	1.256	0.761	0.725	0.447	0.446	1.366	1.359
MSE	0.250	0.253	0.797	0.785	0.560	0.553	0.526	0.525	1.239	1.244

Table 6: The average RMSE value over 10 folds for each data set and a given component. This table summarises the RMSE results obtained across eight similar tables that we have used for recording our experiment results. Table 5 shows an example of what the eight tables looked like, which is used for determining the average RMSE values here.

a fixed probability p_{ij}, all of which consists a transition probability matrix (TPM). In this study, there are four discrete transition states (from 1 – 'perfect' to 4 – 'worst') in the bridge deterioration process according to VicRoads. It is assumed that the bridge condition will either remain in the present state or transit to another worse state. The one-year transition probability matrix is given in Equation 1.

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ 0 & p_{22} & p_{23} & p_{24} \\ 0 & 0 & p_{33} & p_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

The initial condition state of a bridge is denoted as a condition state vector $C_0 = [\,1\,0\,0\,0\,]$. The predicted condition state vector C_t after t years is calculated by Equation 2, where P is the transition probability matrix and C_t^i, $i \in [\,1, 2, 3, 4\,]$ is the probability of the bridge condition being in state i at year t.

$$C_t = C_0 \times P^t = [\,C_t^1\ C_t^2\ C_t^3\ C_t^4\,] \tag{2}$$

The result from Equation 2 is then used to calculate an "overall condition rating" (OCR) using Equation 3, which is the expected value obtained by considering the probability values in the state vector.

$$OCR = \sum_{i=1}^{4} i \times C_t^i \tag{3}$$

The modelling process was repeated based on different rainfall groups (see Figure 1). After the data preparation steps, for each component in each region, the All Rainfall Group (ARG) was split into two sets, i.e. Low Rainfall Group (LRG) and High Rainfall Group (HRG). With the assumption that high rainfall would have greater impact on bridge deterioration, ARG data and HRG data were applied to train the Markov model respectively. 90% of each data set was used for training while both groups predicted on the 10% of HRG data.

As the bridge inspection data were recorded in component-level, the same sets of components from each region should be selected for modelling in order to compare the prediction results. Moreover, the modelling results will be less valid if very little number of records for a component is available for training and testing. Therefore, five components were selected from each region with the number of testing data greater than ten. Table 4 provides the component number with its corresponding component type based on VicRoads inspection manual.

Each model was validated by using a separate testing data set as well as ten-fold validation. Also,

Component	24C	51S	52O	54O	55S
	Model comparison				
All area	-0.002	0.016	0.012	-0.002	0.023
Eastern	-0.001	0.003	0.000	-0.006	0.004
Western	0.194	0.039	-0.042	-0.044	0.047
MNW	0.003	0.004	0.000	0.000	0.003
SW	0.000	-0.015	0.043	0.005	0.009
NE	-0.024	0.006	0.003	-0.002	0.000
N	-0.120	0.032	0.036	0.000	0.006
MSE	-0.003	0.012	0.007	0.001	-0.005

Table 7: A summary of the predictive performance differences (see Table 6) between the baseline and "high rainfall" models. The numbers in this table is obtained by comparing the difference between the RMSE value of the baseline model to the "high rainfall" model. Since a higher RMSE value suggests lower predictive accuracy, a positive number in the difference suggests that the "high rainfall" model has performed better in the region's data set on a specific component. Where the "high rainfall model" performed better, the result is marked in grey. Over all the results show promising results of augmenting external data sources to improve predictive performance.

the performance of each model is validated with the root mean square error (RMSE) as shown in Equation 4, where y and $\hat{y}$ are the "overall condition rating" (OCR) predicted by the baseline and the "high rainfall" model. The closer the RMSE value is to 0, the better the model performance is.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \qquad (4)$$

An example of the RMSE values for baseline data and "high rainfall" data is given in Table 5. This data set contains inspection records from the western regions of Victoria. A high RMSE value means that the error is high and therefore lower predictive performance. In Table 5, each component is tested 10 times (folds) on a different set of test records. The predictive errors are then noted and the test conducted on both the baseline model and the "high rainfall" model for each component. Since we have eight data sets, there are eight such table of results from our experiment so Table 5 gives an idea of what the results look like. To determine if the "high rainfall" model actually performs better, we first obtained the average RMSE values across the 10 runs to arrive at the results in Table 6.

Finally, we compute the difference of these average RMSE scores between the baseline and the "high rainfall" model. This difference, for each component across the eight data sets, are shown in Table 7. What we can see from the experiment results is that components 51S, 52O and 55S see good performance improvements while components 24C and 54O seem not to respond well with our method. Looking at the material used for these components (Table 1, we could draw the following conclusions

- 51S and 55S are components made of steel while 24C is a component made of cast-in-Situ concrete. Consistent with the literature, a steel component is more likely to be impacted by rainfall due to water corrosion (than cast-in-Situ concrete).
- 54O is the waterway part of a bridge made of materials other than steel, precast concrete, cast-in-Situ concrete or timber. This means that this material has to withstand the presence of water and therefore the rainfall data should not have any impact.

5 Summary and Future Work

This paper presents our preliminary results on our work with VicRoads to implement a bridge deterioration model to enable the development of a bridge management system. The challenges of the work lie with the noisiness of the data and access to information required for data cleaning is limited. Therefore an alternative approach to improve the data quality is paramount before considering the various predictive models that one can use. Our strategy was to look at augmenting the data set with sources of data that are known to be reliable. In doing so, the proportion of the noisy and erroneous data is reduced, giving the model a better chance of producing good predictive performance. The use of rainfall data from the BoM verifies the possible viability of this approach. On the five major components of interests to bridge inspectors, the overall performance improvement seen in the model that utilises the high rainfall data suggests that this is a plausible direction to take.

In the near term, we will be conducting further studies on the use of rainfall data from the BoM. This would include

- Verifying if the long term annual rainfall average (600mm) is a good determinant of the high rainfall characteristics. We are keen to investigate this long term average because our own study with the rainfall associated with our bridges average at around the 800mm mark rather than 600mm. It would be important to find out how the "high rainfall" model would perform if the cut-off point for selecting the high rainfall subset is raised to 800mm.
- Considering the use of more advanced models rather than Markov chains with this data to see what impacts the augmented data set could bring to different algorithms for building bridge deterioration models.

The outcomes of these two investigation will eventually lead to insights on the best choice of algorithm for model training and the final augmented data set to use, including the exploration of other data sources to add to the data set, e.g., traffic and weather information, which too contribute to bridge deterioration.

References

Aboura, K., Samali, B., Crews, K. & Li, J. (n.d.), Stochastic deterioration processes for bridge lifetime assessment, *in* 'Broadband Communications, Information Technology & Biomedical Applications, 2008 Third International Conference on', IEEE, pp. 437–442.

Bu, G., Lee, J., Guan, H., Blumenstein, M. & Loo, Y.-C. (2012), 'Development of an integrated method for probabilistic bridge-deterioration modeling', *Journal of Performance of Constructed Facilities* **28**(2), 330–340.

Callow, D., Lee, J., Blumenstein, M., Guan, H. & Loo, Y.-C. (2013), 'Development of hybrid optimisation method for artificial intelligence based bridge deterioration modelfeasibility study', *Automation in Construction* **31**, 83–91.

Cesare, M. A., Santamarina, C., Turkstra, C. & Vanmarcke, E. H. (1992), 'Modeling bridge deterioration with markov chains', *Journal of Transportation Engineering* **118**(6), 820–833.

Guyon, I. & Elisseeff, A. (2003), 'An introduction to variable and feature selection', *The Journal of Machine Learning Research* **3**, 1157–1182.

Huang, Q., Hasan, M. S., Ong, K.-L., Alahakoon, D. & Setunge, S. (2014), 'Beyond modelling: Shifting the focus to a holistic development of analytical models for infrastructure problems'.

Huang, R., Mao, I. & Lee, H. (2010), 'Exploring the deterioration factors of rc bridge decks: A rough set approach', *ComputerAided Civil and Infrastructure Engineering* **25**(7), 517–529.

Huang, Y.-H. (2010), 'Artificial neural network model of bridge deterioration', *Journal of Performance of Constructed Facilities* **24**(6), 597–602.

Jiang, Y. & Sinha, K. C. (1989), 'Bridge service life prediction model using the markov chain', *Transportation research record* (1223).

Karunarathna, W., Zhang, T., Dwight, R. & El-Akruti, K. (2013), 'Bridge deterioration modeling by markov chain monte carlo (mcmc) simulation method'.

Kira, K. & Rendell, L. A. (n.d.), A practical approach to feature selection, *in* 'Proceedings of the ninth international workshop on Machine learning', pp. 249–256.

Lee, J., Sanmugarasa, K., Blumenstein, M. & Loo, Y.-C. (2008), 'Improving the reliability of a bridge management system (bms) using an ann-based backward prediction model (bpm)', *Automation in Construction* **17**(6), 758–772.

Madanat, S. & Ibrahim, W. H. W. (1995), 'Poisson regression models of infrastructure transition probabilities', *Journal of Transportation Engineering* **121**(3), 267–272.

Maovi, S. & Hajdin, R. (2014), 'Modelling of bridge elements deterioration for serbian bridge inventory', *Structure and Infrastructure Engineering* **10**(8), 976–987.

Ng, S.-K. & Moses, F. (1998), 'Bridge deterioration modeling using semi-markov theory', *A. A. Balkema Uitgevers B. V, Structural Safety and Reliability.* **1**, 113–120.

Ozbek, M. E., de la Garza, J. M. & Triantis, K. (2010), 'Efficiency measurement of bridge maintenance using data envelopment analysis', *Journal of Infrastructure Systems* **16**(1), 31–39.

Ranjith, S., Setunge, S., Gravina, R. & Venkatesan, S. (2011), 'Deterioration prediction of timber bridge elements using the markov chain', *Journal of Performance of Constructed Facilities* **27**(3), 319–325.

Sobanjo, J. (n.d.), A neural network approach to modeling bridge deterioration, *in* 'Computing in Civil Engineering (1997)', ASCE, pp. 623–626.

Sobanjo, J. O. & Thompson, P. D. (2011), 'Enhancement of the fdot's project level and network level bridge management analysis tools'.

Thompson, P. D. & Johnson, M. B. (2005), 'Markovian bridge deterioration: developing models from historical data', *Structure and Infrastructure Engineering* **1**(1), 85–91.

Thompson, P., Ford, K., Arman, M., Labi, S., Sinha, K. & Shirol, A. (2012), 'Nchrp report 713: Estimating life expectancies of highway assets: Volume 1: Guidebook', *Transportation Research Board of the National Academies, Washington, DC* .

Tokdemir, O. B., Ayvalik, C. & Mohammadi, J. (n.d.), Prediction of highway bridge performance by artificial neural networks and genetic algorithms, *in* 'Proceeding of the 17th International Symposium on Automation and Robotics in Construction (ISARC), September, Taipei, Taiwan'.

Tran, H. D. (2007), Investigation of deterioration models for stormwater pipe systems, Thesis.

Wakchaure, S. S. & Jha, K. N. (2011), 'Prioritization of bridges for maintenance planning using data envelopment analysis', *Construction Management and Economics* **29**(9), 957–968.

Wellalage, N. K. W., Zhang, T. & Dwight, R. (2014*a*), 'Calibrating markov chainbased deterioration models for predicting future conditions of railway bridge elements', *Journal of Bridge Engineering* **20**(2).

Wellalage, N. K. W., Zhang, T. & Dwight, R. (2014*b*), 'Calibrating markov chainbased deterioration models for predicting future conditions of railway bridge elements', *Journal of Bridge Engineering* **20**(2).

Wellalage, N. W., Zhang, T., Dwight, R. & El-Akruti, K. (2015), *Bridge Deterioration Modeling by Markov Chain Monte Carlo (MCMC) Simulation Method*, Springer, pp. 545–556.

Zhao, Z. & Chen, C. (2002), 'A fuzzy system for concrete bridge damage diagnosis', *Computers & structures* **80**(7), 629–641.

An Improved SMO Algorithm for Credit Risk Evaluation*

Jue Wang[1] **Aiguo Lu**[2] **Xuemei Jiang**[1]

[1] Center for Forecasting Science, Academy of Mathematics and Systems Science
Chinese Academy of Sciences 100190, China,
Email: wjue@amss.ac.cn

[2] Department of Applied Mathematics, Xi'an Shiyou University, Xian, 710065, China.

Abstract

Sequential minimal optimization (SMO) is the most commonly used algorithm for numerical solution of SVM, but traditional SMO is quite limited to the long execution time because of its high computational complexity. We present an improved SMO learning algorithm named FV-SMO in this paper. At each iteration, it jointly optimizes four variables and an theorem is proposed to guarantee an analytical solution of sub-problem. Three credit datasets are selected to demonstrate the performance of FV-SMO, including China credit dataset and two public datasets: German credit dataset from UCI and Darden credit dataset from CD-ROM databases. China credit dataset is generated based on a multi-dimensional and multi-level credit risk indicator system of China credit data. Experimental results demonstrate that FV-SMO is competitive in saving the computational cost and performs best in credit risk evaluation accuracy compared with other five popular classification methods.

Keywords: sequential minimal optimization (SMO), SVM, four-variable working set, credit risk.

1 Introduction

With the diversification of products and services in the fast-moving financial markets, credit risk management is one of the most important issues in the banking industry. Nowadays, the credit risk and its transmission shows complicated characteristics, an early-warning and fast response mechanism becomes the central part in credit risk management. Credit risk evaluation which helps make right credit granting decisions is a vital link in this mechanism. Efficient and accurate credit risk evaluation models based on credit data has attracted increasing attention from both academic and financial institutions and is also inevitable requirement of the meticulous management trend in big data era. Speciailly in China, the China Banking Regulatory Commission (CBRC), who is responsible for regulation of financial industry in China, enables a reporting system for data submission from 2004. In recent years, CBRC has attached much importance to credit risk features mining and risk evaluation models.

Generalized credit risk refers to the risk that a bank borrower or a counterparty fails to meet its obligations in accordance with the agreed terms , so it can be viewed as a classification problem. Numerous methods have been proposed in the literature to develop an accurate classifier model to predict the default risk. Many statistic and optimization models, such as discriminant analysis, logistic regression, linear programming, integer programming and k-nearest neighbor are widely applied. As these traditional models usually fail to capture enough information of nonlinear credit data, recent studies have focused on the research of artificial intelligent (AI) techniques for credit evaluation, such as artificial neural networks (ANN) (Khashman, A 2009, 2011), rough set (You-Shyang Chen 2013), multi-criteria optimization classifier (Zhi 2014), extreme learning machine (Zhou 2012, Jan 2014), feature selection algorithms (Petr 2013, J 2009, Se 2014) and so on.

Support vector machine (SVM) is a promising approach for credit risk evaluation (Se 2014, Terry 2013, Yao 2015). The sequential minimal optimization (SMO) algorithm developed by Platt (Platt 1998) is one of the most efficient solutions for the SVM training phase. It is derived by solving a series of small QP problems, where in each iteration only two variables are selected in the working set, as these small QP problems are solved analytically such as to avoid a time-consuming numerical QP method. The technique is popular and a lot of scholars make efforts to improve on it (Song 2009, lin 2011, Xi 2011, PaiHsuenChen 2006, Keerthi 2001, Cheng 2007). For example, Song (Song 2009) proposes a new strategy by selecting several greatest violating samples set for the next several optimizing steps. Chen(PaiHsuenChen

*This work is supported by Chinese Academy of Sciences(CAS) Foundation for Planning and Strategy Research(KACX1-YW-0906) and the National Natural Science Foundation of China (NSFC No.71271202).

2006) gives a study on SMO-type decomposition methods of choosing the two-element working set under a general and flexible way, which is called Chen-SMO in this paper. However, these improved algorithms still require expensive computational cost in working set selection.

In order to improve the above problem, we present a novel algorithm named FV-SMO in the paper. It is derived by solving a series of the QP problems with four points at each iteration via the way of "maximal violating pair". These QP problems are solved analytically so FV-SMO approaches the optimal solution more quickly. Moreover, a theorem is introduced on SVM-training to guarantee the existence of analytical solution of corresponding sub-problem.

Based on the practical credit data from CBRC, a multi-dimensional and multi-level China's credit risk indicator system is proposed for the first time in this paper, mainly considering macroeconomic environment, enterprises' management ability and credit transaction behavior. Three credit datasets are used, China credit dataset is generated on the credit risk indicator system. German and Darden credit datasets are publicly available. For the numerical experiments, FV-SMO is compared with Chen-SMO in the computational cost. Then FV-SMO is applied for credit risk evaluation compared with other five popular classification approaches. All the experimental results confirm the superiority of FV-SMO in computational cost and classification accuracy.

This paper is organized as follows. Section 2 presents the improved SMO algorithm FV-SMO. Section 3 shows the credit risk indicator system and dataset generation process. Followed by the numerical experiments and result analysis in Section 4. Finally, the conclusion and future research makes up Section 5.

2 An improved SMO algorithm based on four-variable working set

2.1 Solving the four-variable SVM subproblem

Assuming the working set of four-variables as $B = \{i_1, j_1, i_2, j_2\}$, and relatively the non-working set is $N = \{1, \cdots, l\} - B$, α, Q, e and Y can be decomposed:

$$\alpha = \begin{bmatrix} \alpha_B \\ \alpha_N \end{bmatrix}, Q = \begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix}$$

$$e = \begin{bmatrix} e_B \\ e_N \end{bmatrix}, Y = \begin{bmatrix} Y_B \\ Y_N \end{bmatrix}$$

This problem is equivalent to the following sub-problem:

$$\begin{aligned} &\min w(\alpha) = \tfrac{1}{2}\alpha_B^T Q_{BB}\alpha_B + (Q_{BN}\alpha_N - e_B)^T\alpha_B + const \\ &= \tfrac{1}{2}\begin{bmatrix}\alpha_{i_1} & \alpha_{j_1} & \alpha_{i_2} & \alpha_{j_2}\end{bmatrix}\begin{bmatrix} Q_{i_1i_1} & Q_{i_1j_1} & Q_{i_1i_2} & Q_{i_1j_2} \\ Q_{i_1j_1} & Q_{j_1j_1} & Q_{j_1i_2} & Q_{j_1j_2} \\ Q_{i_2i_1} & Q_{i_2j_1} & Q_{i_2i_2} & Q_{i_2j_2} \\ Q_{j_2i_1} & Q_{j_2j_1} & Q_{j_2i_2} & Q_{j_2j_2} \end{bmatrix}\begin{bmatrix}\alpha_{i_1} \\ \alpha_{j_1} \\ \alpha_{i_2} \\ \alpha_{j_2}\end{bmatrix} \\ &+(Q_{BN}\alpha_N - e_B)^T\begin{bmatrix}\alpha_{i_1} \\ \alpha_{j_1} \\ \alpha_{i_2} \\ \alpha_{j_2}\end{bmatrix} + const \\ &s.t \quad y_{i_1}\alpha_{i_1} + y_{j_1}\alpha_{j_1} + y_{i_2}\alpha_{i_2} + y_{j_2}\alpha_{j_2} = -Y_N^T\alpha_N \\ &\quad 0 \le \alpha_{i_1}, \alpha_{j_1}, \alpha_{i_2}, \alpha_{j_2} \le C \end{aligned} \tag{1}$$

When solving (1), Chen (PaiHsuenChen 2006) studied sequential minimal optimization type decomposition methods under a general and flexible way of choosing the two-element working set, the iterative relationship is introduced as:

$$\alpha_i^{k+1} = \alpha_i^k - y_i d, \quad \alpha_j^{k+1} = \alpha_j^k + y_j d \tag{2}$$

Based on Chen's idea, we consider the iterative relationship:

$$\begin{aligned} \alpha_{i_1}^{k+1} &= \alpha_{i_1}^k - y_{i_1}d_1, \quad \alpha_{j_1}^{k+1} = \alpha_{j_1}^k + y_{j_1}d_1 \\ \alpha_{i_2}^{k+1} &= \alpha_{i_2}^k - y_{i_2}d_2, \quad \alpha_{j_2}^{k+1} = \alpha_{j_2}^k + y_{j_2}d_2 \end{aligned} \tag{3}$$

so (1) is rewritten as:

$$\begin{aligned} &\min w(\alpha) = \tfrac{1}{2}\begin{bmatrix}d_1 & d_2\end{bmatrix}\begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22}\end{bmatrix}\begin{bmatrix} d_1 \\ d_2\end{bmatrix} - \begin{bmatrix}b_1 & b_2\end{bmatrix}\begin{bmatrix} d_1 \\ d_2\end{bmatrix} \\ &s.t \quad l_1 \le d_1 \le u_1, \quad l_2 \le d_2 \le u_2 \end{aligned} \tag{4}$$

where

$$\begin{aligned} a_{11} &= K_{i_1i_1} + K_{j_1j_1} - 2K_{i_1j_1} \\ a_{12} &= K_{i_1i_2} - K_{i_1j_2} - K_{j_1i_2} + K_{j_1j_2} \\ a_{22} &= K_{i_2i_2} + K_{j_2j_2} - 2K_{i_2j_2} \\ b_1 &= y_{i_1}\nabla w(\alpha^k)_{i_1} - y_{j_1}\nabla w(\alpha^k)_{j_1} \\ b_2 &= y_{i_2}\nabla w(\alpha^k)_{i_2} - y_{j_2}\nabla w(\alpha^k)_{j_2} \end{aligned}$$

Note $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22}\end{bmatrix}$, Let A be a symmetric positive definite matrix.

Given $y_{i_1} = y_{j_1} = 1$, since $0 \le \alpha_{i_1}^k - y_{i_1}d_1,\ \alpha_{j_1}^k + y_{j_1}d_1 \le C$, so $l_1 = \max(-\alpha_{j_1}^k, \alpha_{i_1}^k - C)$, $u_1 = \min(C - \alpha_{j_1}^k, \alpha_{i_1}^k)$. For other values of y_{i_1}, y_{j_1}, l_1, u_1 has similar results. l_2, u_2 are available by the same analysis. By Theorem 1, problem (1) has the following optimal solution:

when $a_{12} \ge 0$,

$$\begin{cases} d_1^* = \min(\max(l_1, \overline{d_1}, \frac{b_1 - a_{12}u_2}{a_{11}}), \max(\frac{b_1 - a_{12}l_2}{a_{11}}, l_1), u_1) \\ d_2^* = \min(\max(l_2, \overline{d_2}, \frac{b_2 - a_{12}u_1}{a_{22}}), \max(\frac{b_2 - a_{12}l_1}{a_{22}}, l_2), u_2) \end{cases}$$

when $a_{12} < 0$,

$$\begin{cases} d_1^* = \min(\max(l_1, \overline{d_1}, \frac{b_1 - a_{12}l_2}{a_{11}}), \max(\frac{b_1 - a_{12}u_2}{a_{11}}, l_1), u_1) \\ d_2^* = \min(\max(l_2, \overline{d_2}, \frac{b_2 - a_{12}l_1}{a_{22}}), \max(\frac{b_2 - a_{12}u_1}{a_{22}}, l_2), u_2) \end{cases}$$

where $\overline{d_1} = \frac{b_1 a_{22} - b_2 a_{12}}{det(A)}$, $\overline{d_2} = \frac{-b_1 a_{12} + b_2 a_{11}}{det(A)}$.

Finally, the solution of (1) is:

$$\alpha_{i_1}^{k+1} = \alpha_{i_1}^k - y_{i_1} d_1^*, \quad \alpha_{j_1}^{k+1} = \alpha_{j_1}^k + y_{j_1} d_1^*$$
$$\alpha_{i_2}^{k+1} = \alpha_{i_2}^k - y_{i_2} d_2^*, \quad \alpha_{j_2}^{k+1} = \alpha_{j_2}^k + y_{j_2} d_2^*$$

2.2 An improved SMO algorithm based on a four-variable working set

Because the maximal violating pair (MVP) in (Keerthi 2001) is a popular way to select the working set, we choose the working set B via MVP :

1)
$$i_1 \in \arg \max_{t \in I_{up}(\alpha)} -y_t \nabla w(\alpha)_t$$
$$j_1 \in \arg \min_{t \in I_{low}(\alpha)} -y_t \nabla w(\alpha)_t$$
$$i_2 \in \arg \max_{t \in (I_{up}(\alpha) \backslash \{i_1, j_1\})} -y_t \nabla w(\alpha)_t$$
$$j_2 \in \arg \min_{t \in (I_{low}(\alpha) \backslash \{i_1,\ j_1,\ i_2\})} -y_t \nabla w(\alpha)_t$$

2) $B = \{i_1, j_1, i_2, j_2\}$

To sum up, the FV-SMO formal algorithm can be stated as follows:

Algorithm FV-SMO
Given dataset: $x_i, y_i,\ i = 1, 2, ...n$
Result: α_i solved by Lagrange multiplier method
While it does not reach convergence, do:

step1: Given $\varepsilon > 0$ and $\alpha^0 = 0$. Set $k = 0$.
step2: If $m(\alpha^k) - M(\alpha^k) \leq \varepsilon$, stop; Otherwise by the above MVP to find a four-variable working set $B = \{i_1, i_2, j_1, j_2\}$.
Define $N \equiv \{1, \cdots, l\} - B$, α_B^k and α_N^k as sub-vectors of corresponding to B and N, respectively.
step3: Gradient update: $\nabla w(\alpha^{k+1}) = \nabla w(\alpha^k) + ((-K(:, i_1) + K(:, j_1))d_1^* + (-K(:, i_2) + K(:, j_2))d_2^*)Y$.
step4: Set $k = k + 1$ and go to step2.

end

3 China's credit risk indicator system and dataset generation

3.1 China credit risk indicator system

Nowadays, China's economy is in the reform and structural adjustment process, the banking institutions' risk management ability becomes essentially important as it matters the property safety of billions of people. However, traditional management system has been unable to deal with many new problems. Since 2004, CBRC has established an assessment system for monthly monitoring the customers' loan behaviors.

Customer' risk recognition is the precondition of the management system. A multi-dimensional and multi-level credit risk indicator system is constructed for mining the possible characteristics. First of all, the external factors are explored. The majority of China's credit today is accumulated in fields of government infrastructure, real estate construction and large state-owned enterprises. This leads to complicated causal relationship between credit risk, macroeconomic and monetary policy. It is reasonable to take macroeconomic factors as priority into consideration, such as GDP growth rate, M2 growth rate, interest rate and so on. Meanwhile, the overall situation of the industry closely relates to its behavior of credit data, three specific indicators are extracted: industry profit margin, concentration of loan investment and default rate of industry.

Secondly, we focus on the enterprises' management ability. The indicators related to operation situation, solvency and credit level are explored. Taking operation situation into consideration, enterprises' capital scale and long-term viability of operation can be figured out. Generally, the operation situation can be expressed by measuring market capitalization, assets, cash flow and others. Here key financial indicators are investigated, such as asset scale, debt ratio, current ration and so on.

Thirdly, the credit data from CBRC makes the enterprises' transaction data mining possible. Two aspects including loan behavior and association risk are mined. The customers' past loan behavior acts as the most convincing evidence for determining whether a customer should be granted good credit or not. The quantity/quality of loan, the lending bank information and credit extension are explored. In addition, The credit interconnection among enterprises are getting increasingly closer in recent years. The credit association promotes enterprises to share capital, acquire excessive credit or escape from risk investigation. These behaviors increase the difficulty of credit regulation and needs particular attention. Four major association relationship are investigated, including legal person, guarantee, stockholder and other business associates.

Finally, the whole indicator system has three dimension: external factors, management ability and trading behaviors. Then we look into the three dimension at the first level by 14 second level indicators including: macroeconomic, industry, region, operation situation, solvency, credit level, quantity, quality, bank, legal person, stockholder, business associates and guarantee. At last, a third level is extended including 124 detailed indicators. The 124 indicators are not only extracted from the original data. There are also many derived indicators, such as regional default rate, loan bank concentration and so on.

3.2 Generation process of China credit dataset

After the credit risk indicator system analysis, China credit dataset is extracted from the CBRC's credit data. Data preprocessing is implemented for converting the primary data to format. The techniques including cleaning, integration, transformation are used to correct the dirty, incomplete and inconsistent data. Another important issue that needs to be clarified is the definition of default risk, we define a customer default if it is behind hand with its payment for more than three months. Once a default occurs, the customer is marked as a positive sample.

The purpose of feature selection is to filter out unrepresentative features from a given dataset, which is critical for a successful credit default classification model. From the credit risk indicator system as stated in Section 3, T-test and Wilcoxon signed ranks tests are used to distinguish indicators objectively, the criterion is whether the indicator changes significantly prior to the default occurs, 54 indicators are picked out by the single indicator test. Next, stepwise regression pares down the these indicators to eliminate the collinearity. Finally, 17 most representative indicators are chosen for the evaluation model.

After data preprocessing and feature selection, the sample of China credit dataset has 60126 instances, 1822 default (positives) and 58304 non-default (negatives). The number of negatives is almost 32 times the size of positives. Since high imbalance could seriously affect the model performance, downsample method is adopted to construct subsample. Under one to one ratio, 1822 instances are picked out randomly from the sample of negatives. In order to use more information, we repeated sampling process for ten times and got ten subsamples for modeling.

4 Numerical Experiments

4.1 Dataset description

At the beginning, experiments are conducted on China credit dataset as stated in Section 3. China credit dataset contains the whole 3644 samples and 17 indicators, its positives and negatives are balanced.

Another two public real world datasets are also introduced for the credit risk evaluation. German credit consists of 700 examples of creditworthy applicants and 300 examples where credit should not been extended. For each applicant, 24 indicators describe the individual credit history, age, loan amount, account balances, loan purpose, job title, and so on.

For the Darden corporate credit dataset, it is selected from the CD-ROM database and includes 132 companies (66 non-risk cases and 66 risk cases). A total of 25 financial variables are computed for each of the 132 companies using data from the Compustat and from the Moodys Industrial Manual. The information of all datasets is shown in Table 1.

Table 1: Description of testing datasets

Datasets	Number	Negative	Positive	Indicators
China credit	3644	1822	1822	17
German credit	1000	700	300	24
Darden credit	132	66	66	24

4.2 Evaluation criteria

Given a classifier and an instance, there are four possible outcomes: if the instance is positive and it is classified as positive, it is counted as a true positive (TP); if the instance is negative and it is classified as positive, it is counted as a false positive (FP), the definition of TN and FN is the same. Six accuracy criteria are used to evaluate the performance of the classifier, which are defined as follows.

(i) The total classification accuracy rate

$Total\ accuracy = \frac{TP+TN}{TP+FP+TN+FN}$

(ii) The identification rate of "bad" creditors

$Type1\ accuracy = \frac{TP}{TP+FN}$

(iii) The identification rate of "good" creditors

$Type2\ accuracy = \frac{TN}{TN+FP}$

(iv) How accurately of "bad" creditors have been classified

$Precision = \frac{TP}{TP+FP}$

(v) The mixed measure of classification

$F1 - measure = \frac{2}{\frac{1}{Precision}+\frac{1}{Recall}}$

4.3 Complexity comparison

To assess the complexity efficiency of the FV-SMO algorithm , RBF kernel-$K(x,y) = \exp(-\gamma * \|x-y\|^2)$ is chosen to optimize accuracy . $\varepsilon = 1 \times 10^{-10}$ is used as the stopping condition. As the parameters of C and γ will affect the performance of the algorithm, the grid-search algorithm and ten validation are implemented to search the optimal value of (C, γ). The computational cost of the two algorithms in terms of the number of iterations and execution time are compared with Chen-SMO algorithm. All experiments are run on PC (Intel(R)core(TM)5/RAM4.0GB) in MATLAB.2014.a.

Table 2 shows the complexity comparison of Chen-SMO and FV-SMO on the testing datasets. We can see FV-SMO consistently has fewer iterations and shorter training time than Chen-SMO with the same optimal parameters setting. For example, on China credit dataset, the execution time shown in "T" column are 3.250 for Chen-SMO and 2.766 for FV-SMO, yielding a ratio of 1.18, as shown in the "rT" column. Respectively, the numbers of iterations are 7118 for Chen-SMO and 3487 for FV-SMO, yielding

Table 2: Executing complexity and time performance comparison on the three datasets with RBF kernel

Parameters settings	Model	T(s)	rT	Iter	rIter
China Credit	Chen-SMO	3.250	1.00	7118	1.00
C=90.51, $g = 2.8284$	FV-SMO	2.766	1.18	3487	2.04
German credit	Chen-SMO	1.516	1.00	3773	1.00
C=181.01, $g = 0.0009$	FV-SMO	1.404	1.07	2948	1.28
Darden	Chen-SMO	2.075	1.00	4001	1.00
C= 5.66, $g = 0.0078$	FV-SMO	1.919	1.08	2570	1.56

a ratio of 2.04 in the "rIter" column. As for other datasets, similar conclusions can be made. For the three datasets, the training time is reduced by minimum 8% of German credit dataset to maximum 15% of China credit dataset and the iterations number is reduced by 35.7% of German dataset to 51% of China dataset. This proved the proposed algorithm of FV-SMO is more effective in the computational cost.

As m-M can reflect the optimizing convergence rate for current iterating rate more intuitive, Figs.3-5 depicts the curves of m-M versus the iterating steps. The curves with bule and red colour are the results of Chen-SMO and FV-SMO. In most iterating steps, we can see the declines of objective m-M in FV-SMO are superior to that in Chen-SMO. The convergence speed of FV-SMO is significantly faster than that of Chen-SMO, which once again demonstrates the proposed FV-SMO outperforms Chen-SMO in the sense of faster convergence .

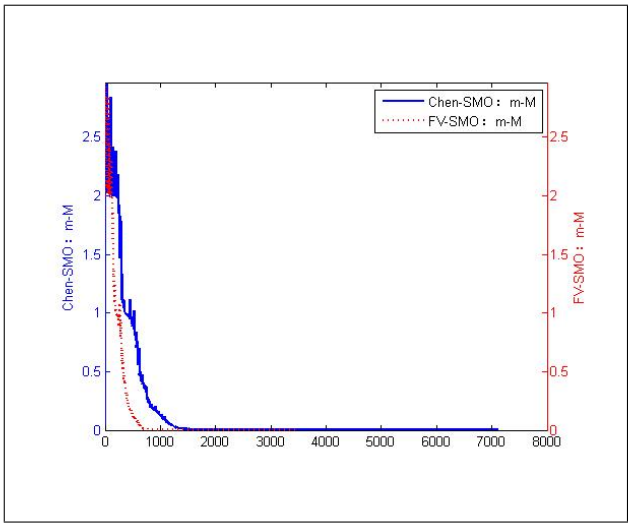


Figure 1: The change of m-M with iterating steps: China credit

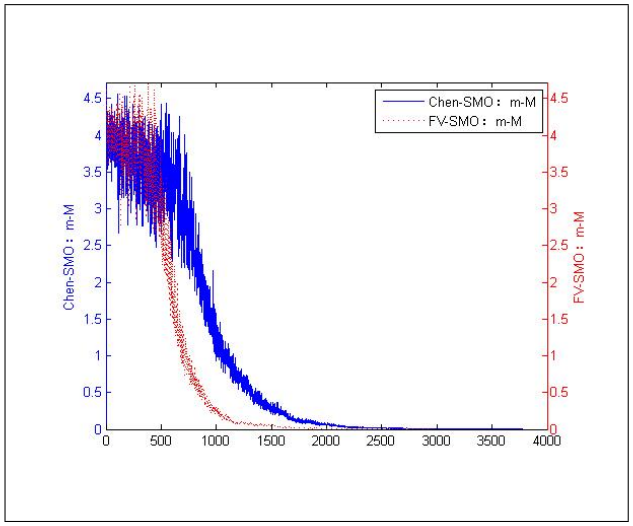


Figure 2: The change of m-M with iterating steps: German credit

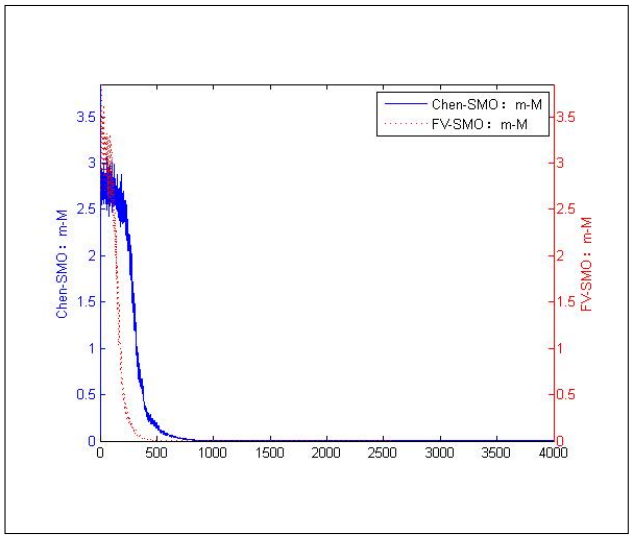


Figure 3: The change of m-M with iterating steps: Darden credit

4.4 Accuracy comparison

Classification accuracy is the basic and decisive aspect in choosing the credit classification model. In order to check the performance of FV-SMO, FV-SMO with five other major popular classification approaches are involved in accuracy comparison. The approaches includes: RBF, Multiplayer-perception, Baysenet, J48 tree, and Logisitic. The results are shown in Table 3-5.

First, the Total accuracy of FV-SMO generally outperforms other classifiers for both China and German datasets, followed by MLP for China dataset and Bayesnet for German dataset. But for Darden dataset, FV-SMO is second only to Logistic. In terms of Type1 accuracy, FV-SMO is superior to other classifiers for China and Darden datasets, ranking second (0.453) on German dataset, the best is Bayenet (0.483). Then from the Type2 accuracy viewpoint, RBF has the best performance (0.887) for China dataset and FV-SMO has the best performance

(0.903) for German dataset. FV-SMO has a relatively poor performance (0.622) compared to best result of Logisitic (0.819) for Darden dataset.

For the measurement of Precision and F1-measure, FV-SMO also yeilds a very good performance, except Precision of FV-SMO ranks second with a tiny gap (0.004) behind RBF on China dataset. F1-measure of FV-SMO ranks second behind Logistic on Darden dataset. For other comparisons, FV-SMO all ranks first. The area under the receiver operating characteristic ROC curve is applied as another performance measurement. Figure 6 to 8 show the performance of the ROC curve for the three datasets. It is obvious that FV-SMO has a better performance of ROC than the others.

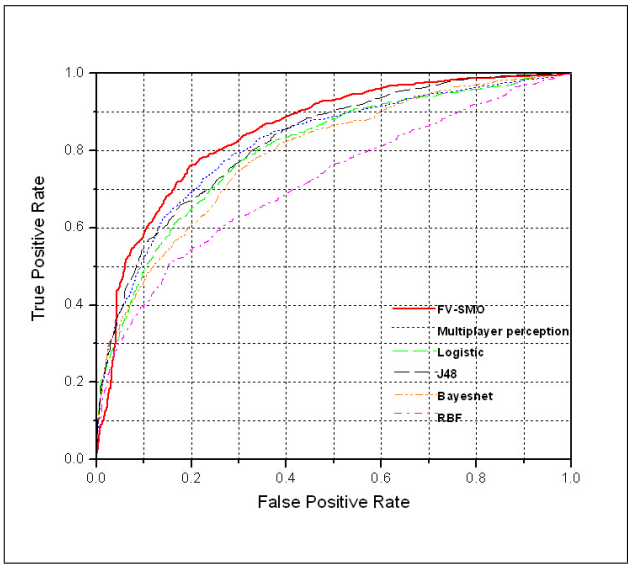


Figure 4: ROC comparison for different models in China dataset

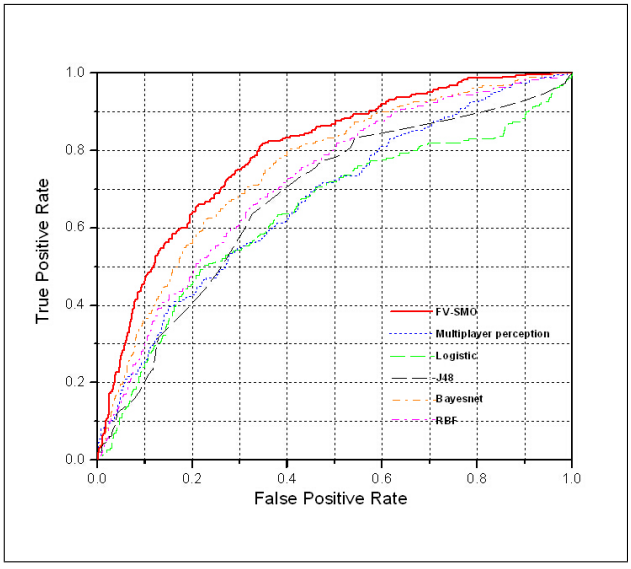


Figure 5: ROC comparison for different models in German dataset

Second, compared with the empirical results of the three datasets, we can find that Type2 accuracy is

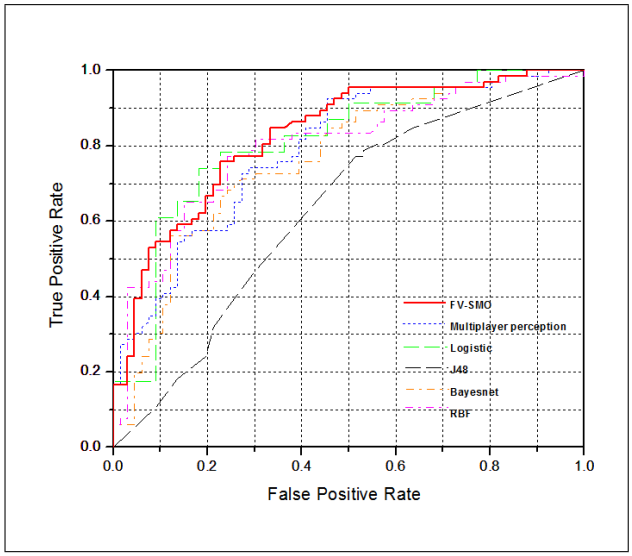


Figure 6: ROC comparison for different models in Darden dataset

better than Type1 accuracy for China and German datasets, which means it is more difficult to catch the "bad" creditors from all the applicants, especially for the unbalanced dataset of German. But the result is inconsistent on Darden dataset. There are two possible reasons. The first reason is different credit markets have different credit characteristics and the second possible reason is that there is more nonlinearity of China and German datasets than Darden dataset.

Third, there is another interesting finding of Type1 accuracy and Type2 accuracy. For example, RBF ranks first of Type2 accuracy (0.887), but performs the worst of Type1 accuracy (0.419) on China dataset. For German dataset, MLP and J48 have a poor performance in terms of Type1 accuracy, only 0.29 and 0.21, but get quite high performance of Type2 accuracy, 0.889 and 0.896, rank top three with a slight difference to the FV-SMO. We can conclude some classifiers have the tendency to get a high recognition rate of majority class by predicting most samples as the "good" ones, especially on the imbalanced dataset, making the classifiers not suitable for the credit risk evaluation. As the recognition rate of minority class is the most important in creidt risk evaluation, FV-SMO is affected at least.

From the above analysis, it can be concluded the proposed method of FV-SMO performs the best in comparison with the other five popular classification approaches.

5 Conclusion and future work

In this paper, we presented a novel SMO learning algorithm on a four-variable working set for classification model and applied it to three credit datasets. Comparisons with Chen-SMO on three datasets reveal that FV-SMO outperforms Chen-SMO significantly in the computational cost. Specifically, the

iterations number is reduced by 35.7% to 51% and execution time is reduced by 8% to 15%. Experimental results also reveal that FV-SMO gets the best performance in the classification accuracy, which provides compelling evidence of the advantages of FV-SMO.

Several future research directions also emerge. Firstly, for the serious skewed origin dataset of China credit dataset, the simple method of dowmsample is adopted, but this will quite limit the exploration of real characteristics and structure of the nonlinear data, more effective strategy will be explored in next step. Secondly, a major limitation of learning methods is in lack of interpretability of the results, interpretability in practical work of the bank operation is quite important, the decision-makers want to know the focus indicators to control the risk and the customer have the rights to know why they are rejected, so improving the interpretability of model is another important understudied direction. Thirdly, the application of FV-SMO discussed in this paper is limited to the binary classification type, and we will extend the algorithm to solve the problem of multi-class and regression problem in future research.

6 References

References

Khashman, A. (2009), 'A neural network model for credit risk evaluation', *International Journal of Neural Systems* **19**, 285–294.

Khashman, A. (2011), 'Credit risk evaluation using neural networks: Emo-tional versus conventional models', *Applied Soft Computing* **11**(8), 5477–5484.

You-Shyang Chen. & Ching-Hsue Cheng.(2013), 'Hybrid models based on rough set classifiers for setting credit rating decision rules in the global banking industry', *Knowledge-Based Systems* **39**, 224–239.

Zhiwang Zhang., Guangxia Gao & Yong Shi. (2014), 'Credit risk evaluation using multi-criteria optimization classifier with kernel, fuzzification and penalty factors', *European Journal of Operational Research* **237**(1), 335–348.

Hongming Zhou. & Yuan Lan . (2014), 'Credit risk evaluation with extreme learning machine', *IEEE International Conference on Systems, Man, and Cybernetics* , 1064–1069.

Jan Chorowski., Jian Wang & JacekM.Zurada. (2014), 'Review and performance comparison of SVM- and ELM-based classifiers', *Neurocomputing* **128**, 507–516.

Petr Hajek.& Krzysztof Michalak. (2013), 'Feature selection in corporate credit rating predictio', *Knowledge-Based Systems journal* **51**, 72–84.

J.VanHulse & T.M.Khoshgoftaar. (2009), 'Feature selection with high-dimensional imbalanceddata', *In: Proceedings of the IEEE International Conference on Data Mining Workshops* , 507–514.

Sebastin Maldonado., Richard Weber. & Fazel Famili. (2014), 'Feature selection for high-dimensional class-imbalanced data sets using support vector machines', *Information Sciences* **286**, 228–246.

Yu L.,Yao X. & Wang S. (2011), 'Credit risk evaluation using a weighted least squares SVM classifier with design of experiment for parameter selection', *Expert Systems with Applications* **38**(12), 15392–15399.

Gang Wang.& Jian Ma. (2012), 'A hybrid ensemble approach for enterprisecredit risk assessment based on support vector machine', *Expert Systems with Applications* **39**(5), 5325–5331.

Terry Harris. (2013), 'Quantitative credit risk assessment using support vector machines: broad versus narrow default definitions', *Expert Systems with Applications* **40**, 4404–4413.

Xiao Yao., Jonathan Crook. & Galina Andreeva. (2015), 'Support vector regression for loss given default modelling', *European Journal of Operational Research* **240**, 528–538.

Platt J. (1998), 'Sequential minimal optimization: A fast algorithm for training support vector machines.Microsoft Research', *Technical Report MSR-TR-98-14*.

X.F.Song. (2009), 'Candidate working set strategybased SMO algorithm in support vector machine', *Information Processing and Management* **45**(5), 584–592.

Yih-Lon Lin. & Jer-Guang Hsieh. (2011), 'Three-parameter sequential minimal optimization for support vectormachines', *Neurocomputing* **74**(17), 3467–3475.

Xili Zhang.,Wei-Guo Zhang. & Wei-Jun Xu. (2011), 'An optimization model of the portfolio adjusting problem with fuzzy return and a SMO algorithm', *Expert Systems with Applications* **38**, 3069–3074.

Chen Pai hsuen.,Fan Rong en.& Lin Chih jen (2011), 'A study on SMO-type decomposition methods for support vector machines', *IEEE Transactions on Neural Networks* **17**(4), 893–908.

Keerthi S, Shevade S K, Bhattacharyya C.& Murthy K (2011), 'Improvements to Platt's SMO algorithm for SVM classifier design', *Neural Computation* **13**, 637–649.

S.Cheng.& F.Y.Shih (2007), ' An improved incremental training algorithm for support vector machines using active query', *Neural Computation* **40**(3), 964–971.

Table 3: Performance comparison for different models in China dataset

Model	Total Accuracy(%)	rank	Type1 Accuracy(%)	rank	Type2 Accuracy(%)	rank
RBF	0.653	6	0.419	6	**0.887**	1
Multiplayer-perception	0.75	2	0.741	2	0.759	5
Bayesnet	0.705	5	0.608	5	0.803	2
J48	0.734	3	0.695	4	0.773	4
Logistic	0.729	4	0.709	3	0.749	6
FV-SMO	**0.778**	1	**0.768**	1	0.788	3
	Precision(%)	rank	F1-measure(%)	rank	ROC curve space	rank
RBF	**0.788**	1	0.547	6	0.718	6
Multiplayer-perception	0.755	3	0.748	2	0.845	2
Bayesnet	0.755	4	0.673	5	0.78	5
J48	0.754	5	0.723	4	0.825	3
Logistic	0.739	6	0.724	3	0.803	4
FV-SMO	0.784	2	**0.776**	1	**0.849**	1

Table 4: Performance comparison for different models in German dataset

Model	Total Accuracy(%)	rank	Type1 Accuracy(%)	rank	Type2 Accuracy(%)	rank
RBF	0.717	3	0.43	3	0.84	5
Multiplayer-perception	0.709	4	0.29	5	0.889	3
Bayesnet	0.735	2	**0.483**	1	0.843	4
J48	0.692	6	0.217	6	0.896	2
Logistic	0.701	5	0.45	4	0.809	6
FV-SMO	**0.768**	1	0.453	2	**0.903**	1
	Precision(%)	rank	F1-measure(%)	rank	ROC curve space	rank
RBF	0.535	3	0.477	3	0.719	3
Multiplayer-perception	0.527	4	0.374	5	0.671	5
Bayesnet	0.569	2	0.523	2	0.755	2
J48	0.471	6	0.297	6	0.676	4
Logistic	0.502	5	0.475	4	0.641	6
FV-SMO	**0.667**	1	**0.54**	1	**0.794**	1

Table 5: Performance comparison for different models in Darden dataset

Model	Total Accuracy(%)	rank	Type1 Accuracy(%)	rank	Type2 Accuracy(%)	rank
RBF	0.742	3	0.848	2	0.742	2
Multiplayer-perception	0.689	5	0.758	4	0.621	5
Bayesnet	0.72	4	0.712	6	0.727	3
J48	0.629	6	0.788	3	0.47	6
Logistic	**0.778**	1	0.739	5	**0.819**	1
FV-SMO	0.743	2	**0.864**	1	0.622	4
	Precision(%)	rank	F1-measure(%)	rank	ROC curve space	rank
RBF	0.767	3	**0.806**	1	0.821	2
Multiplayer-perception	0.667	5	0.709	5	0.791	4
Bayesnet	0.723	4	0.718	4	0.766	5
J48	0.598	6	0.68	6	0.63	6
Logistic	0.81	2	0.77	3	0.808	3
FV-SMO	**0.864**	1	0.773	2	**0.826**	1

An Industrial Application of Rotation Forest: Transformer Health Diagnosis

Tamilalagan Natarajan **Duc-Son Pham** **Mihai Lazarescu**

Department of Computing,
Curtin University,
Perth, Western Australia 6102,
Email: t.natarajan@postgrad.curtin.edu.au

Abstract

We introduce a new approach to solve industrial problems related to equipment health diagnosis. We propose an application of Rotation Forest for diagnosing transformer health conditions. A new and diverse data set is introduced to obtain statistically significant results, which mostly lack in previous works. This new dataset includes real data from transformers used in oil refineries, gas plants and other industries, located in different geographical locations. The paper studies the effectiveness of using feature selection for fault classification in industrial equipment. Principal component analysis is used for feature extraction, followed by the application of a decision tree ensemble for classification. Experiments on various datasets prove that the performance of Rotation Forest is superior to the performance of other artificial intelligence-based techniques such as support vector machines and artificial neural networks. This approach can be adopted in future for fault detection in other types of industrial equipments such as turbines and pumps.

Keywords: dissolved gas analysis, feature selection, Rotation Forest

1 Introduction

Transformer insulating oils are used in transformers for insulation and as a coolant. Transformer electrical faults and thermal stresses break down the insulating oil, which results in the formation of solid and liquid particles, contaminants and small quantities of gases. Moisture and acids are formed due to the thermal and chemical reactions and decomposition. Some of these gases, solid and liquid particles and contaminants remain dissolved in the oil. The concentration of the gases, and solid and liquid particles, is used to detect transformer faults by dissolved gas analysis (DGA) and some of these gases (key gases) are useful in fault detection (*IEEE Guide for the Interpretation of Gases Generated in Oil-Immersed Transformers* 2009),(Duval & dePabla 2001),(IEC 2007). Dissolved gas analysis (DGA) is the most popular, widely used and important method to determine the health condition of the transformer for two main reasons: firstly, the oil sampling process for DGA is a non-intrusive process and does not involve shutting down the transformer; and secondly, DGA can reveal a wide range of faults.

When the key gas concentration levels exceed recommended threshold values (*IEEE Guide for the Interpretation of Gases Generated in Oil-Immersed Transformers* 2009), the transformer is considered to be faulty or unhealthy. The key gases, as given by (*IEEE Guide for the Interpretation of Gases Generated in Oil-Immersed Transformers* 2009) and (IEC 2007) are H2, C2H2, CH4, C2H4, C2H6, CO and CO2. There are different methods to identify the fault types: Duval's triangle, Rogers's ratio (and other ratio based methods) and the key gas method are the most popular of these. Industry guidelines (*IEEE Guide for the Interpretation of Gases Generated in Oil-Immersed Transformers* 2009) and (IEC 2007) and research publications (Rogers 1978) (Duval 2003)(Duval & dePabla 2001)(Wang et al. 2002) provide guidance on the use of concentrations of key gases to determine the transformer fault.

Some of the transformers may be on the border between healthy and faulty conditions: i.e. a small change in gas concentration levels can push the transformer into the faulty category. However, at the time of measurement and test, the transformer may still be healthy as per the ratio methods and on the basis of threshold levels. This problem is addressed by the use of a transformer health index where the transformers are grouped according to their health condition, instead of being simply classified as faulty or healthy. The health index is one of the concepts that have been gaining popularity in recent research (Lapworth 2002),(Ashkezari et al. 2013), (Jahromi et al. 2009),(Scatiggio & Pompili 2013),(Abu-Elanien et al. 2012).

During the last two decades, significant interest has been shown in the application of artificial intelligence-based techniques to determine transformer health condition. Artificial intelligence-based techniques such as fuzzy logic (Abu-Elanien et al. 2012), neural networks (Seifeddine et al. 2012)and support vector machines (SVM) (Ashkezari et al. 2013) have been applied to determine transformer health.

Despite continuing research work, both conventional and artificial intelligence based techniques still suffer from limitations and drawbacks, such as inaccurate and inconsistent diagnosis (Ashkezari et al. 2012), inaccurate classification (both conventional and artificial intelligence based methods) and undefined fault codes, as in the case of ratio methods. Some of the reasons for the limited accuracy and inconsistent classification results are given below.

1. Less efficient classification techniques;
2. Studies and research conducted on limited, less

diverse data sets (for feature selection) do not consider the applicability of the results to other transformers (Ashkezari et al. 2012), (Ashkezari et al. 2014);

3. Determining the correlation between various oil parameters (features), based on data from a small group of transformers, that does not remain true when tested on data from different transformers (Gumilang 2009);

4. Limitations of conventional methods. No single conventional method gives 100 per cent correct diagnosis;

5. Poor quality of input data (caused by inaccuracies in the measuring instruments and human error);

6. Lack of a common framework for expert's interpretation, which results in inconsistent diagnosis (Ashkezari et al. 2014).

This paper attempts to overcome some of the limitations.

The main contribution of this paper is the application of an ensemble-based classifier called the Rotation Forest method. This yields superior classification accuracy when compared with other techniques, because it uses rotated feature space and decision tree groups for training and classification. The results obtained by this approach outperform the results obtained by various other methods such as artificial neural networks, support vector machines, the multilayer perceptron (MLP) and naive Bayes classifiers.

A further contribution of this paper is the study of the effectiveness of feature selection techniques for fault diagnosis of industrial equipment. This paper, through studies and experiments, shows that correlation studies using limited, less diverse data sets result in inaccurate conclusions about correlation between various features (Gumilang 2009).

By experiment, this paper shows that feature selection cannot be effective if the analysis is not based on a statistically significant data set (Ashkezari et al. 2012), (Ashkezari et al. 2014).

An additional contribution made in this paper is the mitigation of the weaknesses of the conventional methods while still making use of their strengths. This paper, while building the health index to form the training database, combines the strengths of various ratio methods and uses the voting process to eliminate the inherent weaknesses such as undefined fault codes. Moreover, this paper adopts and improves the voting process used by (Ashkezari et al. 2013), with the IEC ratio method, Rogers's ratio method and the IEEE condition assessment table (*IEEE Guide for the Interpretation of Gases Generated in Oil-Immersed Transformers* 2009), in addition to Duval's triangle method. Expert decision is another voting component used while building the health index. This approach is robust, as it eliminates the uncertainty arising from the use of one single conventional method to determine transformer health.

The organization of this paper is as follows. Section 2 discusses the data preparation and the research methodology. Section 3 provides a detailed discussion of the concept of feature selection for fault identification, using DGA for transformers. In this section, a critical analysis of the significance of feature selection is given. Section 4 describes the process of developing the training data set and the process of building the various components, namely health index 1 to health index 6. Section 5 describes the Rotation Forest method, and the formulation and implementation of the Rotation Forest ensemble method for classification of the transformers, according to their health conditions. Section 5 further describes the classification trials using the Rotation Forest ensemble method. The test results are compared with classification using other methods. Section 6 provides a summary to conclude the paper.

2 Data preparation and analysis

This section provides a brief overview of the data used in this paper for determination of the transformer health index, using the Rotation Forest method.

2.1 Data Preparation

Data have been collected from transformers used in oil refining over a period of eight years. The transformers range from large transformers with oil reservoirs and conservators to small lighting transformers that are of the nitrogen sealed type. Since data have been collected from transformers operating under different conditions, it is possible to get a good mixture of transformers with various health or fault conditions: this information is helpful for the research. In addition, to introduce diversity, data from transformers located in other geographical regions have also been used for the experiments described in Section 3. Key features that are used to determine the transformer health condition are common across the reports, although they come from different countries and from different facilities (such as oil refineries, mines and LNG plants). To demonstrate the accuracy, applicability and validity of the experiments described in Section 3, this paper also uses data published in other IEEE research papers.

3 Analysis and application of feature selection

This section evaluates the significance of feature selection in the fault identification of industrial equipment such as transformers.

Feature selection is used to select the best features and eliminate unnecessary features for classification. Feature selection is effective and useful for problems where there are numerous features, and where the relationship between the features and class label is not well defined. However, for fault diagnosis of industrial equipment such as transformers, the relationship between the key features and the fault state or class is already given by international standards and these have been used by the industry.

Feature selection is of paramount importance for problems such as malware detection (Alazab et al. 2011), multi-modal affect detection (Hussain et al. 2012) and in detection of diseases such as diabetes (Kelarev et al. 2012).

IEEE and IEC standards are available to establish the relationship between the various transformer features such as H2, C2H2, CH4, C2H4, C2H6, CO and CO2 and transformer faults, which are class labels. In this case, since the relationship between the input features and class labels is already defined, application of feature selection techniques to select the best features is not effective. However, studies have been conducted to use feature selection to minimize the number of input features to the classifier (Ashkezari et al. 2014), (Samirmi et al. 2013) and (Malik et al. 2014). The main aim of the feature selection process is to reduce the number of features needed for

accurate classification and prediction, by applying methods such as correlation and mRMR (minimum-redundancy-maximum-relevance).The logic is that a smaller number of features will need less computing power and can be completed faster. After detailed analysis, this paper concludes that feature selection does not add any significant value to the objective of transformer fault detection and classification: it also increases the risk of incorrect interpretation of the relationship between the various features.

3.1 Case study: correlation between interfacial tension (IFT) and acidity

The experiment below shows that the determination of correlation should not be based on the results of mathematical operations alone, as the correlation might be affected when data collection conditions are not constant. The relationship between the various features will be affected by varying conditions, such as operating loads, the condition of various components (such as paper insulation) and the maintenance status of the transformer. If the data for correlation studies are taken from a small, less diverse data set that does not cover all the possible states of the transformer operation, then the results of correlation studies will not be accurate.

To explain the strong relationship between two features, namely dielectric breakdown value (BDV) and moisture, the correlation between these two features is studied and plotted, as shown in Figure 1. When the value of moisture increases, the dielectric breakdown value (BDV) decreases. The data for this plot is obtained from another research publication (Kohtoh et al. 2010). Similarly, the correlation between

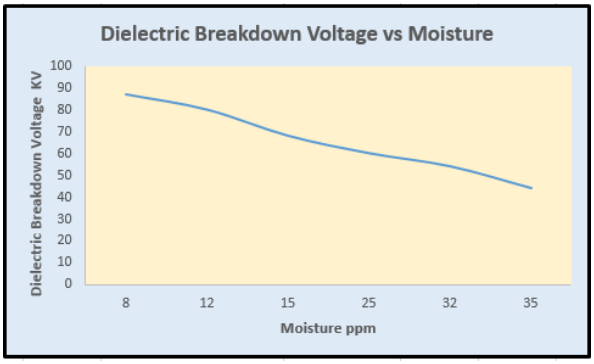


Figure 1: Moisture vs Breakdown voltage

interfacial tension and acidity can be studied and established. Based on the trends shown in Figure 2, it is possible to conclude that there is a linear relationship between acidity and interfacial tension values. The strong correlation between interfacial tension and acidity has been reported in earlier research (Gumilang 2009). Since these two features (interfacial tension and acidity) seem to have strong correlation, researchers might be tempted to apply feature selection by correlation where the intent is to reduce the number of input features. However, the author has encountered real situations where the correlation between the parameters does not conform to the relationship between the parameters shown in Figure 2 and does not conform to the relationship between acidity and interfacial tension (IFT) given by (Gumilang 2009). The key point to note is that it is possible to have a decrease in interfacial tension without a corresponding increase in acid number (Wang et al. 2002). This is supported by the plot shown in Figure

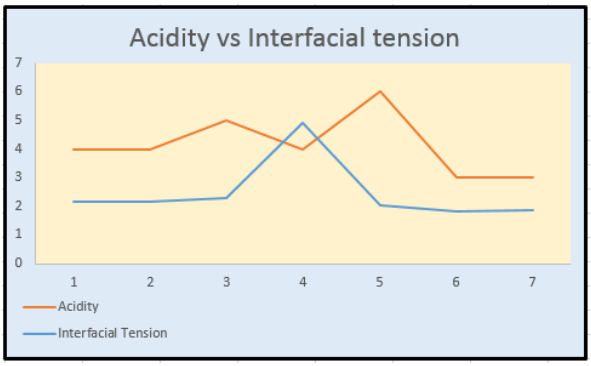


Figure 2: Acidity vs Interfacial Tension

3 and this plot is based on the data from a transformer on which the author has worked in practice. This shows that correlation study and feature selection should be based on a diverse data set that will show all the possible relationships between the various features.

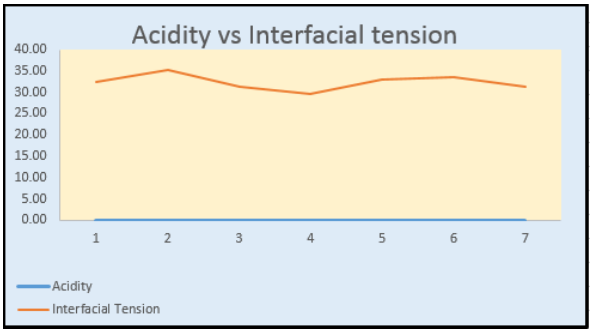


Figure 3: Acidity vs Interfacial Tension

3.2 Explanation for varying relationship between interfacial tension (IFT) and acidity

Acidity may be caused by many factors, such as moisture, sludge formation, paper deterioration, and oxidation. These factors may not always have an effect on interfacial tension, although they might be applicable in certain situations. Corrosion and oxidation of insulating fluids result in acidity (Wang et al. 2002), while interfacial tension (IFT) is affected by many factors such as the presence of polar contaminants, acids, solvents and varnish (Wang et al. 2002). Interfacial tension (IFT) is an indication of the polar contaminants in the insulating oil. An increase in acidity values can be accompanied by a decrease in interfacial tension values if oxidation products are involved. However, interfacial tension values can be high even when acidity values are low, if oil has been contaminated by damaged solid insulation materials, compounds from bushings or even from sources external to the transformer.

From the above discussion it becomes clear that feature selection cannot be based on correlation studies conducted on original features based on a small, less diverse data set. Rather, a more robust technique such as principal component analysis (PCA) should be applied to a large, diverse data set to extract the useful features while maintaining the variance.

3.3 Selection of key gases and other features

According to guidelines from (*IEEE Guide for the Interpretation of Gases Generated in Oil-Immersed Transformers* 2009), (IEC 2007) and numerous other research publications, the most common fault states of the transformer have been identified and the key gases that are known to be the symptoms of these faults have been listed. According to IEEE (*IEEE Guide for the Interpretation of Gases Generated in Oil-Immersed Transformers* 2009), internal faults in oil produce the gaseous by-products hydrogen (H2), methane (CH4), acetylene (C2H2), ethylene (C2H4), and ethane (C2H6). When cellulose is involved, the faults produce methane (CH4), hydrogen (H2), carbon monoxide (CO), and carbon dioxide (CO2). Chemical, electrical and physical properties of oils such as moisture, dielectric breakdown value, acidity, interfacial tension and furans content are used to determine the oil quality (Jahromi et al. 2009), (Naderian et al. 2008), (Ashkezari et al. 2014).

3.4 Analysis

Our investigation and experiments helps us to understand that sufficiently diverse data sets were not used in previous studies.Based on our investigation and experiments , we arrive at the conclusions listed below.

1. The greatest risk of using feature selection techniques such as correlation lies in receiving inaccurate representations of the significance of key gases, as shown in (Ashkezari et al. 2012), where C2H2 has been merged with C2H4 and given a low weight. C2H2 is a very important gas because significant quantities of C2H2 are produced under severe fault (arcing) conditions, where the temperature reaches 1000 degrees centigrade (*IEEE Guide for the Interpretation of Gases Generated in Oil-Immersed Transformers* 2009), (Ding et al. 2011) and C2H2 production starts when hot spot temperatures reach 500 degrees centigrade (Singh & Bandyopadhyay 2010).

2. The gain in processing speed achieved with feature selection (by reducing the number of input features to the classifier) is not significant (Ashkezari et al. 2014) and better processing speeds can be achieved by using other methods such as decision trees.

3. Due to the small size of the data sets, the weights of various features (for feature selection) could not be correctly determined (Ashkezari et al. 2012) and this will affect the efficiency and quality of feature selection.

4. Correlation between features of transformer oil, such as interfacial tension and acidity, needs to be studied on the basis of data from a global data set, rather than on a limited one, to avoid misleading conclusions (Gumilang 2009).

5. No statistically proven methods are currently available to establish the optimal size of the data set and historic database (Ashkezari et al. 2014).

6. Due to the lack of a common framework in building historical databases, the developed algorithm that is trained on a local database may not be able to obtain effective results when applied globally (Ashkezari et al. 2014).

4 Health index

This section explains the use and construction of the health indices. As discussed in (Abu-Elanien et al. 2012)(Jahromi et al. 2009)(Ashkezari et al. 2013)(Ashkezari et al. 2012)(Scatiggio & Pompili 2013)(Hjartarson & Otal 2006), the approach is to build a health index database that groups the transformer into four main categories: Healthy, Fair, Unsatisfactory and Poor. This approach deviates from the traditional approach that focuses on classifying the transformers as healthy or faulty.

The procedures given below (health index 1 to 6) will assign a health index value to each of the transformer based on the factors given in the following paragraphs. Health index value of 1 indicates that the gas concentrations are within a healthy range, indicating that the transformer is healthy and falls into "healthy" condition category. Health index value of 2 indicates that the gas concentrations are not within the normal range and this might mean that the transformer needs to be monitored frequently and the transformer falls into "fair" condition category. Health index value of 3 indicates that the gas concentrations are in the unhealthy range and the transformer falls into "unsatisfactory" category. Health index value of 4 indicates that the gas values are very high and it might mean that the transformer has already failed or it is very close to failure and needs immediate intervention. Transformers with heat index of 4 falls into "poor" category.

This health index database is used by the artificial intelligence-based algorithm for training, testing and classification. Rather than classifying the transformer as healthy or faulty, the transformer will be assigned to one of the four categories mentioned in the preceding paragraph.

4.1 Health index 1

Health index 1 is built by determining the condition of dissolved gases, oil quality and the condition of paper insulation. The concentration of key gases is used to determine the dissolved gas analysis factor. As proposed by (Jahromi et al. 2009),(Ashkezari et al. 2013),(Naderian et al. 2008), this factor is based on the concentrations of the key gases such as CH4, C2H2, C2H4, C2H6, H2, CO and CO2. This section computes the health index by using a similar approach to that proposed by (Ashkezari et al. 2013). The DGA Factor (DGAF) can be defined as:

$$DGAF = \frac{\sum S_i W_i}{\sum W_i}$$

where i denotes seven dissolved gases (H2, CH4, C2H6, C2H4, C2H2, CO and CO2) (*IEEE Guide for the Interpretation of Gases Generated in Oil-Immersed Transformers* 2009), (IEC 2007). *Si* is the score value based on the volume of dissolved gases, and *Wi* represents the weighting factor of each individual dissolved gas (Jahromi et al. 2009),(Ashkezari et al. 2013),(Naderian et al. 2008). DGAF of 1 indicates that the gas concentrations are within a healthy range, indicating that the transformer is healthy. DGAF of 2 indicates that the gas concentrations are not within the normal range and this might mean that the transformer needs to be monitored frequently. DGAF of 3 indicates that the gas concentrations are in the unhealthy range. DGAF of 4 indicates that the gas values are very high and it might mean that the transformer has already failed or it is very close to failure and needs immediate intervention. The oil quality

factor (OQF) is determined by the values of features such as dielectric breakdown value (BDV), water content, dielectric dissipation factor, inter-facial tension, acidity and colour. The oil quality factor (OQF) can be defined as:

$$OQF = \frac{\sum S_i W_i}{\sum W_i}$$

where i denotes four oil tests (BDV, acidity, moisture content and inter-facial tension, *Si* is the score based on the value of oil test results and *Wi* represents the weighting factor of each individual oil test (Jahromi et al. 2009),(Ashkezari et al. 2013),(Naderian et al. 2008). The standard or acceptable values of key parameters are given by (Naderian et al. 2008) and in ASTM standards. The paper insulation factor (PIF) is determined by the quality and condition of the paper dielectric insulation. CO2/CO ratio, furans and DP (degree of polymerization) are the parameters that are used to evaluate the condition of paper insulation. The use of the CO2/CO ratio to determine the condition of paper insulation is a popular method (Wang et al. 2002), (IEC 2007), (Arakelian 2002).

4.2 Health index 2

The Duval's triangle method (Duval & dePabla 2001),(Duval 2002)is an efficient method used to detect faults but it is effective only in cases where it has been confirmed that the dissolved gas values have crossed a certain threshold. Duval's triangle analysis does not include the fault-free condition (Bakar et al. 2014). This paper uses the IEC ratio method, Rogers's ratio method and IEEE techniques for condition assessment in the voting process in addition to Duval's triangle method.

4.3 Health index 3

The health index 3 is created by using the IEC ratio method (IEC 2007). This method is better than Duval's triangle method in some aspects (can detect normal states and better discrimination between discharge and thermal faults). Yet, it can result in "not-diagnosed" or "unknown" states (Ashkezari et al. 2011). This method can produce effective results when gas concentration values are above their typical threshold values.

4.4 Health index 4

Health index 4 is created using Rogers's ratio method (*IEEE Guide for the Interpretation of Gases Generated in Oil-Immersed Transformers* 2009). This method is better than Duval's triangle method in some aspects, as it can detect normal states. It can give better discrimination between discharge and thermal faults. Although ratio methods such as Rogers's ratio offer better diagnostic accuracy, they can sometimes give ratios that result in non-diagnosed or unknown states (Ashkezari et al. 2011), (*IEEE Guide for the Interpretation of Gases Generated in Oil-Immersed Transformers* 2009). Nevertheless, these methods have been widely used in the industry due to their effectiveness in discriminating between the various fault types and their ease of use.

4.5 Health index 5

The IEEE table (*IEEE Guide for the Interpretation of Gases Generated in Oil-Immersed Transformers* 2009) gives an indication about the health condition of the transformer and also gives the threshold values of various gases. This offers greater reliability in determining the transformer health condition, as this method does not give unknown states and it can also be used when historical data are not available.

4.6 Health index 6

Health index 6 is formed by taking expert comments and judgments (Ashkezari et al. 2013).However, there is always room for human error. Based on the author's first-hand experience working with testing laboratories, both as an asset owner and as a testing engineer, in many cases the experts from the testing laboratories are not given all the information. Features such as leakage reactance and winding resistance provide additional information about condition of the transformer and these features can be used to calculate the final health index (Jahromi et al. 2009). If more features are available, then the expert can make a better informed decision on the health of the transformers. However, normally, not all the features are available to the expert to help make the decision. For example, features such as leakage reactance and winding resistance (Jahromi et al. 2009)are not measured regularly and are not available to the experts, except in special circumstances. In many cases, even the transformer loading and maintenance history (Jahromi et al. 2009) are not available to the experts. These reasons justify the use of a majority voting strategy to produce the final health index (Ashkezari et al. 2013). In this research paper, the expert comments fall into three categories and the status code assignment method is shown below:

1. Assigned status code is 1 (good condition) when the expert recommendation is to conduct routine annual checks, indicating the transformer condition is good.

2. Assigned status code is 3 (unsatisfactory) when the expert recommendation is to conduct another test in six months time: that is, the expert has observed an abnormality and, in order to confirm whether that abnormality is persistent or to check whether deterioration is continuing, another test is recommended within six months.

3. Assigned status code is 4 (poor) when the expert recommendation is to conduct another test within a short time frame, such as three months, one month or even less than one month, if the trend shows severe deterioration of gas values and other features. In some cases, it may require immediate shut down and repair of the transformer.

4.7 Final health index

The final health index is formed by applying a majority voting strategy to the values obtained from health indices 1 to 6. If the majority of the indices classify the transformer condition as healthy, then the final health index is 1 (healthy) and if the majority of the indices classify the transformer condition as poor, then the final health index is 4 (poor). In a few cases, where there are an equal number of votes for two different categories, the more severe condition is given priority. For example, if two health indices determine that the condition falls under category 2 (fair) and two other health indices determine the condition as 3 (unsatisfactory) for the same transformer,

the more severe condition, 3, is assigned to the transformer record.

4.8 Training database

The training database is built with transformer data records and the final health index. The following features are input to the classifier: hydrogen, methane, ethylene, acetylene, ethane, carbon monoxide, carbon dioxide, dielectric breakdown voltage, moisture, acidity, interfacial tension (IFT), furans, CO2/CO ratio and class label (final health index). The addition of IEC, Rogers's and IEEE methods to the voting process helps when there is ambiguity between the results of various indices. If Duval's triangle method determines that the transformer is faulty and the expert comment says that the transformer is in fair condition, then the IEEE and other methods help to determine the actual condition and assist in removing the ambiguity in forming the final health index.

5 Rotation Forest approach

Rotation Forest is a new ensemble method (Rodriguez et al. 2006) that has gained popularity in recent years, as this is a more efficient and robust method (Lin et al. 2012). This method uses a group of classifiers, rather than one classifier, each of which is trained independently, using a different set of extracted features (Lin et al. 2012). Feature extraction is carried out using principal component analysis (PCA).

5.1 Principal component analysis(PCA)

To create the training data for a base classifier (decision tree), the feature set is randomly split into K subsets (K is a parameter of the algorithm). Filtering is done using principal component analysis (PCA) and each subset is subject to PCA filtering. As no principal components are lost, diversity is maintained. This approach improves the accuracy and diversity of the method.

5.2 Decision tree ensemble

The J48 classifier for the decision tree was implemented using the C4.5 algorithm. Based on the time taken to build the decision tree and accuracy, the J48 graft type decision tree offers superior performance when compared with other decision trees (Pachghare & Kulkarni 2011). For the feature-based classification of transformers, through experimentation, it was found that decision trees yielded better classification accuracies than other methods. Instead of a single decision tree, a group of decision trees are used and hence this method is called Forest (Rodriguez et al. 2006).

5.3 Feature selection using principal component analysis (PCA)

Principal component analysis (PCA) identifies the features that are important for capturing the variance in the data. This technique is applied on large, high dimensional data sets to reduce the dimensions before the classification techniques are employed. The relationship between the various original features is analysed by PCA and a new, smaller set of variables called principal components are produced. These principal components are ranked by the variance represented by them. As shown in Figure 4, eleven original variables were analysed using PCA and five components were obtained. These five components contain almost 70 per cent of the variance in the data. The component matrix shown in Figure 4 lists all eleven original variables input to PCA. The weights of these eleven variables of the five components are also shown in Figure 5.

Total Variance Explained						
Component	Initial Eigenvalues			Loadings		
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %
1	1.881	17.103	17.103	1.881	17.103	17.103
2	1.837	16.702	33.805	1.837	16.702	33.805
3	1.498	13.618	47.422	1.498	13.618	47.422
4	1.379	12.538	59.961	1.379	12.538	59.961
5	1.047	9.516	69.477	1.047	9.516	69.477
6	.962	8.743	78.220			
7	.806	7.331	85.551			
8	.582	5.295	90.846			
9	.440	4.001	94.847			
10	.303	2.753	97.600			
11	.264	2.400	100.000			
Extraction Method: Principal Component Analysis.						

Figure 4: PCA extraction

Component Matrix[a]					
	Component				
	1	2	3	4	5
Hydrogen	-0.213	0.286	0.2	0.02	0.825
Methane	0.599	0.617	-0.81	0.029	-0.088
Ethylene	0.235	0.56	0.065	0.01	-0.149
Acetylene	-0.163	-0.105	-0.15	0.311	-0.088
Ethane	0.621	0.551	-0.287	0.028	0.114
CarbonMonoxide	-0.073	0.147	0.76	-0.172	-0.422
CarbonDioxide	-0.242	0.456	0.639	-0.272	0.092
B Down kv	0.492	-0.208	0.361	0.598	0.001
Moisture ppm	-0.547	0.413	-0.007	0.542	0.088
Acidity mgKOH/g	-0.487	0.409	-0.148	0.532	-0.309
Interfacial Tension	0.41	-0.366	0.453	0.492	0.163
Extraction Method: Principal Component Analysis					
a. 5 components extracted					

Figure 5: Components table

Classifier	Number of iterations	maxGroup / minGroup	Projection Filter	Removed Percentage Principal Components	Seed
J48-C 0.25	10	3	Principal Components (unsupervised attribute)	50%	1
10 Fold Cross-validation					

Figure 6: Parameters for WEKA test

5.4 Application of Rotation Forest method

Rotation Forest (Rodriguez et al. 2006) is applied to the training database through WEKA. Tenfold cross validation was performed on the data. J48-C0.25 is used as the base classifier and the principal components as the projection filter. The following features are input to the classifier: hydrogen, methane, ethylene, acetylene, ethane, carbon monoxide, carbon dioxide, dielectric breakdown voltage, moisture, acidity, interfacial tension (IFT), furans, CO2/CO ratio and the class label (final health index).The parameters used for running the Rotation Forest algorithm through WEKA are shown in Figure 6. Figure 7 shows the results of the Rotation Forest algorithm, and demonstrates the precision and accuracy

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.879	0.06	0.83	0.879	0.854	0.97	Unsatisfactory
0.889	0.043	0.881	0.889	0.885	0.97	Fair
0.972	0.017	0.978	0.972	0.975	0.99	Good
0.564	0.006	0.815	0.564	0.667	0.97	Poor
0.908	0.034	0.908	0.908	0.907	0.98	Weighed Average

Figure 7: Rotation Forest results

Method	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
LIB SVM	0.567	0.554	0.663	0.57	0.416	0.507
Multilayer Perceptron	0.808	0.158	0.798	0.81	0.795	0.874
Naive Bayes	0.728	0.122	0.729	0.73	0.709	0.878
Rotation Forest	0.909	0.034	0.909	0.91	0.909	0.981

Figure 8: Comparison with other AI techniques

of the Rotation Forest when compared with other techniques using the same data set.

According to the test results shown in Figure 8, it is certain that reliable classification accuracy and prediction capabilities can be achieved using the Rotation Forest method instead of a combination of techniques such as fuzzy logic with SVM (Ashkezari et al. 2013), GA with SVM (JinLiang et al. 2011) and LS-SVM with improved GA(Yanqing et al. 2010).

6 Conclusion

In this paper, we have presented an artificial intelligence-based approach to detect faulty transformers. Our approach uses a machine learning technique to drive the detection process and involves a detailed feature analysis from a large, diverse data set. We acknowledge that conclusions reached in other research papers (Gumilang 2009) about the correlation of the various features and the importance of certain features such as C2H2 gas (Ashkezari et al. 2012) were based on smaller data sets. This paper fulfils the need to understand the correlation between various features using a large, more diverse data set compiled from various sources. To overcome the issues posed by a small, less diverse data set, this paper uses a new, large and more diverse data set. Hence, an improvement in the generalization ability of the applied algorithm has been achieved.

1. The key contribution of this research paper is the application of Rotation Forest for transformer health classification.
2. A further contribution is the study of the significance of feature selection and the finding that the conclusions of correlation studies cannot be applied effectively for feature selection from the point of view of transformer diagnosis.
3. Through the use of additional conventional techniques, an improvement in the voting process has been realized; this has improved the consistency and data diversity of the training database.
4. The use of new, large, diverse data sets compiled from various sources (located in different geographical regions) for experimentation results in improved understanding of the relationship between the various features of transformer oil and results in improved generalization ability of the machine learning algorithm.
5. Experiments on various data sets compiled from various sources (located in different geographical regions) prove that the performance of the Rotation Forest method is superior to other artificial intelligence-based techniques such as the support vector machine (SVM) or artificial neural networks.
6. Feature selection process and correlation studies will not be very useful and effective in the fault diagnosis of industrial equipment such as transformers, as the faults are well defined and the relationship between the faults and the various input features are well defined in IEEE and IEC standards.

References

Abu-Elanien, A., Salama, M. & Ibrahim, M. (2012), 'Calculation of a health index for oil-immersed transformers rated under 69 kv using fuzzy logic', *IEEE Transactions on Power Delivery* **27**(4), 2029–2036.

Alazab, M., Venkatraman, S., Watters, P. & Alazab, M. (2011), Zero-day malware detection based on supervised learning algorithms of api call signatures, *in* 'Proceedings of the Ninth Australasian Data Mining Conference - Volume 121', AusDM '11, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, pp. 171–182.

Arakelian, V. (2002), 'Effective diagnostics for oil-filled equipment', *Electrical Insulation Magazine, IEEE* **18**(6), 26–38.

Ashkezari, A., Ma, H., Ekanayake, C. & Saha, T. (2012), Multivariate analysis for correlations among different transformer oil parameters to determine transformer health index, *in* 'Power and Energy Society General Meeting, 2012 IEEE', pp. 1–7.

Ashkezari, A., Ma, H., Saha, T. & Cui, Y. (2014), 'Investigation of feature selection techniques for improving efficiency of power transformer condition assessment', *IEEE Transactions on Dielectrics and Electrical Insulation* **21**(2), 836–844.

Ashkezari, A., Ma, H., Saha, T. & Ekanayake, C. (2013), 'Application of fuzzy support vector machine for determining the health index of the insulation system of in-service power transformers', *IEEE Transactions on Dielectrics and Electrical Insulation* **20**(3), 965–973.

Ashkezari, A., Saha, T., Ekanayake, C. & Ma, H. (2011), Evaluating the accuracy of different dga techniques for improving the transformer oil quality interpretation, *in* 'Universities Power Engineering Conference (AUPEC), 2011 21st Australasian', pp. 1–6.

Bakar, N., Abu-Siada, A. & Islam, S. (2014), 'A review of dissolved gas analysis measurement and interpretation techniques', *Electrical Insulation Magazine, IEEE* **30**(3), 39–49.

Ding, H., Heywood, R., Lapworth, J. & Ryder, S. (2011), Learning from success and failure in transformer fault gas analysis and interpretation, *in* 'Reliability of Transmission and Distribution Networks (RTDN 2011), IET Conference on', pp. 1–6.

Duval, M. (2002), 'A review of faults detectable by gas-in-oil analysis in transformers', *Electrical Insulation Magazine, IEEE* **18**(3), 8–17.

Duval, M. (2003), 'New techniques for dissolved gas-in-oil analysis', *Electrical Insulation Magazine, IEEE* **19**(2), 6–15.

Duval, M. & dePabla, A. (2001), 'Interpretation of gas-in-oil analysis using new IEC publication 60599 and IEC TC 10 databases', *Electrical Insulation Magazine, IEEE* **17**(2), 31–41.

Gumilang, H. (2009), Unique relationship between interfacial tension test (ift) and neutral number test (acidity) of transformer insulation oil in pln p3b jb - jakarta and banten regional, *in* 'IEEE 9th International Conference on the Properties and Applications of Dielectric Materials, 2009. ICPADM 2009.', pp. 29–32.

Hjartarson, T. & Otal, S. (2006), Predicting future asset condition based on current health index and maintenance level, *in* 'IEEE 11th International Conference on Transmission Distribution Construction, Operation and Live-Line Maintenance, 2006. ESMO 2006.'.

Hussain, M. S., Monkaresi, H. & Calvo, R. A. (2012), Combining classifiers in multimodal affect detection, *in* 'Proceedings of the Tenth Australasian Data Mining Conference - Volume 134', AusDM '12, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, pp. 103–108.

IEC (2007), *Mineral oil-impregnated electrical equipment in service Guide to the interpretation of dissolved and free gases analysis, IEC 60599*, ed.

IEEE Guide for the Interpretation of Gases Generated in Oil-Immersed Transformers (2009), *IEEE Std C57.104-2008 (Revision of IEEE Std C57.104-1991)* pp. 1–36.

Jahromi, A., Piercy, R., Cress, S., Service, J. & Fan, W. (2009), 'An approach to power transformer asset management using health index', *Electrical Insulation Magazine, IEEE* **25**(2), 20–34.

JinLiang, Y., Yongli, Z. & Guoqin, Y. (2011), Power transformer fault diagnosis based on support vector machine with cross validation and genetic algorithm, *in* '2011 International Conference on Advanced Power System Automation and Protection (APAP)', Vol. 1, pp. 309–313.

Kelarev, A. V., Stranieri, A., Yearwood, J. L., Abawajy, J. & Jelinek, H. F. (2012), Improving classifications for cardiac autonomic neuropathy using multi-level ensemble classifiers and feature selection based on random forest, *in* 'Proceedings of the Tenth Australasian Data Mining Conference - Volume 134', AusDM '12, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, pp. 93–101.

Kohtoh, M., Ueta, G., Okabe, S. & Amimoto, T. (2010), 'Transformer insulating oil characteristic changes observed using accelerated degradation in consideration of field transformer conditions', *Dielectrics and Electrical Insulation, IEEE Transactions on* **17**(3), 808–818.

Lapworth, J. (2002), A novel approach (scoring system) for integrating dissolved gas analysis results into a life management system, *in* 'Conference Record of the 2002 IEEE International Symposium on Electrical Insulation, 2002.', pp. 137–144.

Lin, L., Zuo, R., Yang, S. & Zhang, Z. (2012), SVM ensemble for anomaly detection based on rotation forest, *in* '2012 Third International Conference on Intelligent Control and Information Processing (ICICIP)', pp. 150–153.

Malik, H., Mishra, S. & Mittal, A. P. (2014), 'Selection of most relevant input parameters using waikato environment for knowledge analysis for gene expression programming based power transformer fault diagnosis', *Electric Power Components and Systems* **42**(16), 1849–1861.

Naderian, A., Cress, S., Piercy, R., Wang, F. & Service, J. (2008), An approach to determine the health index of power transformers, *in* 'Conference Record of the 2008 IEEE International Symposium on Electrical Insulation, 2008. ISEI 2008.', pp. 192–196.

Pachghare, V. & Kulkarni, P. (2011), Pattern based network security using decision trees and support vector machine, *in* '2011 3rd International Conference on Electronics Computer Technology (ICECT)', Vol. 5, pp. 254–257.

Rodriguez, J., Kuncheva, L. & Alonso, C. (2006), 'Rotation forest: A new classifier ensemble method', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**(10), 1619–1630.

Rogers, R. (1978), 'IEEE and IEC codes to interpret incipient faults in transformers, using gas in oil analysis', *IEEE Transactions on Electrical Insulation* **EI-13**(5), 349–354.

Samirmi, F., Tang, W. & Wu, H. (2013), Feature selection in power transformer fault diagnosis based on dissolved gas analysis, *in* 'Innovative Smart Grid Technologies Europe (ISGT EUROPE), 2013 4th IEEE/PES', pp. 1–5.

Scatiggio, F. & Pompili, M. (2013), Health index: The terna's practical approach for transformers fleet management, *in* 'Electrical Insulation Conference (EIC), 2013 IEEE', pp. 178–182.

Seifeddine, S., Khmais, B. & Abdelkader, C. (2012), Power transformer fault diagnosis based on dissolved gas analysis by artificial neural network, *in* '2012 First International Conference on Renewable Energies and Vehicular Technology (REVET)', pp. 230–236.

Singh, S. & Bandyopadhyay, M. (2010), 'Dissolved gas analysis technique for incipient fault diagnosis in power transformers: A bibliographic survey', *Electrical Insulation Magazine, IEEE* **26**(6), 41–46.

Wang, M., Vandermaar, A. & Srivastava, K. (2002), 'Review of condition assessment of power transformers in service', *Electrical Insulation Magazine, IEEE* **18**(6), 12–25.

Yanqing, L., Huaping, H., Ningyuan, L., Qing, X. & Fangcheng, L. (2010), Transformer fault gas forecasting based on the combination of improved genetic algorithm and LS-SVM, *in* 'Power and Energy Engineering Conference (APPEEC), 2010 Asia-Pacific', pp. 1–4.

Non-Invasive Attributes Significance in the Risk Evaluation of Heart Disease Using Decision Tree Analysis

Mai Shouman[1], Tim Turner[2]
School of Engineering and Information Technology
University of New South Wales at the Australian Defence Force Academy
Northcott Drive, Canberra ACT 2600
[1] mai_shouman@yahoo.com
[2] t.turner@adfa.edu.au

Abstract

Non-invasive attributes are low cost, easy to identify attributes that can have significant influence in the risk evaluation of heart disease. This research formulates a combination of non-invasive attributes that can be used in the risk evaluation of heart disease. It uses the decision tree data mining technique to identify the significance of different single, combined, and calculated non-invasive attribute combinations in the risk evaluation of heart disease against the benchmark Cleveland heart disease dataset and a larger Canberra hospital heart disease dataset. For each non-invasive attribute combination, 10-fold cross-validation is applied to ensure reliable performance measures. Different equations of non-invasive attributes on the Canberra dataset show that the best combination in the risk evaluation of heart disease is age, sex, resting blood pressure and Rohrer's Index equation with a mean accuracy and standard deviation of 73.8% and 4.9% respectively.

Keywords: Non-invasive attributes, Heart disease risk evaluation, Decision tree, Data mining

1 Introduction

Heart disease has been the leading cause of death in the world over the past decade on different continents and in different countries regardless of their income (World Health Organization, 2011b). Early detection of heart disease patients can help in recovering patients' health and decreasing the mortality rate from heart disease (Centers for Disease Control and Prevention, 2013). There is a vital need for accurate systematic tools that identify patients at high risk and provide information for early detection of heart disease (Paladugu, 2010).

Community-level screening tests play an especially important role in the early detection of heart disease (Kotnik, 2010). They can be applied where there is limited availability of resources such as electrocardiogram, stress tests, and cardiac angiogram machines needed for the diagnosis of heart disease. Recent research focuses on discovering new specific, sensitive and cheap community-level screening tests (Kotnik, 2010, Patel et al., 2009).

There are two famous heart disease risk evaluation screening tests commonly used in heart disease diagnosis: the Framingham Heart Disease Risk Evaluation Tool (Framingham Heart Study, 2013) and the Australian Absolute Cardiovascular Risk Calculator (National Heart Foundation of Australia, 2009). Both of these two heart disease risk evaluation tests use a set of attributes such as age, sex, systolic blood pressure, total cholesterol, diabetes and smoking status to identify if a patient is at high, moderate or low risk of heart disease (National Heart Foundation of Australia 2009, Framingham Study. 2013). Although these two tests help in identifying patients at risk of heart disease, they need prior blood-based investigation to identify the cholesterol and diabetes levels. Such tests are both invasive, requiring physical samples to be taken, and relatively expensive. Hence, there is a need to simplify the heart disease risk evaluation tool attributes so that affordable detection strategies can be implemented (Bitton and Gaziano, 2010). There is a need to find less costly tests and accurate systematic tools that can be used for community-level screening to identify patients at high risk of heart disease and provide information to enable early intervention (Paladugu, 2010).

Motivated by the increasing mortality rates of heart disease, researchers are using several data mining techniques to help healthcare professionals in the diagnosis of heart disease patients (Das et al., 2009, Kavitha et al., 2010). The research presented here investigates applying the decision tree data mining technique in the risk evaluation of heart disease patients. The investigation considers different combinations of non-invasive attributes of patient health care records in a benchmark dataset from the Cleveland Heart Disease Dataset and a new, larger dataset drawn from the Canberra Hospital in Canberra, Australia. The investigation identifies the significance of using non-invasive attributes in community-level screening tests for the risk evaluation of heart disease patients.

2 Literature review

2.1 Using data mining techniques in the risk evaluation of heart disease

Data mining in healthcare is an emerging field of high importance for providing diagnosis and prognosis and a deeper understanding of medical data. Data mining applications in healthcare include analysis of health care centres for better health policy-making and prevention of hospital errors, and early detection and prevention of diseases (Canlas Jr, 2009). Motivated by the increasing mortality rates of heart disease, researchers are using several data mining techniques to help healthcare professionals in the diagnosis of heart disease patients.

Different data mining techniques have been used to help health care professionals in the diagnosis of heart disease. Those techniques most frequently used focus on classification: naïve bayes, decision tree, and neural network. Other data mining techniques are also used including kernel density, automatically defined groups, bagging algorithm, and support vector machine. Much of the literature reporting on the efficacy of data mining techniques cannot be compared together as researchers have been using different datasets. However, over time a defacto benchmark data set has arisen in the literature: the Cleveland Heart Disease Dataset (CHDD http://archive.ics.uci.edu/ml/datasets/Heart+Disease). Results of trials on this dataset do allow comparison.

2.2 Non-invasive attributes in heart disease risk evaluation

Heart disease can be detected by several tests, such as chest X-rays, coronary angiograms, electrocardiograms, and exercise stress tests (National Center for Chronic Disease Prevention and Health Promotion, 2013). However, those tests are very costly and typically require sophisticated equipment and a visit to a medical facility for their conduct. There is a vital need for accurate and systematic tools that provide information for early detection of heart disease to identify those patients at high risk (Paladugu, 2010).

It is widely accepted that age, sex, blood pressure, smoking, cholesterol and diabetes are the major risk factors for developing heart diseases (Cupples and D'Agostino, 1987). The Framingham Heart Disease Risk Calculator (Framingham Heart Study, 2013) and Australian Absolute Cardiovascular Risk Calculator (National Heart Foundation of Australia, 2009) are two famous heart disease screening tests that use these major risk factors for identifying degree of risk of heart disease. The Framingham test and the absolute risk calculator use a set of invasive and non-invasive attributes in the risk evaluation of heart disease patients. Although non-invasive attributes are easily known and low cost attributes, the use of only non-invasive attributes in the risk evaluation of heart disease patients has not been investigated before. Moreover, if non-invasive attributes show significant performance in the risk evaluation of heart disease patients then this investigation could be of great benefit to the early detection of heart disease.

The invasive attributes are measured using data from blood tests prior to any evaluation. These measures may be difficult to implement where there are limited resources available (Bitton and Gaziano, 2010): "Can combinations of non-invasive attributes provide reliable performance in the diagnosis of patients at risk of heart disease?"

The non-invasive attributes are those that can be identified easily without complex machines and instruments that are typically found in a hospital or formal medical setting; for example: age, sex, height, weight, smoking habits, and resting blood pressure. Although weight needs a scale to be measured and resting blood pressure needs a blood pressure monitor to be measured, these tools can be available at home or in a pharmacy and do not need a medical practitioner to be measured or physical samples to be taken. These attributes can be collected quite cheaply. Age, sex, blood pressure, and smoking habits are major risk factors for developing heart disease (Cupples and D'Agostino, 1987). Can these non-invasive attributes be used with data mining techniques to cost effectively identify patients at risk of heart disease? Combinations of non-invasive attributes from two datasets (Cleveland and Canberra) provide a basis for examination of their potential.

The performance of diagnostic algorithms using only non-invasive attributes has not previously been assessed. How accurately can data mining techniques using only non-invasive attributes classify patients as sick or healthy? Success here would provide a great opportunity for application to community screening tests, thus enabling early intervention in patients at high risk of heart disease and determining suitable treatment regimes for those patients.

3 Methodology

Decision Tree is one of the successful data mining techniques used in the diagnosis of heart disease patients (Andreeva, 2006, Sitar-Taut et al., 2009, Das et al., 2009). This paper applies the Decision Tree technique to identify the non-invasive attributes combination that will show the best performance in the diagnosis of heart disease patients. The research undertook a systematic investigation of the potential for non-invasive attributes to be effective diagnostic pointers (see Figure 1). First, each non-invasive heath attribute in the Cleveland and Canberra datasets was tested. Then, different combinations of non-invasive attributes were trailed. Finally, equations using different non-invasive attributes were developed and examined to determine if they would enhance Decision Tree performance in the diagnosis of heart disease patients.

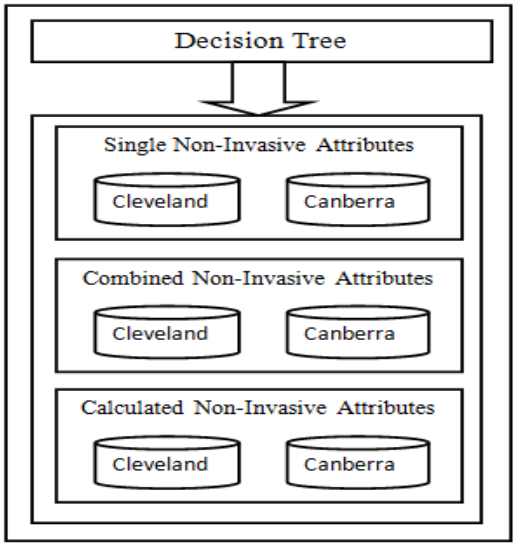


Figure 1: Systematically Applying Decision Tree to Different Heart Disease Datasets Attributes

3.1 Decision Tree

Gain Ratio Decision Tree is one of the most successful types of Decision Trees (Bramer, 2007, Cieslak et al., 2012). It is a relationship between entropy and splitting information.

The entropy selects the splitting attribute that minimizes the value of entropy, thus maximising the Information Gain. To identify the splitting attribute of the Decision Tree, one must calculate the Information Gain for each attribute and then select the attribute that maximizes the Information Gain. The Information Gain for each attribute is calculated using Equation 1 (Bramer, 2007, Han and Kamber, 2006):

$$E = \sum_{i=1}^{k} P_i \log_2 P_i \qquad \text{(Equation 1)}$$

Where k is the number of classes of the target attribute

P_i is the number of occurrences of class i divided by the total number of instances (i.e. the probability of i occurring).

The Information Gain measure is biased toward tests with many outcomes. That is, it prefers to select attributes having a large number of values (Han and Kamber, 2006). To reduce the effect of the bias resulting from the use of Information Gain, a variant known as Gain Ratio was introduced (Bramer, 2007). Gain Ratio adjusts the Information Gain for each attribute to allow for the breadth and uniformity of the attribute values using Equation 2.

Gain Ratio = Information Gain / Split Information (Equation 2)

Where the split information is a value based on the column sums of the frequency table (Bramer, 2007).

3.2 Cleveland and Canberra heart disease datasets

The benchmark dataset used in this study is the Cleveland Clinic Foundation Heart disease dataset The dataset involves 13 data attributes (Table 1). The dataset contains 303 rows of which 297 are complete. Six rows contain missing values and they are removed from the experiments. The comparison dataset used in this study is obtained from the cardiology department of the Canberra Hospital, Canberra, Australia. The dataset involves 13 data attributes (Table 2). The dataset contains 864 rows of which 250 are healthy and 614 are sick. Data mining techniques need to have proportional balance in the target attribute (Diagnosis) (Han and Kamber, 2006). The benchmark dataset contains 54% records of healthy patients and the remainder of sick patients. So, records were selected from the Canberra dataset to match this proportion for comparison investigations. Random selections from the patient records identified as sick were made to create 460 rows of which 250 are healthy and 210 are sick. Ten different random selections of the main Canberra heart disease dataset (864 rows) are made to maintain consistency when applying data mining techniques and there is no difference in accuracy found in the different random combinations.

Name	Type	Description
Age	Continuous	Age in years
Sex	Discrete	1 = male 0 = female
Cp	Discrete	Chest pain type including typical angina, atypical angina, non-angina pain, and asymptomatic
Trestbps	Continuous	Resting blood pressure in (mm Hg)
Chol	Continuous	Serum cholesterol in (mg/dl)
Fbs	Discrete	Fasting blood sugar > 120 in (mg/dl):
Restecg	Discrete	Resting electrocardiographic results
Thalach	Continuous	Maximum heart rate achieved
Exang	Discrete	Exercise induced angina: 1 = yes 0 = no
Old peak ST	Continuous	Depression induced by exercise relative to rest
Slope	Discrete	The slope of the peak exercise segment including up sloping, flat, and down sloping
Ca	Discrete	Number of major vessels colored by fluoroscopy that ranged between 0 and 3.
Thal	Discrete	3 = normal 6= fixed defect 7= reversible defect
Diagnosis	Discrete	Diagnosis classes: 0 = healthy 1= patient who is subject to possible heart disease

Table 1: Cleveland Heart Disease Data Attributes

Name	Type	Description
Age	Continuous	Age in years
Sex	Discrete	1 = male 0 = female
Postcode	Continuous	Residential Post Code
Height	Continuous	Height in centimetres
Weight	Continuous	Weight in kilogram
Diastole	Continuous	Left Ventricle Diastole in centimetres
Systole	Continuous	Left Ventricle Systole in centimetres
Resting Heart Rate	Continuous	Resting Heart Rate in (bpm)
Peak Heart Rate	Continuous	Peak Heart Rate in (bpm)
Resting Blood Pressure High	Continuous	Resting Blood Pressure in (mm Hg)
Resting Blood Pressure Low	Continuous	Resting Blood Pressure in (mm Hg)
Peak Blood Pressure High	Continuous	Peak Blood Pressure in (mm Hg)
Peak Blood Pressure Low	Continuous	Peak Blood Pressure in (mm Hg)
Diagnosis	Discrete	Diagnosis classes: 0 = healthy 1= patient who is subject to possible heart disease

Table 2: Canberra Heart Disease Data Attributes

4 Performance Evaluation

To measure the stability of the data mining technique using a single/set of attribute(s), the data is divided into training and testing data with 10-fold cross-validation. Cross-validation reduces the potential for the training to skew the data mining techniques' accuracy through peculiarities in the training data (Bramer, 2007). To evaluate the performance of the data mining techniques the sensitivity, specificity, and accuracy are calculated.

Sensitivity is the proportion of positive instances that are correctly classified as positive. Equation 3 shows how the sensitivity is calculated.

Sensitivity = True Positive / Positive (Equation 3)

Specificity is the proportion of negative instances that are correctly classified as negative. Equation 4 shows how the specificity is calculated.

Specificity = True Negative/ Negative (Equation 4)

Accuracy is the proportion of instances that are correctly classified (Bramer, 2007, Han and Kamber, 2006). Equation 5 shows how the accuracy is calculated.

Accuracy = (True Positive + True Negative) / (Positive + Negative) (Equation 5)

5 Results and discussion

5.1 Single Non-Invasive Attributes for Cleveland and Canberra Heart Disease Risk Evaluation

The Cleveland dataset contains three non-invasive attributes: age, sex, and resting blood pressure; and the Canberra heart disease dataset contains five non-invasive attributes: age, sex, resting blood pressure, height, and weight. Table 3 shows the mean and standard deviation of sensitivity, specificity, and accuracy of Decision Tree using single non-invasive attributes of the Cleveland dataset where the accuracy ranges between 61.5% and 55.3%. The sex attribute shows best mean accuracy followed by age and resting blood.

Cleveland Data Attributes	Sensitivity		Specificity		Accuracy	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
Age	55.6 %	15.7%	62.8 %	20.1%	56%	7.9%
Sex	83%	10.1%	41.7 %	18.5%	61.5 %	10.1%
Resting Blood Pressure	26.4 %	13.3%	79.5 %	9.4%	55.3 %	12%

Table 3: Applying Decision Tree on Single Non-Invasive Cleveland Heart Disease Data Attributes

Table 4 shows the mean and standard deviation of sensitivity, specificity, and accuracy for single non-invasive attributes of the Canberra dataset with accuracy ranging between 71% and 46.1%. The age attribute shows the best mean accuracy followed by sex, resting blood pressure, height, and weight.

On both Cleveland and Canberra heart disease datasets, the age and sex single data attributes show the best results among other single non-invasive attributes, with the age attribute attaining 56% and 71% mean accuracy respectively. The sex attribute achieves 61.5% and 66% mean respectively. Unsurprisingly, relying on single attributes for the diagnosis of heart disease is insufficient with many of the attributes not better than chance in predictive power. So what is the effect on accuracy of combined non-invasive attributes?

Canberra Data Attributes	Sensitivity		Specificity		Accuracy	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
Age	61.6 %	16.5%	75.3 %	19.7%	71%	8.2%
Sex	67.4 %	14.7%	58.4 %	16.3%	66%	8.3%
Resting Blood Pressure	46.6 %	11.2%	63.6 %	7.3%	55.1 %	5.9%
Height	45.7 %	18.2%	59.7 %	7.9%	50.8 %	10.9%
Weight	24.2 %	26.8%	71.5 %	26.5%	46.1 %	9.3%

Table 4: Applying Decision Tree on Single Non-Invasive Canberra Heart Disease Data Attributes

5.2 Different Combinations of Non-Invasive Attributes for Cleveland and Canberra Heart Disease Risk Evaluation

This section investigates the performance of combined non-invasive attributes in the diagnosis of heart disease patients using the Cleveland and Canberra datasets. Table 5 shows the performance of different combinations of the non-invasive attributes in diagnosis using the Cleveland heart disease dataset. The combinations of age, sex, and resting blood pressure show best mean accuracy of 65.8% followed by age and sex combination showing mean accuracy of 65.2%.

No of Attributes	Cleveland Data Attributes	Sensitivity		Specificity		Accuracy	
		Mean	St Dev	Mean	St Dev	Mean	St Dev
Two	Age, Sex	60.4%	9.1%	67.5%	9.3 %	65.2%	7.5 %
	Age, RBP	46.6%	12.8%	71.3%	14 %	58.3%	9.3 %
	Sex, RBP	69%	15.4%	54.2%	13 %	60 %	8.9 %
Three	Age, Sex, RBP	61.2%	10.1%	69.5%	10.1%	65.8%	8.6 %

Table 5: Applying Decision Tree on Combined Non-Invasive Cleveland Heart Disease Data Attributes

The difference of the mean accuracy between the age and sex combination and age, sex, and resting blood pressure combination is just 0.6%. So, t-test for significance is applied to identify if there is a significant difference between the two combinations. The t-test shows that there is no significant difference between the two combinations (see Table 6). The sensitivity measure is the true positive, meaning sick patients that are identified as sick. The specificity measure is the true negative, meaning healthy patients that are identified as healthy. In this context, sensitivity is the most useful in identifying sick patients to ensure appropriate care. Thus the age, sex and resting blood pressure combination demonstrates best results with the highest mean sensitivity in the diagnosis of Cleveland heart disease dataset.

Cleveland Data Attributes	Accuracy			Sensitivity		
	Mean	St Dev	T-Test Significance	Mean	St Dev	T-Test Significance
Age, Sex	65.2%	7.5%	No (t = -0.911, p <= 0.05)	60.4%	9.1%	No (t = -1.262, p <= 0.05)
Age, Sex, RBP	65.8%	8.6%		61.2%	10.1%	

Table 6: T-Test Significance between Non-Invasive Cleveland Heart Disease Data Combinations

The Canberra heart disease dataset contains five non-invasive attributes: age, sex, resting blood pressure (Systolic and Diastolic), height, and weight. Table 7 shows the performance of different combinations of these non-invasive attributes in diagnosis using the Canberra heart disease dataset.

No of Attributes	Canberra Data Attributes	Sensitivity		Specificity		Accuracy	
		Mean	St Dev	Mean	St Dev	Mean	St Dev
Two	Age, Sex	66.3%	14 %	73.8%	21.4%	74.2%	7.5 %
	Age, RBP	58.8%	15.7%	70.2%	15.3%	65.7%	7.1 %
	Age, Height	71 %	11.3%	69.9%	19 %	72.1%	8.3 %
	Age, Weight	67.5%	12.2%	70.1%	19.2%	69.6%	10.3%
	Sex, RBP	54 %	9.8 %	66.2%	7.7 %	61.3%	6.8 %
	Sex, Height	66.5%	13.7%	58.4%	16.3%	65.5%	8%
	Sex, Weight	54.3%	9%	72 %	13.9%	66.2%	7.7 %
	RBP, Height	35 %	9.6 %	61.3%	11.7%	49.2%	8.9 %
	RBP, Weight	38.4%	11.2%	68.4%	14.9%	53.8%	7.4 %

	Height, Weight	43.3%	11.5%	63.1%	6.7%	53.3%	9.8%
Three	Age, Sex, RBP	66.7%	13.3%	76.5%	13.7%	74.8%	6.5%
	Age, Sex, Height	26.2%	22.4%	87.3%	12.3%	62.9%	8.3%
	Age, Sex, Weight	42.1%	16%	89%	6.4%	68.2%	12.2%
	Age, RBP, Height	60.7%	10.9%	69.4%	18%	67.1%	8.6%
	Age, RBP, Weight	59.9%	13%	70.2%	17.2%	66.3%	9.7%
	Age, Height, Weight	42.9%	7.8%	80.5%	16.6%	64.2%	7.4%
Four	Age, Sex, RBP, Height	61%	14.1%	73.7%	11.2%	70.4%	7.2%
	Age, Sex, RBP, Weight	60.9%	13%	72.6%	13.4%	69.7%	6.5%
	Age, Sex, Height, Weight	51.9%	9.3%	78.7%	12.8%	68.8%	7.1%
	Age, RBP, Height, Weight	63.7%	10.3%	67.8%	19%	67.3%	9.7%
	Sex, RBP, Height, Weight	53.6%	9.7%	73.8%	8.9%	65.4%	6.6%
Five	Age, Sex, RBP, Height, Weight	59.5%	13.4%	72.9%	11.7%	69%	6.6%

Table 7: Applying Decision Tree on Combined Non-Invasive Canberra Heart Disease Data Attributes

The combination of resting blood pressure and height shows the best mean accuracy of 79.2%. However, the mean sensitivity of this combination is 35%, a very small value - this combination result is discarded. The combination of age, sex, and resting blood pressure shows the next best mean accuracy of 74.8% followed by age and sex with mean accuracy of 74.2%. The difference of the mean accuracy between the age and sex combination and the age, sex, and resting blood pressure combination is just 0.6%. A t-test for significance is applied to determine if there is a significant difference between the two combinations. There are no significant differences between the two combinations (see Table 8). Applying the t-test between the sensitivity of the two combinations, there is no significant difference (Table 8). Although the t-test did not provide clear resolution, age, sex, and resting blood pressure combination is selected because it shows better mean accuracy and mean sensitivity with lower standard deviations. Thus age, sex, and resting blood pressure combination shows better results than other combinations in the diagnosis of Canberra heart disease dataset.

Including the height and weight attributes with the age, sex, and resting blood pressure non-invasive attributes combination, shows decrease in the mean accuracy to 69% (Table 7). Does converting height and weight into body mass index (BMI) or Rohrer's Index (RI) equation enhance the accuracy in the diagnosis of Canberra heart disease patients? The resting blood pressure attribute also includes both the high (Systolic) and low (Diastolic) resting blood pressures, so further analysis is needed to determine if using the difference between the resting blood pressure Systolic and Diastolic (called 'pulse pressure') with the age, sex, and resting blood pressures in combination can enhance accuracy. That is, the use of pulse pressure as a non-invasive attribute in the risk evaluation of heart disease needs further investigation.

Canberra Data Attributes	Accuracy			Sensitivity		
	Mean	St Dev	T-Test Significance	Mean	St Dev	T-Test Significance
Age, Sex	74.2%	7.5%	No (t = -1.448, p <= 0.05)	66.3%	14%	No (t = -0.444, p <= 0.05)
Age, Sex, RBP	74.8%	6.5%		66.7%	13.3%	

Table 8: T-Test Significance between Non-Invasive Canberra Heart Disease Data Combinations

The benchmark Cleveland heart disease dataset does not contain the height and weight attributes, so it is not possible to apply the BMI or Rohrer's Index equation to the Cleveland dataset.

5.3 Different Equations of Non-Invasive Attributes for Canberra Heart Disease Risk Evaluation

The Canberra dataset non-invasive attributes combinations (age, sex, and resting blood pressure) shows best results in heart disease diagnosis. Adding height and weight attributes to this combination decreases the mean accuracy. However, the height and weight attributes can be used to calculate BMI and RI values. The impact of BMI and RI equations in combination with age, sex, and resting blood pressure attribute combinations on mean accuracy is investigated. Adding the 'pulse pressure' to different non-invasive attribute combinations is also investigated.

Researchers have compared the RI to the BMI in its ability to predict body fat levels. One study suggested that RI may be a much better choice than BMI at assessing adults overweight (Valdez et al., 1996). However another study found that age specific BMI is better than age specific RI in predicting underweight or overweight (Mei et al., 2002). There is a need to identify how BMI and RI can enhance the ability of Decision Tree in the risk evaluation of heart disease.

Equation 6 uses the weight and height to calculate the BMI (Sultan et al., 2009).

$$BMI = Weight\ [kg] / (Height\ [m])^2 \quad \text{(Equation 6)}$$

Table 9 shows the mean and standard deviation of sensitivity, specificity, and accuracy resulting from adding the BMI equation to different combinations of age, sex, and resting blood pressure non-invasive Canberra dataset heart disease attributes. Integrating BMI with age, sex, and resting blood pressure does not enhance Decision Tree accuracy. The age, sex, resting blood pressure and BMI combination shows mean accuracy and mean sensitivity of 74% and 66.5% respectively. The age, sex, and resting blood pressure combination is showing mean accuracy and mean sensitivity of 74.8% and 66.7 % respectively (see Table 9). However, age, sex, resting blood pressure and BMI combination shows more stability in the standard deviation of the accuracy (5%) which is better than the standard deviation of the accuracy of the age, sex, and resting blood pressure combination (6.5%).

Rohrer's Index is a measure of leanness of a person calculated as a relationship between mass and height. It was first proposed in 1921 as the "Corpulence Index" by Rohrer and hence also known as Rohrer's Index. It is similar to the body mass index, but the mass is normalized with the third power of body height rather than the second power (Ensminger and Ensminger, 1993). Equation 7 uses the weight and height to calculate the Rohrer's Index.

$$Rohrer's\ Index = Weight\ [kg] / (Height\ [m])^3 \quad \text{(Equation 7)}$$

Canberra Attributes	Sensitivity		Specificity		Accuracy	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
Age, Sex, RBP	66.7 %	13.3 %	76.5 %	13.7 %	74.8 %	6.5 %
Age, BMI	61.6 %	12.8 %	72.4 %	16.1 %	66.6 %	8.6 %
Sex, BMI	60.9 %	14.2 %	69.5 %	11.9 %	68.8 %	6.1 %
RBP, BMI	48.8 %	10.8 %	66.8 %	5.2 %	59.1 %	3.8 %
Age, Sex, BMI	24%	17.1 %	93.7 %	8.1 %	62.4 %	9.5 %
Age, RBP, BMI	57%	14.1 %	74.8 %	16%	67.4 %	9.3 %
Sex, RBP, BMI	38.3 %	12.4 %	75.6 %	11.2 %	61.2 %	6%
Age, Sex, RBP, BMI	66.5 %	16.4 %	76.1 %	10.1 %	74%	5%

Table 9: Integrating BMI with Different Non-Invasive Canberra Heart Disease Data Attributes

Table 10 shows the mean and standard deviation of sensitivity, specificity, and accuracy when adding the Rohrer's Index equation to different combinations of age, sex, and resting blood pressure non-invasive Canberra heart disease attributes. Integrating the Rohrer's Index equation with the age, sex, and resting blood pressure attribute combination does not enhance Decision Tree accuracy in the diagnosis of Canberra heart disease patients. However, the mean and standard deviation of the sensitivity and the standard deviation of the accuracy are enhanced. The age, sex, resting blood pressure and Rohrer's Index combination shows mean and standard deviation of the accuracy of 73.8% and 4.9% respectively and mean and standard deviation of the sensitivity of 67.1% and 11.4 % respectively.

Canberra Attributes	Sensitivity		Specificity		Accuracy	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
Age, Sex, RBP	66.7%	13.3%	76.5 %	13.7%	74.8 %	6.5 %
Age, Rohrer's Index	66.8%	12.7%	71%	19.1%	70 %	6.7 %
Sex, Rohrer's Index	59.8%	12.2%	62.4 %	15.7%	64.4 %	6.5 %
RBP, Rohrer's Index	55%	12.1%	70.1 %	10.9%	61.9 %	6.5 %
Age, Sex, Rohrer's Index	33.7%	16.6%	91.8 %	4.8 %	66.3 %	12%
Age, RBP, Rohrer's Index	59.3%	13.3%	74.9 %	16.6%	69.1 %	9.2 %
Sex, RBP, Rohrer's Index	43.4%	10.7%	74%	6.4 %	61.2 %	8.2 %
Age, Sex, RBP, Rohrer's Index	67.1%	11.4%	75.5 %	9.2 %	73.8 %	4.9 %

Table 10: Integrating Rohrer's Index with Different Non-Invasive Canberra Heart Disease Data Attributes

The resting blood pressure attribute also includes both the high and low resting blood pressures, so further analysis is needed to determine if using the difference between the resting blood pressure high and low (called 'pulse pressure') with the age, sex, and resting blood pressure combination can enhance accuracy. The WHO report suggests that the pulse pressure can be used as an indicator of heart disease (World Health Organization, 2005). Table 11 shows the mean and standard deviation of sensitivity, specificity, and accuracy when adding the RBPDiff ('Pulse pressure') Equation (Equation. 8) to combinations of age, sex, and resting blood pressure attributes. The RBPDiff equation used in this investigated is:

RBPDiff = RBP High – RBP Low (Equation 8)

Integrating RBPDiff with age, sex, and resting blood pressure attributes does not enhance Decision Tree accuracy in the diagnosis of Canberra heart disease patients. The age, sex, resting blood pressure and RBPDiff combination shows mean accuracy and standard deviation of 72.6% and 5.2% respectively, and mean sensitivity and standard deviation of 65.3% and 14.9% respectively (see Table 11). Adding the RBPDiff with the age, sex, RBP, and BMI combination and age, sex, RBP, and Rohrer's Index combination does not enhance performance showing mean accuracy of 72.5% and 74.7% respectively (Table 11).

Adding the different BMI, Rohrer's Index, and RBPDiff equations with age, sex, and resting blood pressure attributes combinations, the mean accuracy is not enhanced. However, the standard deviation is decreased demonstrating better stability (see Table 12). In this context, sensitivity is more important because patients who are at high risk of heart disease need to be identified and get appropriate care. The age, sex, RBP, and Rohrer's Index attributes combination shows the best mean and standard deviation sensitivity followed by age, sex, and RBP combination (67.1%, 11.4% and 66.7%, 13.3% respectively – see Table 12).

Canberra Attributes	Sensitivity		Specificity		Accuracy	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
Age, Sex, RBP	66.7 %	13.3 %	76.5 %	13.7 %	74.8 %	6.5 %
Age, RBPDiff	65%	15.6 %	70.4 %	19.2 %	68%	9.4 %
Sex, RBPDiff	62.2 %	13%	60.3 %	16.5 %	63.1 %	8.3 %
RBP, RBPDiff	33.7 %	7.7 %	67.3 %	10.7 %	50.7 %	6.2 %
Age, Sex, RBPDiff	39.3 %	20.1 %	89.6 %	10.3 %	68.6 %	8.4 %
Age, RBP, RBPDiff	56.2 %	13%	73.5 %	17.4 %	67%	7.5 %
Sex, RBP, RBPDiff	26.6 %	8.9 %	83.5 %	6.5 %	57.7 %	11. 9%
Age, Sex, RBP, RBPDiff	65.3 %	14.9 %	73.4 %	13.3 %	72.6 %	5.2 %
Age, Sex, RBP, BMI, RBPDiff	63.9 %	15.6 %	75.1 %	12.5 %	72.5 %	6.5 %
Age, Sex, RBP, Rohrer's Index, RBPDiff	67.6 %	15.6 %	76%	10.1 %	74.7 %	6.3 %

Table 11: Integrating RBPDiff with Different Non-Invasive Canberra Heart Disease Data Attributes

Canberra Attributes	Sensitivity		Specificity		Accuracy	
	Mean	St Dev	Mean	St Dev	Mean	St Dev
Age, Sex, RBP	66.7%	13. 3%	76.5 %	13. 7%	74.8 %	6.5 %
Age, Sex, RBP, BMI	66.5%	16. 4%	76.1 %	10. 1%	74 %	5%
Age, Sex, RBP, Rohrer's Index	67.1%	11. 4%	75.5 %	9.2 %	73.8 %	4.9 %
Age, Sex, RBP, RBPDiff	65.3%	14. 9%	73.4 %	13. 3%	72.6 %	5.2 %

Table 12: Summarizing Integrating BMI, Rohrer's Index, and RBPDiff with Non-Invasive Canberra Heart Disease Data Attributes

The t-test is applied to the accuracy of the two combinations (age, sex, and RBP combination and age, sex, RBP, and Rohrer's Index combination) to identify if there is significant difference and shows significant difference between them (see Table 13). However, applying t-test to the sensitivity of the two combinations shows that there is no significant difference (Table 13). In this context, increasing the mean sensitivity and decreasing its standard deviation is more important. The age, sex, RBP, and Rohrer's Index combination shows better results than other non-invasive attributes combinations for the Canberra heart disease dataset.

Canberra Data Attributes	Accuracy			Sensitivity		
	Mean	St Dev	T-Test Significance	Mean	St Dev	T-Test Significance
Age, Sex, RBP	74.8 %	6.5 %	Yes (t = -2.635 , p <= 0.05)	66.7 %	13.3 %	No (t = 0.490 , p <= 0.05)
Age, Sex, RBP, Rohrer's Index	73.8 %	4.9 %		67.1 %	11.4 %	

Table 13: T-Test Significance for Adding Rohrer's Index to Non-Invasive Canberra Heart Disease Data Attributes

6 Conclusion and future work

This paper investigates applying Decision Tree data mining technique to identify the significance of non-invasive attributes in the diagnosis of heart disease

patients. It also investigates different combinations of non-invassive attributes in the diagnosis of heart disease patients. The results show that the best combination is age, sex, resting blood pressure and Rohrer's Index equation with mean accuracy and standard deviation of 73.8% and 4.9% respectively. The future work will explore a hybrid model Decision Tree in the diagnosis of heart disease patients using non-invasive attributes.

References

Andreeva, P. (2006). "Data modelling and specific rule generation via data mining techniques". International Conference on Computer Systems and Technologies.

Bitton, A. and Gaziano, T. (2010). "The Framingham Heart Study's impact on global risk assessment". Progress in cardiovascular diseases, **53**:68-78.

Bramer, M. (2007). "Principles of data mining", Springer.

Canlas Jr., R. D. (2009). "Data Mining in Healthcare: Current Applications and Issues".

Centers for Disease Control and Prevention. (2013). "Chronic Disease Prevention and Health Promotion" [Online]. Available: http://www.cdc.gov/nccdphp/. Accessed 27 September 2013].

Cieslak, D., Hoens, T. R., Chawla, N. and Kegelmeyer, W. P. (2012). "Hellinger distance decision trees are robust and skew-insensitive". Data Mining and Knowledge Discovery, **24**:136-158.

Cupples, L. and D'agostino, R. (1987). "Some risk factors related to the annual incidence of cardiovascular disease and death in pooled repeated biennial measurements". US Department of Health and Human Services.

Das, R., Turkoglu, I. and Sengur, A. (2009). "Effective diagnosis of heart disease through neural networks ensembles". Expert Systems with Applications, Elsevier, **36** (2009):7675–7680.

Ensminger, M. E. and Ensminger, A. H. (1993). "Foods & nutrition encyclopedia", CRC press.

Framingham Heart Study. (2013). "About the Framingham Heart Study" [Online]. Available: http://www.framinghamheartstudy.org/about/history.html. Accessed 5 October 2013].

Han, J. and Kamber, M. (2006). "Data Mining Concepts and Techniques", Morgan Kaufmann Publishers.

Kavitha, K., Ramakrishnan, K. and Singh, M. K. (2010). "Modeling and design of evolutionary neural network for heart disease detection". International Journal of Computer Science Issues (IJCSI), **7**.

Kotnik, T. (2010). "Prevention Programs". Challenges in Family Medicine.

Mei, Z., Grummer-Strawn, L. M., Pietrobelli, A., Goulding, A., Goran, M. I. and Dietz, W. H. (2002). "Validity of body mass index compared with other body-composition screening indexes for the assessment of body fatness in children and adolescents". The American journal of clinical nutrition, **75**:978-985.

National Center for Chronic Disease Prevention and Health Promotion. (2013). "Know the facts about heart disease" [Online]. Available: http://www.cdc.gov/heartdisease/docs/consumered_heartdisease.pdf. Accessed 9 October 2013].

National Heart Foundation of Australia (2009). "Guidelines for the assessment of Absolute cardiovascular disease risk".

Paladugu, S. (2010). "Temporal mining framework for risk reduction and early detection of chronic diseases". University of Missouri--Columbia.

Patel, V. L., Shortliffe, E. H., Stefanelli, M., Szolovits, P., Berthold, M. R., Bellazzi, R. and Abu-Hanna, A. (2009). "The coming of age of artificial intelligence in medicine". Artificial intelligence in medicine, **46**:5-17.

Sitar-Taut, V., Zdrenghea, D., Pop, D. and Sitar-Taut, D. (2009). "Using machine learning algorithms in cardiovascular disease risk evaluation". Journal of Applied Computer Science & Mathematics, **5**:29-32.

Sultan, K. M., Alobaidy, M. W. and Hussein, A. I. (2009). "The Prevalence of Weight Loss Assessed by Body Mass Index in Patients with Stable Chronic Obstructive Pulmonary Disease". The Iraqi postgraduate medical journal

Valdez, R., Greenlund, K., Wattigney, W., Bao, W. and Berenson, G. (1996). "Use of weight-for-height indices in children to predict adult overweight: the Bogalusa Heart Study". International journal of obesity and related metabolic disorders: journal of the International Association for the Study of Obesity, **20**:715-721.

World Health Organization (2005). "Clinical guidelines for the management of hypertension", WHO regional office for the eastern mediterranean.

World Health Organization. (2011b). "Burden: mortality, morbidity and risk factors" [Online]. Available: http://www.who.int/nmh/publications/ncd_report_chapter1.pdf. Accessed 15 December 2012.

Author Index

Recent Volumes in the CRPIT Series

ISSN 1445-1336

Listed below are some of the latest volumes published in the ACS Series *Conferences in Research and Practice in Information Technology*. The full text of most papers (in either PDF or Postscript format) is available at the series website `http://crpit.com`.

Volume 147 - Computer Science 2014
Edited by Bruce Thomas, University of South Australia and Dave Parry, AUT University, New Zealand. January 2014. 978-1-921770-30-2.
Contains the proceedings of the Australian System Safety Thirty-Seventh Australasian Computer Science Conference (ACSC 2014), Auckland, New Zealand, 20 – 23 January 2014.

Volume 148 - Computing Education 2014
Edited by Jacqueline Whalley, AUT University, New Zealand and Daryl D'Souza, RMIT University, Australia. January 2014. 978-1-921770-31-9.
Contains the proceedings of the Sixteenth Australasian Computing Education Conference (ACE2014), Auckland, New Zealand, 20 – 23 January 2014.

Volume 149 - Information Security 2014
Edited by Udaya Parampalli, University of Melbourne, Australia and Ian Welch, Victoria University of Wellington, New Zealand. January 2014. 978-1-921770-32-6.
Contains the proceedings of the Twelfth Australasian Information Security Conference (AISC 2014), Auckland, New Zealand, 20 – 23 January 2014.

Volume 150 - User Interfaces 2014
Edited by Burkhard C. Wünsche, University of Auckland, New Zealand and Stefan Marks, AUT University, New Zealand. January 2014. 978-1-921770-33-3.
Contains the proceedings of the Fifteenth Australasian User Interface Conference (AUIC 2014), Auckland, New Zealand, 20 – 23 January 2014.

Volume 151 - Australian System Safety Conference 2013
Edited by Tony Cant, Defence Science and Technology Organisation, Australia. May 2013. 978-1-921770-38-8.
Contains the proceedings of the Australian System Safety Conference (ASSC 2013), Adelaide, Australia, 22 – 24 May 2013.

Volume 152 - Parallel and Distributed Computing 2014
Edited by Bahman Javadi, University of Western Sydney, Australia and Saurabh Kumar Garg, IBM Research, Australia. January 2014. 978-1-921770-34-0.
Contains the proceedings of the Twelfth Australasian Symposium on Parallel and Distributed Computing (AusPDC 2014), Auckland, New Zealand, 20 – 23 January 2014.

Volume 153 - Health Informatics and Knowledge Management 2014
Edited by Jim Warren, University of Auckland, New Zealand and Kathleen Gray, University of Melbourne, Australia. January 2014. 978-1-921770-35-7.
Contains the proceedings of the Seventh Australasian Workshop on Health Informatics and Knowledge Management (HIKM 2014), Auckland, New Zealand, 20 – 23 January 2014.

Volume 154 - Conceptual Modelling 2014
Edited by Georg Grossmann, University of South Australia and Motoshi Saeki, Tokyo Institute of Technology, Japan. January 2014. 978-1-921770-36-4.
Contains the proceedings of the Tenth Asia-Pacific Conference on Conceptual Modelling (APCCM 2014), Auckland, New Zealand, 20 – 23 January 2014.

Volume 155 - The Web 2014
Edited by Stephen Cranefield, University of Otago, New Zealand, Andrew Trotman, University of Otago, New Zealand and Jian Yang, Macquarie University, Australia. January 2014. 978-1-921770-37-1.
Contains the proceedings of the Second Australasian Web Conference (AWC 2014), Auckland, New Zealand, 20 – 23 January 2014.

Volume 156 - Australian System Safety Conference 2014
Edited by Tony Cant, Defence Science and Technology Organisation, Australia. May 2014. 978-1-921770-39-5.
Contains the proceedings of the Australian System Safety Conference (ASSC 2014), Melbourne, Australia, 28 – 30 May 2014.

Volume 158 - Data Mining and Analytics 2014
Edited by Xue Li, University of Queensland, Lin Liu, University of South Australia, Kok-Leong Ong, Deakin University and Yanchang Zhao, Department of Immigration and Border Protection, Australia and RDataMining.com, Australia. November 2014. 978-1-921770-17-3.
Contains the proceedings of the Twelfth Australasian Data Mining Conference (AusDM'14), Brisbane, Australia, 27–28 November 2014.

Volume 159 - Computer Science 2015
Edited by David Parry, AUT University, New Zealand. January 2015. 978-1-921770-41-8.
Contains the proceedings of the 38th Australasian Computer Science Conference (ACSC 2015), Sydney, Australia, 27 – 30 January 2015.

Volume 160 - Computing Education 2015
Edited by Daryl D'Souza, RMIT University and Katrina Falkner, University of Adelaide, Australia. January 2015. 978-1-921770-42-5.
Contains the proceedings of the 17th Australasian Computing Education Conference (ACE 2015), Sydney, Australia, 27 – 30 January 2015.

Volume 161 - Information Security 2015
Edited by Ian Welch, Victoria University of Wellington, New Zealand and Xun Yi, RMIT University, Australia. January 2015. 978-1-921770-43-2.
Contains the proceedings of the 13th Australasian Information Security Conference (AISC 2015), Sydney, Australia, 27 – 30 January 2015.

Volume 162 - User Interfaces 2015
Edited by Stefan Marks, AUT University and Rachel Blagojevic, Massey University, New Zealand. January 2015. 978-1-921770-44-9.
Contains the proceedings of the 16th Australasian User Interface Conference (AUIC 2015), Sydney, Australia, 27 – 30 January 2015.

Volume 163 - Parallel and Distributed Computing 2015
Edited by Bahman Javadi, University of Western Sydney and Saurabh Kumar Garg, University of Tasmania, Australia. January 2015. 978-1-921770-45-6.
Contains the proceedings of the 13th Australasian Symposium on Parallel and Distributed Computing (AusPDC 2015), Sydney, Australia, 27 – 30 January 2015.

Volume 164 - Health Informatics and Knowledge Management 2015
Edited by Anthony Maeder, University of Western Sydney, Australia and Jim Warren, University of Auckland, New Zealand. January 2015. 978-1-921770-46-3.
Contains the proceedings of the 8th Australasian Workshop on Health Informatics and Knowledge Management (HIKM 2015), Sydney, Australia, 27 – 30 January 2015.

Volume 165 - Conceptual Modelling 2015
Edited by Motoshi Saeki, Tokyo Institute of Technology, Japan and Henning Kö, Massey University, New Zealand. January 2015. 978-1-921770-47-0.
Contains the proceedings of the 11th Asia-Pacific Conference on Conceptual Modelling (APCCM 2015), Sydney, Australia, 27 – 30 January 2015.

Volume 166 - Australasian Web Conference 2015
Edited by Joseph G. Davis, University of Sydney, Australia and Alessandro Bozzon, Delft University of Technology, The Netherlands. January 2015. 978-1-921770-48-7.
Contains the proceedings of the 3rd Australasian Web Conference (AWC 2015), Sydney, Australia, 27 – 30 January 2015.

Volume 167 - Interactive Entertainment 2015
Edited by Yusuf Pisan, University of Technology, Sydney, Keith Nesbitt and Karen Blackmore, University of Newcastle, Australia. January 2015. 978-1-921770-49-4.
Contains the proceedings of the 11th Australasian Conference on Interactive Entertainment (IE 2015), Sydney, Australia, 27 – 30 January 2015.