AI2002

Workshop Proceedings

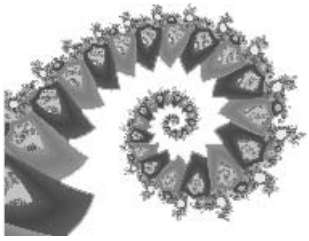
Data Mining

Edited by
Simeon J. Simoff,
Graham J. Williams and
Markus Hegland

The 15th **A**ustralian **J**oint **C**onference
on **A**rtificial **I**ntelligence 2002

Rydges Canberra, Australia

2 - 6 December 2002



ADMO2

Proceedings
Australasian Data Mining Workshop

3rd December, 2002, Canberra, Australia

Edited by
Simeon J. Simoff, Graham J. Williams and
Markus Hegland

in conjunction with
The 15th Australian Joint Conference
on Artificial Intelligence
Canberra – Australia,
2nd - 6th December, 2002

University of Technology Sydney
2002

© Copyright 2002. The copyright of these papers belongs to the paper's authors. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage.

Proceedings of the 1st Australasian Data Mining Workshop – ADM02, in conjunction with the 15th Australian Joint Conference on Artificial Intelligence, 2nd - 6th December, 2002, Canberra, Australia

S. J. Simoff, G. J. Williams and M. Hegland (eds).

Workshop Web Site:
http://datamining.csiro.au/adm02/

Published by the University of Technology Sydney

ISBN 0-9750075-0-5

Foreword

The Australasian Data Mining Workshop is devoted to the art and science of data mining: the analysis of (usually large) data sets to discover relationships and present the data in novel ways that are compact, comprehendible and useful for researchers and practitioners. Data mining projects involve both the utilisation of established algorithms from machine learning, statistics, and database systems, and the development of new methods and algorithms, targeted at large data mining problems. Nowadays data mining efforts have gone beyond crunching databases of credit card usage or stored transaction records. They have been focusing on data collected in the health care system, art, design, medicine, biology and other areas of human endeavour.

There has been an increasing interest in Australian industry, academia, research institutions and centers towards the area of data mining, evidenced by the growing number of research groups (e.g. ANU Data Mining Group, CSIRO Enterprise Data Mining, and UTS Smart e-Business Systems Lab), academic and industry events (e.g. the data mining seminar series organised by PricewaterhouseCoopers Actuarial Sydney) related to one or another aspect of data mining. The workshop is aiming to bring together people from academia and industry that are working in the development and application of data mining methods, techniques and technologies. This workshop aims to bring together researchers and industry practitioners from different data mining groups in Australia and the region, and overseas researchers and practitioners that are working in the development and application of data mining methods, techniques and technologies. The workshop is expected to become a forum for presenting and discussing their latest research and developments in the area. The works selected for presentation at the workshop are expected to facilitate the cross-disciplinary exchange of ideas and communication between industry and academia in the area of data mining and its applications. Consequently, the morning part of the workshop (the sessions on “Practical Data Mining” and “Applications of Data Mining") addresses the data mining practice. The afternoon part of the workshop includes sessions on “Data Mining Methods and Algorithms”, “Spatio-Temporal Data Mining”, and “Data Preprocessing and Supporting Technologies”. The organisers have also reserved a special presentation session for an overview of on-going projects.

As part of the Australian Joint Conference on Artificial Intelligence the workshop follows a rigid peer-review and paper selection process. Once again, we would like to thank all those, who supported this year’s efforts on all stages – from the development and submission of the workshop proposal to the preparation of the final program and proceedings. We would like to thank all those who submitted their work to the workshop. All papers were extensively reviewed by two to three referees drawn from the program committee. Special thanks go to them for the final quality of selected papers depends on their efforts.

Simeon, J. Simoff, Graham J. Williams and Markus Hegland

November 2002

Workshop Chairs

Simeon J. Simoff	University of Technology Sydney, Australia
Graham J. Williams	Enterprise Data Mining, CSIRO, Australia
Markus Hegland	Australian National University, Australia

Program Committee

Sergei Ananyan	Megaputer Intelligence, Russia & USA
Rohan Baxter	Enterprise Data Mining, CSIRO, Australia
John Debenham	University of Technology Sydney, Australia
Vladimir Estivill-Castro	Giffith University, Australia
Eibe Frank	University of Waikato, New Zealand
Paul Kennedy	University of Technology Sydney
Inna Kolyshkina	PricewaterhouseCoopers Actuarial Sydney, Australia
Kevin Korb	Monash University, Australia
Xuemin Lin	University of NSW, Australia
Warwick Graco	Health Insurance Commision, Australia
Ole Nielsen	Australian National University, Australia
Tom Osborn	NUIX Pty Ltd, and The NTF Group, Australia
Chris Rainsford	Enterprise Data Mining, CSIRO, Australia
John Roddick	Flinders University, Australia
David Skillicorn	Queen's University, Canada
Dan Steinberg	Salford Systems, USA

Program for ADM02 Workshop

Tuesday, 3 December, 2002, Canberra, Australia

9:00 - 9:10 Opening and Welcome

9:10 - 10:30 Session 1 – Practical Data Mining

- 09:10 - 10:00 STOCHASTIC GRADIENT BOOSTING: AN INTRODUCTION TO TreeNet™
  Dan Steinberg, Mikhail Golovnya and N. Scott Cardell
- 10:00 - 10:20 CASE STUDY: MODELING RISK IN HEALTH INSURANCE - A DATA MINING APPROACH
  Inna Kolyshkina and Richard Brookes

10:20 - 10:35 Coffee break

10:35 - 12:15 Session 2 – Applications of Data Mining

- 10:35 - 11:00 INVESTIGATIVE PROFILE ANALYSIS WITH COMPUTER FORENSIC LOG DATA USING ATTRIBUTE GENERALISATION
  Tamas Abraham, Ryan Kling and Olivier de Vel
- 11:00 - 11:25 MINING ANTARCTIC SCIENTIFIC DATA: A CASE STUDY
  Ben Raymond and Eric J. Woehler
- 11:25 - 11: 50 COMBINING DATA MINING AND ARTIFICIAL NEURAL NETWORKS FOR DECISION SUPPORT
  Sérgio Viademonte and Frada Burstein
- 11:50 - 12:15 TOWARDS ANYTIME ANYWHERE DATA MINING E-SERVICES
  Shonali Krishnaswamy, Seng Wai Loke and Arkady Zaslavsky

12:15 - 13:00 Lunch

13:00 - 14:00 Session 3 – Data Mining Methods and Algorithms

- 13:00 - 13:20 A HEURISTIC LAZY BAYESIAN RULE ALGORITHM
  Zhihai Wang and Geoffrey I. Webb
- 13:20 - 13:40 AVERAGED ONE-DEPENDENCE ESTIMATORS: PRELIMINARY RESULTS
  Geoffrey I. Webb, Janice Boughton and Zhihai Wang
- 13:40 - 14:00 SEMIDISCRETE DECOMPOSITION: A BUMP HUNTING TECHNIQUE
  S. McConnell and David B. Skillicorn

14:00 - 14:40 Session 4 – Spatio-Temporal Data Mining

- 14:00 - 14:20 AN OVERVIEW OF TEMPORAL DATA MINING
  Weiqiang Lin, Mehmet A. Orgun and Graham. J. Williams
- 14:20 - 14:40 DISTANCES FOR SPATIO-TEMPORAL CLUSTERING
  Mirco Nanni and Dino Pedreschi

14:40 - 14:55 Coffee break

14:55 - 16:10 Session 5 – Data Preprocessing and Supporting Technologies

- 14:55 - 15:20 PROBABILISTIC NAME AND ADDRESS CLEANING AND STANDARDISATION
  Peter Christen, Tim Churches and Justin Zhu
- 15:20 - 15:45 BUILDING A DATA MINING QUERY OPTIMIZER
  Raj P. Gopalan, Tariq Nuruddin and Yudho Giri Sucahyo
- 15:45 - 16:10 HOW FAST IS -FAST? PERFORMANCE ANALYSIS OF KDD APPLICATIONS USING HARDWARE PERFORMANCE COUNTERS ON ULTRASPARC-III
  Adam Czezowski and Peter Christen

16:10 - 17:00 Session 6 – Project Reports, Discussion and Closure

Table of Contents

Stochastic Gradient Boosting: An Introduction to TreeNet

Dan Steinberg, Mikhail Golovnya, N. Scott Cardell
Salford Systems

Stochastic Gradient Boosting

An introduction to TreeNet™

Salford Systems
http://www.salford-systems.com
dstein@salford-systems.com
Dan Steinberg, Mikhail Golovnya, N. Scott Cardell

© Copyright Salford Systems 2001-2002

Introduction to Stochastic Gradient Boosting

- New approach to machine learning and function approximation developed by Jerome H. Friedman at Stanford University
  - Co-author of CART® with Breiman, Olshen and Stone
  - Author of MARS™, PRIM, Projection Pursuit
- Good for classification and regression problems
- Builds on the notions of committees of experts and boosting but is substantially different in key implementation details

© Copyright Salford Systems 2001-2002

Benefits of TreeNet

- Built on CART trees and thus
  - immune to outliers
  - handles missing values automatically
  - selects variables,
  - results invariant wrt monotone transformations of variables
- Resistant to mislabeled target data
  - In medicine cases are commonly misdiagnosed
  - In business, non-responders are occasionally flagged as "responders"
- Resistant to overtraining – generalizes well
- Can be remarkably accurate with little effort
- Trains rapidly; at least as fast as CART

© Copyright Salford Systems 2001-2002

Stochastic Gradient Boosting: Key Innovations -1

- Stagewise function approximation in which each stage models residuals from last step model
  - Conventional boosting models use the original target at each stage
  - Each stage uses a very small tree, as small as two nodes and typically in the range of 4-8 nodes
  - Conventional bagging and boosting use full size trees
  - Bagging works best with massively large trees (1 case in each terminal node)
- Each stage learns from a fraction of the available training data, typically less than 50% to start and often falling to 20% or less by the last stage

© Copyright Salford Systems 2001-2002

Stochastic Gradient Boosting: Key Innovations -2

- Each stage learns only a little: severely downweighted contribution of each new tree (learning rate is typically 0.10, even 0.01 or less)
  - How much is learned in each stage compared to a single tree
- In classification, focus is on points near decision boundary; ignores points far away from boundary even if the points are on the wrong side
  - If we do very badly on certain observations we ignore them
  - Unlike boosting which would upweight such points
    - Explains why boosting is vulnerable to mislabeled data

© Copyright Salford Systems 2001-2002

Combining Trees into "Committees of Experts"

- Idea that combining good methods could yield promising results first was suggested by researchers a decade ago
  - In tree-structured analysis, suggestions made by:
    - Wray Buntine (1991, Bayes style allows cases to go down several tree paths)
    - Kwok and Carter (1990, split nodes several different ways to get alternate trees)
    - Heath, Kasif and Salzberg (1993, split nodes several different ways using different linear combination splitters)
- More recent work introduced concepts of bootstrap aggregation ("bagging"), adaptive resampling and combining ("arcing") and boosting
  - Breiman (1994, 1996, multiple independent trees via sampling with replacement)
  - Breiman (1996, multiple trees with adaptive reweighting of training data)
  - Freund and Schapire (1996, multiple trees with adaptive reweighting of training data)

© Copyright Salford Systems 2001-2002

Trees Can be Combined By Voting or Averaging

- Trees combined via voting (classification) or averaging (regression)
- Classification trees "vote"
  - Recall that classification trees classify
    - assign each case to ONE class only
  - With 50 trees, 50 class assignments for each case
  - Winner is the class with the most votes
  - Votes could be weighted – say by accuracy of individual trees
- Regression trees assign a real predicted value for each case
  - Predictions are combined via averaging
  - Results will be much smoother than from a single tree

© Copyright Salford Systems 2001-2002

Bootstrap Resampling Effectively Reweights Training Data (Randomly and Independently)

- Probability of being omitted in a single draw is (1 - 1/n)
- Probability of being omitted in all n draws is $(1 - 1/n)^n$
- Limit of series as n increases is (1/e) = 0.368
  - approximately 36.8% sample excluded — 0 % of resample
  - 36.8% sample included once — 36.8 % of resample
  - 18.4% sample included twice thus represent ... — 36.8 % of resample
  - 6.1% sample included three times ... — 18.4 % of resample
  - 1.9% sample included four or more times ... — 8 % of resample
  - 100 %
  - Example: distribution of weights in a 2,000 record resample:

0	1	2	3	4	5	6
732	749	359	119	32	6	3
0.366	0.375	0.179	0.06	0.016	0.003	0.002

© Copyright Salford Systems 2001-2002

Bootstrap Aggregation Performance Gains

Statlog Data Set Summary

Data Set	# Training	# Variables	# Classes	# Test Set
Letters	15,000	16	26	5,000
Satellite	4,435	36	6	2,000
Shuttle	43,500	9	7	14,500
DNA	2,000	60	3	6,186

Test Set Misclassification Rate (%)

Data Set	1 Tree	Bag	Decrease
Letters	12.6	6.4	49%
Satellite	14.8	10.3	30%
Shuttle	0.062	0.014	77%
DNA	6.2	5.0	19%

© Copyright Salford Systems 2001-2002

Adaptive Resampling and Combining
(ARCing, a Variant of Boosting)

- Bagging proceeds by independent, identically-distributed resampling draws
- Adaptive resampling: probability that a case is sampled varies dynamically
- Starts with all cases having equal probability
- After first tree is grown, weight is increased on all *misclassified* cases
- For regression, weight increases with prediction error for that case
- Idea is to focus tree on those cases most difficult to predict correctly

© Copyright Salford Systems 2001-2002

ARCing reweights the training data

- Similar procedure first introduced by Freund & Schapire (1996)
- Breiman variant (ARC-*x*4) is easier to understand:
  - Suppose we have already grown *K* trees:
    let $m(j)$ = # times case j was misclassified ($0 <= m(j) <= K$)
  - Define $w(j) = (1 + m(j)^4)$
  - Prob (sample inclusion) = $w(j) \Big/ \sum_{i=1}^{M} w(i)$
  - Weight = 1 for cases with zero occurrences of misclassification
  - Weight = $1 + K^4$ for cases with *K* misclassifications
    - Rapidly becomes large if case is difficult to classify
  - Samples will tend to be increasingly dominated by misclassified cases

© Copyright Salford Systems 2001-2002

Problems with Boosting

- Boosting in general is vulnerable to overtraining
  - Much better fit on training than on test data
  - Tendency to perform poorly on future data
- Boosting highly vulnerable to errors in the data
  - Technique designed to obsess over errors
  - Will keep trying to "learn" patterns to predict miscoded data
- Documented in study by Dietterich (1998)
  - An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization

© Copyright Salford Systems 2001-2002

Stochastic Gradient Boosting

- Building on multiple tree ideas and adaptive learning
- Goal of avoiding shortcomings of standard boosting
- Placed in the context of function approximation
- Trees combined by adding them (adding scores)
  - Friedman calls it Multiple Additive Regressive Trees (MART)
  - Salford calls it TreeNet™

© Copyright Salford Systems 2001-2002

Function Approximation By a Series of Error Corrections

- Our approximation to any function can be written as

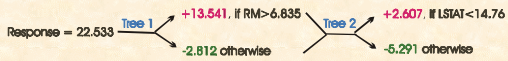
$$F(X) = F_0 + \beta_1 T_1(X) + \beta_2 T_2(X) + \ldots + \beta_M T_M(X)$$

- Where F_0 is the initial guess, usually what we would use in the absence of any model (e.g. mean, median, etc.)
- The approximation is built up stagewise
  - Once a stage is added it is never revised or refit
- Each stage added by assessing model and attempting to improve its quality by, for example, reducing residuals
  - Each stage is a "weak learner" – a small tree

© Copyright Salford Systems 2001-2002

Function Approximation By a Series of Trees

- Consider Boston Housing data set
  - Average neighborhood home value is $22,533
- Start model F(x) with this mean and construct residuals
- Model residuals with two-node tree
  - This is just an error correction based on one dimension of data
  - Model will attempt to separate positive from negative residuals
- Now update model, obtain new residuals and repeat process
- Estimated function will look something like this:



© Copyright Salford Systems 2001-2002

Function Approximation By a Series of Adjustments

- Function is built up through a series of adjustments or considerations
- Each adjustment adds (or subtracts) something from the current estimate of function value
- When we know nothing our home value prediction is the mean
  - Then we take number of rooms into account and adjust upwards for larger houses and downwards for smaller houses
  - Then we take socioeconomic status of residents into account and again adjust up or down
  - Continue taking further factors into account until an optimal model is built
- Similar to building up a score from a checklist of important factors (get points for certain characteristics, lose points for others)

© Copyright Salford Systems 2001-2002

Adjusting Trees Can be Any Size

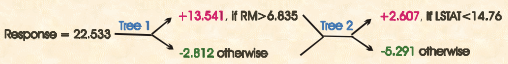
- Two-node, three-node, and larger trees can be used
- Friedman finds that six-node trees generally work well
- A tree with more than two nodes still adjusts the existing model
  - May take several variables into account simultaneously
  - Each tree just partitions data into subsets
  - Each subset gets a separate adjustment

$$F(X) = F_0 + \beta_1 T_1(X) + \beta_2 T_2(X) + \ldots + \beta_M T_M(X)$$

© Copyright Salford Systems 2001-2002

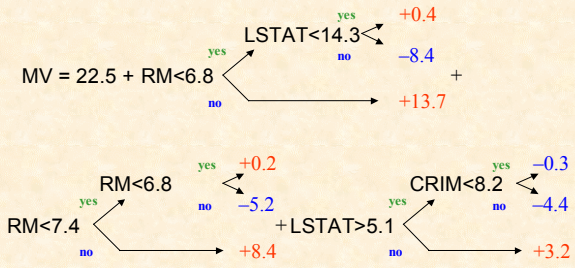
Two-node adjusting trees create main effects-only models

- Consider again the Boston Housing data set model
- Each tree involves only one variable
- Each contribution of any one tree not dependent on which branch a case terminates in any other tree
- High LSTAT reduces estimated home values by same amount regardless of number of rooms in house



© Copyright Salford Systems 2001-2002

TreeNet Model with Three-Node Trees



Each tree has three terminal nodes, thus partitioning data at each stage into three segments

© Copyright Salford Systems 2001-2002

Rationale for Additive Trees

- Want to provide this style of function approximation with some theoretical justification
- Need to specify many details:
  - How to choose tree size to use
  - How many forward steps to take
  - How to identify optimal model
  - How to interpret model and results
  - How much to adjust at a step
- Need to describe practical performance
  - Comparison with conventional boosting and single trees

© Copyright Salford Systems 2001-2002

Predictive Modeling and Function Approximation

GIVEN

Y	⇨ Output or Response Variable
X	⇨ Inputs or Predictors
L(Y, F)	⇨ Loss Function

ESTIMATE

$$F^*(\mathbf{X}) = \arg\min_{F(\mathbf{X})} E_{Y,\mathbf{X}}[L(Y,F(\mathbf{X}))]$$

© Copyright Salford Systems 2001-2002

Classical Function Approximation

- Specify a functional form for F(x), known up to a set of parameters B
- Learn by fitting F*(x) to data, minimizing loss measure L
- Achieved by iterative search procedure in which B is adjusted with reference to gradient (∂L/ ∂F)(∂F/ ∂B)
- Final result is obtained by adding together a series of parameter changes guided by gradient at an iteration
- Think of this as a gradual form of learning from the data

© Copyright Salford Systems 2001-2002

Nonparametric Function Approximation

- General non-parametric case: F(**X**) is treated as having a *separate parameter for each distinct combination of predictors X*
- With infinite data best estimate of F(X) under quadratic loss at any specific data vector Xi would be

$$F^*(X_i) = \frac{1}{N_{X_i}} \sum_{j:X_j=X_i} y_j$$

- With plentiful data accurate estimates of F(X) can be obtained for any X
- But we only have finite data so
  - most possible X vectors not represented in the data
  - lack of replicates means inaccurate estimates at any X
- Direct optimization in N free parameters will result in a dramatic overfitting
  - will somehow have to limit the total number of free parameters

© Copyright Salford Systems 2001-2002

General Optimization Strategy for Function Approximation

- Make an initial guess $\{F_o(\mathbf{X}_i)\}$ – for example, assuming that all $F_o(\mathbf{X}_i)$ are the same for all $\mathbf{X}_i$
- Compute the negative gradient at each observed data point *i*

$$\vec{g} = -\left\{\frac{\partial \hat{L}}{\partial F(X_i)}\right\}_{i=1}^{N}$$

- The negative gradient gives us the direction of the steepest descent
- Take a step in the steepest descent direction

© Copyright Salford Systems 2001-2002

Guarding Against Overfit In the Non-parametric Case

- Literal steepest descent is inadvisable as it would allow free adjustment of one parameter for each data point
- Instead, limit the number of free parameters that can be adjusted to a small number, say *L*.
- Can do this by partitioning data into *L* mutually exclusive groups making a common adjustment within each group
- The challenge is to find a good partitioning of data into *L* mutually exclusive groups
  - Within each group gradients should be similar

© Copyright Salford Systems 2001-2002

Identifying Common Gradient Partitions with Regression Trees

- Our goal is to group observations with similar gradients together so that a common adjustment can be made to the model for each group
- Build an *L*-node regression tree with the target being the negative gradient

© Copyright Salford Systems 2001-2002

Generic Gradient Boosting Algorithm

For the given estimate of LOSS, and iterations M

1. Choose start value $\{F(\mathbf{X}_i)\}=\{F_o(\mathbf{X}_i)\}$ (e.g. mean, for all data)
2. FOR m = 1 TO M
   3. Compute $\mathbf{g}_m$, the derivative of the expected loss with respect to $F(\mathbf{X}_i)$ evaluated at $F_{m-1}(\mathbf{X}_i)$ (e.g. residual, deviance)
   4. Fit an *L*-node regression tree to the components of the negative gradient ⇨ this will partition observations into *L* mutually exclusive groups
   5. Find the within-node update $\mathbf{h}_m(\mathbf{X}_i)$, adjusting each node separately: conventional model updating
   6. Update: $\{F_m(\mathbf{X}_i)\} = \{F_{m-1}(\mathbf{X}_i)\} + \mathbf{h}_m(\mathbf{X}_i)$
7. END

© Copyright Salford Systems 2001-2002

Gradient Boosting for Least Squares Loss

$$\hat{L}(\{F(X_i)\}) = \frac{1}{N}\sum_{i=1}^{N}(Y_i - F(X_i))^2$$

1. Initial guess $\{F_0(\mathbf{X}_i)\}=\{\text{ave}(Y_i)\}$
2. FOR m = 1 TO M
   3. $\mathbf{g}_m \sim \{Y_i - F_{m-1}(X_i)\} = \{\text{Residual}_i\}$
   4. Fit an *L*-node regression tree to the current residuals ⇨ this will partition observations into *L* mutually exclusive groups
   5. For each given node: $h_m(\mathbf{X}_i)$ = node-ave(Residual$_i$)
   6. Update: $\{F_m(\mathbf{X}_i)\} = \{F_{m-1}(\mathbf{X}_i)\} + \mathbf{h}_m(\mathbf{X}_i)$
7. END

© Copyright Salford Systems 2001-2002

Gradient Boosting for Least Absolute Loss

$$\hat{L}(\{F(X_i)\}) = \frac{1}{N}\sum_{i=1}^{N}|Y_i - F(X_i)|$$

1. Initial guess $\{F_0(\mathbf{X}_i)\}=\{\text{median}(Y_i)\}$
2. FOR m = 1 TO M
   3. $\mathbf{g}_m \sim \{\text{sign}(Y_i - F_{m-1}(X_i))\} = \{\text{sign}(\text{Residual}_i)\}$
   4. Fit an *L*-node regression tree to the *signs of the current residuals (+1, -1)*: this will partition observations into *L* mutually exclusive groups
   5. For each given node: $h_m(\mathbf{X}_i)$=node-median(Residual$_i$)
   6. Update: $\{F_m(\mathbf{X}_i)\} = \{F_{m-1}(\mathbf{X}_i)\} + \mathbf{h}_m(\mathbf{X}_i)$
7. END

© Copyright Salford Systems 2001-2002

Gradient Boosting for Classification: Binary Response

- In the case of binary response, the negative log-likelihood function is used in place of the loss function
- Friedman codes Y as {+1, -1} with conditional probabilities

$$P(y \mid X) = \frac{1}{1+e^{-yF(X)}},\ y \in \{-1,+1\}$$

- Here $F(X) = \log\frac{P(Y=1 \mid X)}{P(Y=-1 \mid X)}$ – log-odds ratio at *X*
- *F(X)* can range from - infinity to +infinity

© Copyright Salford Systems 2001-2002

TreeNet and Binary Response

$$L(\{F(X_i)\}) = \sum_{i=1}^{N} \log\left(1 + e^{-y_i F(X_i)}\right)$$

1. Initial guess $F_0(X) = \log\dfrac{1+\bar{y}}{1-\bar{y}}$
2. FOR m = 1 TO M
   3. $\mathbf{g}_m \sim \left\{y_i \Big/ \left(1 + e^{y_i F_{m-1}(X_i)}\right)\right\} = \{\tilde{y}_i\}$
   4. Fit an L-node regression tree to the "residuals" (see the next slide) computed above ⇨ this will partition observations into L mutually exclusive groups
   5. For each node $h_m(X_i) = \sum_{node} \tilde{y}_i \Big/ \sum_{node} |\tilde{y}_i|(1-|\tilde{y}_i|)$
   6. Update: $\{F_m(\mathbf{X}_i)\} = \{F_{m-1}(\mathbf{X}_i)\} + \mathbf{h}_m(\mathbf{X}_i)$
7. END FOR

© Copyright Salford Systems 2001-2002

Interpretation

- Put Y=1 in focus and call p – probability that Y=1
- Then $p_i = 1\Big/\left(1 + e^{-F(X_i)}\right)$
- Initial guess = Log[overall resp. rate / (1 – overall resp. rate)]
- "Residual" $\tilde{y}_i = \begin{cases} 1 - p_i, & \text{if } y_i = 1 \\ -p_i, & \text{otherwise} \end{cases}$
- Update $\mathbf{h}_m(\mathbf{X}_i)$ =(Node resp. rate – Ave. node(p))/Var

$$\text{Ave. node}(p) = \sum_{node} p_i \Big/ N_{node} \quad \text{Var} = \sum_{node} p_i(1-p_i) \Big/ N_{node}$$

© Copyright Salford Systems 2001-2002

A Note on Mechanics

- The tree is grown to group observations into homogenous subsets
- Once we have the right partition our update quantities for each terminal node are computed in a separate step
- The update is not necessarily taken from the tree predictions
- Important notion: tree is used to define a structure based on the split variables and split points
- What we do with this partition may have nothing to do with the usual predictions generated by the trees

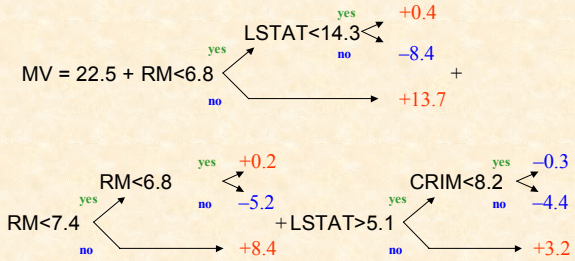
© Copyright Salford Systems 2001-2002

Slowing the learn rate: "Shrinkage"

- Up to this point we have guarded against overfitting by reducing the number of free parameters to be optimized
- It is beneficial to slow down the learning rate by introducing the shrinkage parameter 0<ν<1 into the update step:
  $\{F_m(\mathbf{X}_i)\} = \{F_{m-1}(\mathbf{X}_i)\} + \nu\, \mathbf{h}_m(\mathbf{X}_i)\}$
- With a group of correlated variables, only one variable in the group might enter the model with ν=1, whereas with ν<1 several variables in the group may enter at the later steps.

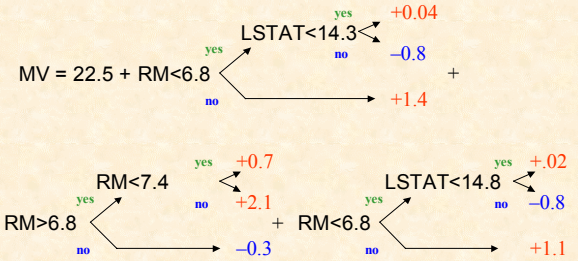
© Copyright Salford Systems 2001-2002

TreeNet Three-Node Trees Model: Learn Rate=1



© Copyright Salford Systems 2001-2002

TreeNet Three-Node Trees Model: Learn Rate= 0.1



Adjustments are smaller and evolution of model differs

© Copyright Salford Systems 2001-2002

Stochastic Training Data

- A further enhancement in performance is obtained by not allowing the learner to have access to all the training data at any one time
  - JHF recommends 50% random sampling rate at any one iteration
- No a priori limit on the number of iterations so there is always plenty of opportunity to learn from all the data eventually
- By limiting the amount of data at any one iteration we reduce the probability that an erroneous data point will gain influence over the learning process
- In complete contrast to standard boosting in which problem data points are "locked onto" with steadily growing weight and influence

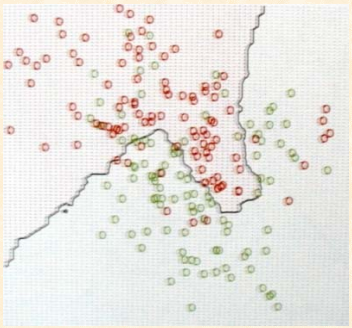
© Copyright Salford Systems 2001-2002

Ignoring data far from the decision boundary in classification problems

- A further reduction in training data actually processed in any update occurs in classification problems
- We ignore data points "too far" from the decision boundary to be usefully considered
  - Correctly classified points are ignored (as in conventional boosting)
  - Badly misclassified data points are also ignored (very different from conventional boosting)
  - The focus is on the cases most difficult to classify correctly: those near the decision boundary

© Copyright Salford Systems 2001-2002

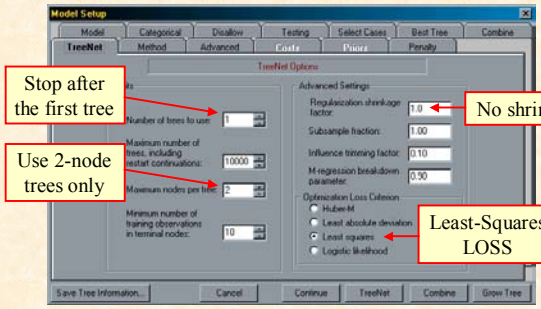
Decision Boundary Diagram



- 2-dimensional predictor space
- Red dots represent cases with +1 target
- Green dots represent cases with –1 target
- Black curve represents the decision boundary

© Copyright Salford Systems 2001-2002

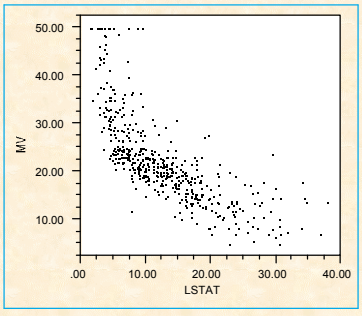
A Simple TreeNet Run



- **BOSTON HOUSING DATA: Target is MV (median neighborhood home value)**
- **One Predictor: LSTAT (% residents low socio-economic status)**

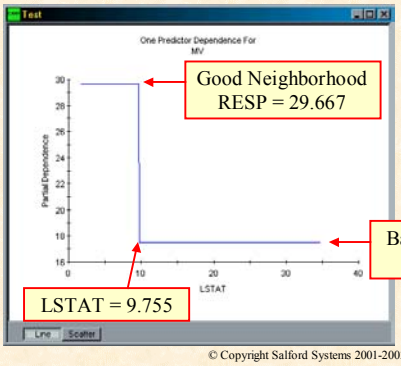
© Copyright Salford Systems 2001-2002

Scatter Plot: MV vs. LSTAT



© Copyright Salford Systems 2001-2002

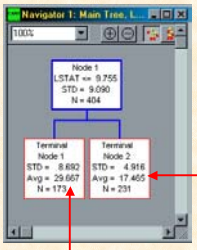
TreeNet Predicted Response



- One-step model
- A regression **tree** with 2 terminal nodes

© Copyright Salford Systems 2001-2002

Identical results from CART Model



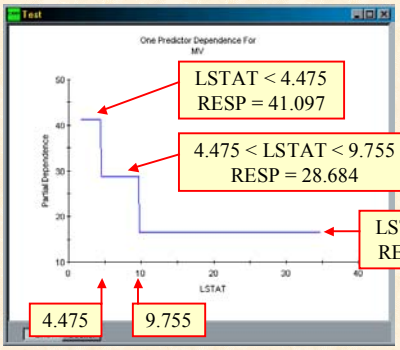
- CART run with TARGET=MV PREDICTORS=LSTAT LIMIT DEPTH=1
- Save residuals as RES1

LSTAT > 9.755
RESP = 17.465

LSTAT < 9.755
RESP = 29.667

© Copyright Salford Systems 2001-2002

TreeNet Model with two 2-node Trees



LSTAT < 4.475
RESP = 41.097

4.475 < LSTAT < 9.755
RESP = 28.684

LSTAT > 9.755
RESP = 16.482

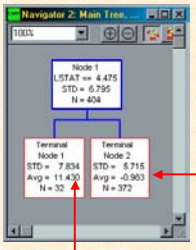
4.475

9.755

- Similar to a regression tree with 3 terminal nodes
- LSTAT is only predictor

© Copyright Salford Systems 2001-2002

Equivalent Two-stage CART Run



- CART run with TARGET=RES1 ← First Run Residuals PREDICTORS=LSTAT LIMIT DEPTH=1
- Save residuals as RES2

LSTAT > 4.475
RESP = -0.983

LSTAT < 4.475
RESP = 11.430

These are within-node adjustments to the 1st run RESPONSE

© Copyright Salford Systems 2001-2002

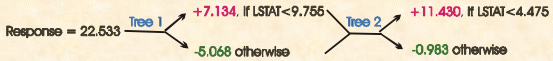
Computing RESPONSE -1

- 1st CART Run produced:
  - IF LSTAT < 9.755 THEN RESP1 = 29.667
  - IF LSTAT > 9.755 THEN RESP1 = 17.465
- 2nd CART Run produced:
  - IF LSTAT < 4.475 THEN ADJUST = 11.430
  - IF LSTAT > 4.475 THEN ADJUST = -0.983
- Combining two CART runs:
  - IF LSTAT < 4.475 THEN RESP2 = 29.667+11.430 = 41.097
  - IF 4.475< LSTAT< 9.755 THEN RESP2=29.667-0.983 = 28.684
  - IF LSTAT > 9.755 THEN RESP2 = 17.465 - 0.983 = 16.482
- This is exactly what was reported by TreeNet

© Copyright Salford Systems 2001-2002

Computing Response -2

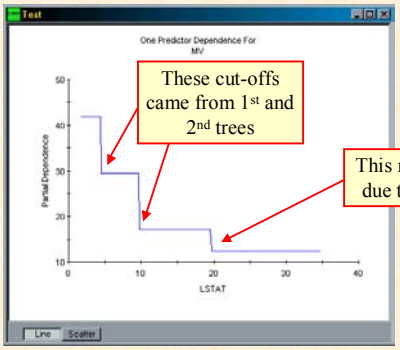
- This process can be schematically shown as



- Each tree in the sequence can be grown on the entire training data set. Unlike a decision tree we do not lose sample size as the learning progresses

© Copyright Salford Systems 2001-2002

TreeNet Run with 3 Trees



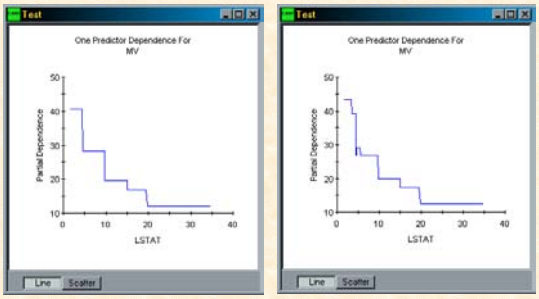
These cut-offs came from 1st and 2nd trees

This new cut-off is due to the 3rd tree

- Still just one predictor in model
- Now we obtain 4 regions

© Copyright Salford Systems 2001-2002

TreeNet Runs with 4 and 9 Trees



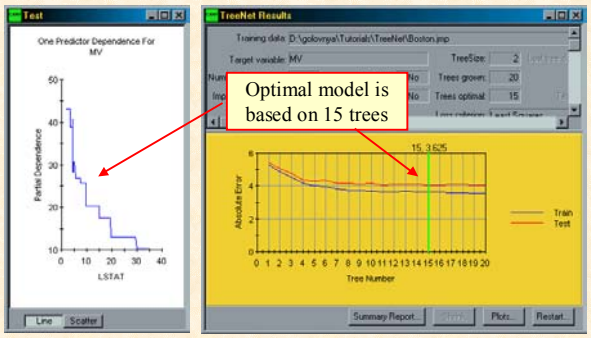
© Copyright Salford Systems 2001-2002

Optimal Tree is identified by reference to test data performance

- A Treenet model can be evolved indefinitely
  - Want to be able to pick the "right-sized" model
  - Although resistant to overfitting the model can overfit drastically in smaller data sets
- All model results refer by default to performance on test data
  - Require independent test sample
  - Cross-validation methods not available (yet)
- For practical real time scoring may also want to select an overly small model

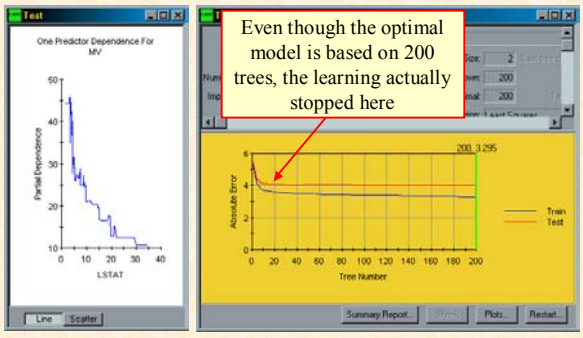
© Copyright Salford Systems 2001-2002

TreeNet Run with 20 Trees



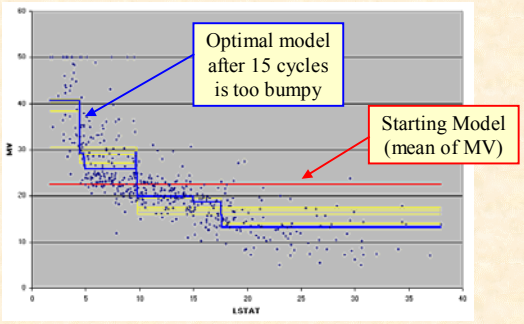
© Copyright Salford Systems 2001-2002

TreeNet Run with 200 Trees



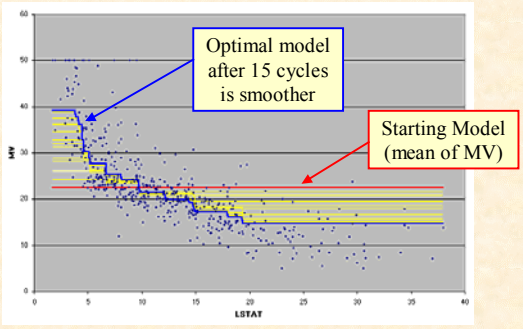
© Copyright Salford Systems 2001-2002

First 15 Runs (No Shrinkage)



© Copyright Salford Systems 2001-2002

First 15 Runs (Shrinkage at .2)



© Copyright Salford Systems 2001-2002

Interactions and TreeNet

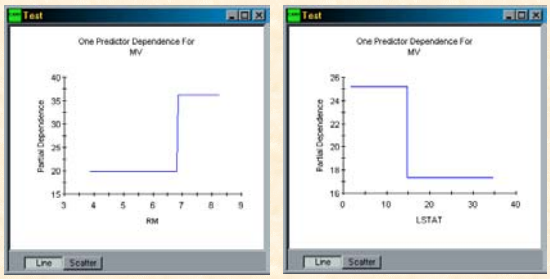
- At m-th cycle, TreeNet model can be represented by the following formula:

$$F_m(\mathbf{x}) = \sum_{i=1}^{m} h(\mathbf{x};\mathbf{a}_i)$$

- Here $h(\mathbf{x};\mathbf{a}_i)$ stands for individual tree at cycle i.
- It now becomes clear that the order of interactions only depends on the complexity of individual terms in the sum above, therefore:
  - "Stumps" (each tree has only one split based on a single variable) always result an additive model
  - Trees with L terminal nodes may allow up to L-1 interactions

© Copyright Salford Systems 2001-2002

TreeNet Run with 2 Predictors and 2 Trees



- First tree uses RM (Number of Rooms)
- Second tree uses LSTAT to update residuals

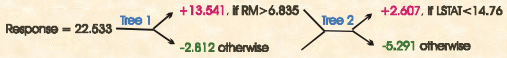
© Copyright Salford Systems 2001-2002

Stumps Produces Additive Model

Jointly, 4 different regions are created:

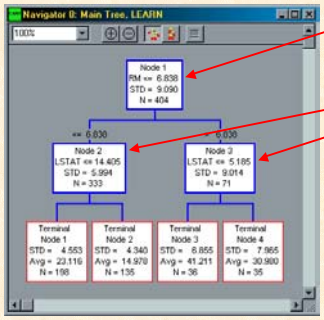
MV	#OBS	
14.43	163	Small houses, bad neighborhood
22.33	256	Small houses, good neighborhood
30.81	4	Large houses, bad neighborhood
38.68	83	Large houses, good neighborhood

This is an additive model:



© Copyright Salford Systems 2001-2002

CART Run with 4 Nodes



The first split is the same as TreeNet

But these two splits are different => the model is no longer additive, RM and LSTAT interact

Conclusion: CART model builds interactions

© Copyright Salford Systems 2001-2002

CART Run with 4 Nodes

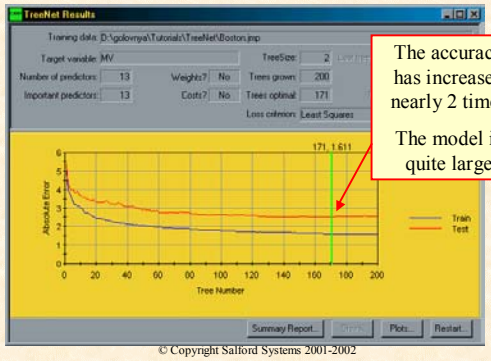
Again, 4 different regions are created:

MV	#OBS	
14.98	174	Small houses, bad neighborhood
23.12	245	Small houses, good neighborhood
30.98	41	Large houses, bad neighborhood
41.21	46	Large houses, good neighborhood

- Similar conclusions, but model is no longer additive
- Completely different counts but the sums within the first and the second pairs are the same as before

© Copyright Salford Systems 2001-2002

Using All 13 Available Predictors

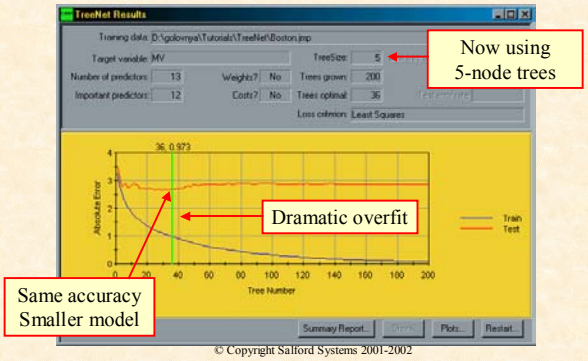


The accuracy has increased nearly 2 times

The model is quite large

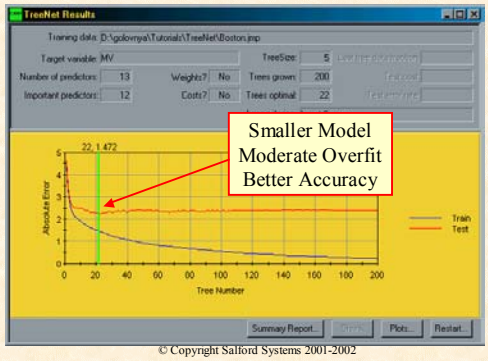
© Copyright Salford Systems 2001-2002

Increase the Base Tree Size



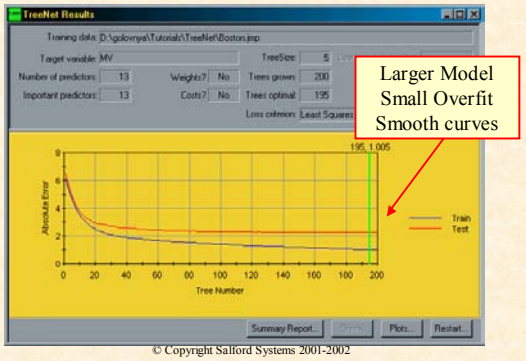
© Copyright Salford Systems 2001-2002

Reduce the Learning Rate to .5



© Copyright Salford Systems 2001-2002

Reduce the Learning Rate to .1



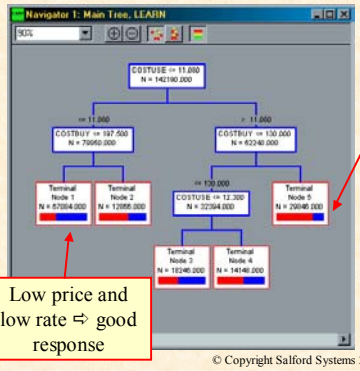
© Copyright Salford Systems 2001-2002

Classification Example

- CELL Phone Data
- RESPONSE: YES/NO to subscribe
  - 126 YES
  - 704 NO
- PREDICTORS:
  - COSTBUY: cost of the hand set (4 levels)
  - COSTUSE: monthly charges (4 levels)
- WEIGHT variable is added to account for non-even distribution of responders and non-responders

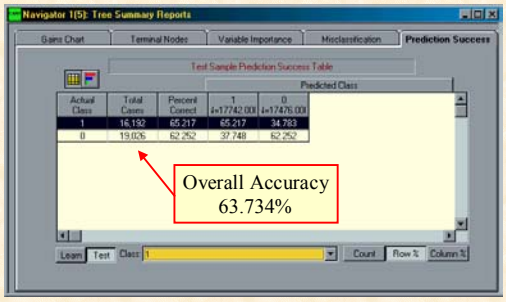
© Copyright Salford Systems 2001-2002

A Single CART Run



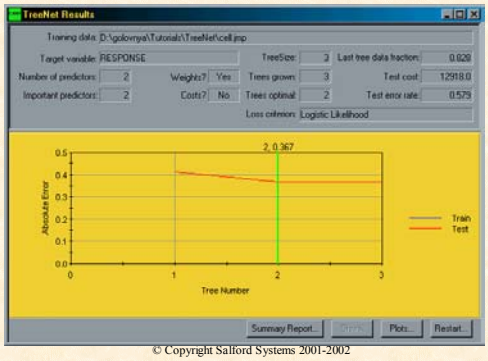
© Copyright Salford Systems 2001-2002

Prediction Success



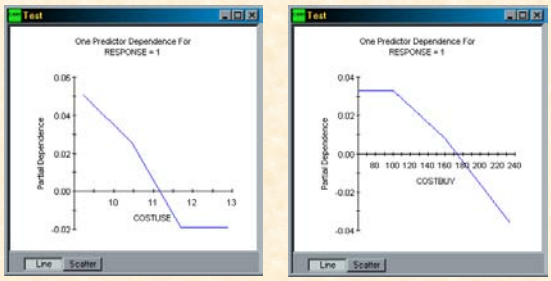
© Copyright Salford Systems 2001-2002

A Simple TreeNet Classification Model



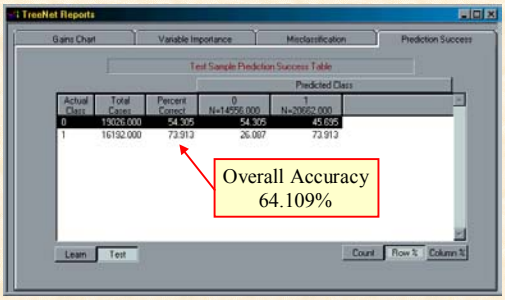
© Copyright Salford Systems 2001-2002

Individual Contributions



© Copyright Salford Systems 2001-2002

Prediction Success



© Copyright Salford Systems 2001-2002

Now some live TreeNet runs

- Official version available May, 2002 from Salford Systems
- Send e-mail to request copy to

support@salford-systems.com

© Copyright Salford Systems 2001-2002

References

- Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24, 123-140.
- Breiman, L. (1996). Arcing classifiers (Technical Report). Berkeley: Statistics Department, University of California.
- Buntine, W. (1991). Learning classification trees. In D.J. Hand, ed., *Artificial Intelligence Frontiers in Statistics,* Chapman and Hall: London, 182-201.
- Dietterich, T. (1998). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, Boosting, and Randomization. *Machine Learning*, 40, 139-158.
- Freund, Y. & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In L. Saitta, ed., *Machine Learning: Proceedings of the Thirteenth National Conference*, Morgan Kaufmann, pp. 148-156.
- Friedman, J.H. (1999). Stochastic gradient boosting. Stanford: Statistics Department, Stanford University.
- Friedman, J.H. (1999). Greedy function approximation: a gradient boosting machine. Stanford: Statistics Department, Stanford University.
- Heath, D., Kasif, S., and Salzberg, S. (1993) k-dt: A multi-tree learning method. *Proceedings of the Second International Workshop on Multistrategy Learning*, 1002-1007, Morgan Kaufman: Chambery, France.
- Kwok, S., and Carter, C. (1990). Multiple decision trees. In Shachter, R., Levitt, T., Kanal, L., and Lemmer, J., eds. *Uncertainty in Artificial Intelligence 4*, North-Holland, 327-335.

© Copyright Salford Systems 2001-2002

Case study: Modelling Risk in Health Insurance - A Data Mining Approach.

Inna Kolyshkina
PricewaterhouseCoopers
201 Sussex Street
SYDNEY NSW 2000
inna.kolyshkina@au.pwcglobal.com

Richard Brookes
PricewaterhouseCoopers
201 Sussex Street
SYDNEY NSW 2000
richard.brookes@au.pwcglobla.com

ABSTRACT

Interest in data mining techniques has been increasing recently amongst the actuaries and statisticians involved in the analysis of insurance data sets which typically have a large number of both cases and variables. This paper discusses the main reasons for the increasing attractiveness of using data mining techniques in insurance. A case study is presented showing the application of data mining to a business problem that required modeling risk in health insurance, based on a project recently performed for a large Australian health insurance company by PricewaterhouseCoopers (Sydney). The data mining methods discussed in the case study include: Classification and Regression Trees (CART), Multivariate Adaptive Regression splines (MARS) and hybrid models that combined CART tree models with MARS and logistic regression. The non-commercially sensitive implementation issues are also discussed.

Keywords

Data analysis in insurance, data mining, Classification and Regression Trees (CART), Multivariate Adaptive Regression splines (MARS), hybrid models.

1. INTRODUCTION

In insurance, like in many other industries (health, telecommunication, banking to name a few) the size of databases today often reaches terabytes. In a dataset like this, with millions of cases and hundreds of variables, finding important information in a dataset is like finding the proverbial need in the haystack. However the need for extraction of such information is very real, and data mining is definitely a technique that can meet that need.

Various data mining methodologies have been used in insurance for risk prediction/assessment, premium setting, fraud detection, health costs prediction, treatment management optimization, investments management optimization, customer retention and acquisition strategies. In fact, recently a number of publications have examined the use of data mining method in insurance and actuarial environment (eg, Francis, 2001, WorkCover NSW News, 2001). The main reasons for the increasing attractiveness of the data mining approach is that it is very fast computationally and also overcomes some well-known shortcomings of traditional methods such as generalised linear models that are often being used for data analysis in insurance. This paper gives an example of the application of data mining methodologies to modelling risk in insurance based on a recent project completed by PwC Actuarial (Sydney) data mining team for a large insurance company client.

2. DATA MINING VERSUS LINEAR METHODS. MODELLING METHODOLOGIES USED: CART DECISION TREES, MARS AND HYBRID MODELS.

The main reasons for the increasing popularity of data mining methods amongst the actuarial community can be briefly summarised as follows. Data mining relies on the intense use of computing power, which results in an exhaustive search for the important patterns, uncovering hidden structure even in large and complex data sets and in many cases a well-performing model. Also, unlike the more traditional linear methods, it does not assume that the response is distributed according to some specified distribution (which is often incorrect for real-life insurance data sets). In contrast, traditional methods take longer to develop models, and have particular trouble selecting important predictors and their interactions. Another very attractive feature, involved in many data mining modeling methodologies is automatic "self-testing" of the model. A model is first built on a randomly-selected portion of the data and then tested and further refined on the remaining data. Finally, most data mining methods allow the inclusion in the model categorical predictors with a large number of categories which are typically present in the insurance data sets (for example, postcode, injury code, occupation code etc). Classical methods cannot deal with such variables effectively, and, as a result, they are either left out of the model, or have to be grouped by hand prior to inclusion.

Each data mining technique has its advantages as well as its drawbacks. These outside of the scope of this paper, but are discussed in detail in the literature (for example, Vapnik (1996) and Hastie *et al*. (2001)). We were very aware of the importance of selecting the method of analysis that is best suited for a particular problem, and, after an extended study of the available data mining techniques, we selected tree-based models and their hybrids for everyday modeling of insurance data. The reasons for such selection are as follows. Tree-based methods are very fast, require less data preparation than some other techniques, can more easily handle missing values or noisy data, are unaffected by outliers, and are easy to interpret.

A useful feature of the software packages we used (CART® and MARS®) is that they are easy to implement in SAS which is the main data analysis software package used by us as well as by the majority of our clients.

We provide below brief introductions to the techniques we used, only complete enough for appreciating the outline of the modelling process we describe. A more detailed description of

them can be found in the literature as indicated in the individual sections.

2.1 Classification and Regression Trees (CART®)

The CART methodology is known as binary recursive partitioning (Breiman et al, 1984). It is binary because the process of modelling involves dividing the data set into exactly two subgroups (or "nodes") that are more homogeneous with respect to the response variable than the initial data set. It is recursive because the process is repeated for each of the resulting nodes. The resulting model is usually represented visually as a tree diagram. It divides all data into a set of several non-overlapping subgroups or nodes so that the estimate of the response is "close" to the actual value of the response within each node (Lewis et al, 1993). CART then ranks all the variables in the order of importance, so that a relatively small number of predictors get a non-zero importance score. This means that it quickly selects the most important predictors out of many possible ones. The model is quickly built, is robust and easily interpretable. However, as any decision tree, it is coarse in the sense that it predicts only a relatively small number of values and all cases within each node have the same predicted value. It also lacks smoothness: a small change in a dependent variable can lead to a large change in the predicted value. Another disadvantage of CART is that it is not particularly effective in modelling the linear structure, and would build a large model to represent a simple relationship. Further details and discussion of decision trees and CART® can be found in literature (Breiman *et al*, 1984 ; Hastie *et al*, 2001).

2.2 Multivariate adaptive regression splines (MARS)

MARS is an adaptive procedure for regression, and can be viewed as a generalisation of stepwise linear regression or a generalization of the recursive partitioning method to improve the latter's performance in the regression setting (Friedman, 1991; Hastie *et al*, 2001). The central idea in MARS is to formulate a modified recursive partitioning model as an additive model of functions from overlapping, (instead of disjoint as in recursive partitioning), subregions (Lewis *et al*, 1993).

The MARS procedure builds flexible regression models by fitting separate splines (or basis functions) to distinct intervals of the predictor variables. Both the variables to use and the end points of the intervals for each variable-referred to as knots-are found via an exhaustive search procedure, using very fast update algorithms and efficient program coding. Variables, knots and interactions are optimized simultaneously by evaluating a "loss of fit" (LOF) criterion. MARS chooses the LOF that most improves the model at each step. In addition to searching variables one by one, MARS also searches for interactions between variables, allowing any degree of interaction to be considered. The "optimal" MARS model is selected in a two-phase process. In the first phase, a model is grown by adding basis functions (new main effects, knots, or interactions) until an overly large model is found. In the second phase, basis functions are deleted in order of least contribution to the model until an optimal balance of bias and variance is found. By allowing for any arbitrary shape for the response function as well as for interactions, and by using the two-phase model selection method, MARS is capable of reliably tracking very complex data structures that often hide in high-dimensional data (Salford Systems, 2002).

2.3 Hybrid Models

The strengths of decision trees and "smooth" modeling techniques can be effectively combined. Steinberg and Cardell (1998a, 1998b) describe the methodology of such combining where the output of CART model in the form of terminal node indicator, of the predicted values or of the complete set of indicator dummies is included among other inputs in the "smooth" model. The resulting model is continuous and gives a unique predicted value for every record in the data. Typically, all strong effects are detected by the tree, and the "smooth" technique picks up the additional weak, in particular linear, effects. Combined, these small effects can very significantly improve the model performance (Steinberg and Cardell 1998a, 1998b).

3. HEALTH INSURER CASE STUDY

3.1 Background

The methodology described above was successfully applied in a recent project completed for a major health insurance company client. It was used for creating the model of overall projected lifetime customer value. The model took into account many aspects influencing customer value such as premium income, reinsurance, changes in the family situation of a customer (births, marriages, deaths and divorce), probability of a membership lapse and transitions from one type of product to another. Each of these aspects as well as hospital claim frequency and cost for the next year and ancillary claim frequency and cost for the next year was modelled separately and the resulting models were combined into a complex customer lifetime value model. In this article we will discuss one of the sub-models, namely the model for hospital claim cost for the next year.

3.2 Data

3.2.1 Data description

De-identified data was available at a member level over a 3 year period. The model used information available over the first 2 years to fit a model based on outcomes over the last year. We excluded from the data those customers who lapsed prior to the end of the 3 year period or joined the health insurer later than 3 years ago. This latter exclusion allowed us to avoid issues related to waiting periods and enabled us to use two years of data history in the modelling.

The data used for analysis can be grouped as follows. Firstly, there were demographic variables, such as age of the customer, gender, family status (about 30 variables). The second variable group was geographic and socio-economic variables such as location of the member's residence and socio-economic indices related to the geographic area of the member's residence such as indices of education, occupation, relative socio-economic advantage and disadvantage (about 80 variables). The third group of variables was related to membership and product details such as duration of the membership, details of the hospital and ancillary product held at present as well as in the past (about 30 variables). The fourth group of variables was related to claim history (both ancillary and hospital), details of medical diagnosis of the member, number of hospital episodes and other services provided to the member in previous years, number of claims in a particular calendar year etc (about 100 variables). The fifth and last group of variables included such information as distribution channel, most common transaction channel, payment method etc (about 50 variables). Overall there were about 300 variables.

3.2.2 Data preparation, cleaning and enrichment.

The data underwent a rigorous checking and cleaning process. This was performed in close cooperation with the client. Any significant data issues or inconsistencies found were discussed with them. Among other things such as statistical summaries, distribution analysis etc, the checking process involved exploratory analysis using CART which was applied to identify any aberrant or unusual data groups.

Some of the variables in the original client data set were not directly used in the analysis. For example instead of the date of joining, we used the derived predictor "duration of membership". In other cases, if a predictor was described by the client as likely to contain unreliable or incorrect information, it was excluded from the analysis.

A number of variables included in analysis were derived by us with the purpose of better describing the customer behaviour. Examples are duration of membership and indicator of whether or not the member had a hospital claim in previous years. Many of such predictors, for example, indicators of whether the member stayed in hospital for longer than one day and whether or not the services received were of surgical or non-surgical nature, were created after consultation with clinical experts.

Other variables were added to the data from various sources such as Australian Bureau of Statistics. These included a number of various socioeconomic indices based on the member's residence, some related to broader geographic areas such as state, others more closely targeting member's location such as postcode-based indicators.

3.3 Modelling Methodology

First, we built a CART tree model. This served purposes of exploration, getting appreciation of the data structure and selection of the most important predictors and provided easily interpretable diagram. The client found CART diagrams easy to understand and informative. To further refine the model, we then built a hybrid model of CART and MARS using the hybrid modelling methodology (Steinberg & Cardell, 1998a, Steinberg & Cardell, 1998b). This was achieved by including CART output in the form of a categorical variable that assigned each record to one of the nodes according to the tree model as one of the input variables into a MARS model. MARS, like CART

In some cases where we wanted to achieve an even higher degree of precision we built a "three-way hybrid model" combining CART®, MARS® and a linear model such as logistic regression or generalized linear model. This was done by feeding MARS® output (in the form of basis functions created by MARS®) as inputs into a linear model.

3.4 Model diagnostic and evaluation

The main tools we used for model diagnostics were gains chart and analysis of actual versus predicted values for hospital cost.

The gains chart for the overall hospital claims cost model presented in Figure 1 shows that we are able to predict the high cost claimants with a good degree of accuracy. As a rough guide, the overall claim frequency is 15%. Taking the 15% of members predicted as having the highest cost by the model, we end up with 56% of the total actual cost. Taking the top 30% of members predicted as having the highest cost by the model, we end up with almost 80% of the total actual cost.

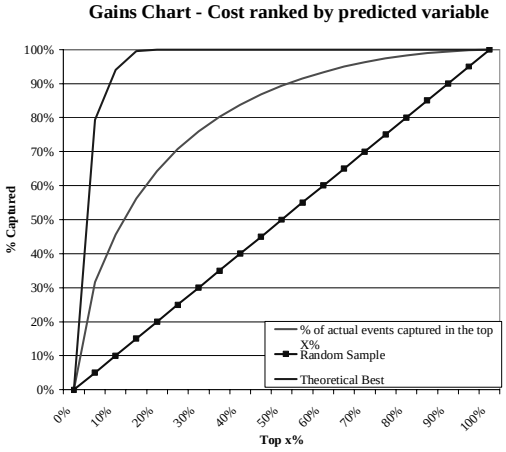


Figure 1 Gains chart for total expected hospital claims cost

A further diagnostic of model performance is analysis of actual versus expected values of probability of claim or claim cost. Such analysis can be pictorially represented by a bar chart of averaged actual and predicted values for overall annual hospital cost. This chart is shown in Figure 2. To create this chart, the members were ranked from highest to lowest in terms of predicted cost, and then segmented into 20 equally sized groups. The average predicted and actual values of hospital cost for each group were then calculated and graphed.

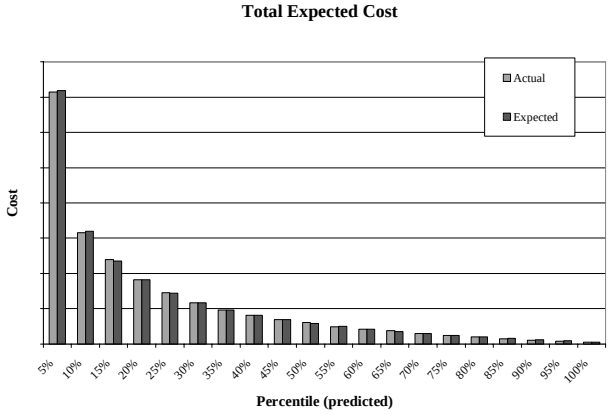


Figure 2 The bar chart of averaged actual and predicted values for overall annual hospital cost

The chart suggests that the model fits well, however slightly over-predicts for the lower expected costs but this was of little business importance for the client.

3.5 Findings and Results

3.5.1 Model Precision

The model achieved high degree of precision as is demonstrated at the actual versus predicted graph (Figure 2) and gains chart (Figure 1) above.

3.5.2 Predictor importance for hospital claims cost

Predictors of the highest importance for overall hospital cost were age of the member, gender, number of hospital episodes and hospital days in the previous years, the type of cover and socio-economic characteristics of the member.

Other important predictors included duration of membership, family status of the member, the type of cover that the member

had in the previous year, previous medical history and the number of physiotherapy services received by the member in the previous year. The fact that the number of ancillary services (physiotherapy) affected hospital claims cost was a particularly interesting finding.

Details of the resulting model are commercially sensitive. However, we can state that many of the potential predictors given above were indeed significant to a degree greater than we had expected. For example, while some health insurance specialists argue that the only main risk driver for hospital claim cost is age of the member, our results have demonstrated clearly that although age is among important predictors of hospital claims cost, a large amount of variation is not explained by age alone. One way of showing this is by means of a graph of predicted cost by age shown in Figure 3. If age were the most important predictor with other predictors not adding much value, the graph would show values scattered closely to a single curve. The fact that it is scattered so widely, shows that there are many other factors contributing significantly to predicted cost. Examples of such factors are socioeconomic indicators, type of hospital product and, for some age groups, the supply of hospitals in the location of the member's residence.

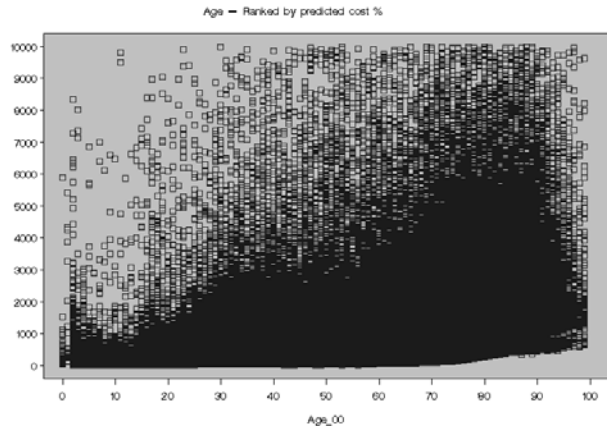


Figure 3. The graph of predicted hospital cost versus age.

We also build models for ancillary claims of various types, including optical, dental and physiotherapy claims. Unsurprisingly, the most important predictor of ancillary claims is the customer's previous claiming pattern. However, there are strong age related effects (for instance the teenage orthodontic peak for dental claims), socio-economic effects and location effects.

3.6 Implementation issues.

The deliverables for the model included a SAS algorithm which takes the required input data and produces a cost score for a given customer so the client could easily implement the model directly in SAS environment.

4. CONCLUSION

The results described above as well as a number of projects completed by PwC Actuarial (Sydney) for large insurer clients demonstrate that data mining methodologies can be very useful for analysis of the insurance data.

5. ACKNOWLEDGEMENTS

We would like to thank Mr John Walsh (PricewaterhouseCoopers Actuarial, Sydney) for support, advice and thoughtful comments on the analysis.

6. REFERENCES

[1] Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984). Classification and Regression Trees. Wadsworth, Pacific Grove, CA.

[2] Francis, L. (2001). Neural networks demystified. *Casualty Actuarial Society Forum*, Winter 2001, 252–319.

[3] Haberman, S. and Renshaw, A. E. (1998). Actuarial applications of generalized linear models. In Hand, D. J. and Jacka, S. D. (eds). *Statistics in Finance*. Arnold, London.

[4] Han , J., and Camber M. (2001) *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers.

[5] Hastie, T., Tibshirani R. and Friedman, J. (2001). *The elements of statistical learning: Data Mining, Inference and prediction.* Springer-Verlag, New York.

[6] Lewis, P.A.W. and Stevens, J.G., "Nonlinear Modeling of Time Series using Multivariate Adaptive Regression Splines," *Journal of the American Statistical Association*, **86**, No. 416, 1991, pp. 864-867.

[7] Lewis, P.A.W., Stevens, J., and Ray, B.K., "Modelling Time Series using Multivariate Adaptive Regression Splines (MARS)," in *Time Series Prediction: Forecasting the Future and Understanding the Past*, eds. Weigend, A. and Gershenfeld, N., Santa Fe Institute: Addison-Wesley, 1993, pp. 297-318.

[8] McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models* (2nd edition). Chapman and Hall, London.

[9] Salford Systems (2000). *CART® for Windows User's Guide*. Salford Systems

[10] Salford Systems (2002). MARS® (Multivariate Adaptive Regression Splines) [On-line] http://www.salford-systems.com, (accessed 08/10/2002).

[11] Smyth, G. (2002). Generalised linear modelling. [On-line] http://www.statsci.org/glm/index.html, (accessed 25/09/2002).

[12] Steinberg, D. and Cardell, N. S. (1998a). Improving data mining with new hybrid methods. Presented at *DCI Database and Client Server World*, Boston, MA.

[13] Steinberg, D. and Cardell, N. S. (1998b). The hybrid CART-Logit model in classification and data mining. *Eighth Annual Advanced Research Techniques Forum*, American Marketing Association, Keystone, CO.

[14] Steinberg, D. and Colla, P. L., (1995). *CART: Tree-Structured Nonparametric Data Analysis*. Salford Systems, San Diego, CA.

[15] Vapnik, V. (1996). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

[16] WorkCover NSW News (2001) Technology catches insurance fraud.[On-line] http://www.workcover.nsw.gov.au/pdf/wca46.pdf (accessed 08/10/02)

Investigative Profile Analysis with Computer Forensic Log Data using Attribute Generalisation

Tamas Abraham
Information Networks Division
Defence Science and Technology Organisation
PO Box 1500, Edinburgh SA 5111, Australia
tamas.abraham@dsto.defence.gov.au

Ryan Kling
Information Networks Division
Defence Science and Technology Organisation
PO Box 1500, Edinburgh SA 5111, Australia
ryan.kling@dsto.defence.gov.au

Olivier de Vel
Information Networks Division
Defence Science and Technology Organisation
PO Box 1500, Edinburgh SA 5111, Australia
olivier.devel@dsto.defence.gov.au

ABSTRACT

Investigative profiling is an important activity in computer forensics that can narrow the search for one or more computer perpetrators. Data mining is a technique that has produced good results in providing insight into large volumes of data. This paper describes the use of a well-known data mining technique, attribute-oriented induction, together with newly designed profile analysis methodology, for the purpose of identifying irregularities in computer logs. The process relies on background knowledge in the form of concept hierarchies, and uses a distance measure to estimate the level of contrast between records generalised from formatted computer log data. Results obtained have shown the process to perform according to expectations.

1. INTRODUCTION

Computer Forensics undertakes the *post-mortem*, or "after-the-event" analysis of computer crime. Of particular importance is the requirement to successfully narrow the potentially large search space often presented to investigators of such crimes. This usually involves some form(s) of guided processing of the data collected as evidence in order to produce a shortlist of suspicious activities. Investigators can subsequently use this shortlist to examine related evidence in more detail [5].

Investigative profiling is an important activity in computer forensics that can significantly narrow the search for the perpetrator and reason about the perpetrator's behaviour. This is analogous to criminal profiling which focuses on establishing personality characteristics of an offender in order to identify the type of person involved in the crime under investigation (e.g., arson). Profiling can also aid in identifying the type of activity the perpetrator is engaged in e.g., e-mail authorship analysis may identify the educational level or gender of the offender and may, consequently, be able to establish if an e-mail has been masqueraded [7].

Data Mining is employed to analyse large data sets, as might occur in a typical computer forensics investigation, in order to discover potentially useful, previously unknown regularities within data. In contrast to other, more conventional technologies, it has been able to produce good results on large data sets where both incompleteness and noise may be present [8].

Data mining for the more specific purpose of constructing personal profiles has been used in the context of customer personalisation. Here, marketing content and services are tailored to an individual on the basis of knowledge about their preferences and behaviour. Applications include content-based and collaborative filtering-based recommendation systems, customer profiling [3; 2; 14], fraud detection [9], web browsing activities [6; 18; 23; 17]. Content-based recommendation systems model the link between data content and a person's preferences for that content whereas collaborative recommendation systems model the link between a person's preferences and other persons' preferences for the given data content [15; 19]. Customer profiling is growing in importance in e-commerce. Both factual and individual behavioural information are derived from the customer's e-transactional history. The personalisation of web browsing activities for the purpose of improving the user's access to the web has also attracted interest recently. Techniques for the modeling of the user's web access behaviour are varied including; the use of a page content interestingness metric (N-grams) for capturing a user's interests [6], web page navigation dependencies for page predictive prefetching [18], web page clustering for deriving aggregate user profiles [17], sequential web page patterns for discovering negatively-correlated components within a web site structure [23]. Although not all, many of these web personalisation applications deal with aggregate or classes of user profiles rather than individual user profiles.

In this paper, we describe techniques to profile and analyse computer forensic data. We use a combination of existing techniques not yet employed in this application domain, modified where necessary to accommodate the particular environment. An earlier attempt by the same authors using the association mining paradigm instead of generalisation is documented in [1]. This paper shares the same motivation and methodology but uses a different set of tools to achieve similar goals, while also adding some new ideas to the previous presentation. In Section 2, we introduce the elements used in our approach to computer forensic profiling. Section 3 describes the algorithms we use and how they have been adopted for our needs. Tests performed on actual computer log data are detailed in Section 4, followed by our

conclusions in Section 5.

2. BACKGROUND TO INVESTIGATIVE PROFILING

An offender profile consists of two components, namely the factual component and behavioural component. The factual profile (**FP**) consists of factual background knowledge about the offender such as their name, employee status, computer user name(s), relationships with other employees and organisations etc. The behavioural profile (**BP**) incorporates knowledge about an offender's crime scene-related behaviour. Behaviour profile knowledge is derived from a variety of sources namely, log file transactions, header and body of e-mails, telecommunications call-record data patterns and so on.

The behavioural profile, **BP**, can be modeled in different ways. For example, a *BP* can be represented as a union of sub-profile hierarchies (PH_j) such as, authorship profile, software application usage profile, log-in profile etc. That is,

$$\mathbf{BP} \leftarrow \bigcup_j^M PH_j$$

A profile hierarchy is a knowledge representation scheme using a hierarchy of multi-slot frames, similar to a concept hierarchy (described in Section 2.2), that characterises a behavioural profile. An example profile hierarchy that describes the hierarchy of application type usage is shown in Figure 1.

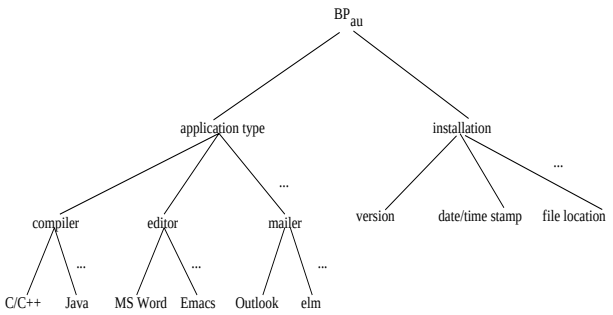


Figure 1: An example profile hierarchy representation, the application type usage PH_{au}.

Alternatively, **BP** can also be modeled as a set of rules:

$$\mathbf{BP} \leftarrow \{R_i | i = 1, 2, \ldots N\}$$

Here, the rule attributes can be obtained from the raw data and/or selected from the profile hierarchy nodes. For example, the rule "If user X is a system administrator, then the application $Y = nmap$ (a stealth port scanner) executed" may be a valid rule in a system administrator profile (assuming that port scanning, as used in his/her current job context, is employed for system hardening), but probably not in a finance contractor profile.

In this paper we study user behavioural profiles derived from event data in log files. These profiles are conveniently represented by a set of conjunctions ("rules") $\{R_i | i = 1, 2, \ldots N\}$ of the form

$$R_i : A_1 \wedge A_2 \wedge \ldots \wedge A_n$$

These rules provide an intuitive and declarative way to describe user behaviour [9]. For example, the rule

$$\begin{aligned} R_0 : \quad & (\mathbf{StaffType} = admin) \wedge \\ & (\mathbf{DayOfWeek} = tuesday) \wedge \\ & (\mathbf{Application} = database) \wedge (\mathbf{Access} = valid) \end{aligned}$$

states that "Administration staff that work on Tuesday have a valid access to a database application". A more complex rule might be

$$\begin{aligned} R_1 : \quad & (\mathbf{Time} = morning) \wedge \\ & (\mathbf{Source} = flyByNightCorp) \wedge \\ & (\mathbf{DayOfWeek} = tuesday) \wedge \\ & (\mathbf{Duration} = lessThanMinute) \wedge \\ & (\mathbf{StaffType} = management) \end{aligned}$$

stating that "A person logging into the computer on a Tuesday morning from a company with a domain name $flyByNightCorp$ is likely to be a person with a management-type profile that logs-in only briefly (less than one minute)".

2.1 Profiling with Generalisation

One of the potential uses for data mining is the building of rule sets that describe behavioural data [3; 2; 4]. This may be data collected about people or the operation of some systems. Often in computer forensic investigations, this information could be found in log files on a computer system. The rules generated from this data can be considered to describe a profile contained within the data set.

Attribute-oriented induction [10] is a technique that replaces low-level attribute data in a database relation with higher level abstractions using pre-defined concept hierarchies. Via this process, the original relation gets reduced to a usually much smaller collection of generalised records that are higher level statements about the data. Each generalised record will have a count value assigned to it depending on how many of the original records contributed to the particular record. The final generalised relation will have a variety of records with different counts, some of which may be higher than others. We hypothesise that records with sufficiently high counts will form a profile, or general description of the data, whilst low count records should be treated as outliers, with the potential to describe evidence contrary to expected behaviour. For example, if an attacker gains super-user privileges on a computer system, he may use them to perform actions not normally instigated by the real super-user(s). This would clearly be a deviation from the super-user profile as supported by data up to the time of intrusion. Both kinds of behaviour will be present in the data, and depending on how (in)frequent the anomalous behaviour is, it can be detected either as an outlier or another profile element contradicting normal behaviour. This observation immediately defines the need for two analogous, but separate comparison processes:

- Intra-profile analysis to compare profile elements to locate contradictions or alternatives
- Profile-to-outlier comparisons to identify contradictions between profile elements and outliers

These particular tasks will be discussed in more detail later. Another important aspect of forensic profiling is that a user profile is generated using available evidence and does not

change once produced. Additional evidence may be added later, but this should be regarded as the incompleteness of initial evidence rather than the evolution of an existing profile. In this case, the profile could be re-generated.

2.2 Concept Hierarchies

Background knowledge in data mining is popularly expressed in the form of concept hierarchies and is often used in rule generation processes [21; 11]. The hierarchies convey a generalisation of concepts from node to root (which is usually the concept any) and can be, for example, represented as a set of parent-child relationships in a data file. An investigator may be prompted with a set of node level concepts found in the data and asked to abstract it to higher level ones according to his/her liking. This hierarchy, which is generally domain-dependent in forensics investigations, can then be used in the mining process to produce a profile that contains rules with elements at an arbitrary level of abstraction. A set of hierarchy fragments is often more desirable as it helps to avoid over-generalisation by not including very high level concepts in the search process.

Concept hierarchies in attribute-oriented induction are a vital pre-requisite. Their structure, contents and completeness directly affect the quality of results produced by the algorithm. In a computer forensic environment, it is also a necessity to employ background knowledge that is as complete and accurate as possible. It is therefore not unusual to request that such hierarchies be available prior to induction. Some of these hierarchies can be supplied by experienced investigators. Others can be automatically generated based on general knowledge about attribute values (eg employment positions). It is also important to add as many of the attribute values into a concept hierarchy as possible. Although the algorithm should, and can, handle missing values in hierarchies, it will not be able to generalise them. An automated facility to add missing values at an appropriate (eg average leaf depth) level could be provided as part of the investigative tool set.

Evolution within a profile is also an important indicator of potentially irregular activities. A profiling algorithm is expected to be able to attach temporal tags to rules indicating intervals of validity if so required. Concept hierarchies, therefore, must accommodate such functionality. This usually happens at two levels – changes in the structure of concept hierarchy over time, and changes in the position of a leaf node value over time. A modified mining algorithm can then produce profile elements valid over given temporal intervals, which may reduce the complexity of post-induction analysis (fewer elements to compare over specific intervals).

3. THE PROFILING PROCESS

The data obtained for computer forensic investigations are usually information stored on computers and networks. They range from system log files to databases, personal user files and other items that may be located on a computer. To build a profile for a particular user, many of these items may need to be examined both individually and as a collection of interrelated items with potential relationships existing between recorded activities. Profiling algorithms are therefore expected to be of varying complexity. A simple algorithm may produce rules based on the sporadic occurrences of data observed in a single file, another may be required to recognise temporal dependencies or causal relationships in user activity recorded across several files.

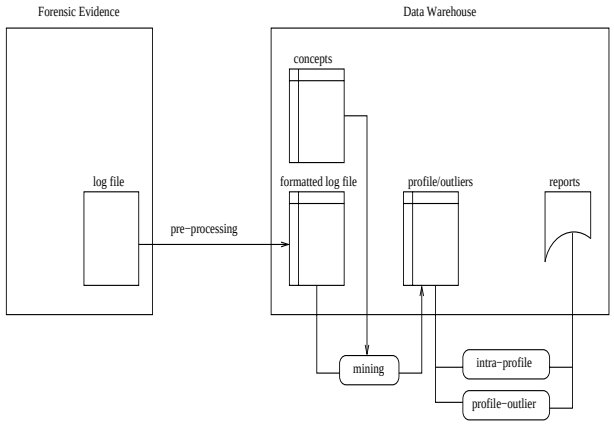


Figure 2: Data flow diagram of the profiling process.

Since computer forensics undertakes the post-mortem analysis of computer crime, much of the analysis is done off-line. Therefore, emphasis is more often on effectiveness than efficiency in order to produce a smaller set of targeted conclusions, and reduce overall human investigation time. For example, it is preferable to achieve a low rate of false negatives at the expense of increased computational time and number of false positives.

Much of the information found on a computer is expected to be in a format not suited for immediate analysis. An investigator must facilitate this by providing details on the subsets of data intended for analysis, their format and conversion requirements, and available background knowledge. Some of this can be achieved through automated means. Filtering, the removal of unwanted information and the aggregation of separate data items are some of the more important activities during this stage of the analytical process.

The data flow diagram in Figure 2 describes the data, rules and processes used for profiling purposes. It incorporates references to background knowledge, and the final conclusions resulting from the forensic analysis (detailed in Section 3.2).

3.1 Induction Algorithm

The attribute-oriented induction algorithm implemented for forensic analysis is designed to generate a profile using a single input file. The usual steps of *data filtering*, *data conversion*, and the creation of *concept hierarchies* precedes the execution of profiling.

Different hierarchies may be used to attain different generalised data. A broad, shallow tree will allow rapid data generalisation, whereas a narrow, deep tree will lead to a slower, more directed approach. The use of both narrow and broad trees in the hierarchies of different fields could be used to rapidly ascend less influential attributes, whilst maintaining as much control on those attributes under scrutiny.

It should be noted that in our implementation, the root concept is not present in the database. Complete ascension of a hierarchy to the root is pointless since removal of that attribute from consideration achieves the same end. As such, the first set of branches from the true root concept can be

considered the roots of sub-trees. Likewise any concept that has no parent can also be assumed to be a single generalisation step away from the root or "ANY" value.

The concept hierarchy should be complete and exclusive to the data present in the data relation. Any data not in the hierarchy will not be ascended. Since the process designed here relies on the hierarchy being correct and complete, an incomplete hierarchy may affect the effectiveness of threshold values. Ideally the hierarchy should not contain additional information as this may lead to under-generalisation.

The attribute-oriented induction approach in this paper, as earlier mentioned, is in effect an extended implementation of the algorithm by [10]. It primarily performs concept hierarchy ascension for each attribute in a database relation. Consolidation, re-evaluation and consequent action upon the data are also involved. There are a number of different mechanisms in place to control the ascensions as described below.

3.1.1 Parameters

The ascension method presented here accepts a number of parameters detailing the attribute priorities, threshold control values, the source of the data and concept hierarchy and which attributes are to be ignored completely.

- *Attribute Priorities* The attribute priorities define the order in which the attributes are ascended. Each attribute is ascended one level at a time. An attribute with a priority of less than or equal to zero will not be ascended, but will retain the distinct values in the database (cf. ignoring a attribute, see *Data Selection* below). Each attribute with a priority greater than zero is ascended either to a threshold value, to the top of its concept hierarchy or is removed from consideration prior to ascension. Ascending each attribute independently of the others prevents over-generalisation and ensures that the smallest possible change is available. All attributes with a priority greater than zero are ascended, halted due to threshold values (allowing no further ascension) or removed from consideration before any attribute is subject to further, subsequent ascensions. This allows for simple, automated ascension control that prevents over-generalisation in some attributes and under-generalisation in others. The approach presented can easily be adapted for user-controlled ascension in order to rapidly traverse one hierarchy and maintain a steady ascension of another. However, as discussed in Section 2.2, this control may be simulated using different concept hierarchies.

- *Thresholds* Many different thresholds are available to control the ascension of the hierarchy by setting threshold limits on different values. Generalisations are considered complete when either the general record threshold is reached or each attribute to be generalised has reached a threshold. A list of the attributes that have reached an attribute specific threshold (including those attributes that have been ignored) is maintained to track which attributes still require generalisation.

  The *record threshold* is a defined number of records that is used to determine if further generalisation is required. It is based solely on the number of records in the generalised relation and is not specific to any attribute or depth. If the threshold exceeds the total number of records in the generalised data, no more generalisations are made. This threshold can be represented as a value (a distinct number of records) and/or as a percentage of the original number of records.

  Attribute thresholds are used to monitor the specific ascent of each individual concept hierarchy. There is an attribute threshold for each attribute in the relation and they are independent of one another. In a similar manner to the record threshold, if the threshold for a particular attribute exceeds the number of distinct values for that attribute, then the corresponding hierarchy is not ascended. Once a attribute has reached its threshold, further generalisation of that attribute is prohibited, however the other attributes remain unaffected. This threshold can be defined as a discrete number of distinct values and/or a percentage of the number of distinct values for that attribute found in the initial data. The number of values at a given depth is the number of possible values at that depth. This value is calculated based on the structure of the concept hierarchy. For this reason, the hierarchy should be as correct and complete as possible.

  Depth thresholds are also available for each individual hierarchy. This control value simply limits how far the hierarchy may ascend. A depth of zero is considered to be the root concept; a depth of one includes the first set of branches; a depth of two is the next set of branches and so on. Section *3.1.2* describes how the ascension of unbalanced trees is handled.

  The *vote threshold* is another general threshold that applies to the complete generalised relation. If the vote attribute of a generalised record exceeds the threshold, a note of the record is made and the ascension is continued. As such, this threshold does not affect the generalisation process, however the final data presented by the algorithm is influenced. The vote attribute and propagation of vote values after generalisation are described in greater detail in Section *3.1.3*.

- *Data selection* The source of the data to be mined and the concept hierarchies are provided by a relational database. Concepts are linked by means of a parent-child relationship. Not all attributes in the data relation are required in the data-mining process. Attributes may be marked for removal prior to any generalisations. The values for these attributes are converted to empty values, the data is merged and the votes are propagated. If the original values of the attribute are required, an empty concept hierarchy, or a priority of zero may be used to prevent generalisation (see *Attribute Priorities*).

3.1.2 Unbalanced-tree Ascension

Some concept hierarchies contain leaf values that are of different depths from the root concept. This is reasonable due to inconsistencies in the natural hierarchy of some attributes and is accommodated in the method presented here.

In these cases the deepest leaf nodes are ascended first. These first level concepts are then ascended along with leaf nodes of equal depth from the root in subsequent cycles and so on. Without this control, unbalanced hierarchies would be ascended incorrectly leading to misrepresentation of the data.

For example, if a concept node has a leaf value as a direct descendant, and another as a "grand-child" (ie a concept as a child which has a leaf value as its own child), then ascending each value simultaneously would have both the node and its concept child co-exist in a generalised relation. Although it is true that the data would still match this generalisation, the representation of the number of records in the original relation that match the generalised relation would be incorrect. This is because the votes propagated to the concept node would not include the votes propagated to its concept child (as ascension stopped at a lower level). This is undesirable from our perspective.

3.1.3 Numerical Data

Numerical data is somewhat different to regular discrete-valued attributes such as names or addresses. Even though in any given data set there are always a limited number of records, meaning that only a limited number of different values may exist for a numeric attribute, we prefer to create groupings of such values bound by pre-defined lower and upper limits. These limits can then be used to collate the initial 'discretisation' of numeric attributes into higher level concepts in a hierarchy.

3.1.4 Vote Propagation

A vote is maintained with each generalised record in the generalised relation. This count refers to the number of records in the original data set that correspond to the current generalised record. This vote can be used for threshold control and for post-mining analysis. Outlying data can be readily identified and isolated from common data by post-mining analysis of unusually low votes. Upon ascending an attribute, the newly generalised relation might contain duplicate records, each with their own vote. Before any thresholds are tested on the generalised data, the duplicate records must be consolidated. The votes are propagated and a single instance of the record is retained.

If the vote of a particular record exceeds the specified vote threshold prior to ascension, a copy of that record may be retained separately for later analysis. The record is then ascended like any other. Any further traversal of the concept hierarchy for that consolidated record will result in all subsequent generalised versions of that record also being stored. The vote totals and generalised records stored in the alternate data structure can be used during post- mining analysis to identify infrequent or unusual data. For example, assume a vote- threshold of 100 records is set and a generalised record that breaches this threshold has a vote of 120 records. If the relation is then generalised further and these 120 records are merged with a single record (the resultant vote being 121), then post mining analysis involving the comparison of these records and consequent analysis of the concept hierarchy can help trace the details of that single record. If the alternate generalised data was not stored then only the final results would be available for interpretation and out-lying data may be missed.

3.1.5 Ascension Algorithm

The mining algorithm AOI-t (*Attribute-Oriented Induction with thresholds*) summarises the basic structure of the approach described above.

Algorithm 1: AOI-t

Inputs: Formatted log data; concept hierarchies; thresholds
Outputs: Generalised data for post mining analysis to identify outlying data, irregularities and common trends

```
Calculate depths and number of concepts at each depth
Consolidate initial data and votes
While not complete
  For each priority P ∈ [1, ..., n]
    For each attribute A_i, P_i = P at depth D_A, i ∈ [1, ..., n]
      Check thresholds (see Note 1)
      If final requirements met (see Note 2)
        Set complete
      else if attribute A_i not to be ascended
        add attribute to completed list
      else attribute A_i clear for ascension
        if vote for record exceeds threshold
          store copy of record in alternate relation
        substitute values with next level concepts
        merge identical records (vote propagation)
        decrement depth counter D_A
```

Note 1 Checks if A_i has already been halted or if all attributes have been halted in addition to checking record, attribute and depth thresholds against the relative values.

Note 2 Final requirements are:

1. Record threshold met
2. Attribute specific threshold (including attribute and depth thresholds) reached for each individual attribute
3. All concept hierarchies ascended completely, or
4. Any combination of the above requirements involving each attribute.

3.2 Profile Separation

Generalisation of data is an important step in the computer forensic analysis process. Analysis of the generalised data is required to interpret the results and generate useful information. There are numerous approaches to identifying the information of interest, and the approach chosen will depend on the type of information required. If outlying data or unusual events are being investigated, those records with a low vote may be of interest. If pattern identification for use in further analysis is the aim then those records with the larger votes are likely to be of greater consequence. The advantage of using an initial generalisation procedure in these types of analyses is that the data are simplified and the results quantified. There are also consequences to be drawn by comparing records with low votes (outliers) and large votes (profile elements). By applying appropriate measures to describe the difference between records, we might find serious discrepancies between data elements. Outlier and profile comparisons are not the only methods that can be useful; profile elements may also be compared. In this situation, different kinds of differences may be of interest, for example, identification of potential alternatives for similar activities.

Outlying data can be an indication of data irregularities. For example, the presence of outlying data in a database containing network traffic and login details may be a sign of suspicious access times or unusual access locations. If a secure network was somehow accessed from an unauthorised remote location, then the details regarding this access may be relevant to an investigation. After the attribute-oriented

induction has generalised the data, analysis of the records in the generalised relation can help identify the outliers.

It is not unreasonable to assume that the majority of data in a database should not be considered anomalous. Thus records with low votes in a generalised relation (infrequent data) are prime targets for analysis to identify outliers. Generalised records with large votes, on the other hand, can be considered indicators of normal operations. By normal, we mean "regular" rather than "correct", as frequent anomalous behaviour will indeed register a large number of votes. All votes initially begin at one and the votes of the individual records, post-induction, will depend on the size of the database, the concept hierarchies used, the generalisation thresholds and the content of the data itself. Generalised records will often share the same number of votes, especially low numbered ones. We assign the concept of *vote frequency* to represent the number of occurrences a particular vote has in th generalised record set.

In order to separate outliers from a desired profile, we have devised a number of strategies. For this purpose, the original n records were represented by m (x = vote, y = vote frequency) pairs ($n = \sum_{i=1}^{m} x_i y_i$) generated from the generalised record set. Our aim is to include as much of the data in the profile as possible on the assumption that most systems should be operating under normal conditions. Hence, we only hope to have generalised records with very low votes in the outlier set.

When examining (vote, vote frequency) pairs from several test runs, we made some observations:

- For low x, we usually get high y, and vice versa
- The scatter plot of (x, y) resembles an inverse polynomial (eg $y = y_0 + a/x + b/x^2$)
- There is a large deviation in maximum values along both axes
- There can be large gaps between x values for larger values

These results are dependent on a number of factors, such as data homogeneity, concept hierarchy structure, no of data records available and so on, as mentioned above. It is therefore difficult to recommend a single strategy for separation of profile and outlier elements.

In data mining, these decisions are often made interactively in consultation with the user. For example, a scatter plot may be presented and the user can decide where to make the point of separation. Some automated strategies can also be employed, some of which we present below. Our single most important requirement in these processes is to find an x-value (vote) that separates the profile from the outliers. That is, for some $x_0 \in \{x\}$, if $x \geq x_0$ then x belongs to profile elements, else x belongs to outlier elements.

3.2.1 Constant Separators $x = x_0$, $y = y_0$

If cut-off values are based on percentages or given values then a decision can be made using given thresholds, where the decision becomes finding the nearest vote (x_0) satisfying the thresholds. Examples:

- A maximum threshold of p% of all data can be outliers. Find x_0 so that $\sum_{i=1}^{x_0} x_i y_i \leq (n * p)/100$.
- Profile elements are expected to have a vote frequency smaller than a threshold y_0. Find first x_0, starting from 1, where $y < y_0$.
- x_0 may be given as a threshold.

Several similar variations of the above examples may exist and can be used instead. All require at least some form of user support to provide adequate results, such as a visual inspection of a scatter plot to check position of x_0.

The following sections describe methods to separate the profile and outliers that do not necessarily rely on any input from the user, and hence may be preferable if higher levels of automation in the knowledge discovery process are required.

3.2.2 Using Line $y = mx + x_0$

This technique is a variation of the previous ones, except that it uses the observation that scatter plots of (x, y) resemble the inverse polynomial. It locates the first x-value where the corresponding y-coordinate is less or equal. In effect, the profile will include all x-values to the right of the first point that lies on or under the line. The advantage of this technique is that it does not usually require a user-defined threshold (we used $m = 1$ and $x_0 = 0$ by default), although results may be poor when the generalised data does not satisfy the above assumption.

3.2.3 Euclidean Distances

In this automatic separation method, we assign each point (x_i, y_i), where $i = 2, \ldots, m-1$ to either (x_1, y_1) or (x_m, y_m), depending on which point they are closest to. Then we find the first x_i that has been assigned to (x_m, y_m) and designate all points (inclusive) to the right as profile elements. This technique is sensitive to data where there is big variance in the order of magnitude of some coordinates. Therefore, we usually normalise each point (x_i, y_i) to $(\log x_i, \log y_i)$.

3.2.4 Rate of Change in Means

This technique involves the development of 2 subsets, or clusters, incrementally, starting with a single element in each (the two extremes in x, x_1 and x_m). The composition of the clusters is determined by monitoring the rate of change in the local mean for the two clusters, following the addition of the next element (the next highest vote value for non-profile and next lowest vote value for profile data). The subset with the smallest change in mean is extended to include that new element and new means are recalculated.

This algorithm tends to be affected by locally unusual frequencies. For example, a vote count may have a much lower (or higher) frequency relative to the frequencies of other nearby vote values. The incremental nature of this algorithm could then lead to a bottleneck at this locally unusual frequency, preventing the ideal profile from being developed, by adding undesirable elements to the profile, or by excluding desirable ones. The logarithmic transformation of the coordinates can reduce the effect of large variances.

3.2.5 Closest Point

This technique is an extension of the Euclidean distance method. It is inspired by the principles of PCA (Principal Components Analysis). PCA is a statistical tool used to reduce high dimensional data. Our coordinate pairs, however, are already of a low dimension. Reduction here would mean

finding a single point that is somehow "closest" to all others. The problem thus becomes the finding of the minimum value of $\sum_{j=1}^{m}(x_i - x_j)^2 + (y_i - y_j)^2$, where $i = 1, \ldots, m$. The resulting x_0 where minimum occurs is the first element of the profile. Again, logarithmic conversion can be applied to reduce large differences in data.

3.2.6 K-means Clustering

An alternate approach to the previous method is to apply clustering to the data. However, rather than employing any of the well-known techniques from statistics, we can tailor it to our special case of locating two clusters with their initial single elements being the two extremities in x_1 and x_m. The clusters are defined by a point of separation located by minimising the variance of both clusters. Again, logarithmic conversion of data values can be utilised.

3.2.7 Regression

This idea is also based on the observation that the scatter plot of the data points resembles an inverse polynomial. We have used statistical plotting tools to calculate best matches to several of our data sets. By inspecting the results, it is obvious that the best point of separation would be close to where the polynomial starts "smoothing" out, where its first derivative nears 1, or alternatively where the curvature is highest. Since for a second order inverse polynomial, these calculations involve solving high-order equations, we can instead calculate values for our discrete data points once the parameters of the approximating (regression) polynomial are available.

3.2.8 Separation in One Dimension

Using the empirical evidence suggesting low votes occur more frequently, we can concentrate on examining those values without taking into consideration what the actual counts were. That is, rather than solve the separation problem in two dimensions, we reduce the problem into separating low vote frequency values from high ones. This is similar to one of the user-controlled separation methods suggested in Section *3.2.1*, where a constant frequency value is used to find the profile ($y < y_0$). Some of the previously suggested methods could be re-used for this problem, but standard statistical techniques should provide fast and effective results. For example, if we rank our values in increasing order, we might divide at a given percentile (eg top quartile). Another possibility is to calculate the mean of all vote frequencies. The first vote corresponding to the value less than the mean may be designated as the start of the profile.

3.3 Element Distance Metrics

Once the profile and and outliers are attained from the generalised record set, the next step is to compare individual elements, be that records from the profile or from the profile and the outliers. Activities at this stage can be summarised into these categories:

- Contrasting *outliers to profiles*. This produces a list or summary of data entries that deviate from the profile. Outliers have low support in the data, thus contradicting records in the profile that generally have high support indicate anomalous behaviour that may require further attention.
- Generating *intra-profile* contrasts. This means element pairs in a profile that are in contradiction with each other. These rules may indicate a shift in behaviour, whose causes may need to be investigated. Generally, if only small deviations are observed in two profile elements, this could be attributed to *alternative* behaviour rather than direct contrast.

To measure difference between generalised records found in either the profile or among the outliers, a distance metric will be required.

In data mining, it is common practice to evaluate the strength of information extracted from data sets. This strength is usually described as "interestingness" and the search for better and more relevant measures has received wide attention in the literature [12; 13]. In our analysis, we are comparing two generalised records at a time, and what interests us is as follows:

- How many attributes are being compared?
- How many attributes are different?
- How different are the individual attributes?

Let us illustrate these questions with some examples that may occur in real-life: In the case of unusual network traffic, a user connecting to a server at the same time they normally do might be justified in using a different machine for some reason. If this data is similar to that found in the profile, yet different enough to be excluded from the profile, does it warrant deeper investigation? Similarly, that same user might connect to the same server on a weekend, from a different machine. In this situation there are two factors that come into consideration: the location and the time. In the latter case, the distance of the data from the profile is greater than the distance of the data in the former, and as such may be more interesting. Alternatively, the user may connect on the weekend, from their home computer. In this case, there are still two factors that must be accounted for, however a connection from home may be more suspicious than a connection from a different, authorised, machine. Thus the distance of each individual attribute from the profile also has an important role in measuring interestingness. Therefore a distance measure that accounts for the number of attributes that vary from the profile, and the extent to which those attributes vary is required.

We have briefly experimented with a value distance metric (VDM) [22] to generate a measure that can be used for any number of attributes and provide a meaningful representation of difference between generalised records. We found that results were greatly influenced by varying value frequencies for different attributes (some attributes would have many attribute valuess with small frequencies, others would have few with much larger frequencies) and therefore the magnitudes assigned to certain comparisons were often out of proportion. Indeed, an important observation made throughout this process was that difference in a single attribute (with the others having the same value) is the single most important situation that can occur for computer forensic analysis purposes. Note that if three attributes are compared and only one value is the same, then this situation can be separated into two smaller problems of two-attributed comparisons with one value being the same. However, the significance of a three-attribute comparison having two equal values is much higher just a single one (averaging

the distance metrics for the two different attributes does not work well).

3.3.1 Metric Properties

Suppose we have two generalised records R_1 and R_2, with k attributes, $k \in [2, n]$. We want to develop a "contrast" measure c_i for each attribute A_i, $i = 1, \ldots, k$, such that $c_i \in [0, 1]$, where 0 indicates no contrast. For two attribute values a_i^1 and a_i^2 (both original leaf-values and higher level concept values in the concept hierarchy for A_i are allowed), we can observe the following relationships based on their relative positions in the hierarchy:

1. $a_i^1 \equiv a_i^2$: the values are the same, thus their contrast $c(a_i^1, a_i^2) = 0$.
2. $a_i^1 = a_i^2$: although the two values are not the same, they are in a parent-child hierarchical relationship. Note that our ascension technique does not allow this to occur in a generalised relation (Section *3.1.2*), therefore we do not need to assign a contrast value to this situation.
3. $a_i^1 \approx a_i^2$: the two values are "similar" to each other, or in other words, are in close relationship with each other (siblings, or "uncle-nephew"). In this case, their contrast should be between the two extreme values, ie $c(a_i^1, a_i^2) \in (0, 1)$, depending on how "similar" they are (smaller value for higher similarity).
4. $a_i^1 \neq a_i^2$: the two values cannot be considered similar, their contrast is thus $c(a_i^1, a_i^2) = 1$.

In establishing a contrast measure for similar values, we defined a number of (possibly obvious) requirements, which included statements such as "A child is closer to its parent than its sibling, uncle or grandparent", "A child is closer to its sibling than its uncle or cousin", "A child is closer to its uncle than its cousin", or "A child is the same distance from any of its siblings", and so on. For this purpose, we use the measure

$$h(n_1, n_2) = p(n_1, n_2) \frac{d(n_1) + d(n_2)}{2}$$

where n_1, n_2 are nodes in a concept hierarchy, $d(n)$ is the depth of node n with the root node having the minimal depth of 1, $p(n_1, n_2)$ is the minimal path length to be traversed between nodes n_1 and n_2. Note that by definition $d(n) = p(root, n) + 1$, and $p(n, n) = 0$. By having components for both path length and depth, $h()$ assigns larger values to comparisons that involve nodes at deeper levels of the hierarchy and/or nodes that are further apart "sideways" (in breadth). The conversion of the above formula into the range $(0, 1)$ by

$$c(n_1, n_2) = 1 - \frac{1}{h(n_1, n_2)}$$

satisfies the instinctive requirement of creating larger contrast values for nodes that are, by our definition, far from each other in the hierarchy.

3.3.2 Metric Strategy

The similarity of nodes mentioned in the previous section is a subjective measure. Using the definition of contrast for all elements in a concept hierarchy will never produce the value of 1[1]. Many hierarchies, however, exhibit a naturally fragmented abstraction of values. For example, the attribute *Origin* in Figure 3 might be represented as addresses belonging to different cities. If these cities are directly below the root node, then these branches may be used as separators in similarity, that is, a member of one branch would be considered dissimilar to any other values not in the same branch. This kind of separation may occur at any level in a hierarchy, and needs to be defined prior to calculating contrast measures in order to allow the maximum value to be attainable by some comparisons[2].

In making comparisons between generalised records (Section *3.3.1*), we are hence looking for the following to occur:

- $\forall c_i = 0$, except for one $c_j > 0$.
- $\exists c_i = 0$, with more than one $c_j > 0$.

In both cases we prefer one of the contrast values to be large with the others much smaller (or zero in the first case) in order to warrant further investigation. Generally, as highlighted earlier, it is difficult to handle more than two attributes with non-zero contrasts at a time, but we can overcome this problem by multiple two-attribute generalisations of the original data set.

User	Console	Origin	Day	Date	Time
samson	ftp	cpe-203-21-225-3	Mon	Apr 16	21:24-21:31
ftp	ftp	c81010.upc-c.che	Mon	Apr 16	20:58-20:59
everett	ttyp1	host-216-252-150	Mon	Apr 16	20:24-22:25
max	ftp	max2.cs.mtu.edu	Mon	Apr 16	15:59-16:00
max	ftp	max2.cs.mtu.edu	Mon	Apr 16	15:22-15:40
evelyn	ttyp1	marlin.mtu.edu.a	Mon	Apr 16	15:07-15:12
joseph	ftp	188.191.47.170	Mon	Apr 16	14:23-14:23
joseph	ttyp1	188.191.47.170	Mon	Apr 16	13:05-14:45
george	ftp	188.191.47.157	Mon	Apr 16	12:49-13:05
george	ftp	188.191.47.157	Mon	Apr 16	12:15-12:38
george	ftp	188.191.47.157	Mon	Apr 16	11:46-12:02
robert	ttyp1	hamilton.cs.mtu.	Mon	Apr 16	11:03-11:03
george	ftp	188.191.47.157	Mon	Apr 16	10:58-11:09
robert	ttyp3	hamilton.cs.mtu.	Mon	Apr 16	10:45-10:58
robert	ttyp1	hamilton.cs.mtu.	Mon	Apr 16	10:15-11:03

Figure 3: Example time slice of past and current user login information as obtained by executing the UNIX **last** command.

4. DATA, EXPERIMENTAL METHODOLOGY AND RESULTS

To evaluate the profiling methodology proposed in this paper, a number of experiments have been performed. To generate data sets, log files captured by executing the UNIX **last** command were used, which searches the *wtmp* system log file and lists past and current user login information for a computer. An example output from executing the **last** command is shown in Figure 3.

Note that the data used in our experiments are actual log data recorded by a UNIX-based computer set as server with remote login access. In order to preserve anonymity, the data attribute name instances have been modified. Furthermore, there was no implication of inappropriate behaviour in the data set.

[1] The maximum minimal path in any tree of maximum depth d_{max} is $2d_{max}$, hence $\lim_{d \to \infty}(c_{max} = 1 - \frac{1}{2d_{max}^2}) = 1$.

[2] It could also be argued that there may not be any need to have contrasts to the value of 1 if we are only looking for sufficiently large values.

GRec1	*GRec2*	$\#\, c_i \neq 0$	*ADiff*	$\sum c_i$
288	139	1	0001	0.75
288	194	1	0001	0.75
288	37	1	0010	0.8333333333333334
288	35	1	0010	1.0
288	36	1	0001	1.0
...	...	...	...	...
288	40	2	0011	1.5833333333333335
288	261	2	0011	1.5833333333333335
288	155	2	1010	2.0
...	...	...	...	...
288	110	3	1011	2.75

Figure 4: Profile-to-outlier summary information for generalised records

Of several columns of information generated, six (*Duration* is not shown in Figure 3) attributes were copied or composed into a table containing formatted input. Some filtering was performed at this stage to remove incomplete (current) and non-user (e.g. *shutdown*) logins. The table, using additional higher level concepts from attribute hierarchies, was then mined to produce generalised records. These records were separated into a profile and a set of outliers using some of the algorithms outlined in Section 3.2. Intra-profile and profile-to-outlier contrasting was then performed. The distance metric of Section *3.3.1* was employed to produce reports.

4.1 Profile-to-outlier Experiments

The first set of experiments were performed in order to locate obvious discrepancies among the data. By comparing the profile with outliers, we implicitly make contrasts between common and unusual occurrences in the data. Therefore, if two records are in contrast according to our metric strategy from Section *3.3.2*, they would indicate a highly unusual event occurring in the outliers. Note that although every event described by outlier records can be considered suspicious, with this strategy we attach high importance to events that are only a little different to well-known behaviour. In fact, performing generalisation of four attributes and finding two records with zero contrasts in three attributes is considered more important than the same generalisation for three attributes and two zero contrasts.

The size and contents of the generalised record set varied greatly based on the threshold and parameter settings we employed. As an indication, an original record set of 2000 records is usually reduced to 300-600 generalised records, with a large percentage of these being outliers (up to 95%). Note that high levels of generalisation were deliberately avoided for reasons outlined earlier.

Figure 4 shows a subset of the summary information for an experiment. The first two columns contain the generalised record identifiers being compared, the third column the number of non-zero contrasts, the fourth a bit-representation of the attributes and where the differences occur,[3] while the last one holds the cumulative contrast.

Figure 5 shows a profile element with some comparisons with outlier records where contrast exists for a single attribute. Of the four attributes, **User** was not generalised, **Console**, **Origin** and **Time** were. We have grouped the comparisons by the attribute where difference occurred. The first grouping (**Time**) contains two separate contrast values. The most interesting example here is the last one with

[3]Here, four attributes were generalised, hence the length of 4, with value(s) of 1 indicating the position where non-zero contrast occurred.

Contrast	**User**	**Console**	**Origin**	**Time**
-	george	ttyp	miami.edu	12am-5.30am
0.75	george	ttyp	miami.edu	7pm-12am
0.75	george	ttyp	miami.edu	5.30am-8.30am
1.0	george	ttyp	miami.edu	1pm-5pm
0.8333	george	ttyp	adsl.miami	12am-5.30am
1.0	george	ttyp	bph.net.au	12am-5.30am
1.0	tptp	ttyp	miami.edu	12am-5.30am

Figure 5: Example profile-to-outlier comparisons for a given profile element

the largest value. Upon further examination one may notice that the reason behind this value is that the time associated with this record describes working hours, whilst the other three (including the value in the profile) are all outside working hours. The correct structuring of the hierarchy for this attribute ensured that this particular discrepancy receives higher attention in the analysis (there is a greater path length between non-working and working hours than between non-working hours). The second group of two outlier records contain contrasts in the **Origin** attribute. Again, the larger value is more interesting, and the explanation is similar to the previous case: the smaller distance is explained by the closer relationship between the values *miami.edu* and *adsl.miami*, and points again to the correct logical structuring of the hierarchy. Note that the smaller value of the two contrasts is larger then the smaller value in the previous grouping. We cannot attach any significance to this observation, as it can be explained by the structural differences between the two hierarchies attached to the different attributes even when node relationships are the same (ie. same path length but nodes at greater depth). The last "group" of a single element shows contrast in the **User** attribute. While this may be a valid observation, from the perspective of establishing unusual behaviour, it may not hold the same level of interest as the previous groups. This argument also raises the possible need for further pruning of results (ie. no contrasts may be reported if there are present only in certain attributes).

4.2 Intra-Profile Experiments

Difference in elements of the profile indicate either an alternative or genuine misuse (if the latter is done often enough, it will form part of the profile). Examination of differences therefore needs to be more targeted, thus intra-profile contrasts were calculated only for two attributes at a time. This involves careful selection of meaningful combinations that are selectable prior to generalisation. Of the $\binom{7}{2}$ possible combinations, only a fraction needed to be tried. Much of this process may be automated, if valid pairings of attributes are available as part of our background knowledge. Then, generalisation can be performed for all pairings, and for each, an ordered set of generalised record pairs can be produced, with higher contrast values requiring more attention. Note that the results need to be organised by individual attribute pairs, as each attribute has its own hierarchy, the structure of which directly affecting the actual contrast values produced. This causes the mathematically possible contrast values to vary greatly in magnitude for each pairing, which therefore cannot be summarised together (indeed, it can be proven that only a limited number of values can be produced for different maximum depths and path lengths). For the same reason, we must avoid over-generalisation of

attributes. We found that the best strategy for profile contrasting purposes was to generalise only one or two levels of the concept hierarchy, and often only for one of the attributes as opposed to both.

One of the more interesting contrasts produced by our tests was the pair (with maximal contrast value 1.0 in *Origin*, meaning the two concepts were on separate branches in the hierarchy)

$GR_0 : (\textbf{User} = george) \wedge (\textbf{Origin} = miami)$
$GR_1 : (\textbf{User} = george) \wedge (\textbf{Origin} = mtu.edu.au)$

which indicates that the same user has been logging in from two very different geographic locations. Further inspection of this contrast revealed that the user in question left his place of work at an Australian university for another in Miami while still regularly accessing his old academic account. For this particular user, we found another profile element with maximal contrast to $\textbf{Origin} = miami$, which, when compared to the university location

$GR_1 : (\textbf{User} = george) \wedge (\textbf{Origin} = mtu.edu.au)$
$GR_2 : (\textbf{User} = george) \wedge (\textbf{Origin} = telstra)$

yielded a contrast value of 0.75. This corresponds to a closer relationship between the two locations. Indeed *telstra* is an Australian service provider and the generalised record described after-hours external logins of the user to the university server.

5. CONCLUSIONS AND FUTURE DIRECTIONS

The initial implementation of the profiling analysis process based on attribute-oriented induction and described in this paper has resulted in promising results capable of identifying irregularities in computer logs that can serve as useful evidence in computer crime investigations. The paper owes a lot to our earlier attempt to apply similar analysis methodology using association rules [1]. Clear differences between the mining algorithm and post-analysis techniques exist between this presentation and that paper. Profile analysis, however, forms only a part of the investigative process and relies heavily on expert knowledge. It is therefore best perceived as a component in a larger collection of tools designed to aid the forensic investigator.

Further opportunities to enhance our methodology exist in several areas. One such area is the handling of multiple log information in a single process. Multi-dimensional mining may offer a solution for this problem, with some interesting work already found in the literature [16; 20]. Alternatively, it may be possible to "flatten" several logs into a sequence of "events", for which more traditional sequential mining techniques can be applied. Further work can also be carried out in the intelligent presentation of results, notably in the provision of appropriate visual interpretation of the profiles and its potential contrasts. Contrast measures currently used are concept hierarchy-based and such depend heavily on the quality of these structures. Measures based on the data alone or in combination with concept hierarchy based ones may strengthen the validity of some of our conclusions. Further investigation of existing techniques summarised in [12; 13] may help in finding new ideas for this purpose.

We also hope to analyse other, larger sets of data than the one that was available to us at the time of the experiments to potentially fine-tune our approach.

6. REFERENCES

[1] T. Abraham and O. de Vel. Investigative profiling with computer forensic data and association rules. In *Proceedings of the IEEE International Conference on Data Mining*, Maebashi City, Japan, December 2002.

[2] G. Adomavicius and A. Tuzhilin. Expert-driven validation of rule-based user models in personalization applications. *Data Mining and Knowledge Discovery*, 5(1/2):33–58, 2001.

[3] G. Adomavicius and A. Tuzhilin. Using data mining methods to build customer profiles. *Computer*, 34(2):74–82, 2001.

[4] C. Aggarwal, Z. Sun, and P. Yu. Online algorithms for finding profile association rules. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM-98)*, Bethesda, MD, USA, 1998.

[5] E. Casey. *Digital Evidence and Computer Crime*. Academic Press, 2000.

[6] P. K. Chan. A non-invasive learning approach to building web user profiles. In B. Masand and M. Spiliopoulou, editors, *Proceedings of the Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*, August 1999.

[7] O. de Vel, A. Anderson, M. Corney, and G. Mohay. Mining e-mail content for author identification forensics. *SIGMOD Record*, 30(4), 2001.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second Int. Conference on Knowledge Discovery and Data Mining*, August 1996.

[9] T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.

[10] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: an attribute-oriented approach. In *Proceedings of 18th Int. Conference on Very Large Databases*, 1992.

[11] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proceedings of 21st VLDB Conference*, September 1995.

[12] R. J. Hilderman and H. J. Hamilton. Heuristic measures of interestingness. In *Proceedings of the 3rd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'99)*, September 1999.

[13] R. J. Hilderman and H. J. Hamilton. Knowledge discovery and interestingness measures: A survey. Technical Report CS-99-04, Dept of Computer Science, University of Regina, October 1999.

[14] M. Hirsh, C. Basu, and B. Davidson. Learning to personalize. *Communications of the ACM*, 43(8):102–106, 2000.

[15] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[16] H. Lu, L. Feng, and J. Han. Beyond intra-transaction association analysis: mining multi-dimensional inter-transaction rules. *ACM Transactions on Information Systems*, 18(4):423–454, 2000.

[17] B. Mobasher, H. Dai, T. Luo, Y. Sun, and J. Wiltshire. Discovery of aggregate usage profiles for web personalization. In *Proceedings of the Workshop on Web Mining for E-Commerce (WEBKDD'00)*, August 2000.

[18] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos. Effective prediction of web-user accesses: a data mining approach. In *Proceedings of the Workshop on Mining Logdata Accross All Customer Touchpoints (WEBKDD'01)*, August 2001.

[19] S. Nesbitt and O. de Vel. A collaborative filtering agent system for dynamic virtual communities on the web. In *Proceedings of the Conference on Learning and Discovery (CONALD98)*, June 1998.

[20] T. Oates and P. R. Cohen. Searching for structure in multiple streams of data. In *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996.

[21] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proceedings of 21st VLDB Conference*, September 1995.

[22] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[23] P. N. Tan and V. Kumar. Mining indirect associations in web data. In *Proceedings of the Workshop on Mining Logdata Accross All Customer Touchpoints (WEBKDD'01)*, August 2001.

Mining Antarctic scientific data: a case study

Ben Raymond and Eric J Woehler
Australian Antarctic Division
Kingston, Tasmania 7050
http://www-aadc.aad.gov.au
ben.raymond@aad.gov.au

ABSTRACT

The Australian Antarctic Data Centre is a web-accessible repository of freely-available Antarctic scientific data. The Data Centre seeks to increase the value and utility of its holdings through data mining analyses and research. We present and discuss analyses of an extensive spatial/temporal database of at-sea observations of seabirds and related physical environmental parameters. Mixture-model based clustering identified two communities of seabirds in the Prydz Bay region of East Antarctica, and characterised their spatial and temporal distributions. The relationships between observations of three seabird species and environmental parameters were explored using predictive logistic models. The parameters of these models were estimated using data from the Prydz Bay region. The generality of the models was tested by applying them to data from a different region (that adjacent to Australia's Casey station). This approach identified regional differences in the at-sea observations of seabird species. The results of these analyses complement those of at-sea studies of seabirds elsewhere around the Antarctic. They also provide insights into possible data errors that were not readily apparent from direct examination of the data. These analyses enhanced ecological understanding, provided feedback on survey strategy, and highlighted the utility of the repository.

1. INTRODUCTION

The Australian Antarctic Data Centre (AADC) was established in 1995 to make scientific observations and results from Antarctica freely available. The free availability of data is one of Australia's obligations under the Antarctic Treaty (article III). The majority of the data collected in Antarctica, while originally collected for a specific investigation, nevertheless have wide potential relevance to other projects and investigators. Many of the AADC's holdings are ecological or environmental in nature, and linkages between databases are extensive.

The AADC plays an active role in the analysis of Antarctic scientific data by mining its holdings. The broad aim is to improve the value of these data to the Antarctic community. Several approaches are being taken, including:

- the direct application of mining and exploratory techniques in order to uncover new information from the data. These analyses are additional to those undertaken as a routine part of Antarctic scientific studies and aim to exploit the multi-disciplinary nature of the data held by the AADC;
- the extraction of actionable information from low-level scientific data. This has direct application to conservation, planning, and legislative activities, as well as producing "end-product" data suitable for use by other scientific investigators; and
- to generate a better understanding of the holdings of the AADC, including the identification of data errors, duplicated data, missing records, linkages between databases, and data acquisition procedures. This information has direct application for data management issues, such as maintaining high data quality and an efficient database structure.

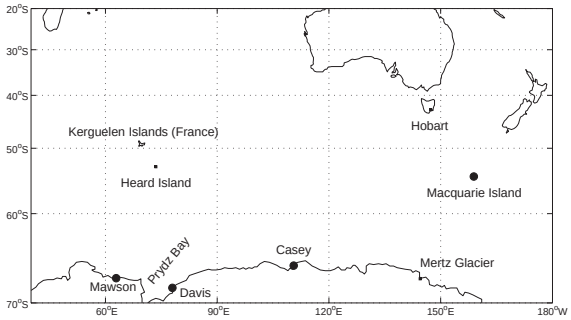


Figure 1: Australian Antarctic research stations (•) and other locations mentioned in the text

We present an overview of the mining of the "Wildlife-on-Voyage" (WoV) database. This database holds an extensive collection of observations of wildlife (comprising birds, whales, and kelp) made from ships during Antarctic voyages. The information within this collection has wide scientific relevance. However, the data present numerous analytical challenges, including spatial and temporal variation (within and across years), missing values, and a lack of balance in sampling. We begin by describing the data and the methods that were used to collect them, and then present and discuss two investigations using these data. These investigations focused on the identification of communities of seabirds and the relationships of the birds with their environment.

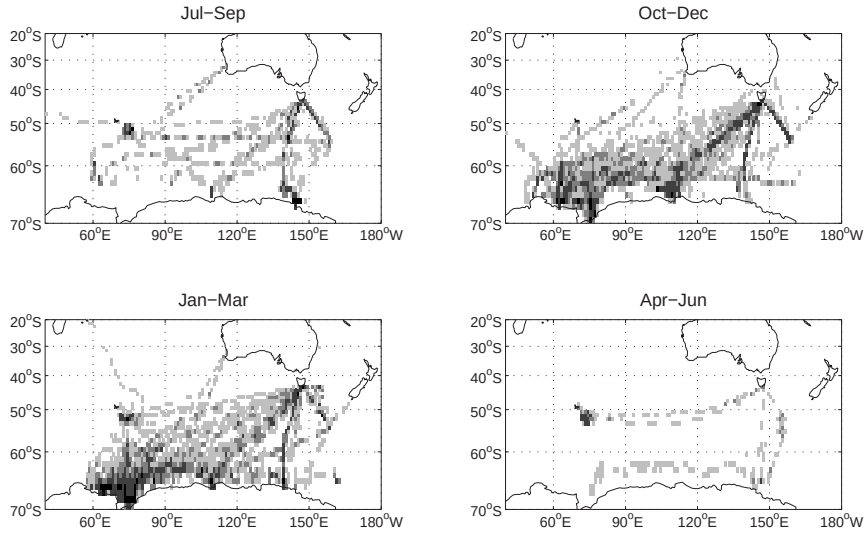


Figure 2: Spatial and temporal distribution of at-sea sightings of seabirds made from Australian Antarctic voyages, 1980-2002. Data from all years have been pooled. Densities are shown in cells of size 1° longitude × 1° latitude. The shade of grey denotes the number of surveys made in the cell (black = more than 30 surveys, white = no data)

2. DATA DESCRIPTION

The seabird component of the WoV database comprises approximately 140 000 observations of 119 species, made on 98 voyages conducted between 1980 and 2002. These voyages were undertaken in the course of Australia's Antarctic scientific research program. The majority of the voyages were for the transportation of personnel and supplies to Australian Antarctic bases (see Figure 1), with a small number of voyages for scientific surveys. While survey voyages attempted to maintain a balanced sampling strategy, the same was not true of the transportation voyages. Observations on these voyages were incidental, with little or no opportunity for balanced survey design. Figure 2 shows the spatial and temporal distribution of the data. The most densely surveyed areas are clearly those adjacent to the Australian Antarctic stations. The temporal distribution of the observations is heavily biased against the winter months, because the extensive sea ice in the Antarctic during winter makes ship travel virtually impossible.

Observations of wildlife were made in surveys of 10 minutes duration, with generally one survey made per hour of the voyage. Physical environmental data collected at the time of each survey included sea surface temperature (°C), sea state (or wave height, recorded on an ordinal scale), cloud cover (categorised as clear, partial, total, or blowing snow), wind force (Beaufort) and direction, and atmospheric pressure (hPa). Sea ice cover was also estimated but, as discussed below, alternative sea ice data derived from satellite images were used in the analyses.

3. PREPROCESSING

3.1 Data cleaning

Data cleaning and error checking consumed a large proportion of the time spent on this study. Prior to the 1992/1993 season all observations were recorded on paper forms and manually entered into the database. On voyages after this season a laptop-based entry system was used where possible, reducing the likelihood of errors in data transcription.

Rule-based techniques were used to detect violations of physical limitations: for example, sea surface temperature cannot be less than -1.8 °C, the approximate freezing point of sea water. Similarly, the differences in time and position of consecutive observations were used to calculate an apparent ship speed, which was then compared to a maximum possible speed of 25 knots. There were instances in which either the time or position stamp of a data record was in error by one digit, suggesting an error during manual entry of the data. Position and time stamp errors were in general more easily identified using graphical methods, particularly where the errors were small (for example, transcription errors in the tenths-of-degrees digit).

The species diversities of Antarctic seabird communities are low. Except for very rare species one could reasonably expect to encounter the same species from year to year in a given region. The identification of species for which there were very few observations in a region therefore proved to be a simple but effective mechanism of finding records that were likely to contain errors in species identification or data entry. For example, we found four observations of Australasian gannets in Prydz Bay (66°S), a species which is not normally

found south of 50°S. Other likely errors in species identification were also identified during the community analyses (see section 4, below).

Errors were corrected using interpolation from surrounding values where possible, or patched using data from the marine science database (see below). In some cases, there were insufficient data to allow interpolation: such entries were deleted from the data set.

3.2 Database linkages

The physical environmental variables in the WoV database (see section 2, above) provide a natural set of linkages to other databases both within and external to the AADC. Of particular interest is a marine science database that holds data collected from onboard sensors during Antarctic voyages. These data include various environmental variables including sea surface temperature, wind speed and direction, and solar radiation, as well as voyage information such as ship speed and position. Marine science data are available only from voyages of the *Aurora Australis*; the other ships used for Australian Antarctic scientific voyages do not have this real-time data logging system installed.

Other, external, environmental databases are also relevant to this study. For example, the National Snow and Ice Data Centre at the University of Colorado (`http://nsidc.org`) maintains a database of satellite-derived sea ice concentration data. This database holds daily Antarctic sea ice concentration data from 1978 onwards, on a spatial grid with a cell size of 25km × 25km. These sea ice data were used in preference to the directly-observed data, in order to avoid the potential bias of ship tracks to areas of open water (i.e. less sea ice).

4. COMMUNITY ANALYSIS

4.1 Motivation

A community can be defined as a group of species that share a habitat. Community analysis can offer a broad view of an ecosystem and allows species-level information to be abstracted and presented in a compact form. Such analyses are therefore of interest for management and conservation purposes, but may also be used to guide more specific scientific investigations of particular species or areas of interest. The concepts and techniques of community analysis are identical to those of market basket analysis in data mining (used in a transaction database context, for example, to identify products that tend to be purchased together).

4.2 Methods

The study area of interest was Prydz Bay, defined as that area of the Southern Ocean between 60 °E and 90 °E, and south of 60 °S to the Antarctic continent (see Figure 1). Prydz Bay was chosen as it has been the focus of numerous studies of seabirds in their colonies [18]. Prydz Bay is the primary seabird breeding locality in East Antarctica, with breeding populations of nine species [18], comprising approximately 30% of that East Antarctic seabird biomass [16]. Furthermore, the WoV data coverage within Prydz Bay is relatively dense, as two of Australia's four permanent research stations (Davis and Mawson) are located along this sector of the Antarctic coastline. Observations were pooled into composite records for the analyses. The pooling was limited so that these composite records contained consecutive observations from a single voyage only, and spanned no more than 12 hours and a 50 km change in ship position. These composite records are referred to here as "sites", which is the usual nomenclature used in the ecological literature. The species composition of each site was compiled in presence/absence format, and the environmental variable values within a composite record were combined using a median (for continuous variables) or mode (for nominal or ordinal variables) operator.

Seabird communities were explored using two complementary cluster analyses. The first examined the clustering of sites based on species composition. The seabird communities were then generated from the species compositions of the resulting site clusters.

The division of ecological data into discrete clusters can be problematic because in many cases the data do not show an inherently grouped structure. Rather, ecological data commonly form a continuum between extremes. The division of such a continuum into distinct entities does not necessarily lead to results that make intuitive sense. Soft clustering algorithms (also known as *fuzzy*, or *probabilistic* clustering), which assign to each datum a membership level in each cluster may therefore be preferable to "hard" clustering algorithms, which allocate each datum exclusively to a single cluster.

We applied a mixture-model approach [4; 13] to the clustering of sites by species composition. This is a soft clustering approach in which the data are modelled by a mixture of probability distributions, with each representing a different cluster. Since the species compositions were in binary (presence/absence) form, the Bernoulli distribution was the natural choice. Mixtures of multivariate Bernoulli distributions have been shown in theory to be non-identifiable [11]; however, in practice, interpretable results can still be obtained [6]. We used maximum-likelihood estimation by expectation-maximisation [9; 20]. Although we do not do so here, the mixture model approach also offers principled methods for the selection of the correct number of clusters [10]. This would be of interest in situations where a large number of cluster analyses were required, with little prior information available to guide the choice of number of clusters. Our choice of number of clusters was based on prior knowledge of the seabird communities along with expert assessment of the properties of the emergent clusters.

The species compositions of the seabird communities were assessed on the basis of the membership of each species to each cluster as well as the constancy of each species within each cluster. The constancy may be calculated as the fraction of sites from a cluster that contain an observation of the species in question. Species with a high membership-constancy product can be considered to be the "indicator" species of an assemblage [8]. Indicator species are useful for characterising the species composition of an assemblage, where such an assemblage contains many species.

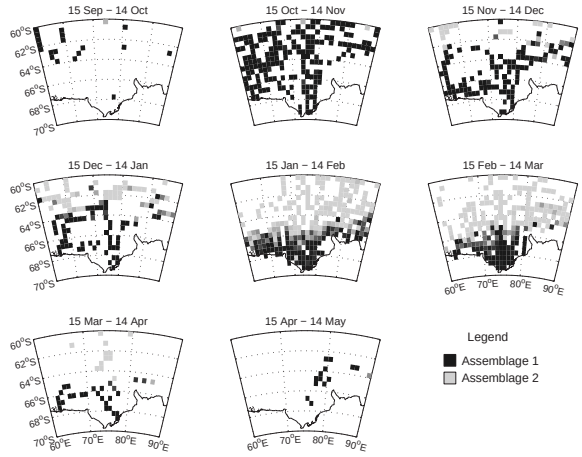


Figure 3: Spatial and temporal distribution of two assemblages of seabirds in the Prydz Bay region of Antarctica. The species composition of each assemblage is shown in Figure 4

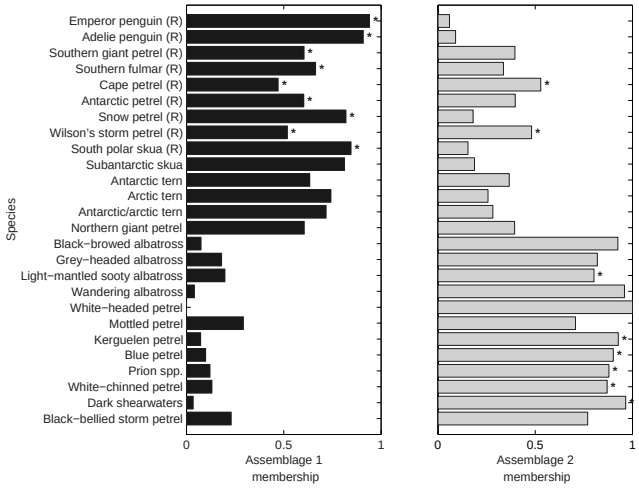


Figure 4: Membership of 26 seabird species to the two assemblages shown in Figure 3. (R) indicates the species that breed in Antarctic locations. Indicator species (see text) are marked with an asterisk

The second cluster analysis grouped seabirds according to their spatio-temporal ranges. In this approach, species that were observed in the same region of the ocean at the same time are grouped together. This yields the seabird communities directly. Dissimilarities between species ranges were calculated using the TwoStep algorithm [2] and clustering computed using a hierarchical complete-linkage algorithm. A hierarchical clustering is more natural in this case because the number of entities is small (26 species within Prydz Bay) and the hierarchy of the dendrogram is itself of interest. Seabird communities identified using this approach are referred to here as "associations". The communities identified by the mixture-model clustering described earlier will be referred to as "assemblages" in order to differentiate the two approaches.

4.3 Results and Discussion

The clustering of sites by species composition revealed a two-group structure in the seabird assemblages. The spatial and temporal distributions of these assemblages (all years combined) is shown in Figure 3, and the species composition of the two assemblages is shown in Figure 4. Assemblage 1 contains all nine species that breed in Prydz Bay in addition to sub-Antarctic skuas, arctic and Antarctic terns, and northern giant petrels. This assemblage was observed close to the Antarctic coast during the middle of the breeding season (January-March, Figure 3). Assemblage 2 contains the remaining 12 species, all of which breed in temperature or sub-antarctic latitudes and forage within Prydz Bay during the southern hemisphere summer. This assemblage was observed during the summer months (December-March), offshore from the Prydz Bay coast. The spatio-temporal ranges of the two assemblages overlap, as can be seen from the mid-grey cells in Figure 3. This overlap is handled transparently by a soft clustering algorithm, because sites which host both assemblages at the same time will have a non-zero membership to both assemblages. In contrast, a hard clustering algorithm would assign such a site exclusively to one of the two assemblages. The overlap is more readily observed using the soft clustering approach. Increasing the number of clusters to three placed this overlap into its own cluster, further highlighting this finding.

Indicator species are marked on Figure 4 with an asterisk. Two species (cape petrels and Wilson's storm petrels) were found to be indicator species in both assemblages. This suggests that their at-sea distributions were quite broad, whereas the other breeding species were generally observed only in relative proximity to the Prydz Bay coast (particularly during the middle of the breeding season; see the distribution of assemblage 1 in Figure 3). This difference is a result of the fact that these two species breed both on the Prydz Bay coast as well as sub-Antarctic locations such as Heard Island (which lies to the north of Prydz Bay; see Figure 1). Thus, individuals observed offshore from the Prydz Bay coast are probably those breeding on Heard Island. The only other species that breeds both in Prydz Bay and on Heard Island is the southern giant petrel.

The hierarchical clustering of species by spatio-temporal range is shown in Figure 5. Cutting the dendrogram at a relatively high dissimilarity level yields two seabird associations (marked as (a) and (b) on the figure) that are identical to the two assemblages shown in Figure 4. Association (a) may be further split into (a1) and (a2). Sub-association (a1) contains southern giant petrels, cape petrels, Wilson's storm petrels and arctic terns: three of these are the species that breed both on the Prydz Bay coast and on Heard Island. Their at-sea distributions are therefore different from the distributions of the remainder of the breeding species. This finding reinforces that obtained from the first cluster analysis, discussed above.

As well as providing direct community information, these analyses yielded additional information relating to issues of species identification. Antarctic terns, arctic terns, and their

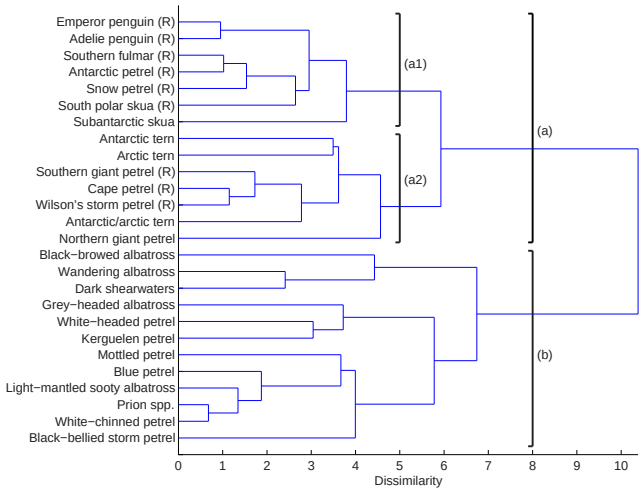


Figure 5: Dendrogram of seabird species, clustered according to similarity of spatio-temporal range. (R) indicates species the that breed in Antarctic locations. Groupings within the dendrogram labelled (a), (b), etc. are discussed in the text

composite (used when specific identification at sea was not possible), are grouped in the same assemblage in Figure 4, and relatively tightly in Figure 5. However, the behaviours of the two species are quite different. Arctic terns breed in the northern hemisphere and migrate to Prydz Bay in the southern hemisphere summer to feed. Antarctic terns breed on sub-Antarctic islands (such as Heard Island) during the summer and migrate north to South Africa during the winter. Antarctic terns breeding on Heard Island feed inshore and do not venture far from land. Thus, our clustering results suggest that at least some of the records of Antarctic terns in Prydz Bay may in fact be arctic terns that have been misidentified. Detailed examination of the distributions of these records would be needed to identify which are likely to be in error. Similarly, northern giant petrels were clustered together with the resident species. Northern giant petrels are a migratory species that are generally found in the northern regions of Prydz Bay [17]. Examination of northern giant petrel records revealed that on one particular voyage, a high number of unlikely northern giant petrel sightings were recorded in the southern part of Prydz Bay. It is possible that these were misidentified southern giant petrels.

5. ENVIRONMENTAL RELATIONSHIPS

5.1 Motivation

The community analyses described above provide a foundation for investigating the relationships between seabirds and their environment. A proper understanding of these relationships is vital for an understanding of the seabirds, the region, and for planning, management, and legislative purposes. Characterising the dependence of the birds on their environment is one of the first steps in assessing the likely impact of global climate change on southern ocean seabirds. It has been suggested [14] that the initial effects of global climate change may be most pronounced in sub-Antarctic regions.

We investigated the use of predictive models as a means of investigating the relationships between seabird observations and environment. Seasonal behaviour and the response to environment are likely to differ among the species within a community, leading to dynamic community compositions. Furthermore, neighbouring communities are not disjoint but rather overlap at the edges of their ranges [19]. The models were therefore built using species-level data rather than community level. Given predictions of individual species ranges, it would be a straightforward matter to combine these into community-level predictions if desired.

The ability to successfully predict the at-sea distributions of seabirds from environmental parameters would be extremely valuable. At-sea survey data for much of the world's oceans are limited due to the logistic difficulties and costs involved. Predictive models that use remotely-sensed environmental data may allow the estimation of seabird distribution in those areas of the ocean not amenable to direct survey.

5.2 Methods

Seabird observations from two different areas were used. Observations from Prydz Bay were used to build and test the models. These models were then applied to data from the Casey station region in order to test the generality of the models. The delineation of Prydz Bay was the same as in section 4, above, except that the northern boundary was extended to 50°S. This extension includes Heard Island (53°5'S, 73°30'E, an important seabird breeding area) in the study. The Casey station region was delimited to the area between 100°E and 120°E, and south of 50°S to the Antarctic continent (see Figure 2). There is no northern land mass equivalent to Heard Island in the Casey station area.

These geographical areas were divided into grids of spatial bins, each spanning 2° longitude by 2° latitude. We assumed that the relationships between bird observations and the physical environment remain constant among years; therefore, data from all years were pooled. However, these relationships do vary with time of year as the bird behaviour is driven by differing processes throughout the season. For each species studied here we have therefore fitted a temporal sequence of models. Each model spanned a 30 day time period and consecutive models overlapped by 15 days.

We present the results of three seabird species: snow petrels (*Pagodroma nivea*, which breed in Antarctic localities including Prydz Bay and the Casey station coastal regions), cape petrels (*Daption capense*, which breed in the Antarctic as well as in sub-Antarctic localities), and white-chinned petrels (*Procellaria aequinoctialis*, which breed on islands in temperate latitudes and forage in Antarctic waters during the southern hemisphere summer). These three species were the three most commonly-observed in each of the three breeding categories described above.

The species compositions of the bins were again compiled in presence/absence format. Logistic regressions were used to relate the distributions of bird observations to four parameters of the physical environment: sea surface temper-

ature (°C), sea state (ordinal scale), sea ice concentration (percent), and distance to coast (km). The model accuracies were assessed using the mean square prediction errors (MSE). For models using the Prydz Bay data, MSE was assessed using cross-validation by voyage: that is, data from half of all available voyages (chosen at random) were used to estimate the model parameters. Data from the remaining voyages were used to assess the model accuracy. Cross-validation is a widely used method of obtaining estimates of model accuracy, particularly when data are limited [5; 15]. All MSE values are presented with reference to the null error rate. This is the mean square prediction error that is obtained with a constant model and reflects the prevalence of the species in question. Any model that fails to predict more accurately than the null is no better than uninformed guessing.

The standard logistic regression assumes that the data are independent. When data are spatial in nature, this assumption is often violated because observations from one location are likely to be similar to observations from nearby locations. This self-similarity is known as spatial autocorrelation [7]. Spatial autocorrelation can often be exploited to improve the predictive accuracy of models. Accordingly, we also applied the spatial autologistic model [1; 3]. This is an extension of the logistic model that explicitly models the spatial autocorrelation of the observations. The estimation of the parameters of the spatial autologistic model is problematic and requires approximate maximum likelihood techniques (see e.g. [12] for a discussion of the estimation of such models). We used a Markov chain Monte Carlo implementation provided by LeSage [12].

The relative importance of each environmental variable in predicting the distribution of observations of each species was assessed. This was achieved by building a model using only a single environmental variable as a predictor. The cross-validation predictive accuracy of this model was compared to the best predictive accuracy obtained using all four predictor variables.

5.3 Results and Discussion

The predictive accuracies of the logistic models are presented in Figure 6. For observations of snow petrels, good predictive accuracies (MSE significantly less than the null rate) were obtained for the entire summer breeding season in both the Prydz Bay and Casey station regions. The model for cape petrel observations was generally adequate in the latter half of the season in both regions. The model for white-chinned petrel observations was better than the null for the majority of the season in Prydz Bay, but was no better than the null during the latter half of the season in the Casey station region.

For snow and cape petrels, the model performance in the Casey station region was similar to that obtained using the Prydz Bay data (Figure 6). We can therefore conclude that the processes linking bird observations and physical environment are similar in the two areas. The same was not true of white-chinned petrels. During the latter half of the season the model error was less than that of the null in Prydz Bay, but not the Casey station region. This result draws attention to the fact that there are differences in the processes linking environment with observations of these birds in the two regions.

The importances of each of the environmental variables in predicting the observations of these three species are illustrated in Figure 7. From late October until approximately January, the most important predictor variables for snow and white-chinned petrels were sea ice concentration and sea state. Sea ice and sea state are co-variates: heavy sea ice will prevent high sea states (wave heights). Snow petrels are known to be an ice-associated species and this is reflected in the positive sign of the model coefficient (marked on Figure 7). The reverse is true of white-chinned petrels. The corresponding variable importances for cape petrels are not relevant because the model was not accurate during this period.

During the latter half of the season the most important predictor variables were sea surface temperature and distance to coast. The model parameter for distance to coast was negative for snow and cape petrels, indicating that these species were observed close to the coast. This matches the known behaviour of the birds: during this time of the breeding season adult birds are feeding the newly hatched chicks and thus forage predominantly close to the coastal colonies.

The autologistic model did not provide substantially better predictive accuracy than the standard logistic model (results not shown). This suggests that the spatial variation in the observations was adequately modelled by the spatial variation in the environmental predictor variables. The additional computational demands of the autologistic model are therefore not justified in this application.

Although in this study we relied on direct observations of sea state and sea surface temperature, these environmental variables may both be estimated using remote sensing technology. The models developed here could therefore potentially be used to estimate at-sea distributions of seabirds in other regions of the Antarctic. Regional differences in the breeding distributions of seabird species would need to be addressed.

6. DISCUSSION AND CONCLUSIONS

The collection of data from polar regions is an expensive and difficult process. Such data are often noisy or incomplete and analyses using conventional statistical (hypothesis-testing) techniques can be extremely difficult. Data mining and exploratory techniques may allow insights into trends and anomalies to be obtained. The relevance of such findings extends beyond intrinsic scientific interest into fields such as conservation and planning. Polar science plays a key role in matters of global importance, including species conservation and global climate change. There are therefore social, scientific, and economic obligations to make the best possible use of Antarctic scientific data.

The investigations presented here used data mining techniques to obtain results of ecological relevance, such as the structures of seabird communities and the relationships of

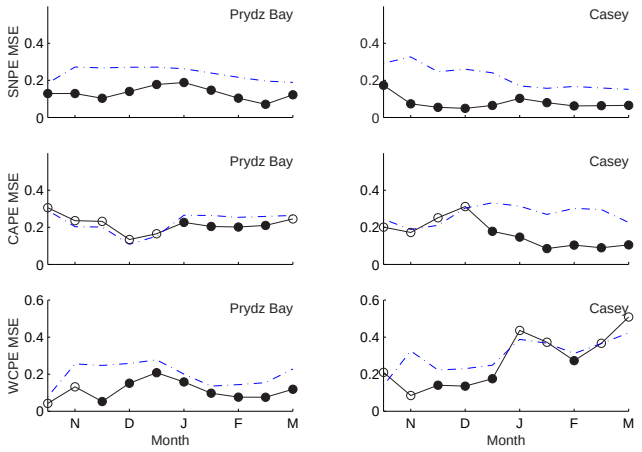


Figure 6: Mean square prediction error (MSE) of logistic models of at-sea observations of three species of seabird in two areas of the Antarctic (Prydz Bay and the Casey station region). The solid line is the logistic model error and the dot-dash line is the null error rate. A filled circle indicates that the logistic MSE is significantly less than the null error at that time (p<0.05, Wilcoxon paired sample test). SNPE=snow petrels, CAPE=cape petrels, WCPE=white-chinned petrels.

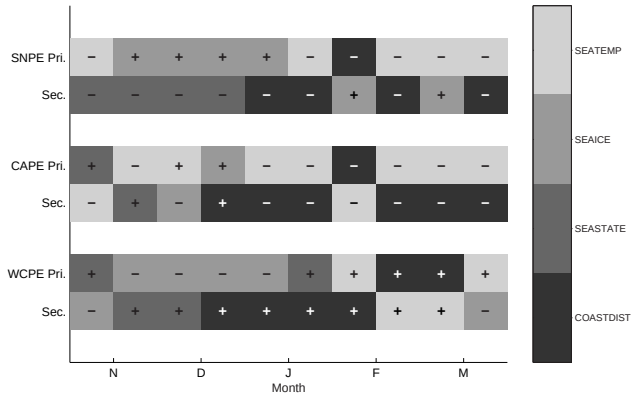


Figure 7: The two most important predictor variables (primary, Pri., and secondary, Sec.) for logistic models of at-sea observations of three species of seabirds. A positive sign indicates that the association was positive (i.e. observations were more likely with increasing values of the environmental variable). SNPE=snow petrels, CAPE=cape petrels, WCPE=white-chinned petrels; SEATEMP=sea surface temperature, SEAICE=sea ice concentration, SEASTATE=sea state (wave height), COASTDIST=distance to nearest coast.

seabird observations with the physical environment. The techniques and findings also addressed matters of data management. Errors in data, which are often difficult to detect through direct inspection, may become apparent in the results of the analyses. This was illustrated by the potential errors in Antarctic tern and northern giant petrel records discussed in section 4.

Spatial considerations are often of concern when dealing with ecological data. In our models of seabird observations and physical environmental parameters, the predictive ability of spatial autologistic models was found to be no better than that of ordinary logistic models (in which spatial autocorrelation is ignored). The additional computational cost of the spatial autologistic model (we used a computationally intensive Markov chain Monte Carlo implementation) is therefore not justified in this application.

Acknowledgments

The authors would like to thank L Belbin and M Riddle for their ongoing support, and all observers who have recorded at-sea observations over the past 22 years. G Cruickshank, C Hodges, B Priest, and F Spruzen entered much of the data in prepraration for the analyses. D Watts constructed and maintains the WoV database. Various freely-available Matlab toolboxes were used: the m_map mapping toolbox (Rich Pawlowicz, `http://www2.ocgy.ubc.ca/~rich/`), the Econometrics toolbox (James P. LeSage, `http://www.spatial-econometrics.com`), and ML estimation of mixtures of multivariate Bernoulli distributions (Miguel Á. Carreira-Perpiñán, `http://cns.georgetown.edu/~miguel/`).

7. REFERENCES

[1] N. Augustin, M. Mugglestone, and S. Buckland. An autologistic model for the spatial distribution of wildlife. *J Appl Ecol*, 33:339–347, 1996.

[2] M. Austin and L. Belbin. A new approach to the species classification problem in floristic analysis. *Aust J Ecol*, 7:75–89, 1982.

[3] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *J Roy Sta B*, 36(2):192–236, 1974.

[4] C. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[5] K. Burnham and D. Anderson. *Model selection and inference*. Springer-Verlag, 1998.

[6] M. Carreira-Perpiñán and S. Renals. Practical identifiability of finite mixtures of multivariate Bernoulli distributions. *Neural Comp*, 12(1):141–152, 2000.

[7] N. Cressie. *Statistics for spatial data revised edition*. Wiley, 1993.

[8] M. Dufrêne and P. Legendre. Species assemblages and indicator species: the need for a flexible asymmetric approach. *Ecol Monogr*, 67:345–366, 1997.

[9] B. Everitt and D. Hand. *Finite Mixture Distributions*. Monographs on Statistics and Applied Probability. Chapman & Hall, 1981.

[10] C. Fraley and A. Raftery. How many clusters? Which clustering method? Answers via model-based cluster analysis. *Computer J*, 41:578–588, 1998.

[11] M. Gyllenberg, T. Koski, E. Reilink, and M. Verlaan. Non-uniqueness in probabilistic numerical identification of bacteria. *J Appl Prob*, 31:542–548, 1994.

[12] J. LeSage. Bayesian estimation of limited dependent variable spatial autoregressive models. *Geogr Anal*, 32(1):19–35, 2000. http://www.spatial-econometrics.com.

[13] G. McLachlan and K. Basford. *Mixture models: inference and applications to clustering*. Marcel Dekker, Inc., New York, USA, 1988.

[14] R. Smith, D. Ainley, K. Baker, E. Domack, S. Emslie, B. Fraser, J. Kennett, A. Leveter, E. Mosley-Thompson, S. Stammerjohn, and M. Vernet. Marine ecosystem sensitivity to climate change. *BioScience*, 49(5):393–404, 1999.

[15] M. Stone. Cross-validatory choice and assessment of statistical predictions (with discussion). *Biometrika*, 64:29–35, 1974.

[16] E. Woehler. The distribution of seabird biomass in the Australian Antarctic Territory: implications for conservation. *Envir Cons*, 17:256–261, 1990.

[17] E. Woehler, C. Hodges, and D. Watts. *An atlas of the pelagic distribution and abundance of seabirds in the southern Indian Ocean, 1981 to 1990*, volume 77 of *ANARE Research Notes*. Australian Antarctic Division, Tasmania, 1990.

[18] E. Woehler and G. Johnstone. Status and conservation of the seabirds of the Australian Antarctic Territory. In J. Croxall, editor, *Seabird status and conservation: a supplement*, pages 279–308. ICBP Cambridge, 1991.

[19] E. Woehler, B. Raymond, and D. Watts. Decadal-scale seabird assemblages in Prydz Bay, East Antarctica. *Mar Ecol Prog Ser (submitted)*, 2002.

[20] J. Wolfe. Pattern clustering by multivariate mixture analysis. *Multiv Be R*, 5:329–350, 1970.

Combining Data Mining and Artificial Neural Networks for Decision Support

Sérgio Viademonte
School of Information Management and Systems
Monash University
PO 197 Caulfield East
3145 Victoria, Australia
sergio.viademonte@sims.monash.edu.au

Frada Burstein
School of Information Management and Systems
Monash University
PO 197 Caulfield East
3145 Victoria, Australia
frada.burstein@sims.monash.edu.au

ABSTRACT

This paper describes an ongoing research project concerned with the application of data mining (DM) in the context of Decision Support. Specifically, this project combines data mining and artificial neural networks (ANN) in a computational model for decision support. Data mining is applied to automatically induce expert knowledge from the historical data and incorporate it into the decision model. The resulting knowledge is represented as sets of knowledge rule bases. An ANN model is introduced to implement learning and reasoning within the proposed computational model. The proposed computational model is applied in the domain of aviation weather forecasting. The paper describes the proposed decision support model, introduces the data pre-processing activities and the data mining approach, the data models used to generate the knowledge rule bases, and their integration with the ANN system. The paper presents evaluation of the performance of the proposed approach and some discussion of further directions in this research.

Keywords:

Artificial neural networks, data mining, decision support, forecasting

1. INTRODUCTION

This paper presents a computational model for decision support based on a combination of data mining and artificial neural network technologies. The proposed computational model has been applied in the domain of aviation weather forecasting, specifically, identifying fog phenomenon at airport terminals.

Weather forecasts are based on a collection of weather observations describing the state of the atmosphere, such as precipitation levels, wind direction and velocity, dew point depression, etc [1, 10]. Access to the past decision situations and knowledge derived from them can provide valuable source of improvement in forecasting rare events, such as fog. Complexity and diversity of the weather observations and large variation in the patterns of weather phenomenon occurrences implies serious problems for forecasters trying to come up with correlation models. Consequently, the area is a potential candidate for KDD purposes [21].

Computational tools for decision support usually incorporate expert knowledge of domain experts together with specific explicit domain knowledge, e.g., factual knowledge. Early attempts in building expert systems revealed the difficulties of capture, represent and incorporate expert knowledge in those systems [4, 19]. One possible approach for the knowledge acquisition problem is to automatically induce expert knowledge directly from raw data [8]. However, this approach brings additional problems as the amount and diversity of data increases and demands specific attention. In this research project, data mining technology is applied to build knowledge from data, specifically inducing domain knowledge from raw data and also ensuring data quality.

In the context of this project, the preprocessed sets of raw data used as input in the data mining algorithm are named *data models*. The knowledge obtained as a result of data mining experiments is termed knowledge models. An artificial neural network (ANN) system provides an interface for the user decision-makers to test and validate hypotheses about the specific application domain. The ANN system learns about the problem domain through the knowledge models, used as training sets.

Section 2 presents the proposed computational model for decision support; section 3 discusses the knowledge discovery process, specifically the data mining phase, and the data and knowledge models. Section 4 presents the applied artificial neural network system. Section 5 discusses the decision support model evaluation and some achieved results, section 6 presents some comments and conclusions.

2. A MODEL FOR DECISION SUPPORT BASED ON DM AND ANN

The purpose of the proposed computational model is to support decision-making by recalling past facts and decisions, hence inducing "chunks" of domain knowledge from past information and performing reasoning upon this knowledge in order to verify hypotheses and reach conclusions in a given situation. The proposed model creates an interactive software environment that uses data mining technology to automatically induce domain knowledge from historical raw data, and an ANN based system as a core for an advisory mechanism (see Figure 1).

The decision support model comprises a database (ideally a datawarehouse), case bases and knowledge rule bases. The database contains raw data from the application domain, in the case of this research project, historical weather observations. The case base contains selected instances of relevant cases from the specific problem at hand. In this project, each case represents a past occurrence and consists of a set of feature/value pairs and a class in which the case belongs. In this research project several case bases were generated and used as input data in the data mining algorithm. The case bases are named *mining data sets* in this research project [20].

Knowledge rule bases are built based on the data mining results; they contain structured knowledge that corresponds to relevant relations found (*mined*) in the case bases. Several rule bases

were generated; according to different parameters used for data mining, e.g., distinct confidence factors and rule support degrees.

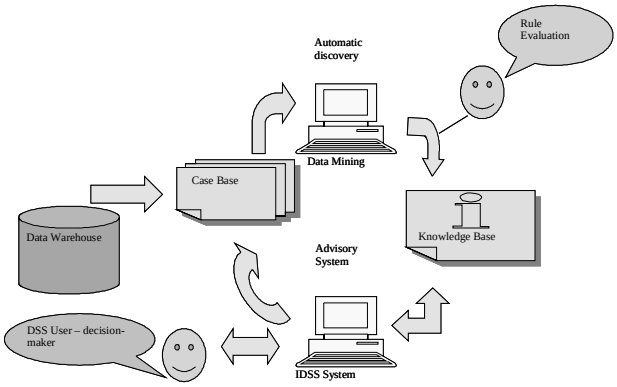


Figure 1 – A computational model for decision support (as described in [21]).

The ANN mechanism is applied to process the obtained knowledge (rule bases). The ANN uses the content of the knowledge bases as learning data source, to build knowledge about the specific application domain through its learning algorithm [11, 13]. After the ANN-based learning procedure has been executed, the advisory system provides an interface through its consult mode for the user to test and validate hypotheses about the current decision situation.

3. INDUCING KNOWLEDGE THROUGH DATA MINING

The database of weather observations used for automatic induction of domain knowledge was generated from Australian Data Archive for Meteorology (ADAM) data repository. It contains weather observations from Tullemarine Airport, Melbourne, Australia, from July 1970 until June 2000, and has 49,901 records and 17 attributes.

3.1 Data preprocessing

The initial database had many problems concerning data quality issues, such as the significant amount of empty and missing values, sparse variables and problems with variability. An extensive pre-processing work was required to make the data appropriate for KDD process, see [20] for detailed discussion about this subject.

The first database used in this project had 75 attributes, some of them related with data quality control and codes describing observations. For example, the *dew point* observation had two associated attributes, one named *DWPT* and other named *DWPT_QUAL*, this last attributes indicates a data quality control information, which was not relevant for our data mining purposes. Many other observations (attributes in the database), like *wind speed*, *wind direction* and *visibility* presented the same problem and had to be removed. The *Year* and *Day* attributes were not necessary for mining purposes, just the *Month* attribute. A derived attribute *previous afternoon dew point* was calculated based in the *date*, *hour* and the *dew point* and inserted in the table. The forecasters recommended this information as very important for fog prognosis.

Several data transformations were performed with Bureau original data set. For example, *Fog Type* attribute has 3 possible values assigned: "*F*" when it refers to a fog event, "*LF*" when is "Local Fog" and null when the event is not fog. This study is concerned with occurrence of fog phenomenon regardless of whether it is a local fog or not. For this reason all the *Fog Type* instances with "*LF*" value were transformed in "*F*" value, meaning a *fog* case. All the instances of *Fog Type* with null value were assigned "*NF*" values, meaning *not fog* case.

The attributes *PastWeather* and *PresentWeather* were transformed from numeric type to non-numeric type. These attributes are qualitative (categorical) attributes; which indicate weather codes.

The *Rainfall* attribute shows two problematic behaviors for data mining: sparsity and lack of variability. It has 21.11 % of null values in *fog class* population and 30.60 % of null values in *not fog class* population. The rainfall volume is initially measured in millimeters and presented to the forecasters; who express their evaluation in codes expressing ranges of millimeters. This procedure makes sense according to the nature of the forecasting task, as it is almost impossible to differentiate precise measurements of rainfall, like 0.3 millimeters and 0.2 millimeters. The numerical values were transformed into categorical codes, which express ranges of rainfall. The instances of null rainfall will be classified into code 0, no rain. To implement this transformation a new attribute was inserted, *Rainfall Range* text attribute. A procedure was implemented to calculate the *Rainfall Range* attribute corresponding to Rainfall attribute values.

The *Wind Direction* is a measure taken by instruments and it is numerically represented in degrees. However, the forecasters do not use detailed numerical measurements when reporting a forecast bulletin but a categorical representation. A categorical description of compass point is used instead, being N for North, S for South and so on. For example, lets consider the compass point 22,1 degrees. Practically, in that case the forecasters assume the compass point NNE, which is the point 22.5 degree. In that case the wind direction is said to be NNE instead of 22.1, as for forecasting purposes the distinction among 22.1, 22.2 and 22.5 degrees is not significant; again a tolerance for imprecision can be observed. It is assumed that each value in degrees belongs to the closest compass point, therefore the middle point between each two compass points was chosen as the boundary point between them, being the middle point itself belonging to the next upper compass point. To implement the transformation of *Wind Direction* attribute from degrees to compass points a new attribute was inserted into the data set, the *WindCompas*, text attribute.

The *not fog class* data set initially had 48.963 instances. After all data transformations and after the nulls instances were removed, the resulted *not fog class* data set has 47,995 instances, and the *fog class* data set has 938 instances. This database was used to select the relevant cases for data mining.

3.2 Generating the data models

The next step was devoted to verify the data dimensionality and class distribution in the database. As we are interested in forecasting fog, the population was discretised into two classes. One class representing fog cases (named *fog class*), and a second class representing cases where fog was not observed (named *not fog class*). The observation database shows a low-prevalence classification, it means that far fewer cases of *fog class* were present comparing to *not fog class*.

The dataset has 938 instances of *fog* class and 47,995 instances of *not fog* class. Figure 2, below, shows the fog classes distribution in the entire weather observations database (population) after data preprocessing. *Fog* class represents 1.92% of the population, and *not fog* class represents 98.08% of the population.

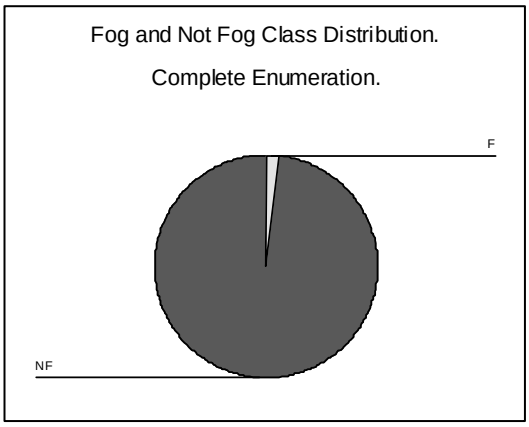


Figure 2 – Fog class distribution in the population

This significant difference in class distribution required the development of a specific sampling strategy in order to have a more homogeneous class distribution in the training set [7, 15, 18]. The sampling approach used in this research project can be classified as stratified multi-stage sampling [7, 23]. The original population was divided in two strata: *fog stratum* and *not fog stratum*. Sampling was separately conducted in stages within each stratum. A random sampling approach was used for fog stratum. Fog stratum was *randomly split without replacement* in 85% for mining data set and 7% for testing and evaluation, respectively.

Not fog stratum was sampled in a different fashion; increased sizes data sets were selected from the whole stratum in 10%, 20% and 100% proportions. The sample being 10% of the whole stratum was named *Model1*, with 4,763 instances. The second sample named *Model2*, 20% of the stratum with 9,572 instances. For comparison purposes the whole stratum was also considered, we call it *Model10*, meaning 100% of the stratum.

The 10%, 20% and 100% percentages were arbitrarily selected based in the size of not fog and fog strata; the aim here is build data models without a significant difference between the numbers of instances from each class. Therefore small percentages were chosen from not fog stratum. In addition, literature review provide useful insights in incremental sampling, according to Weiss and Indurkhya [23] typical subset percentages for incremental sampling might be 10%, 20%, 33%, 50%, 67% and 100%. Using 50% and higher percentages will keep the difference between not fog cases and fog case too big, therefore small percentages were chosen. The 100% subset was selected to verify the mining algorithm performance when using a significantly difference class distribution, the assumption was that this subset will produce very few or either none fog cases rules.

Table 1, below, shows the generated not fog stratum models:

Table 1. Sample models for *not fog* class

Data Model	Sample size	Percentage of the whole stratum
Model1	4.763	10%
Model2	9.572	20%
Model10	47.995	100%

These generated models are called *data models* in the context of this research project, e.g. Model1, Model2 and Model10 are data models. From each data model, three data sets were randomly generated; a *mining* data set (used by the data mining algorithm), an *evaluation* data set (for comparison purposes) and a *test* data set (used as test data by the neural network system). Those data sets were randomly sampled from their original data models in 60% and 80% proportions for mining sets and 10% proportions for evaluation and test.

The final data models are obtained by joining *fog* data sets with *not fog* data sets. Four mining data sets were obtained in this way, named by corresponding sampling proportions: *Mining Model 1-60, Mining Model 1-80, Mining Model 2-60, and Mining Model 10-60*.

For example, *Mining Model 1-60* means that this model was obtained by a sample of 10% out of the overall *not fog* stratum, and 60% of this sample was selected for data mining purposes. The other names follow the same structure.

3.3 Applying data mining

This research project uses an associative rules generator algorithm for data mining, based on AIS algorithm [2]. An association rule is an expression X →Y, where X and Y are sets of predicates; where X is a precondition of the rule in disjunctive normal form and Y - the target post condition. Hence, the outputs of the data mining experiments are associative rules. We chose associative rules to represent the induced knowledge because it is a clear and natural way of knowledge representation that is easy for people to understand; and also because it fits well our neural network system. Section 4 addresses the integration issues between the knowledge models and the ANN system.

This section discusses some procedures that had to be performed for data mining, e.g.: features selection, selection of target attributes and attributes' values in the database, discretization or clustering of attributes, and the selection of mining parameters. Although a detailed description of these procedures is out of the scope of this paper, we believe that it is important to mention them at least briefly. Nearly every data mining project includes the execution of these procedures at a certain level. If incorrectly performed, these procedures can potentially compromise the success of the entire data mining project. For those reasons we decided briefly discuss in this paper some of those procedures we faced in our research project.

Selection of a target attribute procedure requires the selection of an attribute from the case bases that discriminates the class in study, in our project is the attributes that indicates if a particular case corresponds to a fog observation or not; this attribute is named *FogType*. *FogType* attribute was discretised into two values, "F" and "NF"; this represents whether fog phenomenon was or not observed, respectively.

Features selection, means selecting the attributes that form the antecedent part of the rules. In our research almost all attributes were selected for data mining, exceptions were the attributes indicating *Year* and *Day*. Besides that, there is an attribute in our database that indicates the visibility over the airport runaway. Experiments with and without *Visibility* attribute were performed to check whether the *Visibility* attribute might be considered as a synonymous for fog. The assumption was to verify the amount of generated rules in both cases and the prevalence of the *Visibility* attribute.

Selection of attributes values is an important procedure, which addresses dimensionality reduction, together with feature and cases selection. It could happen that some sets of attribute values are not relevant to the survey variable, or have a small frequency of occurrence in the database. In both cases such

attribute values do not add any valuable information in data mining and may be removed.

In our experiment some values or categories of *Hour*, *Rainfall* and *Month* attributes were excluded from the data mining experiments because they had either a small frequency in the database, or because they had a high frequency for either classes, *fog* and *not fog*. It means high sensitivity but low specificity. Sensitivity degree of a finding is defined in relation to a class measures its frequency for that class. Specificity degree of a finding F in relation to a class C, on the other hand, is inversely proportional to the frequency with which the finding F appears in classes other than C

Configuration of mining parameters includes selection of the minimum desired *level of rule confidence*, *support degree* and the *maximum rule order*. It means to choose the ratio of the number of records in the database that support a particular rule. The *maximum rule order* parameter sets the maximum number of antecedents of the rules. For example, in a rule like:

If DRYBULB <= 8.5 And TOTALCLO > 7 And TOTALLOW > 6 And WINDSPEE <= 1.5

Then FOGTYPE = F, Confidence: 88.24%, Support: 9.29%

The rule order is 4, represented by the attributes dry bulb temperature (Drybulb), amount of clouds over the airport runaway (Totalclo), amount of low clouds over the airport runaway (Totallow) and the wind speed (Windspee) at the airport runaway.

In most of the data mining applications the users are usually only interested in rules with support and confidence above some minimum threshold. Thus these parameters are important to be set. Table 2 shows the selected mining parameters in our experiments:

Table 2. Selected mining parameters

Mining Parameter	Value
Confidence Degree	50%, 70%, 80%, 90%
Minimum Support Degree	8%, 6%
Minimum Number of Cases	50
Maximum Rule Order	7, 10

The data mining experiments generated rules with 70%, 80% and 90% confidence degree. As it was impossible to know beforehand the amount of generated rules accordingly with a specific confidence degree, it was decided to use the most frequent percentages in data mining applications [16, 22]. Our goal here is to verify if there is a significant difference in performance accordingly with different combinations of parameters (confidence degree, minimum support degree and maximum order degree). And if so, which combination(s) of these parameters is (are) most appropriate when applying data mining in problems similar to the one we are addressing in this project. Here, we consider as performance measure the amount of rules obtained in each class, together with the amount of item sets in each rule. In general the descriptive capability of a rule is associate with its amount of item sets.

Two sets of data mining experiments were performed. One set of experiments using minimum rule support of 8% and maximum rule order of 7. A second set of experiments, using minimum rule support of 6% and maximum rule order of 10. The first experiments resulted in more restrictive models.

The obtained amount of rules, specifically for fog class, was considered small for a good descriptive model. Hence, it was decided to execute the experiments again with more flexible parameters. Table 3 illustrates this fact; it summarises the amount of associative rules obtained from the mining set *Model1-6* and *Model1-8*. When using 70% rule confidence degree, minimum rule support of 8% and maximum rule order of 7 was obtained 240 associative rules, being 54 in fog class for Model1_6, and 245 associative rules, being 35 in fog class for Model1_8. Keeping the 70% rule confidence degree and changing the minimum rule support to 6% and maximum rule order to 10 we obtained 405 associative rules, being 104 in fog class for Model1_6 and 358 associative rules with 67 in fog class for Model1_8. An increase of 50 rules can be observed in fog class for Model1_6 and 32 rules in fog class for Model1_8. The minimum number of cases, 50, remained constant in all experiments, because it was considered a satisfactory amount of cases, not very restrictive but big enough for a good coverage.

Discretization of numerical attributes is used to determine the granularity of a certain variable. It can be used in general to simplify the data mining problem. Also, most data mining tools and algorithms, mainly those used in classification problems, require discrete values rather than a range of values [9].

Table 3. Mining Model1-6 and Model1-8 with different mining settings

Mining set	Number of rules	Confidence degree	Rule support	Maximum rule order
Model1-6	240	70%	8%	7
Model1-6	405	70%	6%	10
Model1-8	215	70%	8%	7
Model1-8	358	70%	6%	10

Table 4 illustrates three attributes discretization in our experiment. It shows their respective assigned categorical classification in the Data Mining Model1-6. Each attribute has been assigned the same categories in all data models, e.g. *Low*, *Med*, *High* for Dry Bulb. But distinct value ranges occurred in different data models.

Table 4. Discretisation of numerical attributes

Attribute	Mining Model 1-6 Ranges	Categories
Dry Bulb (Celsius degrees)	< = 8.5 > 8.5 and <= 12 > 12	Low Med High
Total Cloud Amount (Eighths)	< = 4 > 4 and < = 7 > 7	Min Med Max
Wind Speed (meters/second)	< = 1.5 > 1.5 and < = 3.6 > 3.6 and < = 6.2 > 6.2	Light Lmode Mode Fmode

The discretization of a particular attribute is measure proportional on the total amount of cases in the database and the frequency of occurrence of each attribute value. Categorical attributes already express a discrete value, however numerical attributes must be discretised in ranges. The used data mining tool automatically discretizes the numerical attributes based on their frequency of occurrence and the amount of their categories.

3.4 Generating knowledge models

Knowledge discovery in databases constitute an interactive and iterative process, having many steps and interrelated fields. We consider knowledge modeling as an important part of the knowledge discovery process. In our research project we distinguish domain modeling, data modeling and knowledge modeling from each other.

We understand domain modeling in the same way as it has been widely used by decision support, expert systems and artificial intelligence community in general. Basically, it is concerned with building a model of a particular domain under investigation for any particular purpose. Data modeling in the context of our project relates to all the activities that transform raw data into the data used for data mining. Such data modeling includes data pre-processing, features selection, reduction and transformation, and data sampling. Knowledge modeling in our context includes the activities related to extracting knowledge from data. This includes the interactive process of mining data, testing and tuning different data mining parameters and data models, e.g., adding or eliminating data features, and even cases. In fact, it is effectively an interactive and iterative process, where we try to build descriptive models as comprehensive as possible for our application domain. The ultimate goal of such a knowledge modeling process is to achieve a good predictive performance of the decision support model. It includes a performance evaluation of the decision support model that will demonstrate how efficient is the descriptive model for this particular case.

The above definitions are important for better understanding of how we generated knowledge models and what they are. The approach we used in this research project to generate knowledge is based on the data models, a data mining algorithm (our descriptive method) and the choice of data mining settings (rule confidence degree, rule support and maximum rule order). For each original data models, and combinations of mining parameters, we obtained a distinct set of associative rules. Not only the amount of rules are different, but also the rules itemsets. Each of these distinct sets of associative rules is identified as a knowledge model, or a knowledge base.

To illustrate our approach, let us consider the data mining Model1-6 with 70% confidence degree, minimum rule support of 6% and maximum rule order of 10. After mining this data set, it generated a particular knowledge base. Similarly, the data mining Model1-6, with 80% confidence degree, minimum rule support of 6% and maximum rule order of 10 generated a different knowledge base. The process follows in this fashion until we execute all data mining sets (case bases) with the selected mining parameters; showed in table 2, sub section 3.3.

Table 5 below shows the generated knowledge models, for each data model accordingly with their respective levels of confidence degree.

Table 5. Generated knowledge models

Mining Data Models	Knowledge Models Generated Rules by Rule Confidence Degree								
	70 %			80%			90%		
	F	NF	Total	F	NF	Total	F	NF	Total
Mining Model1-6V3	104	301	405	37	291	328	16	204	220
Mining Model1-8V3	67	291	358	23	291	314	12	228	240
Mining Model2-6V3	45	283	328	20	283	303	12	274	286
Mining Model10-6V3	10	279	289	9	279	288	9	279	288

This table refers to the experiment using minimum rule support of 6% and maximum rule order of 10; this information is identified by the prefix “V3” in each mining data model. In Table 5, ‘F” relates to fog class and ‘NF” to not fog class. Rules with 50% confidence degree were also generated using the mining Model10-6, due to space limitations it is not included in table 5, but this does not compromise the understandability of the proposed approach.

For the confidence level of 90% too few rules were obtained for fog class; with a maximum amount of 16 rules when using data model Model1-6 and 12 rules when using data Model2-6. These amounts of rules are unlikely to be enough for a satisfactory description of the fog phenomenon. The performance evaluation will show how this affects the predictive capacity of our decision support model.

4. THE ARTIFICIAL NEURAL NETWORKS SYSTEM

We applied an ANN system as the interface of our decision model. The ANN system learns about the problem domain through the knowledge models, used as training sets. Besides implementing learning capability in our decision support model, the ANN system provides an interface for the decision-makers to test and validate hypotheses about the specific application domain.

For the ANN interface we use the Components for Artificial Neural Networks (CANN) framework [4]. The CANN framework is a research project that allows neural networks to be constructed on a component basis. The CANN project relates to the design and implementation aspects of framework architecture for decision support systems that rely on artificial neural network technology [17].

The CANN components are designed in an object-oriented way. It implements a class hierarchy to represent a particular application domain, the domain evidences, classes and the relationships among evidences. Figure 3 presents the screen of the CANN system that represent the evidences (attributes) used to identify fog phenomenon. At the left in figure 3 a list of evidences (attributes) about weather forecasting is presented. We selected the evidence *Dewpoint* to show its properties, for example, it is a string attribute and it is categorized in four categories: *Low*, when the temperature is smaller or equal 4 Celsius degrees; *Med* between 4 and 6 Celsius; *High* between 6 and 9 Celsius and *Max*, temperature higher than 9 Celsius.

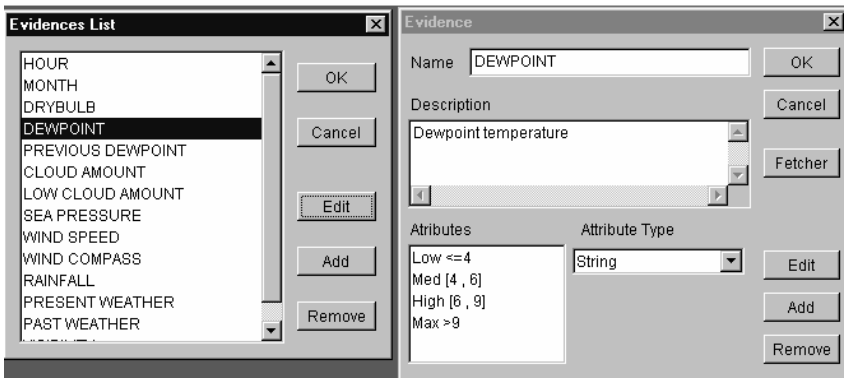


Figure 3: Weather evidences modelled into CANN

CANN system implements a mechanism that associates a data set with a particular ANN model, for example, the *Combinatorial Neural Model (CNM)* [13]. Through the ANN learning algorithm, CANN implements a learning mechanism. Figure 4 illustrates the outcome of the learning process executed by CANN in the meteorological domain.

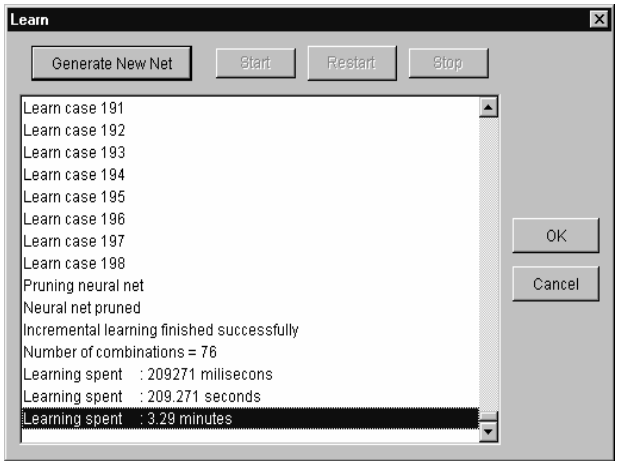


Figure 4: Learning about meteorological domain.

CANN functionality is well suited to the purpose of our project as it is capable of a flexible domain representation, learning and consulting functionalities. A decision-maker interacts with CANN consulting mechanism, in two ways: *case consult* and *case base consult*. A case consult presents to the decision maker a selection of evidences, and their respective evaluation of relevance to the situation at hand. CANN sets up a set of hypotheses based on the presented input data. It evaluates the selected evidences and calculates a confidence degree for each hypothesis. The inference mechanism appoints the hypothesis with the higher confidence degree as the most suitable solution (class) to the problem.

A case base consult is similar to the case consult, however, instead of presenting one single case (or one set of evidences) each time, several cases are simultaneously presented to the ANN system. It evaluates the set of cases in the same way it does for a single case. Figure 5 illustrates a case base consult session.

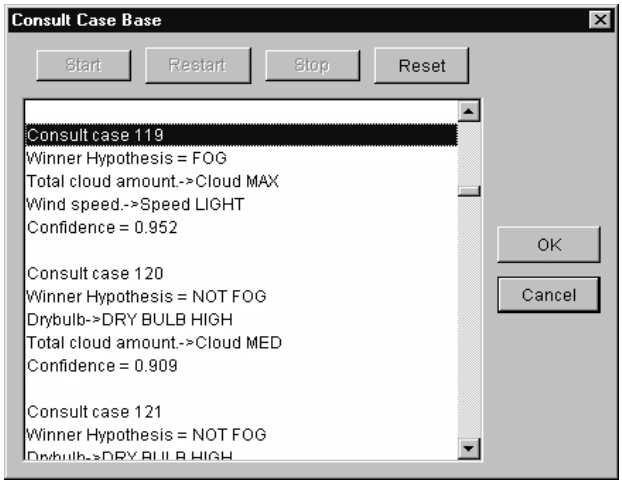


Figure 5: A case base consult session.

For example, case 119 in figure 5 is indicated as a Fog case with confidence degree of 0.952, supported by the evidences *Total cloud amount Max* and *Wind Speed Light*. Case 120 is indicated as a Not Fog case with confidence degree 0.909, supported by the evidences *Drybulb High* and *Total Cloud Amount Med*.

A detailed discussion of the CANN class hierarchy is outside the purposes of this paper as is software engineering design issues. Readers interested in these subjects should refer to [3, 4]. The Combinatorial Neural Model, its algorithm and its learning, pruning and consulting algorithms are presented in [12, 13].

4.1 Mapping associative rules into the ANN topology

The CANN knowledge representation schema reflects the knowledge model structure and content. The rules are directly mapped onto the ANN topology, and simultaneously represented through a symbolic mechanism [14]. Rules describing relations in the weather forecasting domain are represented by neurons and synapses. Figure 6 exemplifies this property. The rule: I3 & I4 & In => F, corresponds to the

strengthened connections among the input nodes I3, I4 and In, the combinatorial node C3, and the output node F of the ANN.

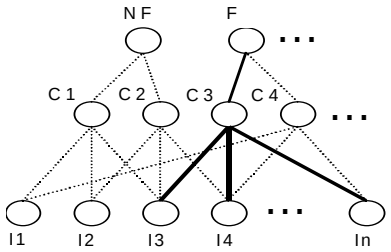


Figure 6. Incorporating rules into ANN topology.

For example, consider the following rule:

```
Rule 1 for Fog Class:
If Total Cloud Amount = Max
       And Wind Speed = Light
       And Wind Direction = SE
Then Fog Type = F.
```

The above rule is mapped into the ANN topology by representing I3 as t*otal cloud amount max*, I4 as *wind speed light* and In *wind direction SE*, and also considering the hypothesis NF as not fog case and F as fog case. Additional information as rule confidence degree will be represented in the ANN topology as confidence level associated with a particular evidence.

5. VALIDATION

The validation of the discovered knowledge is based on the ability of the model to correctly identify meteorological observations, specifically a fog case or a not fog case. The performance of the model relies not only on the applied computation technologies (data mining and ANN), but also on the strategy we applied to obtain the data and knowledge models, e.g., sampling strategy, pre-processing, and mining parameters. Due to the space limitations we cannot discuss all these issues in this paper, but it is important to understand that all those issues have an implication on the performance of our decision support model.

We selected data model Model1-6, with 70%, 80% and 90% rule confidence degrees, 6% of minimum rule support and maximum rule order 10. We identify each case set by adding the rule confidence degree in the data model name, therefore Model1-6-70 corresponds to the data model generated when selecting 70% rule confidence degree; Model1-6-80 using 80% rule confidence degree and Model1-6-90 when using 90% rule confidence degree.

Table 5, in section 3.4, describes the amount of rules for each of these data models. They were used as the ANN learning bases.

For testing we used a subset of the test set generated for data model Model1-6. The test set has randomly selected 120 cases, being 60 cases of not fog and 60 cases of fog.

The results of this experiment are presented in Table 6. They appoint to the efficiency and applicability of the combined approach, data mining and ANN, considering an average of 67.5% of correct classifications.

The ANN system correctly classified 68.3% of the cases when training with rules obtained with 70% of confidence degree. The performance decreased a little, to 67.5% when training with rules obtained with 80% of confidence degree; and the performance decreased to 66.67% when training with rules obtained with 90% of confidence degree. Those results are not a surprise, as increasing the rule confidence degree restricts the amount of obtained rules; therefore a less descriptive model is expected. The worse performance is verified when applying Model1-6-90, it happens because this data model has only 16 rules describing fog what does not represent a enough coverage to describe fog phenomena. Even though, 66.67% of correct classifications can be considered a surprisingly good result considering there are only 16 rules about fog in the rule base.

Table 6. Test Data Model1_6 with different rule confidence degrees

Learning Set	Correct	Misclassified	No conclusion	Total Cases
Model1-6-70	82 (68.30%)	28 (23.3%)	10 (8.3%)	120
Model1-6-80	81 (67.50%)	27 (22.5%)	12 (10.0%)	120
Model1-6-90	80 (66.67%)	25 (20.8%)	15 (12.50%)	120

An average of 67.5% of the cases were correctly classified, what indicates the applicability of the proposed model for decision support in classificatory problems.

Table 7. Data Model1_6 performace discriminating fog and not fog classes

Learning Set	Correct Fog	Correct Not Fog
Model1-6-70	42 (70.0%)	40 (66.67%)
Model1-6-80	39 (65.0%)	42 (70%)
Model1-6-90	39 (65.0%)	41 (68.33%)

Analyzing individually the performance in each class also indicates that the 70% rule confidence degree generates the best set of rules, achieving the highest performance of 70.0% of correct fog cases classified. What basically differentiates each of the training models in our experiment is the number of rules representing *fog cases*.

The change in the number of rules representing *not fog* cases does not represent a significant change in performance, specifically 70.0% in the best case and 66.67% in the worse case. It is because there are enough rules describing *not fog* cases. The same comments cannot be extended to fog class; a decrease in *fog* rules in Model1-6-90 caused a significant lost in predictive performance, with 70.0% of correct classification in the best performance dropping to 65.0%.

Our experiment so far indicates that the 70% rule confidence degree seems to be the best value for this parameter, even when faced with the problem of low prevalence classification. However 70.0% may not be considered a satisfactory performance in many applications. Additional experiments can be carried on to improve the system performance, for example applying different sampling proportions to obtain a more homogeneous class distribution, or applying different data mining parameters. Such as relaxing the minimum rule support to obtain a higher number of rules or even increasing the maximum rule order to generate rules with higher itemsets, therefore better descriptive capabilities.

Further experiments are necessary for more accurate conclusions; however, the results obtained so far indicate the potential applicability of our approach to automatically induce domain knowledge, to handle the problem of low prevalence classification in databases, to incorporate the domain knowledge and implement learning capabilities in the proposed model for decision support.

6. CONCLUSION AND COMMENTS

This paper presents a decision support model and its application to a real world problem. We proposed a decision support model combining data mining and neural networks. Data mining is chosen to automatically induce domain knowledge from raw data and ANN because of its adaptive capabilities, which is important for providing the means for implementation of inductive and deductive learning capabilities [6, 19]. Besides that, this project came up with an efficient sampling strategy to handle problems of dimensionality and class distribution, mainly the low prevalence classification problem, as well as conducted an in-depth investigation of the pre-processing stage to ensure data quality for data mining.

The results obtained so far demonstrate the applicability of the proposed decision support model in aviation weather forecasting, specifically to correct identify fog phenomenon.

The system performance can be further improved through some additional procedures. For example, in our experiments we used neural network topology with maximum order of three. It means that the neural network combinatorial layer associates at maximum three input neurons. Using higher combinatorial order will add more evidences in the neural network learning and evaluation procedures. Considering more evidences for the cases analysis can potentially improve the system performance. Additionally, considering higher number of antecedent itemsets during data mining and relaxing the learning and pruning threshold parameters in the ANN learning algorithm may also potentially improve performance.

In addition, issues concerning system integration may be assessed. Currently case and knowledge bases are stored as relational tables; different technologies are under evaluation for storing the knowledge bases, for example using XML document formats and PMML (http://www.dmg.org), in order to facilitate its integration with the ANN system, based on Java implementation.

7. ACKNOWLEDGEMENTS

This research is partly funded by the Australian Research Council and Monash University grants. We would like to thank the Regional Forecasting Centre from Australian Bureau of Meteorology, Victorian Regional Office for providing meteorological data and support. We also thank Dr Robert Dahni and Mr. Scott Williams from the Regional Forecasting Centre for their help in validation results in relation to aviation weather forecast.

8. REFERENCES

[1] Auer, A. H. J. (1992). Guidelines for Forecasting Fog. Part 1: Theoretical Aspects: Meteorological Service of New Zealand.

[2] Agrawal, R., Imielinski, T., & Swami, A. (1993, May, 1993.). Mining association rules between sets of items in large databases. In Proceedings of Conference on Management of Data., (pp. 207-216). Washington, DC.

[3] Beckenkamp, F. a., & Pree, W. (1999). Neural Network Framework Components. In S. D. C. a. J. R. Fayad M. (Ed.), Object-Oriented Application Framework: Applications and Experiences. (1 ed.): John Wiley.

[4] Beckenkamp, F. a., & Pree, W. (2000, May, 2000.). Building Neural Networks Components. In Proceedings of Neural Computation 2000 - NC'2000, Berlin, Germany.

[5] Buchanan, B., & Feigenbaum, E. (1978). DENDRAL and META-DENDRAL: Their applications dimensions. Artificial Intelligence, 1, 5 - 24.

[6] Carbonell, J. G. (1989, September). Introduction: Paradigms for Machine Learning. Artificial Intelligence, 40, 1-9.

[7] Catlett, J. (1991). Megainduction: Machine learning on very large databases. University of Technology, Sydney, Australia.

[8] Fayyad, U. M., Mannila, H., & Ramakrishman, R. (1997). Data Mining and Knowledge Discovery. (Vol. 3). Boston: Kluwer Academic Publishers.

[9] Howard, C. M., & Rayward-Smith, V. J. (1998). Discovering Knowledge from low-quality meteorological databases. Knowledge Discovery and Data Mining. (Pages: 180-202.).

[10] Keith, R. (1991). Results And Recommendations Arising From An Investigation Into Forecasting Problems At Melbourne Airport. (Meteorological Note 195). Townsville: Bureau of Meteorology, Meteorological Office.

[11] Machado, R. J., Barbosa, V. C., & Neves, P. A. (1998). Learning in the Combinatorial Neural Model. IEEE Transactions on Neural Networks, 9. September, 1998

[12] Machado, R. J., & Rocha, A., F. (1989). Handling Knowledge in High Order Neural Networks: the Combinatorial Neural Model. (Technical Report CCR076). Rio de Janeiro, Brazil.: IBM Rio Scientific Center.

[13] Machado, R. J., & Rocha, A., F. (1990). The combinatorial neural network: a connectionist model for knowledge based systems. In B. B. Bouchon-Meunier, Yager, R. R. & Zadeh, L. A. (Ed.), Uncertainty in knowledge bases. Berlin, Springer Verlag.

[14] Medsker, L. R. (1995). Hybrid Intelligent Systems. (Vol. 1). Boston, USA: Kluwer Academic Publishers.

[15] Mohammed, J. Z., Parthasarathy S., &, L. W., & Ogihara, M. (1996.). Evaluation of Sampling for Data Mining of Association Rules. (Technical Report 617). Rochester, New York. The University of Rochester, Computer Science Dept.

[16] Piatetsky-Shapiro, G., & Frawley, W. (1991). Knowledge Discovery in Database. MIT Press.

[17] Pree, W., Beckenkamp, F. a., & Rosa, S. I. V. (1997, June, 17 - 20, 1997). Object-Oriented Design & Implementation of a Flexible Software Architecture for Decision Support Systems. In Proceedings of 9th. International Conference on Software Engineering & Knowledge Engineering - SEKE'97, (pp. 382 - 388). Madrid, Spain.

[18] Provost, F., Jensen, D. & Oates, T. (2001). Progressive Sampling. In H. L. a. H. Motoda (Ed.), Instance Selection and Construction for Data Mining (Vol. 1, pp. 151 - 170). Norwell, Massachusetts, USA: Kluwer Academic Publishers.

[19] Tecuci, G. a., & Kodratoff, Y. (1995). Machine Learning and Knowledge Acquisition: Integrated Approaches. London, UK.: Academic Press.

[20] Viademonte, S., Burstein, F., Dahni, R. & Williams, S. (2001). Discovering Knowledge from Meteorological Databases: A Meteorological Aviation Forecast Study. In Proceedings of Data Warehousing and Knowledge Discovery, Third International Conference - DaWaK 2001, (pp. 61-70). Munich, Germany: Springer-Verlag.

[21] Viademonte, S. B. & Burstein F.. (2001). An Intelligent Decision Support Model for Aviation Weather Forecasting. In Proceedings of Advances in intelligent data analysis: 4 th international conference / IDA 2001, (pp. 278 - 288). Cascais, Portugal.: Springer-Verlag.

[22] Weiss, S. M., Galen, R. S. a., & Tadepalli, P. V. (1990). Maximizing the predictive value of production rules. Artificial Intelligence, 47 - 71.

[23] Weiss, S. M., & Indurkhya, N. (1998). Predictive Data Mining: A Practical Guide. (Vol. 1). San Francisco, CA: Morgan Kaufmann Publishers, Inc.

About the authors:

Sérgio Viademonte is a Doctoral candidate at the School of Information Management and Systems at Monash University. His research is supported by ORSP and Monash Graduate Scholarships. Sergio has been working on hybrid architectures for expert systems since 1995 when he obtained a Master in Administration, Information Systems Area (by Research) from Federal University of Rio Grande do Sul (UFRGS), RS, Brazil.

Dr Frada Burstein is Associate Professor and Knowledge Management Academic Program Director at the School of Information Management and Systems at Monash University. She is a Chief Investigator for an ARC funded industry collaborative project with Bureau of Meteorology titled ”Improving Meteorological Forecasting Practice with Knowledge Management Systems”. The results reported in this paper address a component of this project.

Towards Anytime Anywhere Data Mining E-Services

Shonali Krishnaswamy
School of Network Computing
Monash University
McMahons Rd, Frankston, VIC 3199, Australia
Shonali.Krishnaswamy@infotech.monash.edu.au

Seng Wai Loke and Arkady Zaslavsky
School of Computer Science & Software Engineering
Monash University
900 Dandenong Road, Caulfield, VIC 3145, Australia
{swloke, arkady.zaslavsky}@csse.monash.edu.au

ABSTRACT

The term *mobile data mining* is becoming prevalent as a consequence of research and development in mining data streams from mobile devices. On another strand of development Application Service Providers (ASP) hosting Internet-based data mining services is being seen as a viable alternative for organisations that value their knowledge resources but are constrained by the high cost of data mining software. This paper advocates the paradigm of *anytime anywhere data mining services* to facilitate delivery of data mining as a service in pervasive environments characterised by ad-hoc task requests and the need for results anytime anywhere. The proliferation of handheld devices and wireless technologies pose new challenges and provide new opportunities for the analysis of data and the delivery of results. We review the current state-of-the-art in data mining ASPs in the commercial domain and discuss our research in enabling a virtual community of data mining e-services. We evaluate the benefits and discuss the issues and challenges in enabling anytime anywhere data mining services.

Keywords

Data Mining Services, Mobile Devices, Pervasive and Ubiquitous Environments

1. INTRODUCTION

Grossman [Gro98] defined the generations of data mining systems starting from the earliest pioneering efforts to the current state-of-the-art and future trends as follows:

- *First Generation Systems* view data mining as a stand-alone application, implemented a single algorithm (or a small collection of algorithms), executed data mining on a single machine and supported vector data.
- *Second Generation Systems* provide integration with databases and data warehouses, support for mining larger datasets and higher dimensionality, incorporation of predictive modelling capabilities and have limited ability to deal with complex data.
- *Third Generation Systems* are characterised by their ability to mine heterogeneous and distributed data sources in a transparent fashion. Agent systems and distributed technologies such as CORBA have been identified as a popular (or enabling) technology for distributed data mining.
- *Fourth Generation Systems* should have the ability to mine "embedded, mobile and ubiquitous" data sources.

It would be pertinent to say that fourth generation data mining systems are no longer an abstract yet-to-emerge concept of the future, but are slowly starting to be a current reality. The widespread use and proliferation of handheld devices and wireless networks and the subsequent emergence of pervasive and ubiquitous environments is leading to active research in *mobile data mining* and the development of systems such as MobiMine [KPP02].

Mobile data mining refers to the spectrum of tasks pertaining to the mining and analysis of data streams emanating from mobile devices [KPP02, GGR02]. We now discuss some typical scenarios and applications that require mobile data mining:

Analysis of stock-market data. Consider a stock-broker who is buying and selling shares using a PDA. The broker should receive incremental analysis of the financial markets to facilitate buying and selling shares incrementally [KPP02] and should be able to post queries in an ad-hoc manner.

Customer profiling support for a mobile sales-person. This is the scenario of a mobile sales person who needs to analyse a customer's likelihood of defaulting on credit. The sales person requires on-line, real-time mining (even if results are not immediate, the query can be issued at the moment the need is conceived) and support for delivery of results to his/her mobile device to help the decision process on the fly.

Delivering personalised recommendations to a "smart" shopper. This scenario pertains to an individual shopper who requires dynamic analysis of customised reports of sales and special offers for a particular item to be delivered to his/her PDA as and when such offers become available. In a more advanced situation, the shopper might wish to be informed of *likely* special offers that will soon be released.

The three scenarios outlined above have several commonalities. These include delivering results to mobile users resource constrained devices with small screens in real-time (or even if results are not immediate from the time of query, the results can be delivered wherever the user is), supporting ad-hoc queries (with appropriate support and leveraging on a whole community of data mining service providers) and they focus on the specific needs of individuals and small entrepreneurial entities. The stock-broker may be an individual running a small business, the shopper has a personal requirement and while the mobile salesperson may be a representative of a multi-national but can also be operating on their own. Data mining has typically been the prerogative of large enterprises that can afford it, however, the growing market segment of business intelligence service providers [Har02, KZL01b] has the potential to bring about a significant change in the perception of and usage of data mining. This paper proposes the paradigm of *anytime anywhere data mining services,* which refers to data mining services that are accessible from a variety of

locations and devices, support ad-hoc queries and task requests and provide results anywhere anytime (and when possible, in almost real-time) to several devices. It aims to provide transparent and seamless access to data mining services hiding the complexity of the process. Finally it propagates the paradigm of service subscription by individuals and small businesses that neither have the resources nor the need to have a data mining infrastructure in place. However these individuals and small businesses stand to benefit from the results of data mining and may be willing to obtain these benefits for a fee paid to a data mining service provider. The concept of anytime anywhere data mining services provides new opportunities and brings with it new challenges for the data mining community. This paper reviews the current status of data mining e-services including our work and discusses some of the challenges and issues that need to be addressed to realise the *anytime anywhere data mining services.*

2. DATA MINING E-SERVICES: CURRENT STATE-OF-THE-ART

The synergy between data mining and electronic commerce has been evident from the increasing number of applications for data mining in the e-commerce domain including customer profiling to support Customer Relationship Management [CRM], supermarket purchasing by making personalised recommendations on hand held devices [LAK01] and to measure the success of e-commerce sites [SpP01]. In fact, it has been recognised that electronic commerce applications have a very high potential to illustrate the practical applicability and benefits of data mining techniques [KoP01]. While the application of data mining techniques to e-commerce represents one strand of the relationship between the two areas, the delivery of data mining as an e-service is another emerging focus. Umesh Dayal [Day01] predicted that "*...data analysis and mining functions themselves will be offered as business intelligence e-services that accept operational data from clients and return models or rules*". The growing number of Application Service Providers (ASPs) like digiMine™ (http://www.digimine.com) and Information Discovery™ (http://www.datamine.aa.psiweb.com) who offer commercial data mining e-services are testimony to this statement [KZL01, KZL02c, Har02]. The increasing focus on data mining e-services can be attributed to the recognition of data mining as an important technology in aiding strategic decision making coupled with the commercial viability of the ASP paradigm that underlines the commercial viability of "renting" software out [Mor99, Cla99]. The option of Internet-delivery of data mining services is emerging as attractive for small to medium range organisations, which are the most constrained by the high cost of data mining software, and consequently, stand to benefit by paying for software usage without having to incur the costs associated with buying, setting-up and training. While the primary focus of the commercial arena has been the delivery of data mining as an e-service using the ASP model, there is an emerging research focus on providing data mining *models* as services on the Internet [SaN00]. Thus rather than the hosting of a data mining system and delivering the results as a service, the aim is to be able to sell data mining models, which can be bought, for example, by start-up organisations operating in a given vertical domain.

The focus of our work in the area of data mining e-services has been as follows. We have proposed and developed a hybrid distributed data mining (DDM) model to support the needs of an e-services environment [KZL00] and an interaction protocol and XML language to support matching, ranking and negotiation in a virtual community of data mining e-services [KZL01, KPH02]. The following sections summarise our on-going research in this area.

2.1 Hybrid DDM Model to Support Data Mining Services

The hybrid DDM model has the following salient features:

Distribution and Heterogeneity. E-services by definition operate in an Internet environment. The service providers and the clients are distributed, thereby implying that the data and data mining system are also distributed. Heterogeneity is also an inherent characteristic in this environment as the client's computational resources (where the data resides) and the service provider's computational resources (where the data mining system resides) may not be the same. The ability for the data mining software to be able to cope with this distribution and heterogeneity transparently is an important feature for operating in this environment. The operational model of the commercial data mining service providers [KZL02c] places less emphasis on this aspect since they provide mechanisms external to the data mining system to collect the data and bring it to their servers. However, the significance of the integration of data mining systems into a distributed environment can have potential benefits such as easier operation as evidenced by the concentrated research in the area of *distributed data mining (DDM)* [Fu01, Kam01]. The hybrid model integrates existing DDM models, namely the client server model and the mobile agent model and therefore operates in distributed and heterogeneous environments.

Location Preferences. Advances in distributed systems, particularly in the area of mobile agents [CHK98, KoG99, Mil99], facilitate performing data mining in remote locations as shown in several research prototype systems including Java Agents for Meta-learning (JAM) [SPT97], Parallel Data Mining Agents (PADMA) [KHS97], Besiezing Knowledge through Distributed Heterogeneous Induction (BODHI) [KPH99], Infosleuth [UMG98] and Papyrus [Ram98]. In the context of data mining e-services, the ability to mine at distributed locations enables clients to choose the option of not shipping their sensitive/confidential data. Alternatively, if clients did not have the adequate computational resources they can use the traditional approach of sending the data across. There is a need for data mining systems operating in the e-services domain to support both methods to provide the highest flexibility and best meeting the needs of clients. The hybrid model can perform mining at the client's site using the mobile agent paradigm or at the service provider's site using the client server paradigm and therefore can support the location preferences of clients in a virtual community of data mining e-services and deliver differentiated services. The hybrid model therefore combines the best features of both the mobile agent model (by performing mining at remote locations and reducing the overheads of data transfer) and the client server model (by incorporating high performance computational resources and having a higher degree of control of the resources).

It can therefore clearly be seen that a hybrid approach has advantages over the mobile agent model and the client server model for data mining e-services.

Quality of Service (QoS). Data mining in the context of e-services environments must take into account and operate within the framework of QoS parameters and Service Level Agreements. In an e-services context, there is a strong need for predictive metrics that estimate the service levels that can be guaranteed [BoS01, Mor01, KSW01]. An important QoS metric for data mining is response time. The ability to estimate the response time of a data mining task is non-trivial. However, it is necessary in order to specify the length of time after which clients will be given their results. It might appear that this can be estimated in an ad-hoc basis using human experience, however, that is not adequate to meet the demands of data mining e-services. In an e-services environment different service providers will quote their respective capabilities. Therefore, it is in the service providers' interest to quote their respective minimum, as this would increase their chances of obtaining the job. On the other hand, quoting an unrealistic minimum would result in lost revenue as SLAs include penalties for not meeting the specified levels of service. Therefore it is imperative for data mining systems to incorporate this dimension into their standard operation. The hybrid model has formalised the cost components of the DDM response time and has the ability to accurately estimate these components to addresses this need of data mining systems delivering e-services. This is an important contribution of the hybrid model's support for operation in a data mining e-services. The accurate estimation of the cost components of the DDM response time is vital for the success of this model and as noted in the literature is a non-trivial task. We have proposed and developed cost formulae for estimating the communication time for different scenarios involving mobile agents (which focus on the mobile agent transfer time) and the client-server approach (which focuses on the dataset transfer time) [KLZ00, KZL02b]. In order to estimate accurately the computation cost of data mining tasks we have proposed and developed a novel rough sets based algorithm for application run time estimation [KLZ02, KZL02a, KZL02b].

Optimisation and Cost-Efficiency. This refers to the need to perform data mining in a manner whereby the utilisation of resources is optimal and cost-efficient and implies task allocation strategies that both minimise the overall response time to best meet a client's needs. The optimal utilisation of resources has a significant bearing the successful operation of a data mining e-services as it leads to the ability to accept more jobs and increase revenue. The e-services concept of "differentiated services" [EgH99], where services are customised to specific requirements of individual client also requires efficient task allocation techniques. While traditionally this notion of task allocation and resource utilisation is not incorporated into data mining systems, as it is not considered to be of significant bearing. However, a consequence of the e-services domain is that it requires such capabilities in data mining systems. The hybrid DDM model uses the *a priori* estimates of the response time as the basis for costing in its task allocation strategy. It uses a combination of the client-server and mobile agent models to minimise the overall response time. We have experimentally shown the improved performance of the hybrid DDM model [KZL03] in comparison to the client-server and mobile agent models.

We have implemented a DDM system *Distributed Agent-based Mining Environment* (DAME) based on the hybrid model [KPH02]. The languages and tools used in the implementation of the DAME system include Java™, C++ and the Aglets SDK™ (for implementing the mobile data mining agents. The data mining algorithms implemented in the WEKA package [WiE99] were incorporated in the DAME system. A combination of network performance/monitoring tools including Iperf™ and Ping were used to obtain network status information for building estimates of the communication component. While Java was the primary language of development, C++ was used to implement certain parts of the system where system-level information or Operating System specific information was required (e.g. current system resource usage required for building profile of servers for estimating the computation component or data mining application run time). C++ was used since the Java Virtual Machine (JVM) abstracts much of the low-level, system-specific aspects from the application programmer. The Java Native Interface (JNI) was used to call the methods of the C++ DLLs. The DAME system architecture is presented in figure 1.

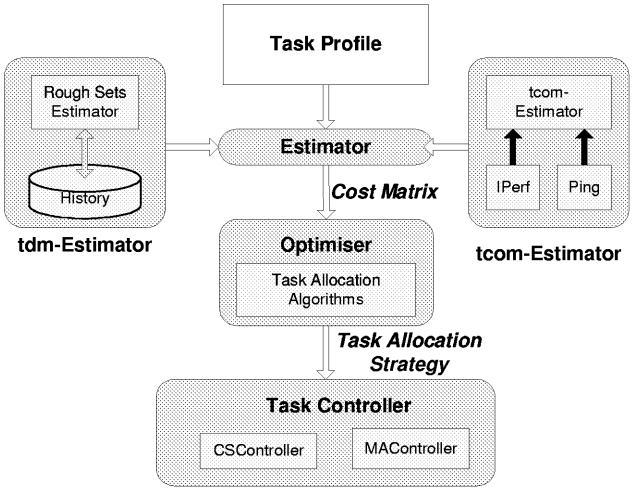


Figure 1. Distributed Agent-based Mining Environment (DAME)

The components of the system are as follows:

Task Profile. This represents a task request with detailed information such as the servers that are available, the location and size of the datasets, the algorithms that must be used and the response time constraints. The Task Profile is given to the Estimator.

Estimator. This component uses the task profile to build estimates for the response times of the alternative communication strategies and the estimates for the data mining task run time required to generate a cost matrix for the task. The Estimator uses two components:

- *tdm-Estimator.* The tdm-Estimator consists of the Rough sets package for estimating application run times and a history of previous tasks for performing estimates of task run times.
- *tcom-Estimator.* The tcom-Estimator uses network monitoring tools including IPerf and Ping to obtain bandwidth and latency measurements used by the cost

formulae for estimating the transfer times of mobile agents and data transfers. A current limitation of the system is the use of IPerf™ since this requires a server to run between nodes. This is not realistic in a data mining e-service context, where estimates must be made without client's having to run an IPerf server. A possible alternative is the use of a tool such as JavaNWS (Java Network Weather Service) [KrW01], which is a tool for measuring network round trip time and bandwidth between a user's desktop and any machine running a web server. We intend to use in the tcom-estimator in future following the next release of JavaNWS. However, for experimental validation of the cost formulae, tools such as IPerf™ are adequate.

Optimiser. The Optimiser contains the task allocation algorithms including the implementation of the Tabu search and uses the cost matrix developed by the Estimator to determine the allocation strategy. The Optimiser can either determine the minimum response time strategy or a strategy that is the closest feasible solution to a client's requested response time.

Task Controller. The Task Controller uses the task allocation strategy determined by the Optimiser to execute the distributed data mining task. The data mining algorithms used are from the WEKA package [WiE99], which is an open-source Java implementation of standard data mining algorithms. The Task Controller uses the MAController and the CSController to perform the DDM task.

- *CSController.* The CSController is responsible for transferring datasets for applying the client-server model. The current implementation uses FTP to transfer data.
- *MAController.* The MAController is responsible for the generation and transfer of mobile agents to remote servers. The Aglets SDK™ package has been used to implement the data mining agents. There are two agents that have been developed to support the DDM process – a *ControlAgent* and a *DMAgent.* The ControlAgent initialises the *DMAgent* by providing it the itinerary. The DMAgent whose operation is shown in figure 2 performs the following tasks:

  1. It converts the data mining algorithms that it needs to carry into a serialised object.
  2. It iteratively determines the next location on its itinerary and transports itself to that location.
  3. It unpacks the algorithm into the specified remote directory.
  4. It uses the itinerary to generate a task file, which is a script that calls the data mining algorithms to process the required data sets.
  5. It forks off a child process to execute the script.
  6. Upon completion, it travels to its next destination and at the final destination can either destroy itself or return to the originating server.

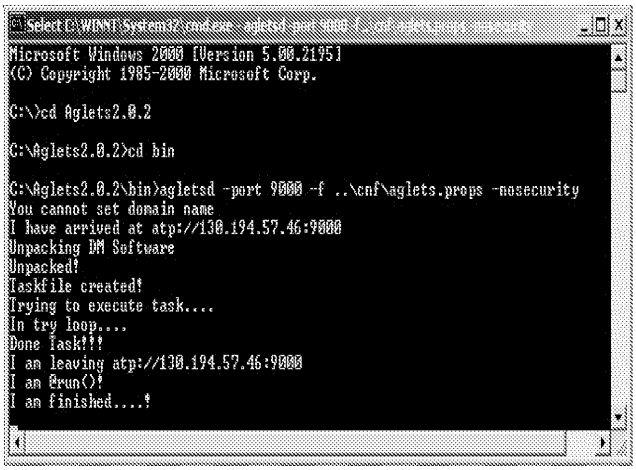


Figure 2. Data Mining Mobile Agent Arriving at a Remote Host

We have discussed the architecture and implementation aspects of the DAME system. The Java packages in the DAME system correspond to the above components. Currently, the DAME system does not have a Graphical User Interface and the operation is through a Command Line Interface CLI. The development of GUI is planned for future work. We have discussed our DDM model to support data mining e-services. We now discuss our interaction protocol and specification language to support data mining e-services.

2.2 An Interaction Protocol and a Specification Language to Support Data Mining Services

In order to support ad-hoc task requests and services as envisaged by e-services, its is necessary to have a well-defined model and language for interaction. Data mining is typically a confidential and sensitive activity and does not lend itself to being broadcast in an open environment. Organisations are hesitant to publicly announce their data mining needs in the quest for identifying a suitable service provider in a virtual community. The question that needs to be addressed in this context is to find the balance between the ability to identify a suitable service provider in a virtual community without having to divulge sensitive and confidential details during the identification and selection phase. The focus of data mining query languages [HFW99, HKS97, MPS96, TUA98, ZaH98, ImV99, ESF00, SZZ01] and specification languages [GBR99, Ole00, KoZ00a, Cri00, PMZ01a] is on describing the data mining task itself (including confidential data descriptions etc.) rather than providing mechanisms for specifying the requirement of the task. For example, in identifying a service provider and organisation would need to state its *preferences* and requirements, such as it has a spatial dataset and that it wants a specific algorithm to be used rather than what fields are in the data. Thus, there is a need for specification of the requirements of a task at the level of abstraction wherein it is possible to state an organisation's needs without divulging confidential details. While e-services environments provide the underlying infrastructure to support discovery and connection to service providers, this process is also largely dependent on the specifications provided by the service providers regarding their capabilities. Thus, the question that needs to be addressed in the context of interactions in data mining

e-services is the specification of the services that can be provided. This not only supports the process of discovery in e-services environments, but also forms the basis for advanced interactions such as ranking and negotiation. Following the selection through a process of identification, ranking and negotiation (either manual or automated) and the signing of the SLA, the interactions need to facilitate the actual performing of the task by specification of data descriptions, access to data, access to servers and delivery of results.

We have proposed and developed an interaction protocol for data mining e-services delineates specification of a task request containing the preferences and requirements of the client from the task specific information, thereby preserving the confidentiality needs of the client with respect to the task. It also facilitates automated selection of the service provider by means of the capability specification of the service providers. Ranking and negotiation have not been completely adopted in e-services frameworks, however, there is on-going research in these areas [BaP01, TeM01]. Our protocol is flexible and can support these emerging trends in e-services as we have shown in our implementation of a prototype virtual community of data mining e-services. The protocol for interaction presented in this section requires an exchange of five messages between clients and service providers:

1. *Data Mining Task Request.* This message is sent from the client to the intermediary (which represents the manager or coordinator of the virtual community), who in turn broadcasts it to the service providers in the virtual community. It specifies the preferences and requirements of the client with respect to the task. However, it does not consist of task specific confidential information.
2. *Service Provider Description.* This message is sent from the service providers in response to the task request. The message can either be forwarded by the intermediary to the client for selection or can be used to support an automated selection process based on matching to identify the subset of service providers who meet the required constraints, ranking to determine the order of service providers that meet the preferences and then negotiation to finalise the selection of the service provider. The selection process is followed by a legally binding SLA between the selected service provider and the client and signals the beginning of the next phase of the interaction.
3. *Data Mining Task Description.* This message is sent from the client to the service provider and details task specific information including data descriptions.
4. *Client Access Information.* This message is sent from the client to the service provider and contains information specifying access to the computational resources and datasets by the service provider.
5. *Service Provider Access Information.* This message is sent from the service provider to the client and specifies how the client can access the service provider for results or to deposit the datasets.

It must be noted that we have formalised the information content of the messages and developed an XML mapping for the specifications. Due to constraints of space we refer readers to [KZL01] for the XML DTDs and detailed description of the specification language.

The interaction protocol and the specification language has been the basis for the implementation of a demonstration prototype of a virtual community of data mining e-services shown in figure 3. The implementation involved the integration of the following technologies and languages: Java™, XML, the Sun Microsystems Java Architecture for XML Binding™ (JAXB) package and C#™. The primary language used was Java. XML was used to store the messages from clients and service providers and for the different components to exchange information, including the transfer of the output from the matching engine to the ranking engine to the negotiation engine. The matching engine used the JAXB package (still in "Early Access" version), which provides an API and tools that automate the mapping between XML documents and Java objects. It works by compiling an XML schema with the DTD into one or more Java classes. The generated classes handle all the details of XML parsing and formatting. The generated classes also ensure that the constraints expressed in the schema/DTD are enforced in the resulting methods. The negotiation engine is the only component developed in C#.

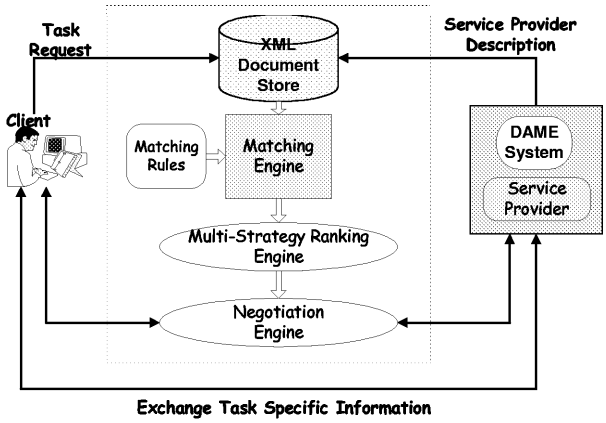


Figure 3 Architecture of the Prototype Virtual Community of Data Mining Services

The prototype consists of the following components:

XML Document Store. This component stores the XML messages sent by the client and service provider including the task request messages and the service provider capability description.

Matching Engine. The matching engine shown in figure 4 applies a set of matching rules that we have developed [KPH02] to process the task request documents sent by the client and match them with the service provider description messages that have been submitted by service providers. In order to test the functioning of the matching engine, XML documents conforming to the DTDs were generated using XML Spy™.

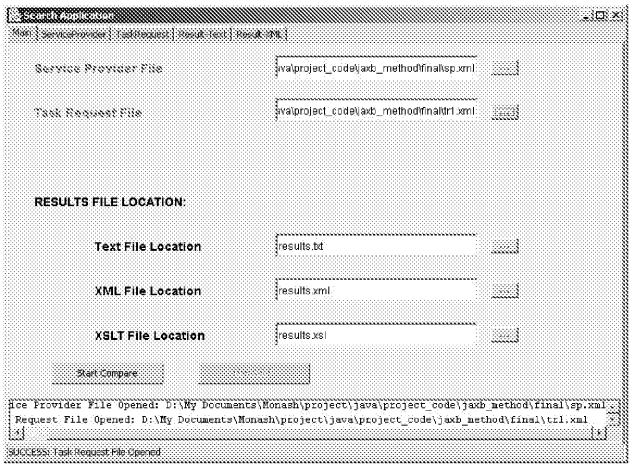


Figure 4. Matching Engine

Multi-strategy Ranking Engine. The ranking engine applies two different ranking strategies to rank the service providers that are determined by the matching engine as being a valid match. Thus, the matching engine determines the service providers that meet the "required" constraints of the client. The ranking engine performs the task of ranking service providers on the basis of the "preferred" constraints of the client. Typically, however, ranking processes require additional information about preferences such as weighting of the different constraints. The ranking engine captures this information from the client. A multi-strategy model was adopted primarily since ranking strategies in e-services is still emerging and it is not obvious from the current work that there is a widely accepted ranking strategy. Further, different ranking strategies result may result in different outcomes and having a strategy that analyses the output from different ranking techniques ensure less bias. The techniques implemented in the ranking engine are:

- *Distance Measures,* where the distance between the clients preferences for the different attributes are captured on a scale of 0-10, with 0 indicating no preference and 10 indicating an absolute requirement. This is used to rank service providers on the basis of their ability to meet preferences using the following techniques: *Euclidean distance, Manhattan distance, Canberra distance, Squared chord distance* and the *Squared Chi-squared distance.*
- *MARI (Multi-Attribute Resource Intermediary)* [TeM01] is a ranking technique based on Multi-Attribute Utility Theory. This technique requires the weight of an attribute to be specified as a range, where the more restricted the range is, the more significant the attribute is for the client.

Negotiation Engine. The negotiation engine shown in figure 5 performs automated negotiation between the client and the highest ranked service provider in the virtual community as determined by the ranking process. The graph in the figure shows the client and provider convergence (reaching agreement), and the side windows the history of proposals for successive negotiation rounds. The negotiation engine, like the ranking engine uses the preferred constraints to drive the concessions in the negotiation process. The negotiation technique that has been implemented in the context of data mining e-services is the Service-Oriented Negotiation Model developed by [SFJ97]. This is a heuristic negotiation strategy that is based on Raffia's basic model for bilateral negotiation [Raf82], and the work covers many aspect of a human-like negotiation model and capable of automating negotiation. The model consists of two protocols for interaction and three decision-making mechanisms. The technique uses a utility function to handle multiple issues and the linear combination of each utility value forms an expected utility value for an offer. The decision for counter-offer happens when the expected utility value of the opponent's counter offer is greater than the currently proposed offer. The technique uses environmental tactics (such as time, resource and behaviour) and an "issue manipulation mechanism" (increase or decrease issues in the set) to prevent a deadlock. Three decision-making mechanisms are used for terminating negotiation or used for converging to an agreement, namely, a "responsive mechanism" and two "deliberative mechanisms". We have implemented two of these mechanisms, namely the responsive mechanism and the trade-off mechanism, which is a deliberative technique. Thus, the specifications developed in this dissertation and the constraints included in this task requests and the ability to match these requests to the capabilities of service providers has enabled the support for automated negotiation techniques.

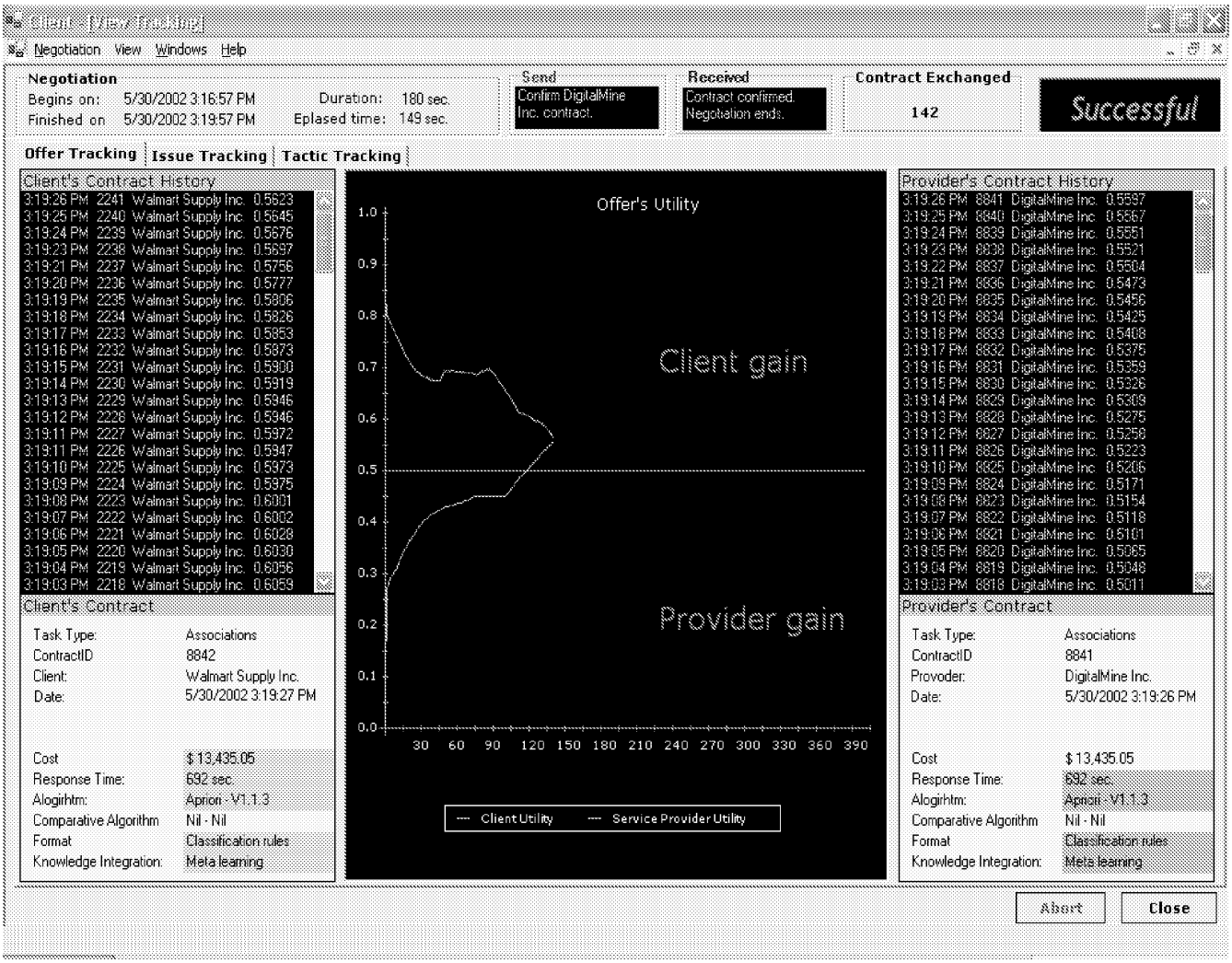


Figure 5. Visualising the Negotiation Process

Distributed Agent-based Mining Environment (DAME). The DAME system is a hybrid DDM system developed to meet the infrastructure requirements of a data mining system operating in an e-services domain.

Our research in the context of delivering data mining e-services is but a small step towards realising the full potential of the anytime anywhere services. While the interaction protocol and specification language provide basic support for the concept of ad-hoc task requests and the techniques for estimating QoS levels in the hybrid DDM model facilitate dynamic construction of Service Level Agreements there are several significant issues and challenges that need to be addressed.

3. ISSUES AND CHALLENGES FOR ANYTIME ANYWHERE DATA MINING SERVICES

The concept of anytime anywhere data mining services envisages the transformation of data mining as technology that supports large enterprises to one that is accessible and available to a wide range of people and organisations of various sizes. There are several open issues and challenges that need to be addressed. Some of these include:

Data as a commodity and Analysis as an e-service. The proposed concept has far-reaching ramifications for the emerging data mining services for it implies going beyond the "analysis as a service" to "data as a commodity and analysis as a service". For instance organisations that perform data mining typically own the data. While this is sufficient in the context of that organisation's business intelligence needs, it does not support the requirements of a small scale stock-broker or a smart shopper, who can never have the data required to perform such analysis. This in turn raises several challenges in some scenarios. While stock-market transactions prices of shares can be obtained publicly, transaction data for retail industries are not. To support the needs of smart shoppers, a data mining service might have to use a monitoring service to determine trends in different retail industries since transaction data are typically a confidential resource.

Mobile Users and Location Awareness. Data mining services must be able to support users who are moving and must take into account the fact that a query/task request might have been submitted from one location and the user may have moved to another location by the time the results are ready. The use of mobile agents that listen for user connections and disconnections is an option could be explored to support mobile users and location awareness.

Mobile agents also allow data to be mined at the client's site (i.e., *in situ mining*). Such flexibility about where data is mined, however, does not come for free – secure agent servers are needed which can host such agents. Indeed, much work is underway to tackle mobile agent security issues, as well as reducing the infrastructure set-up effort by relying on existing (e.g., Web) protocols and we contend that such agent servers, once running, can serve multiple purposes (even as Java applets do). The issue of heterogeneous agent platforms and operating systems are also being partially addressed (e.g., in the agent community via standardization efforts (http://www.fipa.org) and the likely emergence of a small number of popular run-time mobile code environments).

Mobile Devices. Task requests may be posted from small devices with limited computational resources and screen real estate. This needs to be factored into the delivery of results. For example a detailed analysis may be available on-line but an appropriate subset is sent to the user's PDA. There is on-going research in this area [KPP02, LAK01].

Ad-hoc Queries and Timely Results. The needs of stock-brokers or mobile salespersons is highly time-constrained. Further, queries may be posted at any time in an ad-hoc manner. Data mining services of the future would need to support such dynamic task requests for real-time results. Incremental and active data mining techniques such as [Dum02] are applicable in this context. However, the issue of ad-hoc requests in data mining e-services is yet to be addressed though our work involving the specification language for data mining e-services is an initial effort in this context.

Electronic Construction and Management of Service Level Agreements. Since these services are accessible from anywhere and at anytime, there must be support for new users to sign up/ subscribe to a data mining service and seamlessly negotiate SLAs. A related challenge is for the system to help clients to provide specialized information about the data (quirks, etc) to be mined.

We have presented the concept of *anywhere anytime* data mining services as the next step in the convergence of mobile data mining with Internet-based data mining services. The opportunities and potential benefits are many, but so are the technological challenges in the realisation of this concept. The current focus of our research aims to extend our work in data mining services to support the paradigm of *anytime anywhere* data mining services.

4. ACKNOWLEDGMENTS

Our thanks to the ADM reviewers for valuable feedback on the paper.

5. REFERENCES

[BaP01] Bartolini, C., and Priest, C., (2001), *"A Framework for Automated Negotiation"*, Hewlett-Packard Labs Technical Report HPL-2001-96, Available Online: http://www.hpl.hp.com/techreports/2001/HPL-2001-90.html

[BoS01] Bouch, A., and Sasse, A., (2001), *"Why Value is Everything: A User-Centered Approach to Internet Quality of Service and Pricing"*, Proceedings of the Ninth International Workshop on Quality of Service (IWQoS), Lecture Notes in Computer Science (LNCS) 2092, pp. 59-74.

[Cri00] Crisp 1.0 Process and User Guide (CRISP-DM). Available Online: http://www.crisp-dm.org

[Day01] Dayal, U., (2001), *"Data Mining Meets E-Business: Opportunities and Challenges"*, Proceedings of the First SIAM International Conference on Data Mining, Online Proceedings, Abstract of Keynote Available At: http://www.siam.org/meetings/sdm01/html/k4.htm

[Dum02] Dumitriu, L., (2002), *"Interactive Mining and Knowledge Reuse for the Closed Itemset Incremental Mining Problem"*, SIGKDD Explorations, January, Vol. 3, No. 2.

[ESF00] Elfeky,M,G., Saad,A,A., and Fouad,S,A., (2000), *"ODMQL: Object Data Mining Query Language"*, Lecture Notes in Computer Science 1944 (LNCS), pp.128-140.

[Fu01] Fu, Y., (2001), *"Distributed Data Mining: An Overview"*, Newsletter of the IEEE Technical Committee on Distributed Processing, Spring 2001, pp.5-9.

[GBR99] Grossman,R,L., Bailey,S., Ramu,A., Malhi,B., Hallstrom,P., Pulleyn,I., and Oin,X., (1999), *"The Management and Mining of Multiple Predictive Models Using the Predictive Modeling Markup Language (PMML)"*, Information and Software Technology, Vol. 4, pp. 589-595.

[GGR02] Ganti, V., Gehrke, J., and Ramakrishnan, R., (2002), *"Mining Data Streams under Block Evaluation"*, SIGKDD Explorations, January, Vol. 3, No. 2.

[Gro98] Grossman, R., (1998), *"Supporting the Data Mining Process with Next Generation Data Mining Systems"*, Enterprise Systems, August 1998, Available Online: http://www.esj.com/back_issues/toc.asp?MON=8&YR=1998

[Har02] Harney, J., (2002), *"Big-Company Intelligence"*, Enterprise Systems, June 2002, Available Online: http://www.esj.com/features/article.asp?EditorialsID=107

[HFW96] Han,J., Fu,Y., Wang,W., Koperski,K., and Zaiane,O., (1996), *"DMQL: A Data Mining Query Language for Relational Databases"*, In SIGMOD'96 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'96), Montreal, Canada.

[HKS97] Han,J., Koperski,K., and Stefanovic,N., (1997), *"GeoMiner: A System Prototype for Spatial Data Mining"*, Proceedings 1997 ACM-SIGMOD Int'l Conference on Management of Data(SIGMOD'97), Tucson, Arizona, May1997(System prototype demonstration).

[ImV99] Imielinski,T., and Virmani.A., (1999), *" MSQL: A query language for database mining"*, Data Mining and Knowledge Discovery, Vol. 3, pp.373-408.

[KoZ00] Kotasek,P., and Zendulka,J., (2000), *"An XML Framework Proposal for Knowledge Discovery in Databases"*, In: The Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases, Workshop Proceedings Knowledge Management: Theory and Applications, Ecole Polytechnique de Nantes, Lyon, France, 2000, p. 143-156

[Kam01] Kamath, C., (2001), *"The Role of Parallel and Distributed Processing in Data Mining"*, Newsletter of the IEEE Technical Committee on Distributed Processing, Spring 2001, pp. 10-15.

[KHS97] Kargupta, H., Hamzaoglu, I., Stafford, B., Hanagandi, V., and Buescher, K., (1997), *"PADMA: Parallel Data Mining Agents For Scalable Text Classification"*, Proceedings Of The High Performance Computing Conference, The Society For Computer Simulation International, pp. 290-295.

[KoP01] Kohavi, R., and Provost, F., (2001), *"Applications of Data Mining to Electronic Commerce"*, Data Mining and Knowledge Discovery, Special Issue – Applications of Data Mining to Electronic Commerce, (Eds) Ron Kohavi and Foster Provost, Vol. 5, No. 1/2, pp. 5-10.

[KPH02] Krishnaswamy, S., Pin, S, E., Ho, J, N., and Gunawan, W., (2002), *"An XML Specification Language to Support a Virtual Marketplace of Data Mining E-services"*, Accepted for publication at the International Workshop on Data Semantics in Web Information Systems (DASWIS 2002).

[KPM01] Kalogeraki, V., Pruyne, J., and Moorsel, A., (2001), *"A Peer-to-Peer Architecture for Delivering E-Services"*, Hewlett-Packard Labs Technical Report HPL-2001-181, Available Online:

http://www.hpl.hp.com/techreports/2001/HPL-2001-134.html

[KPP02] Kargupta, H., Park, B., Pittie, S., Liu, L., Kushraj, D., and Sarkar, K., (2002), *"MobiMine: Monitoring the Stock Market from a PDA"*, SIGKDD Explorations, January, Vol. 3, No. 2.

[KSW01] Kraiss, A., Schoen, F., Weikum, G., Deppisch, U., (2001), *"Towards Response Time Guarantees for e-Service Middleware"*, IEEE Data Engineering, Vol. 24, No. 1, pp. 59-64.

[KLZ00] Krishnaswamy, S., Loke, S, W., and Zaslavsky, A., (2000b), *"Cost Models for Heterogeneous Distributed Data Mining"*, Proceedings Of the Twelfth International Conference on Software Engineering and Knowledge Engineering, Chicago, Illinois, July 6-8, pp. 31-38.

[KLZ02] Krishnaswamy, S., Loke, S, W., and Zaslavsky, A., (2002), *"Application Run Time Estimation: A QoS Metric for Web-based Data Mining Service Providers"*, Proceedings of the Seventeenth ACM Symposium on Applied Computing (ACM SAC) 2002 in the Special Track on WWW and E-business Applications, Madrid, Spain, March 10-14, ACM Press, pp. 1153-1159.

[KPH99] Kargupta, H., Park, B., Hershberger, D., and Johnson, E., (1999), *"Collective Data Mining: A New Perspective Toward Distributed Data Mining"*, Advances in Distributed Data Mining, (eds) Hillol Kargupta and Philip Chan, AAAI Press, pp. 131-178.

[KZL00] Krishnaswamy, S., Zaslavsky, A., and Loke, S, W., (2000), *"An Architecture to Support Distributed Data Mining Services in E-Commerce Environments"*, Proceedings of the 2nd International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS2000), Milipitas, CA, USA, June 2000, IEEE Press, pp. 239-246.

[KZL01] Krishnaswamy, S., Zaslavsky, A., and Loke, S,W., (2001), *"Towards Data Mining Services on the Internet with a Multiple Service Provider Model: An XML Based Approach"*, Journal of Electronic Commerce Research (Special issue on Electronic Commerce and Service Operations), Vol. 2(3), August. ISSN 1526-6133.

[KZL02a] Krishnaswamy, S., Zaslasvky, A., and Loke, S, W., (2002), *"Predicting Application Run Times Using Rough Sets"*, Ninth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2002), Annecy, France, July, IEEE Press, pp. 455-462.

[KZL02b] Krishnaswamy, S., Zaslavsky, A., and Loke, S, W., (2002), *"Techniques for Estimating the Computation and Communication Costs of Distributed Data Mining"*, Proceedings of International Conference on Computational Science (ICCS2002) – Part I, Lecture Notes in Computer Science (LNCS) 2331, Springer Verlag. pp. 603-612.

[KZL02c] Krishnaswamy, S., Zaslavsky, A., and Loke, S, W., (2002), *"Internet Delivery of Distributed Data Mining Services: Architectures, Issues and Prospects"*, Accepted as Book Chapter in: Architectural Issues of Web-enabled Electronic Business, IDEA Publishing Group.

[KZL03] Krishnaswamy, S., Zaslavsky, A., and Loke, S, W., (2002), *"A Hybrid Model for Optimising Distributed Data Mining"* Submitted to the International Conference on Distributed Computing Systems (ICDCS 2003).

[LaO98] Lange, D., B., and Oshima, M., (1998), *"Programming and Deploying Java Mobile Agents with Aglets"*, Addison-Wesley, 1998.

[LAK01] Lawrence, R, D., Almasi, G, S., Kotlyar, V., Viveros, M, S., and Duri, S, S., (2001), *"Personalization of Supermarket Product Recommendations"*, Data Mining and Knowledge Discovery, Special Issue – Applications of Data Mining to Electronic Commerce, (Eds) Ron Kohavi and Foster Provost, Vol. 5, No. 1/2, pp. 11-32.

[Mor99] Morency, J., (1999), *"Application Service Providers and E-Business"*, Network World Fusion Newsletter, Available Online: http://www.nwfusion.com/newsletters/nsm/0705nm.html

[Mor01] Morsel, A., (2001), *"Metrics for the Internet Age: Quality of Experience and Quality of Business"*, Hewlett-Packard Labs Technical Report HPL-2001-179, Available Online:

http://www.hpl.hp.com/techreports/2001/HPL-2001-134.html

[MPS96] Meo,R., Psaila,G., and Ceri.S., (1996), *"A New SQL-like Operator for Mining Association Rules"*, Proceedings of the Conference on Very Large Databases (VLDB'96), pp. 122-133, Bombay, India, Sept.

[Ole00] OLE DB for Data Mining Specification. Available Online: http://www.microsoft.com/data/oledb/dm.htm

[PMZ01] Punin,J., Krishnamoorthy,M., and Zaki,M,J., (2001), *"LOGML -- Log Markup Language for Web Usage Mining"*, in WEBKDD Workshop 2001: Mining Log Data Across All Customer TouchPoints (with SIGKDD01), San Francisco, August 2001. Available Online: http://www.cs.rpi.edu/~zaki/papers.html

[Raf82] Raffia,H., (1982), "*The Art and Science of Negotiation*", Cambridge, Massachusetts, Harvard University Press.

[Ram98] Ramu, A, T., (1998), "*Incorporating Transportable Software Agents into a Wide Area High Performance Distributed Data Mining Systems*", Masters Thesis, University of Illinois, Chicago, USA.

[SFJ97] Sierra,C., Faratin,P., and Jennings,N., (1997), "*A Service-Oriented Negotiation Model between Autonomous Agents*", Proceedings of the 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-97), Ronneby, Sweden, pp.17-35.

[SaN00] Sarawagi, S., and Nagaralu, S, H., (2000), "*Data Mining Models as Services on the Internet*", SIGKDD Explorations, Vol. 2, No. 1. Available Online: http://www.acm.org/sigkdd/explorations/

[SpP01] Spiliopoulou, M., and Pohle, C., (2001), "*Data Mining for Measuring and Improving the Success of Web Sites*", Data Mining and Knowledge Discovery, Special Issue – Applications of Data Mining to Electronic Commerce, (Eds) Ron Kohavi and Foster Provost, Vol. 5, No. 1/2, pp. 85-114.

[SPT97] Stolfo, S, J., Prodromidis, A, L., Tselepis, L., Lee, W., Fan, D., and Chan, P, K., (1997), "*JAM: Java Agents for Meta-Learning over Distributed Databases*", Proceedings of the Third International Conference on Data Mining and Knowledge Discovery (KDD-97), Newport Beach, California, (eds) David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, AAAI Press, pp. 74-81.

[SZZ01] Sadri,R., Zaniolo,C., Zarkesh,A., Adibi,J., (2001), *"A Sequential Pattern Query language for Supporting Instant Data Mining for e-Services"*, 27th International Conference on Very large Databasese (VLDB-2001), September, 11-14, 2001, Roma, Italy.

[TeM01] Tewari,G., and Maes,P., (2001), *"A Generalized Platform for the Specification, Valuation, and Brokering of Heterogeneous Resources in Electronic Markets"*, in Lecture Notes in Artificial Intelligence (LNAI) 2033, Springer-Verlag, pp.7-24.

[TUA98] Tsur,D., Ullman,J.D., Abitboul,S., Clifton,C.., Motwani, R., and Nestorov. S., (1998*), "Query Flocks: A Generalization of Association-rule Mining"*, Proceedings of ACM SIGMOD'98, Seattle, Washington, June 1998.

[WiE99] Witten, I, H., and Eibe, F., (1999), "*Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*", Morgan Kauffman.

[WoM01] Wolter, K., and Moorsel, A., (2001), "*The Relationship between Quality of Service and Business Metrics: Monitoring, Notification and Optimization*", Hewlett-Packard Labs Technical Report HPL-2001-96. Available Online: http://www.hpl.hp.com/techreports/2001/HPL-2001-96.html

[ZaH98] Zaine,O., and Han,J., (1998), "*WebML: Query the World-Wide Web for Resources and Knowledge",* Proceedings (CIKM'98) Int'l Workshop on Web Information and Data Management (WIDM'98), Bethesda, Maryland, November, pp. 9-12.

About the authors:

Shonali Krishnaswamy is a lecturer in the School of Network Computing, Monash University. Her research interests are in the areas of e-services, mobile and distributed data mining and rough sets. Shonali's PhD thesis proposed a hybrid distributed data mining model for delivering Internet-based data mining services. Her current research projects include modelling, predicting and ensuring quality of service levels for application services and accessing data mining e-services from mobile devices.

Seng Wai Loke is a Research Fellow in the School of Computer Science & Software Engineering of Monash University. He received his PhD in Computer Science. His research interests include pervasive computing and commerce, intelligent and mobile agents, intelligent e-services, e-communities, and declarative programming.

Arkady Zaslavsky received MSc in Applied Mathematics majoring in Computer Science from the Tbilisi State University (Georgia, USSR) in 1976 and PhD in Computer Science from the USSR Academy of Sciences in 1987. He is holding a position of Associate Professor in the School of Computer Science & Software Engineering of Monash University. His research interests include mobile and pervasive computing; distributed and mobile agents and objects; distributed computing and database systems; distributed object technology and mobile commerce. He is a member of ACS, ACM and IEEE Computer and Communication Societies.

A Heuristic Lazy Bayesian Rule Algorithm

Zhihai Wang
School of Computer Science and Software Engineering
Monash University
Vic. 3800, Australia
zhihai.wang@infotech.monash.edu.au

Geoffrey I. Webb
School of Computer Science and Software Engineering
Monash University
Vic. 3800, Australia
webb@infotech.monash.edu.au

ABSTRACT

LBR has demonstrated outstanding classification accuracy. However, it has high computational overheads when large numbers of instances are classified from a single training set. We compare LBR and the tree-augmented Bayesian classifier, and present a new heuristic LBR classifier that combines elements of the two. It requires less computation than LBR, but demonstrates similar prediction accuracy.

1. INTRODUCTION

The naive Bayesian classifier [1] is known to be optimal and efficient for classification when all the attributes are mutually independent given the class and the required probabilities can be accurately estimated from the training data. Assume X is a finite set of instances, and $A = \{A_1, A_2, \cdots, A_n\}$ is a finite set of n attributes. An instance $x \in X$ is described by a vector $< a_1, a_2, \cdots, a_n >$, where a_i is a value of attribute A_i. C is called the class attribute. Prediction accuracy will be maximized if the predicted class $L(< a_1, a_2, \cdots, a_n >) = argmax_c(P(c| < a_1, a_2, \cdots, a_n >)$. Unfortunately, unless $< a_1, a_2, \cdots, a_n >$ occurs enough times within X, it will not be possible to directly estimate $P(c| < a_1, a_2, \cdots, a_n >)$ from the frequency with which each class $c \in C$ co-occurs with $< a_1, a_2, \cdots, a_n >$ within X. Bayes' theorem provides an equality that might be used to help estimate $P(c| < a_1, a_2, \cdots, a_n >)$ in such a circumstance:

$$P(c_i| < a_1, a_2, \cdots, a_n >) = \frac{P(c_i)P(< a_1, a_2, \cdots, a_n > |c_i)}{P(< a_1, a_2, \cdots, a_n >)}. \quad (1)$$

If the n attributes are mutually independent within each class value, then the probability is directly proportional to:

$$P(c_i| < a_1, a_2, \cdots, a_n >) \propto P(c_i) \prod_{k=1}^{n} P(a_k|c_i). \quad (2)$$

Classification selecting the most probable class as estimated using (1) and (2) is the well-known naive Bayesian classifier. The naive Bayesian classifier has been shown in many domains to be surprisingly accurate compared to alternatives including decision tree learning, rule learning, neural networks, and instance-based learning. Domingos and Pazzani [2] argued that the naive Bayesian classifier is optimal even when the independence assumption is violated, as long as the ranks of the conditional probabilities of classes given an example are correct. However, previous research has shown that semi-naive techniques and Bayesian networks that explicitly adjust the naive strategy to allow for violations of the independence assumption, can improve upon the prediction accuracy of the naive Bayesian classifier in many domains. This suggests that the ranks of conditional probabilities are frequently not correct. One approach to improving the naive Bayesian classifier is to relax the independence assumptions. Kononenko [3] proposed a semi-naive Bayesian classifier, which partitioned the attributes into disjoint groups and assumed independence only between attributes of different groups. Pazzani |4| proposed an algorithm based on the wrapper model for the construction of Cartesian product attributes to improve the naive Bayesian classifier. The naive Bayesian tree learner, $NBTree$[5], combined naive Bayesian classification and decision tree learning. It uses a tree structure to split the instance space into sub-spaces defined by the paths of the tree, and generates one naive Bayesian classifier in each sub-space. $NBTree$ frequently achieves higher accuracy than either a naive Bayesian classifier or a decision tree learner. Although $NBTree$ can alleviate the attribute inter-dependence problem of naive Bayesian classification to some extent, $NBTree$ suffers from the replication and fragment problem as well as the small disjunct problem due to the tree structure. Friedman, Geiger and Goldszmidt [6] compared the naive Bayesian method and Bayesian network, and showed that using unrestricted Bayesian networks did not generally lead to improvements in accuracy and even reduced accuracy in some domains. They presented a compromise representation, called Tree-Augmented naive Bayes (TAN), in which the class node directly points to all attributes nodes and an attribute node can have only at most one additional parent to the class node. Based on this presentation, they utilized the concept of mutual information to efficiently find the best tree-augmented naive Bayesian classifier. Zheng and Webb [7] proposed the lazy Bayesian rule (LBR) learning technique. LBR can be thought of as applying lazy learning techniques to naive Bayesian rule induction. At classification time, for each test example, it builds a most appropriate rule with a conjunction of conditions as its antecedent and a local naive Bayesian classifier as its consequent.

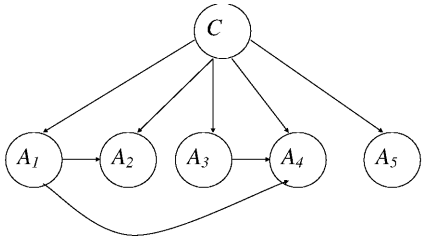


Figure 1: An example of a tree-augmented Bayesian network

Among these approaches of relaxing the attribute independence assumption, *LBR* has demonstrated remarkably low classification error rate. Zheng and Webb [7] experimentally compared *LBR* with a naive Bayesian classifier, a decision tree classifier, a Bayesian tree learning algorithm, a constructive Bayesian classifier, a selective naive Bayesian classifier, and a lazy decision tree algorithm in a wide variety of natural domains. In their extensive experiments, *LBR* obtained lower error than all the alternative algorithms. However, *LBR* is computationally inefficient if large numbers of objects are to classified from a single training set. In this paper, we compare the *LBR* and *TAN* techniques. A heuristic strategy for selecting attribute values to form the antecedent of a lazy Bayesian rule will be presented, which can be thought of as an application of *TAN*. Experimental comparisons and analysis of this heuristic lazy learning of Bayesian rules algorithm with the naive Bayesian classifier, *LBR* and *TAN* show that the heuristic algorithm has the almost same prediction accuracy as *LBR* with much lower computational requirements.

2. TAN AND LBR

Bayesian networks have been a popular medium for graphically representing and manipulating attribute interdependencies. Bayesian networks are directed acyclic graphs (*DAG*) that allow for efficient and effective representation of joint probability distributions over a set of random variables. Each vertex in the graph represents a random variable, and each edge represents direct correlations between the variables. Each variable is independent of its non-descendants given its parents in the graph. Bayesian networks provide a kind of direct and clear representation for the dependencies among the variables or attributes. A tree-augmented Bayesian network is a restricted form of Bayesian network [8], which can be defined by the following conditions:

- Each attribute has the class attribute as a parent;
- Attributes may have at most one other attribute as a parent.

Fig. 1 shows an example of a tree-augmented Bayesian network.

In a tree-augmented Bayesian network, a node without any parent, other than the class node, is called an *orphan*. Given a tree-augmented Bayesian network, if we extend arcs from node A_k to every orphan node A_i, then node A_k is said to be a *super parent*. For any node v, we denote its parents by $Parents(v)$. If v is an orphan, then $Parents(v) = \emptyset$.

LBR uses lazy learning to learn at classification time a single Bayesian rule for each instance to be classified. *LBR* is similar to *LazyDT* (Lazy Decision Tree learning algorithms) [9], which can be considered to generate decision rules at classification time. For each instance to be classified, *LazyDT* builds one rule that is most appropriate to the instance by using an entropy measurement. The antecedent of the rule is a conjunction of conditions in the form of attribute-value pairs. The consequent of the rule is the class to be predicted, being the majority class of the training instances that satisfy the antecedent of the rule. *LBR* can be considered as a combination of the two techniques *NBTree* and *LazyDT*. Before classifying a test instance, it generates a rule (called a Bayesian rule) that is most appropriate to the test instance. Alternatively, it can be viewed as a lazy approach to classification using the following variant of Bayes theorem,

$$P(C_i|V_1 \wedge V_2) = P(C_i|V_2)P(V_1|C_i \wedge V_2)/P(V_1|V_2) \quad (3)$$

Here any test instance can be described by a conjunction of attribute values V, and V_1 and V_2 are any two conjunctions of attribute values such that each v_i from belongs to exactly one of V_1 or V_2. At classification time, for each instance to be classified, the attribute values in V are allocated to V_1 and V_2 in a manner that is expected to minimize estimation error. The antecedent of a Bayesian rule is the conjunction of attribute-value pairs from the set V_2. The consequent is a local naive Bayesian classifier created from those training instances that satisfy the antecedent of the rule. THis local naive Bayesian classifier only uses those attributes that belong to the set V_1. During the generation of a Bayesian rule, the test instance to be classified is used to guide the selection of attributes for creating attribute-value pairs. The values in the attribute-value pairs are always the same as the corresponding attribute values of the test instance. The objective is to grow the antecedent of a Bayesian rule that ultimately decreases the errors of the local naive Bayesian classifier in the consequent of the rule. Leave-one-out cross validation is used to select the attribute values to be moved to the left of a lazy Bayesian rule. The structure of a Bayesian network for a lazy Bayesian rule is shown in Fig. 2, here $V_1 = A_1, A_2, \cdots, A_k$ and $V_2 = A_{k+1}, A_{k+2}, \cdots, A_n$. The general form of this lazy Bayesian rule can be simply expressed as $(A_{k+1} \wedge A_{k+2} \wedge \cdots \wedge A_n) \rightarrow NaiveBayesClassifier(A_1, A_2, \cdots, A_k)$. Both *LBR* and *TAN* can be viewed as variants of naive Bayes that relax the attribute independence assumption. *TAN* relaxes this assumption by allowing each attribute to depend upon at most one other attribute in addition to the class. *LBR* allows an attribute to depend upon many other attributes, but all attributes depend upon the same set of other attributes.

3. DESCRIPTION OF HEURISTIC LAZY BAYESIAN RULE ALGORITHM

The principle cause of LBR's inefficiency when large numbers of instances are to be classified is the selection for each such instance of the attributes to place in the antecedent of the rule. Our strategy in the new algorithm is to move as much of this computation to training time as possible, performing as much of the computation as possible once only at the time when the training data is first analysed. To this end we seek at training time to identify attributes that should not be considered as candidates for inclusion in an

Table 1: The heuristic lazy Bayesian rule algorithm

ALGORITHM: HLBR $(X, V, C, test, \alpha)$
INPUT: 1) X is the set of training instances,
2) V is the set of attributes,
3) C is the set of class values,
4) T is a test instance,
5) α is the significance level.
OUTPUT: a predicted class for the test instance.

$Candidates = \emptyset$; /* The candidate atrributes */
$GlobalNB = NB$ trained using X, V and C;
$Errors$ = leave-one-out errors on X of $LocalNB$;
FOR each attribute a DO
$ThisErrors$ = leave-one-out errors on X of LBR with a as the antecedent;
IF $ThisErrors < Errors$
THEN $Candidates = Candidates + a$;
FOR each instance $test \in T$ DO
$Cond = true$;
$BestNB = GlobalNB$;
$BestErrors = Errors$;
REPEAT
FOR each A in $Candidates$ whose value v_A on $test$ isn't missing DO
X_{subset} = instances in X with $(A = v_A)$;
$TempNB = NB$ trained using $(V - \{A\})$ on X_{subset};
$TempErrors$ = (leave-one-out errors on X_{subset} of $TempNB$)
+ (errors from $Errors$ for instances in $(X - X_{subset})$);
IF (($TempErrors < BestErrors$) AND ($TempErrors$ is significantly lower than $Errors$)
THEN $BestErrors = TempErrors$;
$BestNB = TempNB$;
$ABest = A$;
IF (an $ABest$ is found)
THEN $Cond = Cond \wedge (ABest = v_{ABest})$;
$LocalNB = BestNB$;
$X_{training} = X_{subset}$ corresponding to $ABest$;
$V = V - \{ABest\}$;
$Errors$ = leave-one-out errors on $X_{training}$ of $LocalNB$;
ELSE EXIT from the REPEAT loop;
Classify $test$ using $LocalNB$;

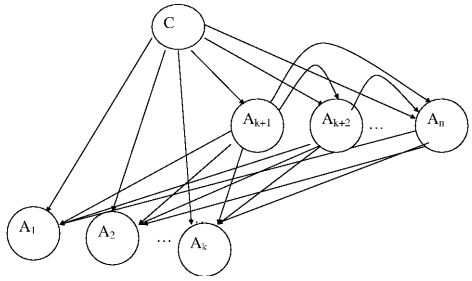


Figure 2: The structure of a Bayesian network for an example *LBR*

antecedent at classification time. To achieve this we perform leave-one-out cross validation for each attribute assessing the error when lazy Bayesian rules are formed using that and only that attribute in the antecedent. We restrict the candidates for consideration at classification time to those for which the cross validation error on this test is less than the cross validation error of naive Bayes. Our reasoning is that if there are harmful interdependencies between this and other attributes then this test will succeed. If there are no such harmful interdependencies then we should not consider the attribute as a candidate for inclusion in an antecedent. The heuristic lazy Bayesian rule algorithm is described in Table 1.

4. EXPERIMENTS

We compare the classification performance of four learning algorithms: the naive Bayesian classifier, *LBR*, *TAN* and our heuristic lazy Bayesian rule algorithm (*HLBR*). We use the naive Bayes classifier implemented in the Weka system, simply called Naive. We implemented a lazy Bayesian rule (*LBR*) learning algorithm and a tree-augmented Bayesian network (*TAN*) learning algorithm in the Weka system. All the experiments are run in the Weka system [10].

Thirty-five natural domains are used in the experiments shown in Table 2. Twenty-nine of these are all the data sets used in [7], the remaining are six larger data sets (German, Mfeat-mor, Satellite, Segment, Sign, and Vehicle). In Table 2, "Size" means the number of instances in a data set. "Class" means the number of values of a class attribute. "Attr." means the number of attributes, not including the class attribute. The error rate of each classifier on each domain is obtained by running 10-fold cross validation, and the random seed for 10-fold cross validation takes on the Weka default value. We also use the Weka default discretization method "weka.filters.DiscretizeFilter," an implementation of MDL discretization [11], as the discretization method for continuous values.

All experimental results for the error rates of the algorithms are shown in Table 3. The final two rows present the mean error across all data sets and the geometric mean error ratio. The latter measure is the geometric mean of the ratio for each data set of the error of respective algorithm divided by the error of HLBR. The geometric mean is used as the appropriate average for ratios. The average is at best a crude measure of overall performance as error rates on different data sets are incommensurable. The error ratio attempts to correct this problem by standardising the outcomes. Both

Table 2: Descriptions of Data

	Domain	Size	Class	Atts.
1	Annealing Processes	898	6	38
2	Audiology	226	24	69
3	Breast Cancer(Wisconsin)	699	2	9
4	Chess (KR-vs-KP)	3196	2	36
5	Credit Screening(Australia)	690	2	15
6	Echocardiogram	74	2	6
7	Germany	1000	2	20
8	Glass Identification	214	7	10
9	Heart Disease(Cleveland)	303	2	13
10	Hepatitis Prognosis	155	2	19
11	Horse Colic	368	2	22
12	House Votes 84	435	2	16
13	Hypothyroid Diagnosis	3163	2	25
14	Iris Classification	150	3	4
15	Labor Negotiations	57	2	16
16	LED 24(noise level=10%)	1000	10	24
17	Liver Disorders(bupa)	345	2	6
18	Lung Cancer	32	3	56
19	Lymphography	148	4	18
20	Mfeat-mor	2000	10	6
21	Pima Indians Diabetes	768	2	8
22	Post-Operative Patient	90	3	8
23	Primary Tumor	339	22	17
24	Promoter Gene Sequences	106	2	57
25	Satellite	6435	6	36
26	Segment	2310	7	19
27	Sign	12546	3	8
28	Solar Flare	1389	2	9
29	Sonar Classification	208	2	60
30	Soybean Large	683	19	35
31	Splice Junction Gene Seq.	3177	3	60
32	Tic-Tac-Toe End Game	958	2	9
33	Vehicle	846	4	18
34	Wine Recognition	178	3	13
35	Zoology	101	7	16

Table 3: Average Error Rate for Each Data Set

	Domain	Naive	LBR	TAN	HLBR
1	Annealing Processes	5.46	5.46	4.01	5.46
2	Audiology	29.20	29.20	29.20	29.20
3	Breast Cancer(Wisconsin)	2.58	2.58	2.58	2.58
4	Chess (KR-vs-KP)	12.36	3.57	5.07	3.85
5	Credit Screening(Australia)	15.07	14.64	14.35	14.64
6	Echocardiogram	27.48	27.48	28.24	27.48
7	Germany	24.60	24.70	24.80	24.60
8	Glass Identification	11.68	9.81	6.07	9.81
9	Heart Disease(Cleveland)	16.50	16.50	16.50	16.50
10	Hepatitis Prognosis	16.13	16.13	16.13	16.13
11	Horse Colic	20.11	19.29	18.48	19.29
12	House Votes 84	9.89	7.13	6.90	7.13
13	Hypothyroid Diagnosis	2.94	2.78	2.88	2.90
14	Iris Classification	6.67	6.67	6.00	6.67
15	Labor Negotiations	3.51	3.51	3.51	3.51
16	LED 24(noise level=10%)	24.50	24.70	24.50	24.70
17	Liver Disorders(bupa)	36.81	36.81	40.29	36.52
18	Lung Cancer	46.88	43.75	50.00	43.75
19	Lymphography	14.19	14.19	15.54	14.19
20	Mfeat-mor	30.65	29.95	30.10	29.90
21	Pima Indians Diabetes	25.00	24.87	25.39	24.87
22	Post-Operative Patient	28.89	28.89	30.00	28.89
23	Primary Tumor	48.97	49.85	49.85	49.85
24	Promoter Gene Sequences	8.49	8.49	8.49	8.49
25	Satellite	18.90	13.27	12.63	13.40
26	Segment	11.08	6.41	6.28	6.45
27	Sign	38.58	20.93	26.85	20.98
28	Solar Flare	18.57	15.69	16.85	15.62
29	Sonar Classification	25.48	25.96	23.56	28.37
30	Soybean Large	7.17	7.17	7.03	7.17
31	Splice Junction Gene Seq.	4.66	4.06	4.69	4.25
32	Tic-Tac-Toe End Game	29.54	14.61	28.81	13.99
33	Vehicle	39.48	31.44	31.68	31.44
34	Wine Recognition	3.37	3.37	3.37	3.37
35	Zoology	5.94	5.94	5.94	5.94
	Mean	**19.18**	**17.14**	**17.90**	**17.20**
	Geo. Mean	**1.14**	**0.99**	**1.02**	**1.00**

Table 4: Comparison of *LBR* to others

	WIN	LOSS	DRAW	p
Naive	16	4	15	0.012
TAN	15	11	9	0.557
HLBR	7	5	23	0.774

Table 5: Comparison of TAN to others

	WIN	LOSS	DRAW	p
Naive	17	9	9	0.168
LBR	11	15	9	0.557
HLBR	12	14	9	0.845

measures suggest that all of LBR, TAN, and HLBR enjoy substantially lower error than naive Bayes. The differences between LBR, TAN, and HLBR are much smaller, ordered from lowest to highest error LBR, then HLBR, then TAN.

The win/loss/draw record provides a more robust evaluation of relative performance over a large number of data sets. Tables 4, 5, and 6 present the win/loss/draw records for *LBR*, *TAN* and *HLBR*, respectively. This is a record of the number of data sets for which the nominated algorithm achieves lower, higher, and equal error to the comparison algorithm, measured to two decimal places. The final column presents the outcome of a two-tailed sign test. This is the probability that the observed outcome or more extreme would be obtained by chance if wins and losses were equiprobable. *LBR* and *HLBR* both achieve lower error than naive Bayes with frequency that is statistically significant at the 0.05 level. No win/loss/draw record indicates a significant difference in performance. This suggests that *LBR*, *HLBR* and *TAN* demonstrate comparable levels of error rate. *LBR* has a higher error rate than *TAN* in eleven data sets, and lower error rate in fifteen. *HLBR* has a higher error rate than *TAN* in twelve data sets, and lower error rate in fourteen. *LBR* has a lower error rate than the naive Bayes classifier in sixteen out of the thirty-five data sets, and a higher error rate in only four data sets. *HLBR* has a lower error rate than the naive Bayes classifier in seventeen out of the thirty-five data sets, and a higher error rate in only three data sets.

These results suggest that HLBR performs, in general, at a similar level of prediction accuracy to *LBR*. This comparable accuracy is obtained with far lower computation than LBR. The runtimes on all datasets of *LBR* and *HLBR* are shown in Table 7. Both *LBR* and *HLBR* were run on a dual-processor 1.7GHz Pentium 4 Linux computer with 2GB RAM. Runtimes less than one second are recorded as 1 second. Note that there is considerable variance in run times on the machine on which the experiments were run. The run time of LBR was higher than that of HLBR on 19 data sets and lower on 8. We calculated the ratio derived by dividing the run time of LBR by the run time of HLBR for each data set. The appropriate form of average for such ratio values is the geometric mean. The geometric mean was 1.4, indicating a substantial average advantage to HLBR.

Table 6: Comparison of HLBR to others

	WIN	LOSS	DRAW	p
Naive	17	3	15	0.002
LBR	5	7	23	0.774
TAN	14	12	9	0.845

Table 7: Runtime of LBR and HLBR (Unit: Seconds)

	Domain	LBR	HLBR
1	Annealing Processes	177	94
2	Audiology	1028	470
3	Breast Cancer(Wisconsin)	16	3
4	Chess (KR-vs-KP)	18468	6516
5	Credit Screening(Australia)	66	40
6	Echocardiogram	1	1
7	Germany	164	56
8	Glass Identification	5	2
9	Heart Disease(Cleveland)	6	4
10	Hepatitis Prognosis	4	5
11	Horse Colic	15	18
12	House Votes 84	26	28
13	Hypothyroid Diagnosis	3905	399
14	Iris Classification	1	1
15	Labor Negotiations	1	1
16	LED 24(noise level=10%)	696	186
17	Liver Disorders(bupa)	2	2
18	Lung Cancer	3	20
19	Lymphography	5	5
20	Mfeat-mor	156	117
21	Pima Indians Diabetes	25	9
22	Post-Operative Patient	1	1
23	Primary Tumor	192	50
24	Promoter Gene Sequences	16	131
25	Satellite	48923	40624
26	Segment	4652	896
27	Sign	11821	9670
28	Solar Flare	103	94
29	Sonar Classification	32	155
30	Soybean Large	1172	247
31	Splice Junction Gene Seq.	12406	4391
32	Tic-Tac-Toe End Game	34	36
33	Vehicle	106	116
34	Wine Recognition	3	3
35	Zoology	5	5

5. CONCLUSIONS

We present a heuristic variant of the lazy Bayesian rules classifier. HLBR seeks to reduce classification time when there are large numbers of instances to be classified by identifying some attributes that should never be considered as candidates for inclusion in the antecedent of a lazy Bayesian rule. Our experimental results suggest that HLBR is successful in this aim while also managing to retain a similar level of classification accuracy to the original LBR.

6. REFERENCES

[1] Mitchell, T. M.: Machine Learning. New York: The

McGraw-Hill Companies, Inc.. (1997) 154-199

[2] Domingos, P., Pazzani, M.: Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. In: Proceedings of the Thirteenth International Conference on Machine Learning. San Francisco, CA: Morgan Kaufmann Publishers, Inc. (1996) 105-112

[3] Kononenko, I.: Semi-Naive Bayesian Classifier. In: Proceedings of European Conference on Artificial Intelligence, (1991) 206-219

[4] Pazzani, M.: Constructive Induction of Cartesian Product Attributes. Information, Statistics and Induction in Science. Melbourne, Australia. (1996)

[5] Kohavi, R.: Scaling up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybird. In: Simoudis, E., Han, J.-W., Fayyad, U. M. (eds.): Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. Menlo Park, CA: AAAI Press. (1996) 202-207

[6] Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian Network Classifiers. Machine Learning, 29 (1997) 131–163

[7] Zheng, Z., Webb, G. I.: Lazy learning of Bayesian Rules. Machine Learning. Boston: Kluwer Academic Publishers.(2000) 1-35

[8] Keogh, E. J., Pazzani, M. J.: Learning Augmented Bayesian Classifiers: A Comparison of Distribution-Based and Classification-Based Approaches. In: Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics. (1999) 225-230

[9] Friedman, N., Kohavi, R., Yun, Y.: Lazy Decision Tree. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence. Menlo Park, CA: The AAAI Press. (1996) 717-724

[10] Witten, I. H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Seattle, WA: Morgan Kaufmann Publishers. (2000)

[11] Fayyad, U. M., Irani, K. B.: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence. (1993) 1022-1027

Averaged One-Dependence Estimators: Preliminary Results

Geoffrey I. Webb
School of Computer Science and Software Engineering
Monash University
Vic. 3800, Australia

Janice Boughton
School of Computer Science and Software Engineering
Monash University
Vic. 3800, Australia

Zhihai Wang
School of Computer Science and Software Engineering
Monash University
Vic. 3800, Australia

webb@infotech.monash.edu.au

ABSTRACT

Naive Bayes is a simple, computationally efficient and remarkably accurate approach to classification learning. These properties have led to its wide deployment in many online applications. However, it is based on an assumption that all attributes are conditionally independent given the class. This assumption leads to decreased accuracy in some applications. AODE overcomes the attribute independence assumption of naive Bayes by averaging over all models in which all attributes depend upon the class and a single other attribute. The resulting classification learning algorithm for nominal data is computationally efficient and achieves very low error rates.

1. INTRODUCTION

Naive Bayes has gained widespread application due to its simplicity, computational efficiency, direct theoretical basis, and competitive classification accuracy. It is particularly attractive for interactive applications due to the speed with which it can be applied. This speed is derived from its use of a simplifying attribute-independence assumption.

Naive Bayes has a long history of application in information retrieval [12] and has gained some popularity in the machine learning community. It has numerous desirable features. It is extremely efficient. It is provably optimal bar only for two explicit assumptions, the attribute independence assumption and the assumption that the estimates based on frequency are accurate. Finally, for many classification tasks its accuracy is extremely competitive with much more computationally intensive techniques [8; 11], especially when sample sizes are small [21].

However, where the attribute independence assumption is violated, its accuracy may suffer. It should be noted that not all violations of the attribute independence assumption matter. Naive Bayes will remain an optimal classifier so long as its probability estimate for the most probable class is higher than its probability estimate for any other class [3]. Nonetheless, harmful violations of the attribute independence assumption appear to be frequent in real-world application domains. In consequence, there is a sizeable body of literature addressing measures that can be applied to maintain the desirable features of naive Bayes while reducing the harmful effects of the attribute independence assumption [4; 6; 7; 9; 10; 11; 13; 14; 15; 18; 17; 21; 22].

Of these approaches, two in particular have demonstrated remarkable prediction accuracy, LBR [21] and TAN [4]. However, both these algorithms have high computational requirements reducing their utility in many data mining applications. We present a new algorithm that results from our efforts to attain the prediction accuracy of LBR and TAN without the computational overheads.

2. CLASSIFICATION BY CONDITIONAL PROBABILITY ESTIMATION

We assume a joint probability distribution X, Y, a sample X^*, Y^* drawn from X, Y, where each $x \in X$ is an object described by a vector of k attribute values $\langle x_1, x_2, \ldots x_k \rangle$ and each $y \in Y$ is a class label. The classification task is to assign a $y \in Y$ to a given $x \in X$.

Assuming that x is drawn at random from X, Y, error can be minimized by selecting $argmax_y P(y \mid x)$. As $P(y \mid x)$ is not generally known, one approach is to estimate it from X^*, Y^*. In general, $P(Z \mid W) = P(W \wedge Z)/P(W)$. If X^*, Y^* is a representative sample of X, Y then for any predicates W and Z, $P(W \wedge Z) \approx F(W \wedge Z, X^*, Y^*)$ and $P(W) \approx F(W, X^*, Y^*)$, where $F(Z, X, Y)$ denotes the frequency with which Z occurs in the reference distribution X, Y. However, to guard against problems associated with sampling error, when estimating population frequency from the sample frequency the exact sample frequency is usually adjusted providing a function on the sample frequency, $f(Z, X^*, Y^*)$, such as the Laplace error estimate [5] or an m-estimate [1]. In consequence,

$$P(y \mid x) = P(x \wedge y)/P(x) \quad (1)$$

$$\approx f(x \wedge y, X^*, Y^*)/f(x, X^*, Y^*). \quad (2)$$

However, the accuracy of this approximation will deteriorate as the values of $P(x \wedge y)$ and $P(x)$ decrease. Where k is large, the probabilities $P(x)$ and $P(x \wedge y)$ are likely to be extremely low, and hence the approximation in (2) will be poor. This problem can be circumvented with respect to estimating $P(x)$ by observing that $P(x) = \sum_{y \in Y} P(x \wedge y)$ and hence that it is necessary only to estimate the numerator for each class and the denominator in (1) can be calculated therefrom. Alternatively, if we wish only to identify the most probable class, we can observe that $P(x)$ is invariant across

classes and hence for any x, $P(y \mid x) \propto P(y)P(x \mid y)$ thus relieving us from the need to estimate $P(x)$. Nonetheless, it remains necessary to estimate $P(x \wedge y)$. In this context it is necessary to use less direct approximations for $P(y \mid x)$.
One approach is to use Bayes theorem:

$$P(Z \mid W) = \frac{P(Z)P(W \mid Z)}{P(W)}. \quad (3)$$

Bayes theorem holds irrespective of whether one adopts a Bayesian or a frequentist definition of probability. Hence its use need not imply a commitment to one or the other theoretical approach to probability. When estimating probabilities from data it can be used to replace the term $P(Z \mid W)$ with $\frac{P(Z)P(W \mid Z)}{P(W)}$ in contexts where the base terms in the latter can be estimated more reliably or accurately than direct estimation of the former.
However, using Bayes theorem to replace the estimation of $P(y \mid x)$ by the estimation of $P(y)P(x \mid y)/P(x)$ does not directly resolve the problem. If direct estimation of $P(y \mid x)$ is inaccurate then direct estimation of $P(x \mid y)$ will also be inaccurate because both require estimation of $P(x \wedge y)$.
Naive Bayes resolves this problem by assuming that the attributes are conditionally independent given the class. In consequence, the following equality is assumed:

$$P(x \mid y) = \prod_{i=1}^{k} P(x_i \mid y). \quad (4)$$

This allows the following approximation to be applied to estimate the conditional probabilities:

$$P(y \mid x) \approx \frac{f(y, X^*, Y^*) \prod_{i=1}^{k} \frac{f(x_i \wedge y, X^*, Y^*)}{f(y, X^*, Y^*)}}{\sum_{y' \in Y} \left(f(y', X^*, Y^*) \prod_{i=1}^{k} \frac{f(x_i \wedge y', X^*, Y^*)}{f(y', X^*, Y^*)} \right)} \quad (5)$$

where the denominator may be omitted when the objective is only to identify the most probable class, as it is invariant across classes.

3. LBR AND TAN

LBR and TAN reduce harmful effects of the attribute independence assumption by allowing models to be formed that represent limited attribute interdependencies. For each x to be classified, LBR selects a subset w of the values in x and assumes only that the remaining attributes are independent given $w \wedge y$. Thus, each x is classified using

$$P(y \mid x) \approx \frac{\frac{f(y \wedge w, X^*, Y^*)}{f(w, X^*, Y^*)} \prod_{i=1}^{k} \frac{f(x_i \wedge y \wedge w, X^*, Y^*)}{f(y \wedge w, X^*, Y^*)}}{\sum_{y' \in Y} \left(\frac{f(y' \wedge w, X^*, Y^*)}{f(w, X^*, Y^*)} \prod_{i=1}^{k} \frac{f(x_i \wedge y' \wedge w, X^*, Y^*)}{f(y' \wedge w, X^*, Y^*)} \right)} \quad (6)$$

This process is lazy, using a wrapper approach at classification time to select the $w \subseteq x$ that appears to best reduce error on the training data.
The models that LBR forms have all attributes dependent upon the same group of attributes. In contrast, TAN forms models in which each attribute depends on at most one other attribute. For attribute-value v we will denote the attribute upon which its attribute depends by $parent(v)$. TAN classifies each x using

$$P(y \mid x) \approx \frac{f(y, X^*, Y^*) \prod_{i=1}^{k} \frac{f(x_i \wedge x_{parent(x_i)} \wedge y, X^*, Y^*)}{f(x_{parent(x_i)} \wedge y, X^*, Y^*)}}{\sum_{y' \in Y} \left(f(y', X^*, Y^*) \prod_{i=1}^{k} \frac{f(x_i \wedge x_{parent(x_i)} \wedge y', X^*, Y^*)}{x_{parent(x_i)} \wedge f(y', X^*, Y^*)} \right)} \quad (7)$$

TAN selects the parent of each attribute at training time. A variant of TAN with high accuracy uses a wrapper to select the parents that appear to best reduce error [6]. We utilize this variant of TAN in the following work.
LBR and TAN have demonstrated comparable classification accuracy [16]. In separate evaluation, LBR has demonstrated classification accuracy comparable to that of boosting decision trees [22]. However, their very strong classification accuracy comes at considerable computational cost. While the lazy strategy adopted by LBR has negligible training costs, the cost to classify each object at classification time is relatively high. In contrast, TAN is relatively efficient at classification time, but model selection imposes substantial computational overheads at training time.

4. EFFICIENT CLASSIFICATION

We seek to retain the accuracy of LBR and TAN while reducing the computational overheads. These overheads result from two types of activity. The first is model selection, performed at training time by TAN and at classification time by LBR. The second is estimation of the required conditional probabilities.
One way to tackle the latter problem is to restrict the allowed interdependencies to each attribute depending upon the class and one other attribute, in other words to the use of probabilities $P(x_i|x_j \wedge y)$. That is, we restrict the space of models that we will consider to 1-dependence Bayesian classifiers [14], the space of TAN models. 1-dependence conditional probabilities can be estimated efficiently by using joint frequencies stored in a three dimensional table, indexed in one dimension by x_i, in another by x_j and in the third dimension by y. This table can be formed at training time in a single scan through the data. Utilizing greater interdependencies requires higher dimensional tables, potentially resulting in infeasible memory requirements.
By the product rule, for any $x_i \in x$,

$$P(x \wedge y) = P(y \wedge x_i)P(x \mid y \wedge x_i). \quad (8)$$

Hence,

$$P(y \mid x) = \frac{P(y \wedge x_i)P(x \mid y \wedge x_i)}{P(x)}. \quad (9)$$

Utilizing (9) instead of (1) allows us to make a weaker, and hence potentially less harmful, attribute independence assumption than (4),

$$P(x \mid y \wedge x_i) = \prod_{j=1}^{k} P(x_j \mid y \wedge x_i). \quad (10)$$

This allows the approximation,

$$P(y \mid x) \approx \frac{f(x_i \wedge y, X^*, Y^*) \prod_{j=1}^{k} \frac{f(x_j \wedge x_i \wedge y, X^*, Y^*)}{f(x_i \wedge y, X^*, Y^*)}}{\sum_{y' \in Y} \left(f(x_i \wedge y', X^*, Y^*) \prod_{j=1}^{k} \frac{f(x_j \wedge x_i \wedge y', X^*, Y^*)}{f(x_i \wedge y', X^*, Y^*)} \right)}. \quad (11)$$

(10) is potentially less harmful than (4) because it assumes independence between fewer attributes (x_i is not considered independent of the other attributes) and independence is assumed under strong constraints (the remaining attributes are independent given y and x_i). The estimate (11) can only be less accurate than the estimate (5) if the estimates of the individual probabilities are less accurate. The only systematic effect that might cause this to occur is that the frequencies will be lower and hence derived from fewer examples. We can reduce the impact of this effect by restricting the application of (11) to attribute values x_i which occur with sufficient frequency for us to have confidence that estimation accuracy will not be significantly affected. Note that this restriction implies the use of lazy learning, as the value of an attribute is only determined with respect to a given object that we wish to classify.

Use of (5) reduces the problem of model selection to selecting between up to k models, the models in which every attribute depends upon the class and the same single attribute such that the value of that attribute x_i occurs with sufficient frequency for us to have confidence in the accuracy of its estimation. This leaves us with the problem of how to select the appropriate model. We note that there are two problems associated with model selection. The first is that it can be expected to lead to variance. The second is that it must entail computational overheads, the performance of the necessary computations to determine which is the preferred model. This reasoning led us to seek models that allowed for attribute interdependencies without a process of selecting which interdependencies to represent. We achieve this by averaging across all models of the form (11) such that x_i occurs at least 100 times in the training data. Note that averaging across all such models is justified as (9) holds for any value of i. Hence,

$$P(y \mid x) = \frac{\sum_{i=1}^{k} \frac{P(y \wedge x_i) P(x \mid y \wedge x_i)}{P(x)}}{k}. \quad (12)$$

When we replace each $P(y \wedge x_i) P(x \mid y \wedge x_i)$, averaging multiple estimates of this quantity with different values of i offers the opportunity to average out estimation errors. If the estimation errors are normally distributed with mean 0, the greater the number of values of i included, the lower the expected estimation error.

The resulting system classifies using the following.

$$P(y \mid x) \approx \frac{\sum_{i:C(x_i) \geq 100} \frac{f(x_i \wedge y, X^*, Y^*) \prod_{j=1}^{k} \frac{f(x_i \wedge x_j \wedge y, X^*, Y^*)}{f(x_i \wedge y, X^*, Y^*)}}{\sum_{y' \in Y} \left(f(x_i \wedge y', X^*, Y^*) \prod_{j=1}^{k} \frac{f(x_i \wedge x_j \wedge y', X^*, Y^*)}{f(x_i \wedge y', X^*, Y^*)} \right)}}{|\{i : C(x_i, X^*, Y^*) \geq 100\}|} \quad (13)$$

where $C(x_i)$ is the number of training objects that have attribute value x_i.

As $\sum_{y' \in Y} \left(f(x_i \wedge y', X^*, Y^*) \prod_{j=1}^{k} \frac{f(x_i \wedge x_j \wedge y', X^*, Y^*)}{f(x_i \wedge y', X^*, Y^*)} \right)$ is invariant across classes, to select the class that maximizes the estimate of $P(y \mid x)$ it is necessary only to select the class that maximizes the following.

$$\frac{\sum_{i:C(x_i, X^*, Y^*) \geq 100} f(x_i \wedge y, X^*, Y^*) \prod_{j=1}^{k} \frac{f(x_i \wedge x_j \wedge y, X^*, Y^*)}{f(x_i \wedge y, X^*, Y^*)}}{|\{i : C(x_i, X^*, Y^*) \geq 100\}|} \quad (14)$$

Table 1: Training time algorithm

INPUTS: training set X^*, Y^*,
number of attributes k, and
number of classes m

OUTPUTS: joint frequency vector $freq$,
class frequency vector $cfreq$,
attribute frequency vector $afreq$,
attribute-value frequency vector $vfreq$, and
item count $count$

Initialize frequencies
$count \leftarrow 0$
Initialize all elements of $freq$, $cfreq$, $afreq$, and $vfreq$ to 0
Accumulate frequencies
FOR EACH $x, y \in X^*, Y^*$
  $count \leftarrow count + 1$
  $cfreq[y] \leftarrow cfreq[y] + 1$
  FOR $i \leftarrow 1$ **TO** k
    IF x_i is known
      $afreq[i] \leftarrow afreq[i] + 1$
      $vfreq[x_i] \leftarrow vfreq[x_i] + 1$
      FOR $j \leftarrow 1$ **TO** k
        IF x_j is known
          $freq[y, x_i, x_j] \leftarrow freq[y, x_i, x_j] + 1$
        END IF
      END FOR
    END IF
  END FOR
END FOR

5. AVERAGED ONE-DEPENDENCE ESTIMATORS

Averaged One-Dependence Estimators (AODE) use (14) for classification. At training time they produce a three dimensional joint frequency table, indexed in one dimension by the values of the class and in the remaining two dimensions by the values of the attributes. The algorithm for this process is presented in Table 1. Note that we allow for missing attribute values but not for missing class values. To apply the algorithm to allow for missing class values it is possible to add a preprocess that removes all such objects.

At classification time we utilize the joint frequency table and (14) for classification as follows. We use the m-estimate [1] to produce conservative estimates of conditional probabilities from joint frequencies. To estimate $P(x_i | x_j \wedge y)$ we use an m-estimate with weight 0.5 and a prior defined by the frequency of the attribute value in the data as a whole. We allow for the possibility that some attribute values are missing, resulting in the following.

$$\hat{P}(x_i | x_j \wedge y) = \frac{freq[y, x_i, x_j] + 0.5}{freq[y, x_j, x_j] + 0.5 \times vfreq[x_i]} \quad (15)$$

To estimate $P(x_i \wedge y)$ we use an m-estimate with weight 0.5 and a prior defined by the product of the frequencies of the two terms.

$$\hat{P}(x_i \wedge y) = \frac{freq[y, x_i, x_i] + 0.5}{afreq[i] + 0.5 \times vfreq[x_i] \times cfreq[y]} \quad (16)$$

The algorithm to return a vector containing probability estimates $\hat{P}(y|x)$ for each $y \in Y$ is presented in Table 2.

AODE can be viewed as Bayesian averaging with uniform priors over all plausible models that have all attributes depend upon a single attribute and the class. Another perspective from which AODE can be viewed is as ensemble learning.

Note, that while there is some superficial similarity between AODE and exact model averaging with naive Bayesian classifiers [2], the differences are very substantial. Both predict by averaging the estimated class probabilities of multiple models. Exact model averaging averages over naive Bayes (zero-dependence) models formed with different subsets of the available attributes. Hence, it cannot successfully model attribute inter-dependencies. In contrast, AODE averages over models that directly model attribute inter-dependencies and hence has the potential to overcome the attribute independence problem. Exact model averaging aggregates models formed by feature subset selection. AODE aggregates models formed by modelling alternative attribute inter-dependencies.

5.1 Computational complexity

The time complexity of training is $O(nk^2)$, where n is the number of training objects and k is the number of attributes. Further, the operations performed in the inner loops involve minimal computation, resulting in minor computational overheads for training for the numbers attributes normally encountered (up to hundreds of attributes). Training time complexity is linear with respect to training set size. Thus, AODE is an efficient algorithm for classification from large datasets.

The time complexity of classifying an object is $O(mk^2)$, where m is the number of classes and k the number of attributes. Again the operations performed within the inner loops involve little computation, resulting in computationally efficient classification for the numbers of classes and attributes normally encountered.

The main space requirement for the algorithm is for the table containing the joint frequencies, which is $O(mv^2)$, where m is the number of classes and v is the number of attribute values. For typical learning problems involving up to tens of classes and hundreds of attributes this entails relatively minor space requirements. Note that training can be performed by a single sequential scan through the data. There is no need to retain training data in memory.

5.2 Sensitivity to large numbers of attributes

Abstract analysis suggests that the performance of AODE will decline as the numbers of attributes increase. One dimension of the algorithm's sensitivity to large numbers of attributes is the time and space complexity of the algorithm. Another dimension is that the benefit of the algorithm's reduction in the attribute independence assumption may be diluted as the number of attributes increases. Consider a situation where there is a pairwise interdependence between two attributes A and B. The probability estimates in the

Table 2: Probability estimation algorithm

INPUTS: object x,
number of attributes k,
number of classes m,
vector of the number of values for each attribute v,
joint frequency vector $freq$,
class frequency vector $cfreq$,
count of objects for which a value is known for an attribute $afreq$,
attribute-value frequency vector $vfreq$, and
item count $count$

OUTPUT: conditional probability vector $prob$,

Initialize values
FOR $i \leftarrow 1$ **TO** m, $prob[i] \leftarrow 0.0$
$sum \leftarrow 0.0$
Calculate probabilities
FOR $y \leftarrow 1$ **TO** m
  $prob[y] \leftarrow 0.0$
  $attcount \leftarrow 0$
  FOR $i \leftarrow 1$ **TO** k
    IF x_i is known **AND** $vfreq[x_i] \geq 100$
      $attcount \leftarrow attcount + 1$
      $p \leftarrow \hat{P}(x_i \wedge y)$
      FOR $j \leftarrow 1$ **TO** k
        IF x_j is known, $p \leftarrow p \times \hat{P}(x_j|x_i \wedge y)$
      END FOR
      $prob[y] \leftarrow prob[y] + p$
    END IF
  END FOR
  If no attribute value occurs with sufficient frequency, revert to naive Bayes
  IF $attcount = 0$
    $prob[y] \leftarrow NB(x, y)$
  ELSE
    $prob[y] = prob[y]/attcount$
  END IF
  $sum \leftarrow sum + prob[y]$
END FOR
Normalize to obtain probabilities
FOR $y \leftarrow 1$ **TO** m, $prob[y] \leftarrow prob[y]/sum$

two submodels formed when all attributes depend on either A or B will be correct with respect to the interdependency. However, in the other submodels, each formed with all attributes depending upon another attribute C, the probability estimates will treat A and B as independent, except in so far as their interdependence is captured by their mutual interdependencies with C. In many cases interdependencies between two variables will be captured in part by their mutual interdependencies with other variables. For example, if we assume that *senility* and *nocturia* are both correlated with *age*. In this case *senility* and *nocturia* will be interdependent. However, for any given value of *age* they might be independent. Nonetheless, as the number of other attributes increases and hence the number of submodels averaged increases, the advantage gained from undoing individual interdependencies in individual submodels might be diluted.

6. EXPERIMENTS

To evaluate the performance of AODE, it was implemented in Weka [19] and compared to existing Weka implementations of naive Bayes (NB), LBR, TAN, and C4.5 (J48). Note that the implementations of LBR and TAN are our own implementations, created for previous research. The implementations of naive Bayes and C4.5 are part of the Weka distribution.

All four algorithms were applied to 39 data sets. These data sets are formed around a core of twenty-nine data sets used in previous related research [3; 21] augmented by a variety of larger data sets. These larger datasets were added as the original twenty-nine datasets were all relatively small and the techniques of LBR and TAN have greatest scope to improve upon NB when more data is available. Note that three further large datasets, Musk, Census-income, and Cover-type, are not included in these results because at least one of LBR or TAN could not complete processing for them within a one-fortnight time limit.

Each algorithm was applied to each data set using ten-fold cross validation on a dual-processor 1.7Ghz Pentium 4 Linux computer with 2Gb RAM. However, due to the long compute times of TAN and LBR, a small number of computations had to be performed on an alternative machine. This has not affected the error results obtained. These computations are identified and excluded from time analyses.

Numeric attributes were discretized using ten-bin discretization. Note that the decision tree learner is applied to the discretized data, rather than being allowed to select its own cut-points. Its error would be lower if it were allowed to select its own cut-points. However, we believe that the error of the probabilistic techniques could also be reduced by application of alternative discretization techniques [20]. All algorithms have been applied to the same discretized data in order to compare their performance on nominal data.

6.1 Relative Error

Table 3 presents the data sets used, the number of objects in each data set, the number of attributes by which each object is described, and the mean error of each algorithm on the data set. At the foot of the table the mean error across all data sets is provided together with the geometric mean of the error ratio obtained by dividing the error of AODE by the error of the alternative algorithm. AODE achieves the lowest mean error across all data sets. However, this is at best a gross measure of performance, as error rates across data sets are incommensurable. The error ratio corrects for this. The geometric mean is the appropriate approach to averaging ratio values such as the error ratio. A value less than 1.0 indicates an advantage to AODE. On this measure AODE registers very large advantage compared with NB and J48 and sizeable advantage compared to TAN and LBR.

Table 4: Win–Draw–Loss Records

	NB	**TAN**	**LBR**	**J48**
AODE WDL	23–8–8	18–6–15	20–4–15	27–2–10
p	0.005	0.364	0.250	0.004

Note that we do not perform statistical tests of significance on differences in performance on individual data sets as the large number of such tests that would be entailed would result in extremely high levels of expected type 1 error. Instead we perform statistical tests on the win-draw-loss records, as presented in Table 4. This table presents the number of data sets for which AODE obtains lower error, equal error (measured to one decimal place), and higher error, with respect to each alternative algorithm. The outcome of a binomial sign test is also presented. This is the probability that the observed ratio of wins to losses or higher would be obtained by chance if both were equi-probable. As can be seen, AODE enjoys highly significant ratios of wins to losses against NB and J48. While AODE wins more often than it loses against TAN and LBR, these ratios are not significant at the 0.05 level.

Our abstract analysis of AODE identified a potential vulnerability of the algorithm when processing data sets utilizing large numbers of attributes. The two data sets with the most attributes are syncon and sonar. For the former AODE achieves substantially lower error than any of the other algorithms. For the latter AODE achieves lower error than all alternatives other than TAN. The next seven data sets in descending order of number of attributes are promoters, lung-cancer, chess, anneal, satellite, pioneer, and ionosphere. For one of these data sets, satellite, AODE achieves lower error than all alternatives. For two, promoters, and lung-cancer, AODE shares the lowest error with NB, LBR, and, in one case, TAN. For none of the remaining four data sets does the error of AODE reach the level of the highest of the alternatives. These outcomes suggest that the expected vulnerability of AODE to large numbers of attributes is not apparent for the numbers of attributes investigated in these experiments. The question of whether the expected vulnerability becomes a problem for larger numbers of attributes is an important area for future investigation.

6.2 Relative Compute Time

Table 5 presents the total compute time of each algorithm by data set. These times are the total time to run Weka to complete the cross-validation task and hence include data input as well as learning and classification. Note that due to the long compute times of LBR and TAN some computations were performed on an alternative computer and hence are excluded as they are incommensurable with the times listed. Note also that compute times on the compute server used are highly variable from run to run. Further, our implementations of AODE, TAN, and LBR are not optimized. Finally, the structure of this task is unfavorable to LBR,

Table 3: Mean error

	objects	**atts**	**AODE**	**NB**	**TAN**	**LBR**	**J48**
waveform	100000	22	3.6	6.9	4.4	3.7	3.9
adult	48842	15	16.7	18.0	16.0	15.0	15.7
letter-recognition	20000	17	13.2	30.0	16.5	16.0	19.4
sign	12546	9	29.3	38.6	26.6	20.9	20.4
pendigits	10992	17	2.2	12.9	3.6	3.3	9.8
pioneer	9150	37	4.1	9.8	3.5	4.8	4.3
satellite	6435	37	11.3	18.9	12.4	13.3	15.3
hypothyroid	3163	26	2.9	2.9	2.9	2.8	2.5
segment	2310	20	5.7	11.1	6.3	6.4	6.5
mfeat-mor	2000	7	30.2	30.7	30.0	30.0	30.1
german	1000	21	24.4	24.6	24.8	24.7	30.4
led	1000	8	26.4	26.2	25.9	26.2	27.1
ttt	958	10	26.1	29.5	28.6	14.6	15.1
anneal	898	39	5.0	5.5	4.0	4.1	9.2
vehicle	846	19	27.9	39.5	31.3	31.4	30.3
breast-cancer-wisc.	699	10	2.7	2.6	2.6	2.6	5.7
crx	690	16	13.6	15.1	14.4	14.6	15.5
balance-scale	625	5	9.9	8.6	8.6	8.6	36.6
syncon	600	61	1.0	3.0	3.0	2.5	23.0
chess	551	40	11.4	12.7	10.0	11.1	8.3
house-votes-84	435	17	6.4	9.9	6.7	7.1	4.1
horse-colic	368	22	20.4	20.1	18.5	19.0	15.0
ionosphere	351	35	9.4	9.1	9.4	9.1	13.7
bupa	345	7	36.5	36.8	39.7	36.8	39.7
primary-tumor	339	18	48.1	49.0	49.9	49.9	57.8
cleveland	303	14	19.1	16.5	16.5	16.5	21.5
hungarian	294	14	15.7	15.3	15.7	16.0	20.4
heart	270	14	15.6	15.2	15.2	15.2	23.7
new-thyroid	215	6	7.0	8.4	7.4	8.4	7.0
glass	214	10	24.8	25.2	24.3	25.7	23.8
sonar	208	61	24.5	25.5	23.6	26.0	32.2
wine	178	14	3.4	3.4	3.4	3.4	21.4
hepatitis	155	20	15.5	16.1	16.1	14.8	16.8
iris	150	5	6.7	6.7	6.0	6.7	4.0
echocardiogram	131	7	27.5	27.5	28.2	28.2	35.9
promoters	106	58	8.5	8.5	8.5	8.5	21.7
post-operative	90	9	28.9	28.9	30.0	30.0	28.9
labor-neg	57	17	3.5	3.5	3.5	10.5	29.8
lung-cancer	32	57	46.9	46.9	50.0	46.9	53.1
Mean error			16.3	18.4	16.6	16.3	20.5
Geometric mean error ratio			1.00	0.82	0.96	0.95	0.72

which is relatively efficient for small test sets and relatively inefficient for large test sets. Hence, these results should be regarded as broadly indicative only. The mean CPU time across all data sets is also presented. Finally, we present the geometric mean of the time for AODE divided by the time for each alternative algorithm. These latter values are presented for all algorithms excluding results for waveform and pioneer as times are not available for all algorithms on these data sets. They are also presented including all results for AODE, NB, and J48 only.

Where there are large numbers of attributes the compute time of AODE increases significantly from that of naive Bayes. This is especially apparent for pioneer. However, while up to 12 times slower than naive Bayes, the compute times are still low, in no case exceeding two minutes to complete ten-fold cross validation. For most data sets the compute time is lower than the decision tree learner. For no data set does the compute time of AODE exceed that of TAN or LBR. For most data sets it is dramatically faster.

The geometric mean time ratios indicate that naive Bayes holds a substantial advantage over AODE, but that AODE holds a substantial advantage over the decision tree algorithm and a massive advantage over TAN and LBR.

7. CONCLUSION

AODE is a new classification learning algorithm that weakens the attribute independence assumption of naive Bayes without undue increase in computational complexity. In these preliminary investigations, this algorithm is demonstrated to deliver accuracy at least comparable to the state-of-the-art LBR and TAN algorithms without their high computational cost. The training time of the algorithm is linear with respect to data set size. Training requires only a single sequential scan of the data. These features make the algorithm highly attractive for processing very large data sets.

Our theoretical analysis of the algorithm leads us to expect that it is more suited to data sets described by fewer rather than more attributes. However, no vulnerability to large numbers of attributes was apparent in our experiments using data sets with as many as 61 attributes.

We believe that AODE is successful in achieving our aim of significantly and substantially reducing the error of naive Bayes without substantially increasing compute time. We believe that the resulting algorithm is extremely well suited to applications that require efficient computation, such as on-line applications, as well as to applications requiring processing of very large training sets.

8. REFERENCES

[1] B. Cestnik, I. Kononenko, and I. Bratko. ASSISTANT 86: A knowledge-elicitation tool for sophisticated users. In I. Bratko and N. Lavrač, editors, *Progress in Machine Learning*, pages 31–45. Sigma Press, Wilmslow, 1987.

[2] D. Dash and G. F. Cooper. Exact model averaging with naive Bayesian classifiers. In *Proceedings of the Nineteenth International Conference on Machine Learning, ICML 2002*, pages 91–98, Sydney, July 2002. Morgan Kaufmann.

[3] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 105–112, Bari, Italy, 1996. Morgan Kaufmann.

[4] N. Friedman and M. Goldszmidt. Building classifiers using Bayesian networks. In *AAAI-96*, pages 1277–1284, 1996.

[5] I. J. Good. *Probability and the Weighing of Evidence.* Charles Griffin and Co. Ltd., London, 1950.

[6] E. Keogh and M. Pazzani. Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *International Workshop on Artificial Intelligence and Statistics*, pages 225–230, 1999.

[7] R. Kohavi. Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. In *KDD-96*, Portland, Or, 1996.

[8] I. Kononenko. Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber, and M. van Someren, editors, *Current Trends in Knowledge Acquisition.* IOS Press, Amsterdam, 1990.

[9] I. Kononenko. Semi-naive Bayesian classifier. In *ECAI-91*, pages 206–219, 1991.

[10] P. Langley. Induction of recursive Bayesian classifiers. In *Proceedings of the 1993 European Conference on Machine Learning*, pages 153–164, Vienna, 1993. Springer-Verlag.

[11] P. Langley and S. Sage. Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406, Seattle, WA, 1994. Morgan Kaufmann.

[12] D. D. Lewis. Naive Bayes at forty: The independence assumption in information retrieval. In *ECML-98: Proceedings of the Tenth European Conference on Machine Learning*, pages 4–15, Chemnitz, Germany, April 1998. Springer.

[13] M. J. Pazzani. Constructive induction of Cartesian product attributes. In *ISIS: Information, Statistics and Induction in Science*, pages 66–77, Melbourne, Aust., August 1996. World Scientific.

[14] M. Sahami. Learning limited dependence Bayesian classifiers. In *Proceedings 2nd International Conference on Knowledge Discovery and Data Mining*, pages 334–338. AAAI Press, 1996.

[15] M. Singh and G. M. Provan. Efficient learning of selective Bayesian network classifiers. In *Proceedings of the 13th International Conference on Machine Learning*, pages 453–461, Bari, 1996. Morgan Kaufmann.

[16] Z. Wang and G. I. Webb. Comparison of lazy bayesian rule and tree-augmented bayesian learning. In *To appear in Proceedings of the IEEE International Conference on Data Mining, ICDM-2002*, 2002.

Table 5: Compute time in CPU seconds

	objects	**atts**	**AODE**	**NB**	**TAN**	**LBR**	**J48**
waveform	100000	22	108	33	*	1548801	702
adult	48842	15	19	10	4792	183699	522
letter-recognition	20000	17	34	8	8710	147454	302
sign	12546	9	4	3	170	5376	112
pendigits	10992	17	16	4	2714	12395	73
pioneer	9150	37	98	8	*	*	18
satellite	6435	37	25	4	20902	15176	1
hypothyroid	3163	26	6	2	474	2937	6
segment	2310	20	4	2	409	1384	8
mfeat-mor	2000	7	1	1	9	95	4
german	1000	21	2	1	39	100	2
led	1000	8	1	1	8	145	1
ttt	958	10	1	1	10	31	1
anneal	898	39	5	1	1323	1001	6
vehicle	846	19	2	1	128	60	3
breast-cancer-wisconsin	699	10	1	1	2	13	1
crx	690	16	1	1	26	71	2
balance-scale	625	5	1	1	1	3	1
syncon	600	61	6	1	510	235	7
chess	551	40	1	1	414	227	2
house-votes-84	435	17	1	1	28	15	1
horse-colic	368	22	1	1	82	19	2
ionosphere	351	35	2	1	58	16	2
bupa	345	7	1	1	2	2	1
primary-tumor	339	18	1	1	18	145	1
cleveland	303	14	1	1	4	4	1
hungarian	294	14	1	1	7	6	1
heart	270	14	1	1	3	4	1
new-thyroid	215	6	1	1	1	1	1
glass	214	10	1	1	3	2	1
sonar	208	61	2	1	330	18	2
wine	178	14	1	1	2	2	1
hepatitis	155	20	1	1	7	4	1
iris	150	5	1	1	1	1	1
echocardiogram	131	7	1	1	1	1	1
promoters	106	58	1	1	52	5	1
post-operative	90	9	1	1	1	1	1
labor-neg	57	17	1	1	2	1	1
lung-cancer	32	57	1	1	44	3	1
Mean CPU time^{+}			4	2	1116	10018	29
Geometric mean time ratio^{+}			1.00	1.51	0.05	0.04	0.68
Mean CPU time			9	3	*	*	46
Geometric mean time ratio			1.00	1.64	*	*	0.69

* indicates computation performed on alternative computer and hence included.
$^{+}$ Values exclude waveform and pioneer for which some values are not available.

[17] G. I. Webb. Candidate elimination criteria for Lazy Bayesian Rules. In *Proceedings of the Fourteenth Australian Joint Conference on Artificial Intelligence*, pages 545–556, Adelaide, December 2001. Springer.

[18] G. I. Webb and M. J. Pazzani. Adjusted probability naive Bayesian induction. In *Proceedings of the Eleventh Australian Joint Conference on Artificial Intelligence*, pages 285–295, Brisbane, Australia, 1998. Springer.

[19] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, CA, 1999.

[20] Y. Yang and G. I. Webb. Proportional k-interval discretization for naive-Bayes classifiers. In *12th European Conference on Machine Learning (ECML'01)*, pages 564–575. Springer, 2001.

[21] Z. Zheng and G. I. Webb. Lazy learning of Bayesian Rules. *Machine Learning*, 41(1):53–84, 2000.

[22] Z. Zheng, G. I. Webb, and K. M. Ting. Lazy Bayesian Rules: A lazy semi-naive Bayesian learning technique competitive to boosting decision trees. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-99)*, pages 493–502, Bled, Slovenia, 1999. Morgan Kaufmann.

SemiDiscrete Decomposition: A Bump Hunting Technique

S. McConnell
School of Computing, Queen's University, Kingston, Canada.
mcconnell@cs.queensu.ca

D.B. Skillicorn
School of Computing, Queen's University, Kingston, Canada, and
Faculty of Information Technology, University of Technology, Sydney.
skill@cs.queensu.ca

ABSTRACT

Semidiscrete decomposition (SDD) is usually presented as a storage-efficient analogue of singular value decomposition. We show, however, that SDD actually works in a completely different way, and is best thought of as a bump-hunting technique; it is extremely effective at finding outlier clusters in datasets. We suggest that SDD's success in text retrieval applications such as latent semantic indexing is fortuitous, and occurs because such datasets typically contain a large number of small clusters.

1. INTRODUCTION

A dataset in tabular form, with n rows representing *objects*, and m columns representing *attributes*, has a natural geometric representation. Each object can be considered as a point in the m-dimensional space spanned by the attributes. A common data mining approach is to *cluster* these points as a way to resolve structure in the dataset. For example, objects that are similar may fall in the same cluster.

When m is large, this geometrical insight is not as useful as it might be because high-dimensional spaces are hard to work with. For example, the distance from a point to its nearest and farthest neighbour tends to be almost the same. Hence, although many clustering techniques are known, they are not necessarily effective as m becomes large (say > 15).

Singular Value Decomposition (SVD) [7] is a dimension-reduction technique that can be used to generate a low-dimensional representation of a high-dimensional dataset. Clusters may be immediately apparent in this representation, or it may be used as input to other clustering techniques. We present SVD in some detail because it is, in some ways, analogous to semidiscrete decomposition.

SVD transforms an m-dimensional space into a new m-dimensional space whose axes have two useful properties: they are orthonormal, and there is an ordering of the axes such that the variation in the original data is concentrated, as far as possible, along the earlier axes. Hence ignoring dimensions associated with later axes provides a faithful representation of the original data in a lower dimensional space. SVD has been used successfully, under the name *latent semantic indexing* [1; 2], for information retrieval in text. In this application, the early dimensions capture much of the information about the correlated use of terms, and retrieval performance is very strong.

Given an $n \times m$ matrix A with $n \geq m$, the singular value decomposition of A is

$$A = U\Sigma V^T$$

where U is $n \times m$, Σ is $m \times m$, and V is $m \times m$. U and V are orthonormal, and Σ is a diagonal matrix whose elements are a set of non-negative, decreasing singular values, σ_1, σ_2, ..., σ_r (where r is the rank of A, $r \leq m$). The rows of U represent coordinates of the corresponding rows of A in a space spanned by the columns of V. Symmetrically, the rows of V represent the coordinates of the corresponding columns of A in a space spanned by the columns of U. A rank k approximation to A can be computed by multiplying the matrix consisting of the first k columns of U, the upper left $k \times k$ submatrix of Σ and the first k rows of V^T.

The *semidiscrete decomposition* (SDD) [12; 16] was developed for image compression, but has been used more often as a substitute for latent semantic indexing [9; 10; 11]. It is usually presented as a weak analogue of SVD, in which the axes of the transformed space are no longer orthonormal, and the coordinates of points in the transformed space are taken only from the set $\{-1, 0, 1\}$. The transformation itself is a mixed programming optimization problem. Its complexity is $(n+m)^3$, which makes exact solutions feasible for some data mining problems. There is also a heuristic approximation algorithm which is described below. The benefit of SDD over SVD is usually considered to be the compactness of the representation of the transformed space, since each coordinate can be stored in 2 bits.

Given a matrix A, the semidiscrete decomposition of A of dimension k (now unrelated to the rank of A) is

$$A_k = X_k D_k Y_k$$

where the entries of X_k and Y_k are from $\{-1, 0, 1\}$, and D_k is a diagonal matrix. X is $n \times k$, D is $k \times k$, and Y is $k \times m$. We drop the subscripts if they are not important.

Let x_i be the ith column of X, d_i the ith diagonal element of D, and y_i the ith row of Y. The standard algorithm for computing the SDD generates a new column, diagonal element, and row on each step. Let A_0 by the $n \times m$ matrix of zeroes. The algorithm is, for each step i:

1. Subtract the current approximation, A_{i-1}, from A to get a residual matrix R_i.

2. Find a triple (x_i, d_i, y_i) that minimizes

$$\| R_i - d_i x_i y_i \|^2 \qquad (*)$$

where x_i is $n \times 1$ and y_i is $1 \times m$. The standard algorithm uses the following heuristic:

(a) Choose an initial y_i.

(b) Solve $(*)$ for x_i and d_i using this y_i.

(c) Solve $(*)$ for y_i and d_i using the x_i from the previous step.

(d) Repeat until some convergence criterion is satisfied.

3. Repeat until $i = k$.

Matlab and C code for this algorithm is available from `www.cs.umd.edu/users/oleary/SDDPACK/`.

The rows of X are the coordinates of an object in the space defined by the new axes described by Y. However, these coordinates can be interpreted in a different, and useful, way. If we divide the objects according to whether they have a -1, 0, or 1 in the first column of X, we have separated them into three classes. The second and subsequent columns can be used to further subdivide the objects, producing a ternary decision tree structure. This decision tree can be used for prediction of new unseen objects, but also provides a similarity metric for training objects based on various forms of distance in the decision tree. Note that, unlike conventional decision trees, this is an *unsupervised* process – we do not need to know class labels for training data.

The resulting ternary tree is a full tree of depth k, but we can reasonably truncate it whenever a set of objects is not separated by the remaining levels. The resulting set of leaves defines a clustering of the data, and also, in an obvious way, a hierarchical clustering or dendrogram.

However, we have not discussed the order in which the axes (of Y) are chosen so, while we know that the dataset can be partitioned into subsets, we do not know which of these partitions are most significant. This turns out to be a difficult question, which we address in the next section.

2. WHAT SDD IS DOING

We illustrate the way in which SDD constructs axes of the transformed space using a small matrix with an obvious structure. Suppose that

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 & 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 & 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The (first 5 columns of the) X matrix produced by SDD are

$$X = \begin{bmatrix} 1 & 0 & -1 & 0 & -1 \\ 1 & 0 & -1 & 0 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ 1 & 0 & -1 & 0 & -1 \\ 1 & 0 & -1 & 0 & -1 \\ 1 & 0 & -1 & 0 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ 1 & 0 & -1 & 0 & -1 \end{bmatrix}$$

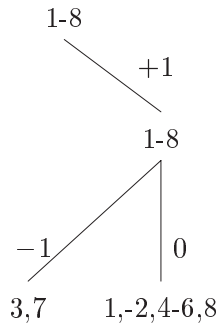


Figure 1: A decision tree based on the example showing partitioning by row number (i.e. object)

and the first 5 columns of (the transposed matrix) Y are

$$Y^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

The values of D are

$$D = \begin{bmatrix} 1.0625 \\ 0.9375 \\ 0.058594 \\ 0.058594 \\ 0.0036621 \end{bmatrix}$$

The decision tree that corresponds to the matrix X is shown in Figure 1. Unlike a conventional decision tree, this one is ternary. It also does not necessarily contain every possible branch – in other words, for some new data, the only possible classification implied by this decision tree is 'unknown'.

Consider the approximation matrices that are subtracted from A at each stage of the algorithm. On the first step, this matrix is the product $d_1 x_1 y_1$, where x_1 is the first column of X and y_1 is the first column of Y^T above. The product of x_1 and y_1 is an 8×8 matrix of 1's, and d_1 is 1.0625, so the matrix that is subtracted from A is the 8×8 matrix, all of whose entries are 1.0625.

In other words, if the matrix is represented in a side view as a histogram as in Figure 2, then the effect of the subtraction has been to remove a slice from the original matrix, leaving four positive peaks and the rest of matrix slightly negative.

On the second step, the matrix that is subtracted is given by the product of x_2 and y_2, which is

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In other words, this product picks out the shape of the remaining peaks. Note also that the value of d_2 is 0.9575,

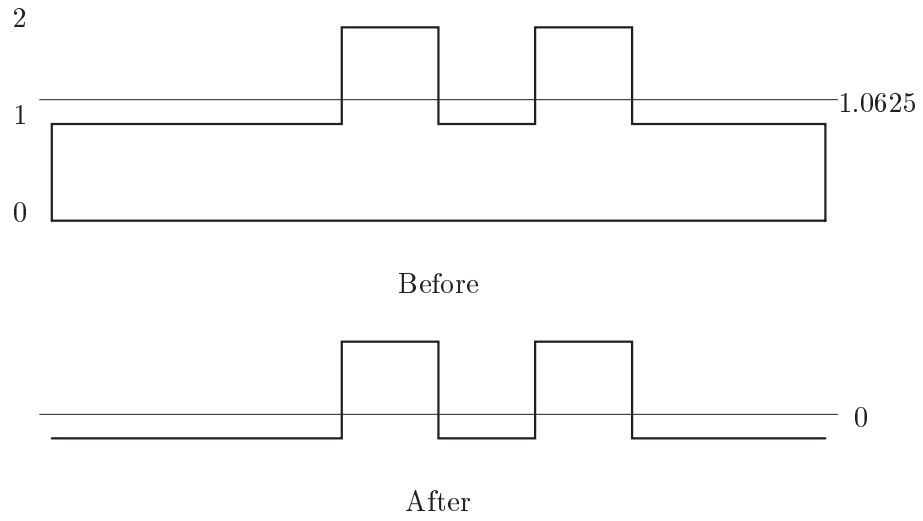


Figure 2: A side view of A showing the effect of the first subtraction

which is exactly the height of the peaks. The effect of the subtraction of the second matrix is to leave a matrix that is almost filled with small negative values.

We can see from this example that the general effect of SDD is to find regions of the matrix in which the magnitude of the values is relatively large. In fact, it is the *volume*, in the sense of Figure 2, that determines which region is selected, since both the magnitude of the values and the size of the region over which they occur are used. SDD is a form of *bump hunting* – on each step, it finds a region of the dataset that sticks out above (below) the rest of the data when values are considered as heights. The product of x_i and y_i selects the position of the region, while d_i selects the height of the bump. (This is slightly simplified because SDD can also remove bumps with *negative* correlation with a selected pattern of rows and columns.)

The use of the 2-norm in the objective function being maximized means that height has more effect on bump selection than the size of the dataset covered by the bump. This means that the algorithm tends to first remove high, but local, bumps in preference to flatter but more widespread bumps. In other words, it tends to find outlier clusters in the data. This explains the order in which axes are chosen by SDD. This relationship means that, unlike SVD, SDD is not scale-independent – multiplying the entire dataset by a constant will produce a different clustering. This property can be used to bias SDD to select primarily small but large clusters or larger but lower clusters, but the normalization used to produce the natural or neutral scale needs to be carefully considered.

Given this discussion, it seems odd that the first bump removed in the example above is a large flat one, rather than the more localized bump that is removed second. The reason is that the localized bump is not very high, so the number of locations of the large flat bump overwhelm the height of the small number of locations in the smaller bump. When the height of the peaks are increased, then the localized bump is removed first. For example, if we start with this matrix:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 9 & 1 & 9 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 9 & 1 & 9 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

then the X matrix becomes

$$X = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & -1 & 1 & -1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & -1 & 1 & -1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

and Y becomes

$$Y = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

with the values of D

$$D = \begin{bmatrix} 9 \\ 0.9375 \\ 0.9375 \\ 0.058594 \\ 0.058594 \\ 0.0036621 \end{bmatrix}$$

It is clear that now the peaks are removed first, and the large flat slice from the bottom second. It is important to note that the order in which slices are removed matters –

if the flat slice had been removed first, its corresponding d value would have been 2 (= 128/64, the average value of the matrix) and the second d value would have been 7.

Although SDD was originally developed as a storage-efficient representation approximating SVD, these examples make it clear that it is actually computing something quite different. When the natural structure of a dataset is many small clusters then SVD and SDD will tend to produce similar results: SVD because it captures a faithful low-dimensional representation of the cluster structure of the data, but SDD because each cluster looks like a small bump. This is shown in Figure 3 which uses a dataset typical of text retrieval applications – many zeroes, the remaining values small positive integers.. Here the position of points (in 3 dimensions) is determined by SVD but the shape used to render them is determined by SDD. It is clear that each cluster (in the SVD sense) is homogeneously labelled by SDD. Hence the two techniques agree.

The critical fact is that any congruence between outcomes is a form of coincidence, since each technique works in a completely different way. This explains both the success of SDD as a substitute for SVD in latent semantic indexing (because document-text matrices tend to contain many small clusters) and the lack of success of SDD we had experienced in other data mining applications. SDD is an *outlier detector* and so will tend to emphasize the most unusual patterns in a dataset – using it as if it were an analogue of SVD leads to frustration.

The heuristic embedded in the basic SDD step leads to two problems, one of them solvable, the other not. The algorithm is quite sensitive to the initial choice of y_i. This means that it does not always find the largest possible slice to remove from the matrix at each step. Hence later steps can find a large slice that was missed on previous steps. It is therefore possible that the values of D are not always decreasing.

We apply the following modification to the algorithm. After the X, Y, and D matrices have been computed, we do the following:

1. Form the product of the d_is with the number of nonzero entries in the corresponding columns of Y.

2. Sort the columns of X, elements of D and rows of Y into decreasing order of the products from the first step.

3. Form a decision tree using the new arrangement of the columns of X.

This has the effect of reordering the slices so that those with the largest volume appear first, in other words, the strongest outliers appear closest to the top of the decision tree.

The second problem occurs because the height of a slice removed depends on the current contents of the matrix, which depends on the order in which previous slices were removed. Reordering at the end cannot therefore reproduce exactly the effect of having chosen a different removal order during the algorithm's execution. The problem occurs because the height of a slice is determined by the *average* height of the locations that will be removed. In the second example above, the peaks of height 9 are completely removed at the first step. However, if the matrix is changed slightly like this:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 9 & 1 & 8 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 9 & 1 & 9 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

then the slices are removed in exactly the same places and orders, but the values of D are now

$$D = \begin{bmatrix} 8.75 \\ 0.9375 \\ 1.6875 \\ 0.51562 \\ 0.51562 \end{bmatrix}$$

in other words, slightly less of the peaks is removed. In some fundamental sense, the way in which the d_i are computed prevents the original matrix from being partitioned as cleanly as it might otherwise be. There seems to be no simple solution to this problem.

3. AN APPLICATION

We illustrate the results obtained using SDD for a dataset of stream sediment geochemical samples collected in Northern N.S.W. The dataset describes 1670 catchment sample sites and, at each site, the concentrations of 33 elements [5]. The dataset has also been analyzed using an unsupervised neural network to cluster the samples [4].

Many of the standard datasets used to illustrate data mining algorithms are simple in the sense that they are based on only a few underlying processes. Hence, in supervised mining, predictors can often achieve prediction accuracies in the 90 percents; and in unsupervised mining there are only a few hidden variables that determine good clusterings. In contrast, geochemical data are the result of a large number of chemical processes, many of whose precise pathways and energetic drivers are poorly understood. Datasets of such measurements therefore capture overlapping processes involving subsets of elements at geographically related sites. Such datasets should be expected to challenge conventional data mining algorithms, especially when the number of attributes is large. On the other hand, SDD is appropriate because it selects regions of datasets in which there is strong local correlation between subsets of attributes and subsets of objects.

For the samples in the NRAC dataset, the decision tree produced by SDD contains only single side branches for the top seven levels. The samples included in each of these outlier clusters are (in order from the top):

- A cluster of size 3 containing high concentrations of Ce, Hf, La, Sr, Th, and Zr – these appear to be samples from pegmatites, which are of interest because they may contain gemstones;
- A cluster of size 2 containing high concentrations of Co, Cr, and Ni – these suggest mafic rocks and perhaps the presence of a nickel or platinum ore body;
- A single sample containing high concentrations of Cd and Sb – this probably represents a

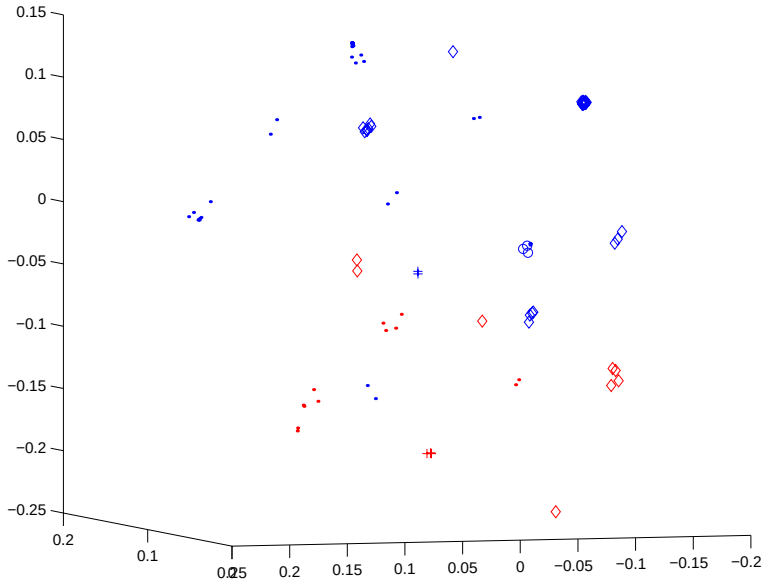


Figure 3: Plot of sparse clusters, position from SVD, shape from SDD. Clusters labelled with the same symbol are distinguished at the next level of the decision tree.

measurement problem since concentrations of cadmium are notoriously hard to measure accurately;

Two samples containing high concentrations of Sn;

A cluster of size 8 containing high concentrations of Cu, Pb, and Zn – this represents typical sulphide mineralization;

A cluster of size 3 containing all the samples in the dataset with high concentrations of gold;

These clusters appear to have some inherent significance. In experiments using a number of other techniques (neural networks, genetic algorithms, decision trees, Autoclass, k-means, and support vector machines) no other technique was able to find outlier clusters such as these in such a complex and highly correlated dataset.

Figure 4 shows a plot of the SVD for the attributes of this dataset. This plot shows that the attributes can be divided into three groups, two across the top of the figure and one in the lower right hand corner that appears more distinct (in fact, this group of elements is strongly related to the outlier cluster of samples). It also shows that attributes 15 and 27 (Lu and Yb) are very similar, and so are attributes 5, 12 and 14 (Ce, Hf, La). However, Figure 5 shows the same plot with the points labelled by SDD. This plot suggests that attribute 4 (Cd) and attributes 13 and 20 (K and Rb) are most anomalous, and that there is a large cluster of the attributes labelled with a circle (Br, Co, Cu, Eu, Fe, Ga, Sc, Sr, and Zn) that are distinctive. Cd has been observed to behave unusually in this dataset. The association of K and Rb is due to their presence in granites. The larger group reflects the presence of feldspars, although Br and Ga separate from the group at deeper levels of the tree as expected. The presence of Zn in this group is also unusual, but Zn might be expected to associate with Cd and so this perhaps reflects some unusual chemistry involving these two elements.

SDD refines the similarity structure defined by SVD, adding further, and sometimes different, discriminations. In extensive experiments with other datasets, reported elsewhere, our experience is that SVD and SDD classification of objects generally correlates at about 0.75 – but the classification of outlier objects and attributes tends to be different, though complementary.

4. RELATED WORK

There are three main approaches to outlier detection for high-dimensional datasets:

1. **Density-based**: "If a tree isn't close to very many other trees, then it's not in a wood". This approach includes Friedman and Fisher's PRIM [6] as well as SDD. PRIM is a top-down technique that finds bumps by removing slices in one dimension at a time, looking for locally dense regions remaining. In contrast, SDD finds bumps using a heuristic approximation to a mixed programming problem. Objects are outliers if they fall into small and/or remote bumps.

2. **Boundary-based**: "If a tree is outside a rope placed around each of the woods, then it's not in a wood". The best example of this approach are variations of 1-class support vector machines [14; 13; 8], in which a dataset is wrapped by (say) a Gaussian in a sufficiently large number of dimensions. Selecting an appropriate 'core' of the Gaussian defines the 'normal' part of the dataset. Objects are outliers if they fall outside this core of the Gaussian.

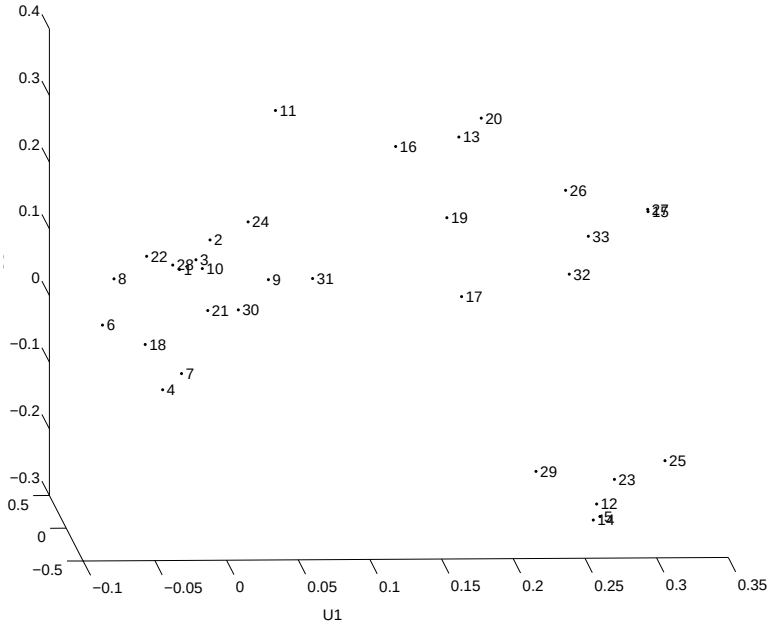


Figure 4: Plot of attributes from SVD

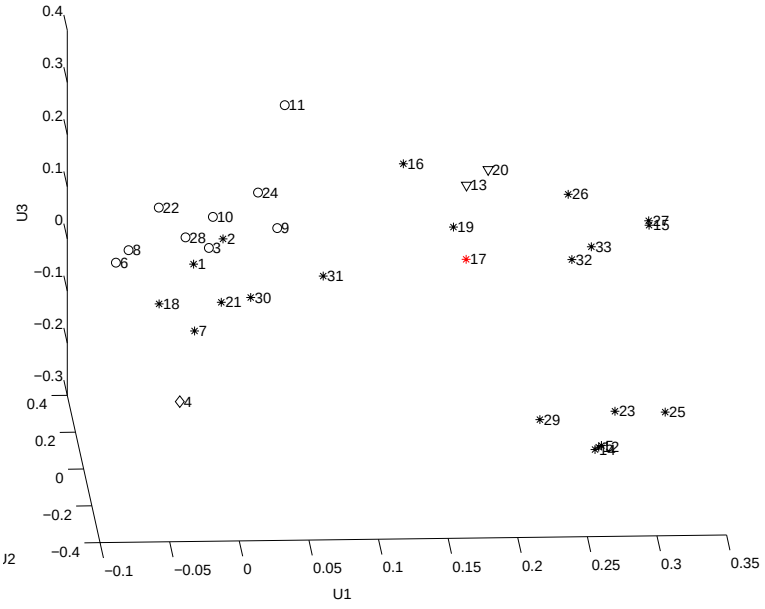


Figure 5: Plot of attributes from SVD, labelling from SDD. o = 1, 0; * = 0, 0; $\diamond = -1, 1$; $\bigtriangledown = -1, 0$

3. **Reconstruction-based**: "If a tree remains a tree when it is cut into sawdust and glued together then it's in the wood". In this approach, a detector for normal objects is built by reconstructing each object and seeing how much the output resembles the input [15]. For example, a neural network with a small hidden layer must necessarily encode properties of normal objects in a compact way. This encoding becomes a kind of check on the normalcy of other objects. Other representations, such as wavelets or minimum description length autoencoders can also be used. Objects are outliers if their reconstructions differ enough from the originals.

Of course, any technique for outlier detection is fundamentally looking for similarity and dissimilarity in the dataset, so any principled clustering technique can also be regarded as an outlier detector. Some are better than others, however. For example, some clustering algorithms try to draw all objects into the neighbourhood of some cluster centre, so will tend to hide small clusters of outliers. Singular values decomposition is particularly effective because it reduces the dimensionality of a dataset to the point where ordinary Euclidean distance can be used as a measure of similarity.

Rule-based techniques can also be considered as finding bumps (regions of the dataset with high confidence but small support). Some of these techniques (i.e. association rules) use exhaustive search, and therefore do not necessarily scale well. Others use various heuristics (e.g. beam search) to reduce the computational cost.

There are also a class of methods based on SVD that, although unsupervised, produce decision trees, notable Boley's *principal direction divisive partitioning* (PDDP) [3], which separates a dataset based on its variation in the direction of the first singular value axis, and then repeats this process on each partition of the dataset independently.

We conclude by comparing the classifications produced by SDD and PDDP on a small dataset with an obvious structure:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The tree produced by PDDP is shown in Figure 6. The SDD classification is shown as a decision tree in Figure 7. The decision tree produced by SDD is clearly more discriminating than the tree produced by PDDP.

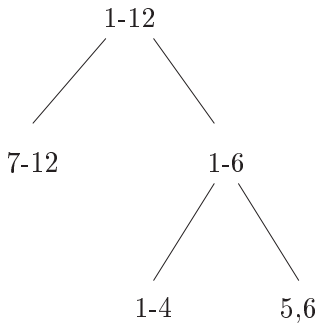


Figure 6: The decision tree produced by PDDP, partitioned by rows

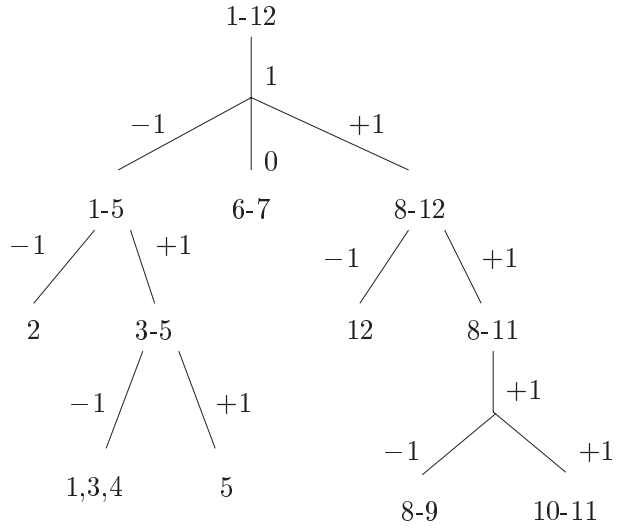


Figure 7: The decision tree produced by SDD, partitioned by rows

5. CONCLUSION

We have shown that SDD works by finding regions of a dataset with anomalously large (positive or negative) values and extracting them. It is therefore best regarded as a bump hunting technique. SDD removes bumps with the largest 'volume', a quantity that is based both on the mag-

nitude of values and the number of positions in the dataset where they occur. This has the general effect of selecting outlier clusters early in the process. We have demonstrated how this happens in a high-dimensional dataset of practical interest. We have also suggested a modification to the basic algorithm designed to make outliers even more obvious.

The SDD technique works in marked contrast to singular value decomposition, which is a space-transforming technique that attempts to model decreasing amounts of variation. SDD and SVD agree on datasets that contain many small clusters, but do not agree when datasets contain a few large clusters. This explains the observation that SDD can be used quite effectively in latent semantic indexing.

Acknowledgements: We would like to thank D.R. Cohen, T.K. Kyser and H.E. Jamieson for assistance with interpreting results using the geochemical dataset.

6. REFERENCES

[1] M. Berry, S. Dumais, and G. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[2] M. W. Berry and R. D. Fierro. Low-rank orthogonal decompositions for information retrieval applications. *Numerical linear algebra with applications*, 3(4):301–327, 1996.

[3] D. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[4] A. Clare and D. Cohen. An unsupervised neural network approach to the analysis of multi-element stream sediment data, northeastern N.S.W., australia. *Geochemistry: Exploration, Environment, Analysis*, 1:119–134, 2001.

[5] D. Cohen, N. Rutherford, D. Garnett, and H. Waldron. Geochemical survey of the upper N.E. region of N.S.W. Geological Survey of New South Wales and the Natural Resources Audit Council, 1995.

[6] J. Friedman and N. Fisher. Bump hunting on high-dimensional data. *Statistics and Computation*, 1997.

[7] G. Golub and C. van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[8] P. Hayton, B. Scholkopf, L. Tarassenko, and P. Anuzis. Support vector novelty detection applied to jet engine vibration spectra. In *NIPS*, pages 946–952, 2000.

[9] G. Kolda and D. O'Leary. A semi-discrete matrix decomposition for latent semantic indexing in information retrieval. *ACM Transactions on Information Systems*, 16:322–346, 1998.

[10] T. Kolda and D. O'Leary. Computation and uses of the semidiscrete matrix decomposition. *ACM Transactions of Information Processing*, 1999.

[11] T. Kolda and D. O'Leary. *Latent Semantic Indexing Via A Semi-Discrete Matrix Decomposition*, volume 107 of *IMA Volumes in Mathematics and Its Applications*, pages 73–80. Springer Verlag, 1999.

[12] D. O'Leary and S. Peleg. Digital image compression by outer product expansion. *IEEE Transactions on Communications*, 31:441–444, 1983.

[13] B. Schoelkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson. Estimating the support of a high-dimensional distribution. Technical Report 99-87, Microsoft Research, 1999.

[14] D. Tax. *One-Class Classification*. PhD thesis, Technische Universiteit Delft, June 2001.

[15] G. Williams, R. Baxter, H. He, S. Hawkins, and L. Gu. A comparative study of rnn for outlier detection in data mining. In *Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM02)*, Maebashi City, Japan, December 2002.

[16] S. Zyto, A. Grama, and W. Szpankowski. Semi-discrete matrix transforms (sdd) for image and video compression. Technical report, Department of Computer Science, Purdue University, 2000.

An Overview of Temporal Data Mining

Weiqiang Lin
Department of Computing
I.C.S., Macquarie University
Sydney, NSW 2109, Australia
wlin@ics.mq.edu.au

Mehmet A. Orgun
Department of Computing
I.C.S., Macquarie University
Sydney, NSW 2109, Australia
mehmet@ics.mq.edu.au

Graham J. Williams
CSIRO Data Mining
GPO Box 664
Canberra ACT 2601, Australia
Graham.Williams@csiro.au

ABSTRACT

Temporal Data Mining is a rapidly evolving area of research that is at the intersection of several disciplines, including statistics, temporal pattern recognition, temporal databases, optimisation, visualisation, high-performance computing, and parallel computing. This paper is first intended to serve as an overview of the temporal data mining in research and applications.

1. INTRODUCTION

Temporal Data Mining is a rapidly evolving area of research that is at the intersection of several disciplines, including statistics (e.g., time series analysis), temporal pattern recognition, temporal databases, optimisation, visualisation, high-performance computing, and parallel computing. This paper is intended to serve as an overview of the temporal data mining in research and applications. In addition to providing a general overview, we motivate the importance of temporal data mining problems within Knowledge Discovery in Temporal Databases (KDTD) which include formulations of the basic categories of temporal data mining methods, models, techniques and some other related areas.
The paper is structured as follows. Section 2 discusses the definitions and tasks of temporal data mining. Section 3 discusses the issues on temporal data mining techniques. Section 4 discusses two major problems of temporal data mining, those of similarity and periodicity. Section 5 provides an overview of time series temporal data mining. Section 6 moves onto a discussion of several important challenges in temporal data mining and outlines our general distribution theory for answering some those challenges. The last section concludes the paper with a brief summary.

2. DEFINITION AND TASKS OF TEMPORAL DATA MINING

The temporal data mining component of the KDTD process is concerned with the algorithmic means by which temporal patterns are extracted and enumerated from temporal data. Some problems for temporal data mining in temporal databases include questions such as: How can we provide access to temporal data when the user does not know how to describe the goal in terms of a specific query? How can we find all the time related information and understand a large temporal data set? and so on.

2.1 Definition and Aims

In this section, we first give basic definitions and aims of Temporal Data Mining. The definition of Temporal Data Mining is as follows:

DEFINITION 1. *Temporal Data Mining is a single step in the process of Knowledge Discovery in Temporal Databases that enumerates structures (temporal patterns or models) over the temporal data, and any algorithm that enumerates temporal patterns from, or fits models to, temporal data is a Temporal Data Mining Algorithm.*

Basically temporal data mining is concerned with the analysis of temporal data and for finding temporal patterns and regularities in sets of temporal data. Also temporal data mining techniques allow for the possibility of computer-driven, automatic exploration of the data. Temporal data mining has led to a new way of interacting with a temporal database: specifying queries at a much more abstract level than say, Temporal Structured Query Language (TSQL) permits (e.g., [17], [16]). It also facilitates data exploration for problems that, due to multiple and multi-dimensionality, would otherwise be very difficult to explore by humans, regardless of use of, or efficiency issues with, TSQL.
Temporal data mining tends to work from the data up and the best known techniques are those developed with an orientation towards large volumes of time related data, making use of as much of the collected temporal data as possible to arrive at reliable conclusions. The analysis process starts with a set of temporal data, uses a methodology to develop an optimal representation of the structure of the data during which time knowledge is acquired. Once Temporal knowledge has been acquired, this process can be extended to a larger set of the data working on the assumption that the larger data set has a structure similar to the sample data.

2.2 Temporal Data Mining Tasks

A relevant and important question is how to apply data mining techniques on a temporal database. According to techniques of data mining and theory of statistical time series analysis, the theory of temporal data mining may involve the following areas of investigation since a general theory for this purpose is yet to be developed:

1. Temporal data mining tasks include:
   - Temporal data characterization and comparison,
   - Temporal clustering analysis,

- Temporal classification,
- Temporal association rules,
- Temporal pattern analysis, and
- Temporal prediction and trend analysis.

2. A new temporal data model (supporting time granularity and time-hierarchies) may need to be developed based on:
   - Temporal data structures, and
   - Temporal semantics.
3. A new temporal data mining concept may need to be developed based on the following issues:
   - the task of temporal data mining can be seen as a problem of extracting an interesting part of the logical theory of a model, and
   - the theory of a model may be formulated in a logical formalism able to express quantitative knowledge and approximate truth.

In addition, temporal data mining needs to include an investigation of tightly related issues such as temporal data warehousing, temporal OLAP, computing temporal measurements, and so on.

3. TEMPORAL DATA MINING TECHNIQUES

A common form of a temporal data mining technique is rule (or functions) discovery. Various types of temporal functions can be learnt, depending upon the application domain. Also, temporal functions (or rules) can be constructed in various ways. They are commonly derived by one of the two basic approaches, bottom-up or top-down induction.

3.1 Classification in Temporal Data Mining

The basic goal of temporal classification is to predict temporally related fields in a temporal database based on other fields. The problem in general is cast as determining the most likely value of the temporal variable being predicted given the other fields, the training data in which the target variable is given for each observation, and a set of assumptions representing one's prior knowledge of the problem. Temporal classification techniques are also related to the difficult problem of density estimation.

In recent years, a lot of the work has been done in non-temporal classification areas by using "Statistical Approaches to Predictive Modelling". Some techniques have been established for estimating a categorical variable, e.g., [26; 5; 20]: kernel density estimators [20; 11] and K-nearest-neighbor method [20]. These techniques are based upon the theory of statistics. Some other techniques such as in [7; 8; 6] are based upon the theory of databases. Temporal classification techniques have not been paid much attention so far. In recent years, the main idea in temporal classification is the straightforward use of sampling techniques within time series methods (distribution) to build up a model for temporal sequences.

3.2 Temporal Cluster Analysis

Temporal clustering according to similarity is a concept which appears in many disciplines, so there are two basic approaches to analyze it. One is the measure of temporal similarity approach and the other is called temporal optimal partition approach.

In temporal data analysis, many temporal data mining applications make use of clustering according to similarity and optimization of temporal set functions. If the number of clusters is given, then clustering techniques can be divided into three classes: (1) Metric-distance based technique, (2) Model-based technique and (3) Partition-based technique. These techniques can be used occasionally in combination, such as Probability-based vs. Distance-based clustering analysis. If the number of clusters is not given, then we can use Non-Hierarchical Clustering Algorithms to find their k.

In recent years, temporal clustering techniques have been developed for temporal data mining, e.g., [23]. Some studies have been done by using EM algorithm and Monte-Carlo cross validation approach (e.g.,[12; 22; 13]).

3.3 Induction

A temporal database is a store of temporally related information but more important is the information which can be inferred from it([3; 4]. There are two main inference techniques: temporal deduction and temporal induction.

1. Temporal deduction is a technique (e.g., in [24] to infer the information that is a temporal logical consequence of the information in the temporal database.
2. Temporal induction can be described as a technique (e.g., in [25]) to infer temporal information that is generalised from the temporal database. Induction has been used in the following ways within data mining: 1) Decision Trees and 2) Rule Induction.

4. TWO FUNDAMENTAL TEMPORAL DATA MINING PROBLEMS

In recent years, two kinds of fundamental problems have been studied in temporal data mining area. One is the Similarity Problem which is to find a time sequence (or TDB) similar to a given sequence (or query) or to find all pairs of similar sequences. The other is the Periodical Problem which is to find periodic patterns in TDB.

4.1 Similarity Problems

In temporal data mining applications, it is often necessary to search within a temporal sequence database (e.g: TDB) for those sequences that are similar to a given query sequence. Such problems are often called Similarity Search Problem. This kind of a problem involves search on multiple and multidimensional time series sets in TDBs to find out how many series are similar to one another. It is one of the most important and growing problems in Temporal Data Mining. In recent years, we still lack a standard definition and standard theory for similarity problems in TDB.

Temporal data mining techniques can be applied in similarity problems. The main steps for solving the similarity problem are as follows:

- define similarity: allows us to find similarities between sequences with different scaling factors and baseline values.

- choose a query sequence: allows us to find what we want to know from large sequences (TDB) (e.g, character, classification)

- processing algorithm for TDB: allows us to apply some statistical methods (e.g, transformation, wavelet analysis) to TDB (e.g, remove the noisy data, interpolate the missing data).

- processing an approximate algorithm: allows us to build up a classcification scheme for the TBD according to the definition of similarity by using some data mining techniques (e.g, visualisation).

The result of the Similarity Problem search in TDB can be used for temporal association, prediction, etc.

4.2 Periodical Problems

The periodicity problem is the problem of finding periodic patterns or, cyclicity occurring in time-related databases (TDB). The problem is related to two concepts: *pattern* and *interval*. In any selected sequence of TDB, we are interested in finding patterns which repeat over time and their recurring intervals (period), or finding the repeating patterns of a sequence (or TDB) as well as the interval which corresponds to the pattern period. For solving a Periodical Problem in TDB, the main steps are as follows:

- determining some definitions of the concept of a period under some assumptions: this step allows us to know what kind of a periodicity search we want to perform from TDB.

- building up a set of algorithms: this step allows us to use properties of periodic time series for finding periodic patterns from a subset of TDB by using algorithms.

- processing simulation algorithms: this step allows us find patterns from whole TDB by the algorithms.

A lot of techniques have been involved in these kind of problems by using pure mathematical analysis such as function analysis, data distribution analysis and so on, e.g.,[9].

4.3 Discussion

In a time-series TDB, sometimes similarity and periodical search problems are difficult even when there are many existing methods, but most of the methods are either inapplicable or prohibitively expensive. There is also another difficult problem: how we can combine multiple-level similarity or periodical search in a multiple-level model? With the reference cube structure, such difficult problems can be solved by extending the methods mentioned in previous subsections, but the problem of combining multiple-level similarity and periodicity in a multiple-level model is still unsolved. Also, more sophisticated techniques need to be developed to reduce memory work-space.

In fact, similarity and periodical search problems can be combined into the problem of finding interesting sequential patterns in TDBs. In recent years, some new algorithms have been developed for "fast mining of sequential patterns in large TDBs":

- generalized sequential pattern (GSP) algorithm: it essentially performs a level-wise or breadth-first search of the sequence lattice spanned by the subsequence relation,

- sequential pattern discovery using equivalence classes (SPADE) algorithm: it decomposes the original problem into smaller sub-problems using equivalence classes on frequent sequences [15].

With any new algorithm, there is one important question that has often been asked: How can we implement the new algorithm directly on top of a Time-series TDB?

5. TIME SERIES TEMPORAL DATA MINING

Statistics has been an important tool for data analysis for a long time. For example, Bayesian inference is the most extensively studied statistical method for knowledge discovery (e.g, [2], [10], [18]) and Markov Model, Hidden Markov Model (e.g., [14; **?**]) also have made their way into temporal knowledge discovery process.

Time series is a record of the values of any fluctuating quantity measured at different points of time. One characteristic feature which distinguishes time series data from other types of data is that, in general, the values of the series at different time instants will be correlated[1]. Application of time series analysis techniques in temporal data mining is often called Time Series Data Mining. A great deal of work has been done into identifying, gathering, cleaning, and labeling the data, into specifying the questions to be asked of it, and into finding the right way to view it to discover useful temporal patterns.

Time series analysis method has been applied into following major categories in temporal data mining:

1. Representation of Temporal Sequence: This refers to the representation of data before actual temporal data mining techniques take place. There are two major methods:

   - General representation of data: representation of data into time series data in either continuous or discontinuous, linear/non-linear models, stationary/non-stationary models and distribution models (e.g., Time domain representation and Time series model representation).
   - General transformation of representation of data: representation of data into time series data in either continuous or discontinuous transformation (e.g., Fourier transformation, Wavelet transformation and Discretization transformation).

2. Measure of Temporal Sequence: measuring temporal charactersistic element in given definitions of similarity and/or periodicity in a temporal sequence (or, two subsequence in a temporal sequence) or between temporal sequences. There are two methods:

[1]Time Analysis Theory can be found in any standard textbook of time series analysis, e.g., [1].

- Characteristic distance measuring in time domain: measuring distance between temporal characterstics in either continuous or discontinuous time domain (e.g., Euclidean squared distance function).
- Characteristic distance measuring in other than time domain: measuring distance between temporal characterstics in either continuous or discontinuous domain other than time (e.g., distance function between two distributions).

3. Prediction of Temporal Sequence: the main goal of prediction is to predict some fields in a database based on TIME domain. The techniques can be classified into two models.
   - Temporal classification models: the basic goal is to predict the most likely state of a categorical variable (the class) in TIME domain.
   - Temporal regression models: the basic goal is to predict a numeric variable in a set by using different transformations (e.g, linear or non-linear) on databases to find temporal information (or, patterns) of the different (or the same) categorical data sets (class).

Recently, there are various results to date on discovering temporal information which have offered forums to explore the temporal data mining progress and future work concerning temporal data mining. But the general theory and general method of temporal data analysis of discovering temporal patterns for temporal sequence data analysis are not well known.

6. CHALLENGES AND RESEARCH DIRECTIONS

Recent advances in data collection and storage technologies have made it possible for companies, administrative agencies and scientific laboratories to keep vast amounts of temporal data relating to their activities. Data mining refers to such an activity to make automatic extraction of different levels of knowledge from data feasible. One of the main unresolved problems, often called General Analysis Method of Temporal Data Mining, that arise during the data mining process is treating data that contains temporal information.

6.1 Challenge Questions

Data mining is a step in the knowledge discovery in databases, although successful data mining applications continue to appear but the fundamental problems are still as difficult as they have been for the past decade. One such difficult and fundamental problem is the development of a general data mining analysis theory. Temporal data mining researchers have paid some attention to this problem but results still remain in their infancy. One of the important roots in data mining analysis is statistical analysis theory. The general temporal data mining analysis theory includes two important analysis methods:

- Data structural temporal knowledge analysis method: This method involves the discovery of data prior temporal knowledge, and the exploitation of the knowledge into a data analysis model to establish the link between the present temporal knowledge and the future temporal knowledge.
- Data temporal measure analysis method: This method involves the transformation of initial data temporal domain (or space) into another domain (or space), then the use of this new domain to represent the original temporal data.

The techniques involved in the above two methods can be divided into following classes:

1. **Temporal data clustering:** temporal clustering targets separating the temporal data into subsets that are similar to each other. There are two fundamental problems of temporal clustering:
   - To define a meaningful similarity measure, and,
   - To choose the number of temporal clusters(if we do not know the cluster numbers).
2. **Temporal data prediction:** the goal of temporal prediction is to predict some fields based on other temporal fields. Temporal data prediction also involves using prior temporal patterns (or, models, knowledge) for finding the data attributes relevant to the attribute of interest.
3. **Temporal data summarization:** the purpose of temporal data summarization is to describe a subset of temporal data by representing extracted temporal information in a model or, in rules or in patterns. It provides a compact description for a temporal dataset. It could also involve a logic language such as temporal logic, fuzzy logic and so on.
4. **Temporal data dependency:** Temporal dependency modelling describes time dependencies among data and/or temporal attributes of data. There are two dependency models: qualitative and quantitative. The qualitative dependency models specify temporal variables (e.g., time gap) that are locally dependent on a given state-space $\mathcal{S}$. The quantitative dependency models specify the value dependencies (e.g., using numerical scale) in a statistical space $\mathcal{P}$.

6.2 Some Answers of the Challenge Questions

During the past few years, we have proposed a formal framework for the definitions and general hidden distribution theory of temporal data mining. We have also investigated applications in temporal clustering, temporal classification and temporal feature selection for temporal data mining. The major work we have done in answering the temporal data mining challenge questions are:

- We have established a General Hidden Distribution-based Analysis Theory for temporal data mining. The general mining analysis theory is based on the statistical analysis method but traditional statistical assumptions only come from the data itself. There are two important concepts in the theory: 1) data qualitative set, data quantitative set and 2) data hidden conditional distribution function. The data qualitative set is the set to decide the data moving structure such as data

periodicity and similarity. In other words, data qualitative set is a base of the data. The data quantitative set is the set to decide the numerical range of the data moving structure. The data hidden conditional distribution function is built on the characteristics of data qualitative and data quantitative sets. Another feature of the general mining analysis method is that we can use (extension of) all existing statistical analysis methods and techniques for mining temporal patterns.

- We have proposed an algorithm which is called The Additive Distributional Recursion Algorithm (ADRA) in General Hidden Distribution-based Analysis Theory for building up temporal data models. The algorithm uses the sieve method[2] to discover temporal distribution function (models, pattern).

- We have extended a normal measure method to a new Temporal Measure Method, which is called Time-gap Measure Method. The new measure method has brought "time length" (which is between temporal events) or "time interval" (which is within a temporal event) into a time point (or, time value) variable. After a temporal sequence is transformed, it can be measured in both state-space $\mathcal{S}$ and probability space $\mathcal{P}$. The time-gap is used as a temporal variable in time distribution function $f(t_v)$ or temporal variable functional equations embedded in temporal models of the sequence.

- We have extended and built up a new application of fundamental mathematics techniques for dealing with large temporal datasets, massive temporal datasets and distributed temporal datasets. The new application is called Temporal Sequence Set-Chains (A special case of the Temporal Set-Chains is a Markov Set-Chains). The key issue in temporal sequence set-chains is the use of stochastic matrices of samples to build up a moving kernel distribution. The temporal sequence set-chains sequence can be used for mining a large temporal sequence, massive temporal sequence and distributed temporal sequence such as Web temporal data sequence (e.g., Web content sequence, Web usage sequence and Web strutural sequence).

- We have proposed a framework of Temporal Clustering method for discovering temporal patterns. In our temporal clustering method, there are three stages of temporal data mining in temporal clustering analysis: 1) the input stage: what an appropriate measure of similarity to use, 2) the algorithm stage: what types of algorithms to use, and 3) the output stage: assessing and interpreting the results of cluster analysis. In the second stage, we also proposed a framework of Distribution-based Temporal Clustering Algorithm. The algorithm is based on our general analysis method.

- We have proposed a framework of Temporal Classification. This temporal classification is generated by our Temporal Clustering method. According to our general analysis theory for Temporal Sequence Mining and its application in temporal clustering, there are also the following three steps for constructing a temporal classification: 1) Provide a definition of temporal classification, 2) Define a distribution distance function and 3) Provide the weighting of temporal objects for changing their class membership. For large numbers of classification, we proposed a *discriminant coordinates of time gap distribution* to deal with such kinds of problems.

- We have proposed a framework of Temporal Feature Selection for discovering Temporal patterns. There are three steps for feature selection in the temporal sequence. The first step of the framework employs a distance measure function on time-gap distributions between temporal events for discovering structural temporal features. In this step, only rough shapes of patterns are decided. The temporal features are grouped into temporal classifications by employing a distribution distance measure. In the second step, the degree of similarity and periodicity between the extracted features are measured based on the data value distribution models. The third step of the framework consists of a hybrid model for selecting global features based on the results of the first two steps.

- We have established the main steps of applying our general temporal data mining theory to real world datasets with different methods and models. There are three steps of applications of our general analysis for discovering knowledge from a temporal sequence: 1) preprocessing data analysis including solving data problems and transforming data from its original form into its quantitative set and qualitative set, 2) temporal pattern searching including qualitative-based pattern searching, quantitative-based pattern searching and discovering global temporal patterns (models), and 3) the interpretation of the global temporal patterns (models) and future prediction.

6.3 Future Research Directions

As we mentioned earlier temporal data mining and knowledge discovery have emerged as fundamental research areas with important applications in science, medicine and business. In this section, we describe some of the major directions of research from recent general analysis theory of temporal data mining research:

1. An extension of this temporal sequence measure method to general temporal points (e.g., temporal interval-based gap function) allowing an arbitrary interval between temporal points may lead to a very powerful temporal sequence transformation method.

2. An extension of the notion of Temporal Sequence Set-Chains on different temporal variables, or different components of a temporal variable, can be applied to deal with following problems of temporal data mining:

   - the number of temporally related attributes of each observation increases,
   - the number of temporally related observations increases, and
   - the number of temporally related distribution functions increases.

[2]The sieve method is an important method in number theory.

3. An important extension of the general temporal mining theory is the development of distributed temporal data mining algorithms.

4. In applications of temporal data mining, all new temporal data mining theories, methods and techniques should be developed on/with privacy and security models and protocols appropriate for temporal data mining.

5. In general data mining theory, we may need to develop fundamental mathematical techniques of fuzzy methods for mining purposes (e.g., temporal fuzzy clustering and algorithms, temporal fuzzy association rules and new types of temporal databases.).

7. CONCLUDING REMARKS

Temporal data mining is a very fast expanding field with many new research results reported and many new temporal data mining analysis methods or prototypes developed recently. Some articles of overview of temporal data mining have discussed in different frameworks for coveing research and application in temporal data mining. In [19], for example, Roddick and Spiliopoulou have presented a comprehensive overview of techniques for the mining of temporal data.

In this report we have provided an overview of the temporal data mining process and some background to Temporal Data Mining. Also we discussed a difficult and fundamental problem, a general analysis theory of temporal data mining and provided some answers to the problem. This leads into a discussion on why there was a need for Temporal Data Mining in industry, which has been a major factor in the efforts that have gone into building the present generation of Temporal Data Mining Systems. We have presented a number of areas which are related to Temporal Data Mining in their objectives and compared and contrasted these technologies with Temporal Data Mining.

Acknowledgements

This research has been supported in part by an Australian Research Council (ARC) grant and a Macquarie University Research Grant (MURG).

8. REFERENCES

[1] D. Brillinger, editor. *Time Series: Data Analysis and Theory*. Holt, Rinehart and Winston, New York, 1975.

[2] P. Cheeseman and J. Stutz. Bayesian classification (AUTOCLASS): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. AAAI Press / MIT Press, 1995.

[3] T. Fulton, S. Salzberg, S. Kasif, and D. Waltz. Local induction of decision trees: Towards interactive data mining. In Simoudis et al. [21], page 14.

[4] B. R. Gaines and P. Compton. Induction of meta-knowledge about knowledge discovery. *IEEE Trans. On Knowledge And Data Engineering*, 5:990–992, 1993.

[5] C. Glymour, D. Madigan, D. Pregibon, and P. Smyth. Statistical inference and data mining. *Communications of the ACM*, 39(11):35–41, Nov. 1996.

[6] F. H. Grupe and M. M. Owrang. Data-base mining - discovering new knowledge and competitive advantage. *Information Systems Management*, 12:26–31, 1995.

[7] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: An attribute-oriented approach. In *Proceedings of the 18th VLDB Conference*, pages 547–559, Vancouver, British Columbia, Canada, Aug. 1992.

[8] J. W. Han, Y. D. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *Ieee Trans. On Knowledge And Data Engineering*, 5:29–40, Feburary 1993.

[9] J. W. Han, Y. Yin, and G.Dong. Efficient mining of partial periodic patterns in time series database. *IEEE Trans. On Knowledge And Data Engineering*, 1998.

[10] D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors. *Learning bayesian networks: the combineation of knowledge and statistical data*. AAAI Press, 1994.

[11] D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors. *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*. AAAI Press, 1997.

[12] E. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. page 126.

[13] A. Ketterlin. Clustering sequences of complex objects. In Heckerman et al. [11], page 215.

[14] C. Li and G. Biswas. Temporal pattern generation using hidden markov model based unsuperised classifcation. In *Proc. of IDA-99*, pages 245–256, 1999.

[15] M.J.Zaki. Fast mining of sequential patterns in very large databases. *Uni. of Rochester Technical report*, 1997.

[16] S. a. O.Etzion, editor. *Temporal databases: Research and Practice*. Springer-Verlag,LNCS1399, 1998.

[17] B. Padmanabhan and A. Tuzhilin. Pattern discovery in temporal databases: A temporal logic approach. In Simoudis et al. [21], page 351.

[18] P.sprites, C.Glymour, and R.Scheines. *Causation, Prediction and Search*. Springer-Verlag, 1993.

[19] J. Roddick and M. Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on Knowledge and Data Engineering, 2002*.

[20] R.O.Duda and P. Hart. *Pattern classification and scene analysis*. John Wiley and Sons, 1973.

[21] E. Simoudis, J. W. Han, and U. Fayyad, editors. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. AAAI Press, 1996.

[22] P. Smyth. Clustering using monte carlo cross-validation. In Simoudis et al. [21], page 126.

[23] T.Oates. Identifying distinctive subsequences in multivariate time series by clustering. In *5th International Conference on Knowledge Discovery Data Mining*, pages 322–326, 1999.

[24] J. D. Ullman and C. Zaniolo. Deductive databases: achievements and future directions. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 19(4):75–82, Dec. 1990.

[25] D. Urpani, X. Wu, and J. Sykes. RITIO - rule induction two in one. In Simoudis et al. [21], page 339.

[26] P. Usama Fayyad and O.L.Mangasarian. Data mining: Overview and optimization opportunities. *INFORMS*, Special issue on Data Mining, 1998.

Distances for Spatio-temporal clustering

(Extended Abstract)

Mirco Nanni
ISTI - Institute of CNR
Via Moruzzi 1 – Loc. S. Cataldo, 56124
Pisa, Italy
nanni@guest.cnuce.cnr.it

Dino Pedreschi
Dipartimento di Informatica, Università di Pisa
Via F. Buonarroti 2, 56127
Pisa, Italy
pedre@di.unipi.it

ABSTRACT

While the general problem of clustering and clustering analysis has received a lot of attention, and plenty of different techniques have been developed, relatively little research has addressed the specific problem of clustering objects that are both spatially and temporally referenced, which is precisely the scope of this paper. After setting the stage, the paper identifies two major issues that are worth studying: (1) how to define suitable distance functions for spatio-temporal objects, and (2) how to deal with the computational cost raised from the adoption of such distance functions within various clustering techniques. The main contributions of our work are a general schema for distance functions, which is provided to tackle the first point, and a repertoire of optimisation techniques, based on properties of the general schema, which are developed and experimentally evaluated.

Note: Both authors are members of the Pisa KDD Laboratory, `http://www-kdd.cnuce.cnr.it`, a research group involving ISTI-CNR (an institute of National Council of Research) and the Computer Science Department of University of Pisa. This extended abstract is based on the work [9].

Keywords

Data mining, Clustering, Spatial and temporal databases

1. INTRODUCTION

Remote sensing systems, regional sales systems, e-commerce and other data collection tools, actually lead to the collection of huge amounts of data, which make it crucial to develop analysis tools able to extract valuable hidden knowledge from the large and complex databases where such data are organised. In particular, we observe that in many cases the information stored in such databases has a dynamic spatial nature: not only spatial databases often associate spatial data with temporal information, such as time-stamps and versioning information, but also relational databases in many cases contain both temporal information (usually simple dates) and spatial references (most of the times in the form of geo-referentiable information, such as zip codes, addresses). This facts, together with the actual interest of the database community in the developing of next-generation databases with full support of spatio-temporal data, make it clear that a satisfactory set of knowledge discovery tools is needed, able to deal with spatio-temporal information.

Among the various forms of spatio-temporal data, individual trajectories in space and time play an important role, due to the growing wide-spreading of mobile GIS technology. Geographic Information Systems (GIS) can provide the basis for mobile location information provision to individuals through cell phones, PDAs and other hand-held devices. Mobile GIS-based information services can enhance the awareness of a person's locality, thus helping to satisfy location-related needs and facilitate interactions among individuals. The technology for mobile GIS is becoming rapidly available through breakthroughs in miniaturisation, exponentially increasing processing power and improvements in wireless connectivity, while social trends are also leading to acceptance and use of such technology. This enormous amount of space-time trajectory data that will be generated through mobile GIS creates tremendous opportunities for data mining techniques to discover knowledge about individuals' activities in time and space.

Among the basic data mining tools, clustering has always received the interests of the research community. Indeed, with the growing availability of spatial data, many efforts have been spent to provide efficient algorithms able to deal with the large dimensions of spatial databases and with some of the specific issues that this kind of data raised or emphasised (e.g. noise, outliers, non-convex shapes). In particular, it has been made clear that in the spatial context, some of the common assumptions underlying traditional clustering algorithms can be violated: as an example, some spatial situations are better modelled by density-based algorithms, i.e. the common intuition that far objects should belong to different clusters is inadequate.

Spatio-temporal data are basically different from both spatial and temporal data, and thus they require a conceptually different treatment. In particular, clustering methodologies should be revised and adapted to the peculiarities of spatio-temporal databases, in a similar way to what happened with the introduction of spatial data. However, so far the most common approach to perform clustering on spatio-temporal data is essentially the converse, i.e. the peculiarities of spatio-temporal data are simply sacrificed (either through some kind of preprocessing, or treating time and space as simple/unrelated attributes) in order to adhere to some pre-existing clustering schema. A few examples are listed below:

Example 1 (Animal sociology). *Let consider a data-*

base containing information on the animals living within some wide area, together with a description of their movements along time (e.g., the set of their positions collected at regular intervals by means of radio-collars). One of the questions a domain expert would like to answer is "do these animals form couples and/or herds?".

The standard approach projects out the temporal component: it makes use of GIS facilities, and reconstructs a "visited area" for each object, such that all positions of the animal fall within its corresponding "area". Thus some form of clustering (also a simple visual one) is applied on such areas.
We can easily see that, whenever the land area is relatively small and the animals are not territorial (i.e., a herd dominates an area and keep all other animals outside), such approach is misleading: typically, a predator and a prey can live in the same spaces (thus having the same "visited area"), but almost never at the same time. Thus, in this case an approach which takes into consideration space and time together would be advisable.

EXAMPLE 2 (OBJECTS WITHIN IMAGES). *Some image processing algorithms are able to recognise where some spots (such as the edge of an object) appear in some images of a stream. As a result, each spot is associated with a position on some of the images of the stream, which can be stored as an incomplete description of their movement. One problem arises: which spots belong to the same object?*

In this case, even points far from each other could be parts of the same object, so the usual distance between (moving) points should be revised. Here, two spots are "similar" not when they are close to each other, but when their relative distance does not change along time: a typical example is a large wheel with a spot on the hinge and one on the external border: the two spots are always far from each other, but their distance never changes, also if the wheel rotates.

1.1 Aim of the paper

In this paper we give some contributions to the development of clustering methodologies tailored around the peculiarity of a spatio-temporal domain. More in detail, the contributions of our work are the following:

- First, we give a modular definition of a spatio-temporal general distance, i.e. a general schema for defining distances between spatio-temporal objects. In particular, the schema deals with moving objects having a clear identity, such as animals and vehicles, not considering cases where identities of objects are ambiguous or cannot be assigned at all, e.g. dealing with moving clouds. By means of a few examples it is shown how several problems can be modelled as clustering problems with an appropriate instantiation of the schema proposed. Moreover, some useful properties of the schema (and thus of all the concrete dissimilarity functions that can be obtained through instantiation) are provided; a special focus will be put on the dissimilarity functions which obey the metric properties.
- The computational issues related to the distances obtained from the schema are analysed in detail. The cost of computing a single distance is studied, and a centre-based clustering algorithm (k-means) is analysed in order to evaluate its computational behavior when applied to the spatio-temporal context with our distance schema, in comparison with the family of computationally expensive matrix-based clustering algorithms. This provides a precise measure of the computational impact of our spatio-temporal general distance on classical clustering techniques.
- The computational cost of computing pairwise distances of spatio-temporal objects motivates the study of ad hoc optimisation techniques which make clustering feasible. A set of such optimisations is proposed and implemented for k-means, exploiting the properties of metrics. These optimisations try to reduce the number of distances to compute at the cost of a negligible overhead. Thus, (i) it is *possible* to apply them not only in the spatio-temporal context, but also in all the situations where the distance used is a metric; and (ii) it is *convenient* to apply them whenever computing a distance is an expensive task (and thus, saving the computation of some distances is worth a little overhead). The spatio-temporal context developed in this paper satisfies all the requirements for applying the proposed optimisations, so we are in the position of experimenting them with some datasets. At the end of the paper an empirical analysis of the performances is given, with respect to a set of experiments on synthetic datasets.

This extended abstract is based on the work [9], where the details of the results presented in the paper can be found, as well as the extension of the optimisation techniques to other families of clustering algorithms.

The rest of the paper is organised as follows. Section 2 briefly summarises related work. In Section 3 a model of data is chosen, which will be taken as reference throughout the paper. In Section 4 a family of distance measures is defined and some basic (mathematical and computational) properties are presented. Section 5 investigates the application of such distances to some clustering algorithms, while Section 6 shows some preliminary experimental results. Finally, Section 7 contains some conclusive remarks and a short plan of future works.

2. RELATED WORK

Only a small part of literature on metrics and clustering explicitly deals with spatio-temporal objects, and can be summarised by three main proposals:
(i) In [6; 7] hierarchical clustering of complex objects is considered. Here, objects can have *simple* attributes – such as thematic ones and spatial coordinates – and *composite* attributes, i.e. *sequences* of complex objects, recursively. Clustering over complex features, such as trajectories in a multi-dimensional space, is performed by the COBWEB clustering algorithm [3], which defines both the distance measure and the cluster representatives via *generalisation* of objects. However, we can notice that sequences are only a simplification of time-varying objects, since they do not take into account the magnitude of the time component (so, for example, the duration of a change is not considered). Moreover,

the generalisation-based approach proposed there can provide only a very simple distance measure for spatial objects, defined as finite sets of points [8].
(ii) A model-based approach is proposed in [4] for clustering trajectories. Here, each cluster collects all objects which can be obtained by a core trajectory (i.e. the regression function, which can be linear/polynomial or a more general non-parametric model) by adding noise with normal distribution. Such a framework is very general, deals with (samples of) continuous trajectories and has solid statistical foundations. However, the distance measure implicitly considered is essentially fixed and Euclidean-based. The normal noise assumption too, although flexible, can be considered as a constraint.
(iii) A very general solution to the problem of working in non-metric spaces is provided by FastMap [2], where each object is mapped into a point of an Euclidean space trying to preserve mutual distances. Such a technique is applied for instance by the Bubble-FM [5] clustering algorithm. The main drawback of this approach is due to the approximations induced by the mapping: in some applications (such as clustering) it can force the coexistence of computations over the original space (to obtain precise results) and over the mapped metric space (because the computation of distances is usually cheaper), adding a lot of complexity to the whole algorithm.
The discovery of similar trends in time series (i.e. subsequences which are repeated either in the same time series, or in two distinct ones) represents a problem close to our topics, but with a different focus. Since the number of possible patterns to search for is potentially very large, some authors propose to predefine a fixed set of patterns to look for, i.e. shapes of interest. For example, Agrawal et al. [1] defined a shape definition language (SDL) to describe patterns or shapes occurring in historical data, while in [10], shapes in time series are similarly captured by an arbitrary gradient alphabet for the description of movement directions.

3. A DATA MODEL FOR TRAJECTORIES

In this paper we consider databases composed by a finite set of spatio-temporal objects. From an abstract point of view, a spatio-temporal object o is represented by a continuous function of time which, given a time instant t, returns the position at time t of the object in a d-dimensional space (typically $d \in \{2, 3\}$). Formally: $o : \mathbb{R}^+ \to \mathbb{R}^d$.
In a real-world application, however, objects are given by means of a finite set of observations, i.e. a finite subset of points taken from the real continuous trajectory. Moreover, it is reasonable to expect that observations are taken at irregular rates within each object, and that there is not any temporal alignment between the observations of different objects.
This calls for an approximate reconstruction of the original trajectory. Among the several possible solutions, we can devise three simple yet interesting categories:

- Global regression: if the trajectories are known to follow a simple regular behavior which can be represented by some analytical time-dependent expression, o can be replaced by a single regression function.
- Local interpolation: objects are assumed to move between the observed points of the (real) trajectory following some rule. For instance, a linear interpolation models a straight movement with constant speed, while other polynomial interpolations can represent smooth changes of direction.
- Domain knowledge-based reconstruction: specific domain knowledge can help to understand how objects behave between observed points, exploiting information about either the objects or the space they move in. For instance, information on the maximum speed that some objects can reach, may strongly narrow the range of choices of their movements.

In this work we choose to follow the simple (yet not trivial) parametric 2-spaghetti approach, thus modelling the movement of objects through linear interpolation between control points and assuming stationarity beyond the extreme control points. This solution is flexible enough to model most of the real situations and, at the same time, it simplifies our study on computational properties for the distance schema we propose in the next section.

4. A FAMILY OF DISSIMILARITY MEASURES

In this section we introduce a very simple schema whose components can be instantiated to obtain several different definitions of distance between spatio-temporal objects.

4.1 General definition and example instances

The main idea simply consists in considering the spatial and temporal components in a modular fashion. In particular, we consider distances which can be computed by analysing the way the distance between the objects varies along time. That implies we restrict to consider only couples of *contemporary* instantiations of objects, therefore excluding subsequence matching and other similar operations typical of time series mining.

DEFINITION 1 (GENERAL DISTANCE SCHEMA). *A* (general) spatio-temporal distance *is a function D defined in the following way:*

$$D(o_1, o_2) = \Phi(d_{o_1,o_2})\big|_T$$

where $d_{o_1,o_2}(t)$ is a distance measure between $o_1(t)$ and $o_2(t)$, $T \subset \mathbb{R}^+$ is the time definition domain of the objects in our database, and $\Phi(f)|_T$ is a functional computed over function f and domain T.

We notice that such a definition requires to have a time definition domain which is common to all objects. This (apparently strong) assumption can be justified in two ways. First, if objects are allowed to have distinct existence intervals, several problems arise in computing their distances: two objects can be compared only over the intersection of their existence intervals, which could be empty (thus not allowing to compute any meaningful distance) or disjunct w.r.t. other existence intervals (thus compromising the coherence of mutual distances between three or more objects). Second, in our context trajectories of objects are reconstructed from sets of observations, so the above assumption simply requires to perform reconstruction over a common time interval, possibly needing to extrapolate trajectories outside

the limit control points, which seems a quite reasonable requirement.
Standard instances of function $d_{o_1,o_2}(.)$ are Minkowski's metrics, in particular the Euclidean and the Manhattan distances, but depending on the application domain other (unusual) choices can range from orientation measures (such as the anomaly of vector $o_1(t) \rightarrow o_2(t)$) to metrics weighted w.r.t. space characteristics (for instance by shrinking distances within fast surfaces and growing them on slow ones).
Analogously, the most usual instance of function $\Phi(.)$ is the *average* functional:

$$\Phi(f)|_T = \frac{\int_T f(t)dt}{|T|},$$

but other choices are possible, such as weighted average (i.e. average of $f(t)w(t)$ for some weight function $w : \mathbb{R}^+ \rightarrow [0,1]$), *minimum*, *maximum*, and *max - min*.

EXAMPLE 3 (OBJECTS MOVING TOGETHER). *A measure D describing whether two objects move together (such as animals within the same herd, or male-female couples), should compare their relative positions along time, following the intuition that they should* usually *be close to each other. So a simple and reasonable instantiation of the above schema for this application is $d_{o_1,o_2}(.)$ = Euclidean distance, and $\Phi(.)$ = simple average.*

EXAMPLE 4 (OBJECTS WITH SOLIDAL MOVEMENTS). *The main characteristic of two objects which move in a solidal way, is that their mutual distance – although possibly large – does not change significantly along time. Instantiating $d_{o_1,o_2}(.)$ to the Euclidean distance and $\Phi(f)$ to $\max f - \min f$, we obtain a reasonably good implementation of that concept.*

4.2 Mathematical properties

Here we collect a few properties of the distances defined in the previous section, focusing on the basic properties of metrics. Proofs are omitted for lack of space, and the interested reader is addressed to [9]. We recall that a distance function $d()$ is a metric if:

1. $\forall i.\ d(i,i) = 0$ (*"zerosity"*)
2. $\forall i \neq j.\ d(i,j) > 0$ (positivity)
3. $\forall i,j.\ d(i,j) = d(j,i)$ (symmetry)
4. $\forall i,j,k.\ d(i,j) + d(j,k) \geq d(i,k)$ (triangularity)

PROPOSITION 1. *If d_{o_1,o_2} is a metric and $\Phi(f)|_T$ is the* (weighted) average *functional, then the resulting distance D is a metric.*

PROPOSITION 2. *If d_{o_1,o_2} is a metric and $\Phi(f)|_T$ is the* maximum *functional, then the resulting distance D is a metric.*

PROPOSITION 3. *If d_{o_1,o_2} is a metric and $\Phi(f)|_T = \min f$ or $\Phi(f)|_T = \max_T f - \min f$, then the resulting distance D* **is not** *a metric. In particular, they hold only the following properties:*

- $\forall i.d(i,i) = 0$ *(zerosity)*
- $\forall i \neq j.d(i,j) \geq 0$ *(***semi-***positivity)*
- $\forall i,j.d(i,j) = d(j,i)$ *(symmetry)*

PROPOSITION 4. *Let functions $D_1,\ldots,D_n$ be metrics and $k_1,\ldots,k_n \in \mathbb{R}^+$. Then we have that the linear combination $D(o_1,o_2) = k_1 D_1(o_1,o_2) + \cdots + k_n D_n(o_1,o_2)$ is a metric.*

4.3 Computational properties

An essential issue is the cost of computing the distance between two objects. As said in Section 3, each object o_i is represented by means of a set of n_i observations. Since all the $\Phi(.)|_T$ functionals introduced so far can be obtained in a compositional way (i.e. by first computing *contributions* for some sub-intervals of T and then composing them), they can be computed by a single scan of the known points of the objects. Therefore, their corresponding spatio-temporal distances will have a complexity $O((n_1+n_2)C_d)$, where C_d is the cost of computing the contribution of each sub-interval.

PROPOSITION 5. *Let o_1 and o_2 be two spatio-temporal objects of size n_1 and n_2, reconstructed via local linear interpolation (see Section 3). Let d_{o_1,o_2} be the Euclidean distance and $\Phi(.)|_T$ one of the following functionals: maximum, minimum,* $\max f - \min f$*, average. Then the resulting D can be computed in $O(n_1+n_2)$.*

5. EFFECTS ON SOME CLUSTERING ALGORITHMS

In this section we briefly show how the representation model of the dataset contributes to the complexity of two basic classes of clustering algorithms. For one of them, moreover, it is shown how its computational cost can be reduced by means of optimisation techniques which exploit some general properties of the defined distances.

5.1 Dissimilarity Matrix-based

In this class of algorithms we include all techniques which make use only of the matrix of distances between all couples of objects contained in the database.
Basic examples are: most of the hierarchical algorithms (couples of clusters are iteratively merged, and the distance of some object o from the new cluster is obtained simply manipulating the distances of o from the two old clusters), k-medoid clustering (the representative of a cluster is always an object of the database), and graph-based clustering (the dissimilarity matrix is used as an adjacency matrix).
Typically all distances between objects are explicitly needed, so that the dissimilarity matrix has to be completely filled. This adds a complexity of $O(N_o^2 C_d)$ to the cost of the algorithm, where N_o is the number of objects in the database and C_d is the cost to compute each distance. This is usually the most expensive step of the whole algorithm.
As stated in the previous section, computing a single distance obtained by instantiating $D(.)$ typically costs $O(n_1+n_2) = O(n_p)$, where n_p is the average number of points for the objects in the database. Therefore, applying our schema to a typical dissimilarity matrix-based algorithm will lead to a complexity of $O(N_o^2 n_p)$.

5.2 K-means

When N_o is very large, computing all N_o^2 distances can become unfeasible. Therefore, algorithms such as k-means try

to compute only a small subset of distances. In particular, a representative for each of the k clusters is computed, and at each iteration of the algorithm all distances between objects in the database and cluster representatives are evaluated. This requires to compute kN_o distances at each step, and – assuming that each distance costs some factor C_d – leads to a global complexity of $O(IkN_oC_d)$, where I is the number of iterations required to reach convergence (usually quite small). Typically $Ik << N_o$, so k-means algorithms are expected to be faster than those based on the dissimilarity matrix.
In our specific context, the cost of each distance depends on the complexity of the objects involved, i.e. the number of observations/control points it is composed by. Since cluster representatives are not objects of the database, the above approximation $C_d = O(n_p)$ does not hold anymore.

PROPOSITION 6. *In a local linear interpolation context, the representative of a cluster* $C = \{o_1, \ldots, o_m\}$ *has a* $O(n_1 + \cdots + n_m) = O(mn_p)$ *complexity.*

We see then that the complexity of each iteration of the algorithm is:

$$O(\sum_{i=1}^{N_o}\sum_{j=1}^{k} m_j n_p) = O(n_p N_o \sum_{j=1}^{k} m_j) = O(n_p N_o^2)$$

where m_j is the number of objects in cluster C_j. Therefore, we can conclude that the global complexity becomes $O(IN_o^2 n_p)$, i.e. a factor I greater than the dissimilarity matrix-based algorithms.

5.3 Optimisations

If the adopted distance $D(.)$ is a metric, we can exploit the triangularity property to avoid the computation of some (expensive) distances.
The main idea is that at each iteration of the algorithm the centers of the clusters sometimes *move* very slightly, so that we need not to explicitly compute the distances between the new representative and all the objects in the database. In most of the cases the new center is still too far or too close to an object to move the latter from its old cluster.
Let call c the old cluster representative and c' the new one. For any object o, if we know its distance from the old center, the triangularity property ensures that:

$$D(o,c') \leq D(o,c) + D(c,c')$$
$$D(o,c) \leq D(o,c') + D(c',c)$$

Now, putting them together and exploiting the symmetry property (so $D(c',c) = D(c,c')$) we obtain that:

$$D(o,c) - D(c,c') \leq D(o,c') \leq D(o,c) + D(c,c')$$

This way, for each object we can obtain a set of distance intervals $[low_c, up_c]$ w.r.t. all clusters c. Comparing such intervals, for several clusters we can usually deduce that they cannot be the closest one to object o, without explicitly compute $D(o,c)$.
Figure 1 depicts a simple example where vertical segments represent the interval of values (i.e. the approximation) available for each candidate centre c', and we are looking for the centre which is closest to object o, i.e. the centre with the smallest value of $D(o,c')$: the most selective upper bound in the filtering process described above is the one

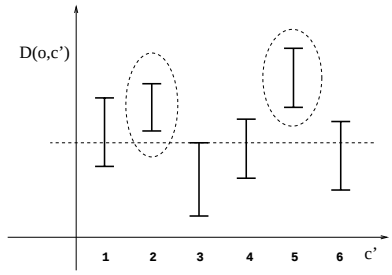


Figure 1: Filtering out far centres by approximations

with the smallest value, corresponding to the centre number 3. Then, we observe that centres 2 and 5 have a lower bound (i.e. their best possible value for $D(o,c')$) which is bigger than the upper bound (i.e. the worst possible value for $D(o,c')$) of centre 3. Thus, we are sure that such centres (circled in the picture) are farer from o than centre 3, so they can be be safely filtered out. For all other centres we cannot derive such information, so their distance w.r.t. o has to be exactly computed.
Section 6 shows the experimental performance improvements introduced by this optimisation.

6. EXPERIMENTATIONS

In order to quantify the impact of the optimisation described in Section 5.3, based on the triangularity property, we measured the performance of a k-means algorithm with an Euclidean distance as $d_{o_1,o_2}(.)$ and an average functional as $\Phi(.)$.

6.1 Sythesised dataset: The "leader" model

In the scenario simulated in our experimentations, data are populated by objects moving on a 2-dimensional space, each of them belonging to some group. Objects in the same group are characterised by the fact that they move *tendencially* together, in the way a heard of nomad animals would. This means, in particular, that at each time instant the objects in a group are not too far from each other, and when the group moves in some direction all of its objects do the same thing, although each one possibly in a slightly different way.
There are several ways to concretely model this behavior, and among them we choose a simple approach that we call *the leader model*. Essentially, the movement of the group as a whole is determined by a single object, the leader. There is exactly one leader for each cluster to model. All other objects of a group move with random speed and direction, giving preference to directions which make itself closer to the leader. This way, it is possible (or even probable, depending on the specific parameters of the simulation) that an object sometimes moves a bit far from the group, but for the next movements the trend will always be to try moving back close to the leader (and thus close to the group).
Figure 2 gives an example of a small group of objects. The graph shows a single coordinate of their positions as function of time, and the trajectory of the leader is plotted with a thicker line.

6.2 Effects of Optimisation

A set of experimentations has been performed over datasets of different sizes. Such datasets are generated following the

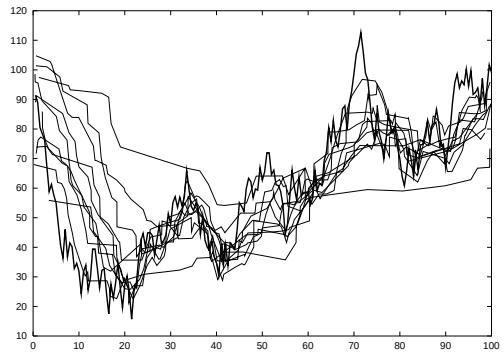


Figure 2: Trajectories

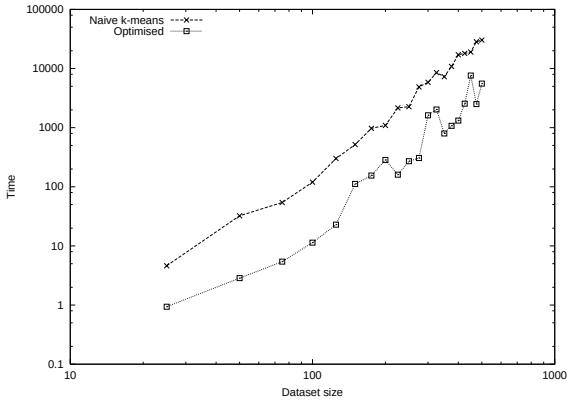


Figure 3: Naive vs. optimised k-means algorithm

methodology described in the previous section, with a size which ranges from 25 to 500 objects with a step of 25. The number of leaders, and then the number of real clusters hidden in the generated data, has been fixed to 8. Since a random component is present in the k-means algorithm for the choice of initial centres, each experiment has been executed multiple times: the values plotted in the graphs presented below represent the average of measured performances.

For the k-means clustering algorithm, two versions are tested and compared: the standard naive version and an optimised implementation developed as suggested in Section 5.3. The two algorithms are identical, sharing the same code for all steps but for the one where the closest centre is chosen for each object in the database: the first algorithm, at each iteration simply computes all $n \cdot k$ object-to-centre distances (n being the number of objects in the database and k being the requested number of clusters) and takes the smallest value; the second one, on the contrary, at each iteration computes the k distances between the old and the new centre of each cluster: this way, the object-to-centre distances computed at the previous iteration can be used to approximate the new ones, so that some of them can be labelled as "too large" without an exact computation. Having exactly the same structure, any difference in the computation time of the two algorithm is due exclusively to the specific optimisation technique.

K-means algorithms require to specify how many clusters to extract from the data. However, usually the user has not a precise *a priori* knowledge of the real number of clusters hidden in data, and so only some approximation of such number can be used. In order to model this phenomenon – at least in a very limited form – we chose to make use of a non-perfect guess of the k parameter (i.e., the number of required clusters) and fixed it to some value close but different from the real number of clusters. More precisely, all experiments described in this section have been performed requiring that the algorithms return 10 clusters, i.e. a number slightly bigger than the real one.

Figure 3 compares the execution time of both algorithms over a log-scale graph: the two curves are almost rectilinear and parallel, that is to say there exist some constants a, b, b' such that:

$$\log y = a \cdot \log x + b$$
$$\log y' = a \cdot \log x + b'$$

where y and y' represent the running time of the two versions of the algorithms for a dataset of size x. That can be rewritten as:

$$y = e^b \cdot x^a$$
$$y' = e^{b'} \cdot x^a$$

meaning that the two algorithms have a polynomial computational complexity of the same order, while the (almost) constant displacements in the graph means that the optimisations reduce the complexity of the algorithm essentially of a constant factor. More precisely, this factor corresponds to a percentage of execution time saved by the optimisations which ranges between 60.1% and 93.7%, with an average of 84.7%.

Summarising, the optimisations proposed for k-means algorithms improve performances of almost an order of magnitude. Such a result clearly proves that the overhead needed to handle the approximated distances is abundantly compensated by the amount of time saved avoiding the exact computation of distances between objects and cluster centres. Moreover, we notice that the complexity of the objects in the database (i.e., the number of observations which describe each object) does not affect in any way the overhead of our optimisation, since the latter is not concerned with the internal representation of objects. On the contrary, the time needed to compute each single object-to-object distance grows linearly w.r.t. such complexity (Proposition 5). This means that if we consider objects with larger complexity, the per cent contribution of the overhead to the cost of the algorithm will decrease, and so the per cent global improvement yielded by our optimisation is expected to grow. Saying that with other words, the relative performances of the optimised k-means algorithm are a growing monotonic function of n_p (the average number of points per object).

7. CONCLUSIONS

In this paper, we focused on the extension of the (general) clustering task to the domain of spatio-temporal objects. We introduced a general framework for defining a significant family of distances, and showed that the complexity of the data we are dealing with has a great impact on the computational cost of distances, and thus on the computational cost of the clustering algorithms. Henceforth, a set of optimisation techniques have been developed, which reduce the incidence of such computational cost, and thus increase the level of applicability of the clustering algorithms. Such optimisations have been designed for the classical k-means

algorithms and have been applied to spatio-temporal data. However, their applicability is much wider, since the only requirement is the use of metrics as distance functions, and are potentially useful for all the contexts where computing pairwise distances is an expensive task. In this sense, the spatio-temporal context is just one motivating example, albeit an important one. Other examples might include clustering of semi-structured or text documents, e.g. in XML document databases or digital libraries, as well as very high-dimensional databases.

Along the research process described in this work, several new issues have risen, which so far received only partial answers or were not discussed at all. Among the most important ones, we mention the following, which are intended to be tackled as a short term continuation of this work:

- Often, the observations which describe trajectories are given together with a measure of their approximation, therefore it would be useful to have some means to evaluate the precision of the trajectories obtained by the reconstruction process and, as a further development, the precision of the distances between objects computed following our schema;
- an appealing approach for dealing with the approximations mentioned in the previous point is the adoption of a more general domain knowledge-based trajectory reconstruction policy, and in particular a constraint-based one. As a result, the means for a full integration of the distance schema (and derived clustering algorithms) within a single Constraint Database System would be possible;
- several additional theoretical properties of our distance schema would be useful in practice, and in particular (i) convergence properties of the clustering algorithms that use it, and (ii) the development of a formal support for the definition of *well-founded centres*, together with a study of their impact on the output quality;
- so far, the optimisations introduced in this work have been implemented only for two classical clustering algorithms: k-means (in this paper), and single link hierarchical clustering (in [9]). In order to confirm the positive results obtained with such algorithms, the natural continuation of our research is to apply the underlying ideas of our optimisations to faster, subquadratic, clustering algorithms dealing with metrics;
- finally, a set of experiments over real-world datasets is planned, to test both the expressiveness of our schema (i.e., its ability to model useful notions of distance) and the performances of our optimised algorithms.

8. REFERENCES

[1] R. Agrawal, K-I. Lin, H.S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Int. Conf. on Very Large Databases (VLDB'95)*, pages 490–501, 1995.

[2] C. Faloutsos and K.-I. Lin. Fastmap: a fast algorithm for indexing, datamining and visualization of traditional and multimedia databases. In *Proceedings of SIGMOD 1995*, 1995.

[3] D. H. Fischer. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[4] S. Gaffney and P. Smyth. Trajectory clustering with mixture of regression models. In *KDD-99*, 1999.

[5] V. Ganti, R. Ramakrishnan, J. Gehrke, A. L. Powell, and J. C. French. Clustering large datasets in arbitrary metric spaces. In *Proceedings of ICDE 1999*, pages 502–511, 1999.

[6] A. Ketterlin. Clustering complex objects: the case of sequences. Technical report, LSIIT, University L. Pasteur, Strasbourg, France, 1997.

[7] A. Ketterlin. Clustering sequences of complex objects. In *Proc. of the 3rd Int'l Conf. On Knowledge Discovery and Data Mining*, Newport Beach, California, 1997.

[8] A. Ketterlin, P. Gancarski, and J. Korczak. Hierarchical clustering of composite objects with a variable number of components. In *D. H. Fisher and P. Lenz, editors, Proceedings of the Fith International Workshop on Artificial Intelligence and Statistics*, pages 229–238, Fort Lauderdale, FL, USA, 1995.

[9] Mirco Nanni. *Clustering methods for spatio-temporal data (Draft)*. PhD thesis, Dipartimento di Informatica – Universitá di Pisa, 2002.

[10] Y. Qu, C. Wang, and S.X. Wang. Supporting fast search in time series for movement patterns in multiple scales. In *CIKM'98*, pages 251–258, 1998.

Probabilistic Name and Address Cleaning and Standardisation

Peter Christen
Department of Computer Science
Australian National University
Canberra ACT 0200, Australia
Peter.Christen@anu.edu.au

Tim Churches
Centre for Epidemiology and Research
New South Wales Department of Health
Locked Mail Bag 961, North Sydney NSW 2059, Australia
tchur@doh.health.nsw.gov.au

Justin Xi Zhu
Department of Computer Science
Australian National University
Canberra ACT 0200, Australia

ABSTRACT

In the absence of a shared unique key, an ensemble of non-unique personal attributes such as names and addresses is often used to link data from disparate sources. Such data matching is widely used when assembling data warehouses and business mailing lists, and is a foundation of many longitudinal epidemiological and other health related studies.

Unfortunately, names and addresses are often captured in non-standard and varying formats, usually with some degree of spelling and typographical errors. It is therefore important that such data is transformed into a clean and standardised format before it is further processed.

Traditional approaches for cleaning and standardisation of personal information have been based on domain-specific rules that need considerable configuration by highly skilled end users. In this paper we describe an alternative approach based on probabilistic hidden Markov models. Experiments on various health-related administrative data sets show that, compared to a rules-based approach, the probabilistic system is less cumbersome and more flexible to use and, for more complex data, produces more accurate results.

Keywords

Hidden Markov models, data cleaning, data mining, record linkage, biomedical informatics, epidemiology.

1. INTRODUCTION

Most real world data collections contain noisy, incomplete and incorrectly formatted information. Thus data cleaning and standardisation are important first steps in data pre-processing, before such data can be stored in data warehouses or used for further analysis [8; 21]. The cleaning and standardisation of personal names and addresses is especially important for data integration, to make sure that no misleading or redundant information is introduced (e.g. duplicate records). Related to data integration is data linkage, the task of linking together records belonging to the same entity (patient, customer, business) from one or more data sets. Also called record linkage [7], data linkage is important in many domains such as longitudinal epidemiological studies, census related statistics, cleaning of mailing lists, and fraud and crime detection systems.

The main task of data cleaning and standardisation is the conversion of the raw input data into well defined, consistent forms and the resolution of inconsistencies in the way information is represented or encoded. Personal data is often captured and stored with typographical and phonetical variations. Moreover, such data may be recorded or captured in various, possibly obsolete, formats, and data items may be missing or contain errors. For personal data to be useful and valuable, it needs to be cleaned and standardised into a well defined format. For example, nicknames should be expanded into their full names, various abbreviations should be converted into standardised forms, and postcodes should be validated using official postcode lists. In most settings it is desirable to be able to detect and remove duplicate records from a data set, in order to reduce costs for business mailings or to improve the accuracy of a data analysis. De-duplication corresponds to linking a data set with itself.

Where a unique entity identifier or key is shared by all the data sets to be linked or deduplicated, the process of record linkage is trivial. However, this is often not the case. In many data analysis and data mining projects the data required for analysis is contained in two or more separate databases, which do not share a common unique entity identifier. In such cases, probabilistic or other record linkage techniques need to be used to merge the data [5; 25]. Typically a range of non-unique personal data items such as names, addresses and dates (e.g. date of birth) are used to link together records belonging to the same entity (e.g. patient or customer). Data linkage can be used to improve data quality and integrity, to allow reuse of existing data sources for new studies, and to reduce costs and efforts in data acquisition. In order to maximise the likelihood of successful data linkage, data must be cleaned and standardised. For example, comparing the name component from Figure 1 as one string `'Doc Peter Miller'` with a variation of the same name, like `'Mr. Miller, Peter'` would result in a non-match, whereas properly cleaned and standardised into the three name fields title, given name and surname, only the title `'mister'` would differ from the title `'doctor'` thus resulting in a partial match.

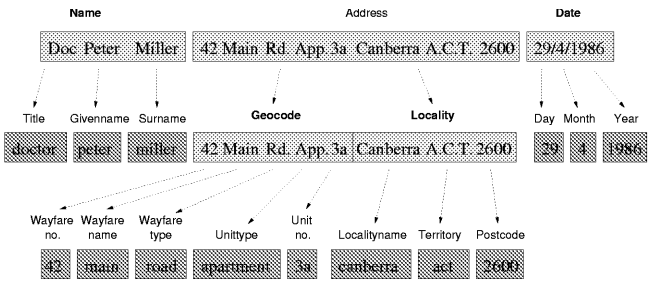


Figure 1: Example name and address standardisation.

Historical collections of administrative health data and transactional business databases nowadays contain many tens or even hundreds of millions of records, with new data being added at the rate of millions of records per annum. Although computing power has increased tremendously in the last few decades, large-scale data cleaning and linking is still a resource-intensive process. There have been relatively few advances over the last decade in the way in which data linkage is undertaken. Only recently [1; 6; 13; 21; 25] researchers started to explore this area, and the first encouraging results on using machine learning techniques are now being reported [2; 4; 15; 23; 24].

The processes of data standardisation and data linkage have various names in different user communities. While statisticians and epidemiologists speak of record or data linkage [5; 7], the same process is often referred to as data scrubbing, pre-processing, or data cleaning [6; 13; 21] by computer scientists and in the database community, whereas it is sometimes called merge/purge processing [9], data integration [3] or ETL (extraction, transformation and loading) in commercial processing of customer databases or business mailing lists. Historically, the statistical and the computer science community have developed their own techniques, and until recently few cross-references could be found.

This paper reports on a project that aims to developing new and improved techniques for data standardisation and data linkage for biomedical data sets. The focus is on improved performance, allowing to standardise and link larger data sets, and on improved quality by using techniques from machine learning and data mining. The prototype software **Febrl**, for *Freely extensible biomedical record linkage*, is currently under development[1]. **Febrl** is written in the free, open-source Python [12] programming language and is itself published under a free, open-source license allowing researchers to customise the software for their particular needs. We hope that **Febrl** will allow biomedical and other researchers to standardise and link their data sets at reduced cost, due to both the free availability of the software and the reduction of human resources needed to use it.

In Section 2 we introduce the task of data cleaning and standardisation of personal names and addresses in more detail. While traditional approaches have been based on rules that need to be customised by the user according to her or his data sets and linkage needs, in Section 3 we present an alternative technique using probabilistic hidden Markov models (HMMs). Instead of having to write sophisticated and complex rules that have to be adjusted for various data sets, the HMM approach needs training records from the data set to be standardised. In Section 3.1 we show how such training data can be created semi-automatically with much less effort and skill than is required to write rules, and in Section 4 we present first results showing that the probabilistic approach can result in highly accurate standardised data by using only small training sets. Finally, Section 5 presents an overview of related work, and Section 6 gives an outlook on current and future directions for this project.

2. CLEANING AND STANDARDISING PERSONAL INFORMATION

The aim of the data cleaning and standardisation process is to transform the raw input data records containing names, addresses and other personal information (like date of birth or gender) into a well defined and consistent form, as shown in Figure 1. Personal information can be categorised into five broad classes: **names**, **addresses**, **dates** (such as date-of-birth), **categorical attributes** (such as sex or country-of-birth) and **identifying numbers** (such as tax file or Medicare numbers). This paper deals with the first two of these classes.

Addresses can be further separated into two parts, namely the **geocode** part which contains street and postal address information, and the **locality** part which contains town or suburb names, postcode, as well as state and country information. As can be seen from Table 1, each component is further split into several output fields each containing a basic piece of information.

Name	Geocode	Locality
title	wayfare_number	postcode
gender_guess	wayfare_name	locality_name
givenname	wayfare_type	locality_qualifier
alt_givenname	wayfare_qualifier	territory
surname	unit_number	country
alt_surname	unit_type	
	property_name	
	institution_name	**Date**
	institution_type	day
	postaddress_number	month
	postaddress_type	year

Table 1: Supported output fields.

We assume that the raw input data records are stored as text files or database tables and each of the input components is made of one or more text strings. The task is then to allocate the words and numbers from the raw input into the appropriate output fields, and to clean and standardise the values in these output fields.

Our approach to data cleaning and standardisation is based on the following three steps. Firstly, the input strings are *cleaned*. Secondly, they are split into a list of words, numbers and characters, which are then *tagged* using look-up tables (mainly for names and addresses) and some hard-coded rules (e.g. for numbers, hyphens or commas). Finally these tagged lists are *segmented* into output fields using probabilistic hidden Markov models. We discuss each step in more detail in the following three sections.

[1] See project web site: `http://datamining.anu.edu.au/linkage.html`

2.1 Cleaning

The cleaning step involves converting all letters into lower case followed by various general corrections of sub-strings using correction lists, which are stored in text files and can be modified by the user. Such correction lists are made of pairs of strings `original:replacement`. If an `original` string is found in the raw input sting, it is replaced by the corresponding `replacement` string. For example, variations of *'known as'*, such as `'a.k.a.'` or `'aka'` are all replaced with the string `'known as'`. Various kinds of brackets and quoting characters are all replaced with a vertical bar `'|'`, which facilitates tagging and segmenting in the subsequent steps. Correction lists also allow the replacement string to be an empty string, in which case an original string found in the input is removed. For example, the original string `'abbrev'` can be removed from an input record, by setting is replacement string to `''`. Note that the **name** component has a different correction list than the **geocode** and **locality** components, because some corrections are specific to names or addresses. The output of the cleaning step is a cleaned string ready to be tagged in the next step.

2.2 Tagging

After an input component string has been cleaned, the next step is to split it at white-space boundaries into a list of words, numbers, characters, punctuation marks and other possible separators. Using look-up tables and some hard-coded rules each of the list elements is assigned one or more *tags*. The hard-coded rules include, for example, tagging an element as a hyphen, a comma, a slash, a number or an alphanumeric word, while most of the other tags (titles, given names, surnames, postcode, locality names, wayfare and unit types, countries, etc.) are assigned to words if they are listed in one of the look-up tables. A list of all supported tags is given in Table 2. These look-up tables are loaded from text files, which can be modified by the user. If a word (or a word sequence) is found in a look-up table, it is not only tagged, but it is also replaced by its corresponding corrected entry in the look-up table. It is possible that a word is listed in more than one look-up table. Consequently, it will be assigned more than one tag (see for example the name word `'peter'` below). Words which are not found in any look-up table and which do not match any of the hard-coded tagging rules are assigned the `'UN'` (unknown) tag. Title words like `'doctor'`, `'doc'`, `'md'` and `'phd'` for example will all be assigned the title word tag `'TI'`, and they will be replaced with the standardised word `'dr'`.

The look-up tables are searched using a *greedy* matching algorithm, which searches for the longest tuple of elements that match an entry in the look-up tables. For example, the tuple of words `('macquarie','fields')` will be matched with an entry in a look-up table with the locality name `'macquarie fields'`, rather than with the shorter entry `'macquarie'` from the same look-up table.

The output of the tagging step is a list of elements (words, numbers and separators) and a corresponding list of tags.

Example: Assume the raw input string of the name component is `'Doc. peter Paul MILLER'`, which is converted by the cleaning step into the string `'doc peter paul miller'`. Assuming that `'doc'` is listed in the title look-up table (and corrected into `'dr'`), it will be assigned a `'TI'` tag. Further assuming `'peter'` is listed in both the male given name and the surname look-up tables, it is assigned the two tags `'GM'` and `'SN'`. If the next word `'paul'` is only found in the male given name look-up table, it will only be assigned a `'GM'`. Finally, assume the name `'miller'` is only found in the surname look-up table, so it will be assigned the tag `'SN'`. Because `'peter'` has been assigned two tags, the following two permutations of tag sequences are possible:

```
['dr', 'peter', 'paul', 'miller']
['TI',   'GM',   'GM',    'SN'  ]
['TI',   'SN',   'GM',    'SN'  ]
```

The question is now which of these two tag sequences is the most likely one, and how should the elements of the word list be assigned to the appropriate output fields? This problem is solved using probabilistic hidden Markov models in the segmentation step as discussed below.

Tag	Description	Name	Geocode/ Locality
LQ	Locality qualifier words	–	LT
LN	Locality (town) names	–	LT
TR	Territory (state) names	–	LT
CR	Country names	–	LT
IT	Institution type words	–	LT
IN	Institution names	–	LT
PA	Postal address type words	–	LT
PC	Postcodes	–	LT
UT	Unit type words	–	LT
WN	Wayfare names	–	LT
WT	Wayfare type words	–	LT
ST	Saint names	LT	LT
TI	Title words	LT	–
SN	Surnames	LT	–
GF	Female given names	LT	–
GM	Male given names	LT	–
PR	Name prefix words	LT	–
SP	Name separators (like *'known as'*)	LT	–
BO	*'baby of'* and similar sequences	LT	–
NE	The word *nee* (surname or *'born'*)	LT	–
II	One-letter words (initials)	–	HC
HY	Hyphen *'-'*	HC	HC
CO	Comma *','*	HC	HC
SL	Slash *' /'*	HC	HC
N4	Numbers with four digits	–	HC
NU	Other numbers (not four digits)	HC	HC
AN	Alpha-numeric words	HC	HC
VB	Vertical bar *' |'* (various brackets)	HC	HC
RU	Rubbish	LT	LT
UN	Unknown	LT	LT

Table 2: Supported tags, based either on look-up tables (LT) or hard coded rules (HC).

2.3 Segmenting

Having a list of elements (words, numbers, characters and separators) and one or more corresponding tag lists, the task

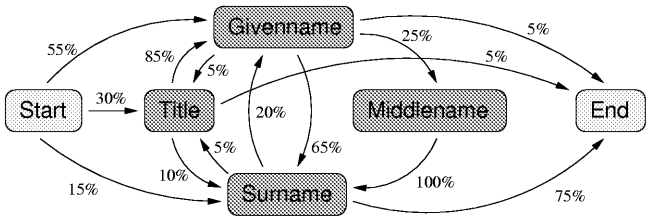


Figure 2: Simple example hidden Markov model for name component.

From state	To state: *Start*	Title	Given name	Middle name	Surname	*End*
Start	–	0.30	0.55	0.0	0.15	–
Title	–	0.0	0.85	0.0	0.10	0.05
Given-name	–	0.05	0.0	0.25	0.65	0.05
Middle-name	–	0.0	0.0	0.0	1.00	0.0
Surname	–	0.05	0.20	0.0	0.0	0.75
End	–	–	–	–	–	–

Obser. symbol	State: *Start*	Title	Given name	Middle name	Surname	*End*
TI	–	0.96	0.01	0.01	0.01	–
GM	–	0.01	0.35	0.33	0.15	–
GF	–	0.01	0.35	0.27	0.14	–
SN	–	0.01	0.09	0.14	0.45	–
UN	–	0.01	0.20	0.25	0.25	–

Table 3: Example name HMM transition and observation probabilities.

is now to assign these elements to the appropriate output fields. Traditional approaches have used rules (such as *'if an element has a tag 'TI' then the corresponding word is assigned to the* `'title'` *output field.'*). Instead, we use probabilistic hidden Markov models [20]. The advantages of hidden Markov models are their robustness with respect to previously unencountered input sequences, as well as the fact that they can be trained by clerical staff, rather than requiring high level analysis and programming skills to create complex rules that accommodate all special cases for a given data set.

3. DATA SEGMENTATION USING HIDDEN MARKOV MODELS

Traditional data cleaning and standardisation systems apply various rule-based approaches to the task of parsing raw data. For example, **AutoStan** [14] uses an initial lexicon-based tokenisation phase followed by a re-entrant rule-based parsing and token re-writing phase. The approach presented in this paper also uses lexicon-based tokenisation (or tagging, as described above), but then uses a probabilistic approach based on hidden Markov models to assign each element in the cleaned and tagged input list to a particular output field.

Hidden Markov models [20] (HMMs) were developed in the 1960s and 1970s and are widely used in speech and natural language processing. They are a powerful machine learning technique, able to handle new forms of data in a robust fashion. They are computationally efficient to develop and evaluate. Only recently have HMMs been used for name and address standardisation [2; 22].

A HMM is a probabilistic finite state machine made of a set of states, transition edges between these states and a finite dictionary of discrete observation (output) symbols. Each edge is associated with a transition probability, and each state emits observation symbols from the dictionary with a certain probability distribution. Two special states are the *start* and *end* state. Beginning from the start state, a HMM generates a sequence of observation symbols $O = o_1, o_2, \ldots, o_k$ by making $k-1$ transitions from one state to another until the end state is reached. Observation symbol $o_i, 1 \leq i \leq k$ is generated in state i based on this state's probability distribution of the observation symbols. The same output sequence can be generated by various paths through a HMM with different probabilities. Given an observation sequence, one is often interested in the most probable path through a given HMM that generated this sequence. The *Viterbi* [20] algorithm is an efficient way to compute this most probable path for a given observation sequence.

The distribution of both transition and observation probabilities are learned using training data. Each training record is an example path and observation sequence. While the dictionary of output symbols can be created using the training data, the states of a HMM are normally fixed and have to be defined before training. Because training data often does not cover all possible combinations of states and observations, during testing and application of a HMM unseen and unknown data is encountered. To be able to deal with such cases, *smoothing* techniques [2] need to be applied which enable unseen data to be handled more efficiently. These techniques – such as *Laplace* or *absolute discounting* – basically assign small observation probabilities to all unseen observations symbols in all states. For states that have more distinct words during training, they are also expected to encounter unknown symbols more frequently. Smoothing techniques reflect this fact by assigning unseen symbols in those states a higher relative probability [2].

Figure 2 shows a simple HMM example for the name component with six states. The *start* and *end* states are both virtual states that are not actually stored in a HMM, as no symbols are emitted in these states. Instead of the *start* state a list of initial state probabilities is used (i.e. probabilities that give the likelihood of a sequence starting in a certain state). In the given example, a name starts with a 55% likelihood with a *Givenname* and is followed with a (conditional) probability of 65% by a *Surname*, or 25% probability with a *Middlename*, and so on.

Instead of using the original words, numbers and other elements from the input records directly, the tag sequences (as discussed in Section 2.2) are used as HMM observation symbols in order to make the derived HMMs more general, i.e. to allow HMMs to be trained on one data set and then be used with other similar, but distinct data sets, with little or no loss of performance. Using tags also limits the size of the observation dictionary. Once a HMM is trained, sequences of tags (one tag per input element) as generated in the tagging step can be given as input to the *Viterbi* algorithm,

which returns the most likely path (i.e. state sequence) of the given tag sequence through the HMM, plus the corresponding probability. The path with the highest probability is then taken and the corresponding state sequence will be used to assign the elements of the input to the appropriate output fields.

Example: Assuming the same name example as above, the input name string `'Doc. peter Paul MILLER'` is cleaned and tagged as explained in Section 2.2 into the following word list and tag sequences:

```
['dr', 'peter', 'paul', 'miller']
['TI',   'GM',   'GM',    'SN'  ]
['TI',   'SN',   'GM',    'SN'  ]
```

These two tag sequences are given to the *Viterbi* algorithm and using the name HMM from Figure 2 with transition and observation probabilities as listed in Table 3, the first tag sequence `['TI','GM','GM','SN']` is assigned to the following path through the HMM (with the corresponding observation symbols in brackets):

```
Start -> Title (TI) -> Givenname (GM) ->
 Middlename (GM) -> Surname (SN) -> End
```

The resulting probability of this path is:

```
0.30 * 0.96 * 0.85 * 0.35 * 0.25 * 0.33 * 1.00 *
          0.45 * 0.75 = 0.0023856525
```

with `0.30` being the transition probability from state `Start` to state `Title`, then `0.96` being the probability that the symbol `'TI'` is observed in state `Title`, `0.85` being the transition probability from the `Title` to the `Givenname` state, and so on. The most probable path for the second sequence `['TI','SN','GM','SN']` is:

```
Start -> Title (TI) -> Givenname (SN) ->
 Middlename (GM) -> Surname (SN) -> End
```

which would result in a probability of:

```
0.30 * 0.96 * 0.85 * 0.09 * 0.25 * 0.33 * 1.00 *
          0.45 * 0.75 = 0.0006134534
```

Thus the first tag sequence is the most likely one, as expected. Using the HMM states, the elements of the input word list are then associated with the corresponding output fields. In this example `'dr'` will become the title, `'peter'` will become the given name, `'paul'` the middle name (or alternative given name) and `'miller'` the surname.

3.1 Hidden Markov Model Training

In our data cleaning and standardisation system we are using one HMM for names and one for addresses (geocode and locality) with the corresponding states as listed in Table 4 and Table 5. Many of the output fields can contain more than one word, therefore the HMMs appropriately have two states for these elements which are then both assigned to the corresponding output field.

Both the transition and observation probabilities for the HMMs have to be trained using collections of records that have been annotated manually, and which are taken from the same (or a similar) data set which will be used for data standardisation. Using these *training records* a HMM *learns* the characteristics of a data set.

State	Description
`titl`	Title state
`baby`	State for *baby of*, *son of* or *daughter of*
`knwn`	State for *known as*
`andor`	State for *and* or *or*
`gname1`	Given name state 1
`gname2`	Given name state 2
`ghyph`	Given name hyphen state
`gopbr`	Given name opening bracket state
`gclbr`	Given name closing bracket state
`agname1`	Alternative given name state 1
`agname2`	Alternative given name state 2
`coma`	State for comma
`sname1`	Surname state 1
`sname2`	Surname state 2
`shyph`	Surname hyphen state
`sopbr`	Surname opening bracket state
`sclbr`	Surname closing bracket state
`asname1`	Alternative surname state 1
`asname2`	Alternative surname state 2
`pref1`	Name prefix state 1
`pref2`	Name prefix state 2
`rubb`	Rubbish state, for elements to be thrown away

Table 4: Name hidden Markov model states.

Thus, instead of requiring highly trained programming staff to maintain a large number of rules that cover all kinds of special cases and exceptions, and that have to be modified for each new given data set, the data to train a HMM can be created by clerical staff within a couple of days for a new data source. Furthermore, this training process can be accelerated by *bootstrapping* it with training data sets derived from other, similar data sources.

The training data consists of sequences with one or more `tag:hmm_state` pairs. Each sequence is a training record that is given to the HMM, and the HMM learns the characteristics of a data set by using all training examples that it is given during training. Maximum likelihood estimates (MLEs) for the matrix of transition and observation probabilities for a HMM are derived by accumulating frequency counts of each type of transition and output symbol (tag) from the training records. Because frequency-based MLEs are used, it is important that the records in the training data set(s) are reasonably representative of the overall data set(s) to be standardised. However, HMMs are quite robust to unseen data and are not overly troubled if the records in the training data set(s) do not represent an unbiased sample of records from the target data. For example, it is possible to add training records which represent unusual records without degrading the performance of the HMMs on more typical records. HMMs also degrade gracefully, in that they still perform well even with records with a previously unencountered structure. A simple set of training examples for a name component might look like:

```
GF:gname1, SN:sname1
UN:gname1, SN:sname1
GF:gname1, GM:gname2, UN:sname1
GF:gname1, GM:sname1
GF:gname1, UN:gname2, SN:sname1
```

Each line in the example above corresponds to one training record, and contains a sequence that corresponds to a partic-

State	Description
wfnu	Wayfare number state
wfna1	Wayfare name state 1
wfna2	Wayfare name state 2
wfql	Wayfare qualifier state
wfty	Wayfare type state
unnu	Unit number state
unty	Unit type state
prna1	Property name state 1
prna2	Property name state 2
inna1	Institution name state 1
inna2	Institution name state 2
inty	Institution type state
panu	Postal address number state
paty	Postal address type state
hyph	State for hyphen
sla	State for slash
coma	State for comma
opbr	Opening bracket state
clbr	Closing bracket state
loc1	Locality name state 1
loc2	Locality name state 2
locql	Locality qualifier state
pc	Postcode state
ter1	Territory name state 1
ter2	Territory name state 2
cntr1	Country name state 1
cntr2	Country name state 2
rubb	Rubbish state, for elements to be thrown away

Table 5: Address hidden Markov model states.

ular path through the various (hidden, unobserved) states of the HMM together with the corresponding observation symbols (tags).

For a new data set HMMs for names and addresses can be created in four main steps. First, a small number (around 100) of records from the original data set are selected randomly and tagged as described in Section 2.2. The output of this step is one or more tag sequences for each input record (name or address component). These training records then have to be edited manually by the user by choosing the correct tag sequence (if more than one has been created for one input record) and by adding the appropriate HMM state to each of the tags in the selected tag sequence. The resulting small number of training records (like the examples above) are then used in the second step to train a first rough HMM, which in the third step is used to create a larger set of training records, again randomly selected from the original data set. These training records now already have a tag sequence that includes HMM states. The user then has to go through these training records and correct tags or HMM states if they are wrong. In the file containing the training records, the original input is commented out, so the user can easily check if a training record is correct or needs to be modified. The two examples of address training records below show the original input string, the cleaned input string at the end of the tagging routine (the tagged word list concatenated back into a string) and the corresponding tag sequence which will be taken as training example. Note that both the original input and the tagged input are commented using the Python comment character hash.

```
# '2 richard st lewisham 2049 nsw'
# '2 richard street lewisham 2049 new_south_wales'
NU:wfnu, UN:wfna1, WT:wfty, LN:loc1, PC:pc, TR:ter1

# '42/131 miller pl manley 2095 nsw'
# '42 / 131 miller place manly 2095 new_south_wales'
NU:unnu, SL:sla, NU:wfnu, UN:wfna1, WT:wfty,
LN:loc1, PC:pc, TR:ter1
```

Using this *bootstrapping* approach a single user can create and validate in one or two days work enough training records to be able to standardise real world data sets with a high accuracy as shown in the following section.

4. STANDARDISATION RESULTS

We developed and evaluated the **Febrl** system using three routinely-collected health data sets which contain personal identifying details. Access to these data sets for the purpose of this project was approved by the Australian National University Human Research Ethics Committee and by the relevant data custodians within the NSW Department of Health. The data sets used in this project were held on secure computing facilities at the Australian National University and the NSW Department of Health head offices. Access to the data sets used in this project was strictly limited to the investigators. All investigators, as well as the system administrators of the computing facilities, were required to sign a confidentiality agreement which apprised them of their responsibilities as well as the legislative protection (and associated criminal penalties for misuse) afforded to the data.

In order to minimise the invasion of privacy which is necessarily associated with every use of identified data, all medical and health status details and other personal details apart from name (on two of the data sets), date-of-birth (on one of the data sets), sex and residential address were removed from the data sets used in this project. The **Febrl** software under development is multi-platform capable, and we were able to tag, clean and segment names and address on both 64-bit Unix and 32-bit Windows platforms without problems or modifications.

4.1 Address Standardisation

The performance of the **Febrl** system with typical Australian address data was evaluated using two data sets. The first was a subset of approximately 1 million addresses taken from uncorrected electronic copies of NSW death certificates as completed by medical practitioners and coroners in the years 1988 to 2002. The information systems which captured these data underwent a number of major changes during this period. The majority of these data were entered from handwritten forms.

The second data set was a random sample of 1,000 records of residential addresses drawn from the NSW Inpatient Statistics Collection for the years 1993 to 2001. This collection contains abstracts for every admission to a public- or private-sector acute care hospital in NSW. Most of the data are extracted from a range of computerised hospital information systems, with a smaller proportion entered from paper forms.

A number of tests were carried out:

1. An initial *bootstrap* hidden Markov model (HMM) was trained using 100 random death certificate (DC) records, and this was used to form a larger training set of 1,100 randomly chosen DC records. A HMM derived from this second training set was then used to standardise 50,000 randomly chosen DC records, and records with unusual patterns of observation symbols (with a frequency of six or less) were examined, corrected and added to the training set if the results produced by the HMM were incorrect. A new HMM was then derived from this augmented training set and the process repeated a further three times, resulting in the addition of approximately 250 extra training records (bringing the total number of training records to 1,450). The HMM which emerged from this process, designated HMM1, was used to standardise 1,000 randomly chosen DC test records and the accuracy of the standardisation was assessed. Laplace smoothing used in this and all subsequent tests.

2. HMM1 was then used to standardise 1,000 randomly chosen Inpatient Statistics Collection (ISC) records, and the accuracy was assessed. In other words, a HMM trained using one data source (DC) was used to standardise addresses from a different data source (ISC) without any retraining of the HMM.

3. An additional 1,000 randomly chosen address training records derived from the Midwives Data Collection (MDC) were then added to the 1,450 training records described above, and this larger training set was used to train HMM2. HMM2 was then used to re-standardise the same sets of randomly chosen test records described in steps 1 and 2 above, and the results were evaluated.

4. Approximately 60 training records, based on archetypes of those records which were wrongly standardised in all of the preceding tests, were then added to the training set to produce HMM3. HMM3 was then used to re-standardise the same DC and ISC test sets. Thus, HMM3 could be considered an "overfitted" model for the particular records in the two test sets, although in practice researchers are likely to use such "overfitting" to maximise standardisation accuracy for the specific data sets used in their particular study.

5. Finally, by way of comparison, the same two 1,000 record test data sets were standardised using the commercial **AutoStan** [14] software in conjunction with a rule set which had been developed and refined by two of the investigators (TC and KL) over the course of several years for use with ISC (but not DC) address data.

For all tests, records were judged to be accurately standardised when all of the elements present in the input address string, with the exception of punctuation, were allocated to the correct output field, and the values in each output fields were correctly transformed to their canonical form where required. Thus, a record was judged to have been incorrectly standardised if any element of the input string was not allocated to an output field, or if any element was allocated to the wrong output field. Due to resource constraints, the investigators were not blinded to the process used to standardise the records. Results are shown in Table 6.

	HMM/Method			
Test Data Set (1,000 records each)	HMM 1	HMM 2	HMM 3	Auto Stan
Death Certificates	95.7	96.8	97.6	91.5
Inpatient Statistics Collection	95.7	95.9	97.4	95.3

Table 6: Proportion of correctly standardised address records (all numbers are percentages).

These results indicate that the HMM approach described in this paper produces standardisation accuracies which are comparable to those produced by a well-established rule-based system when used on the data set for which those rules were developed, and superior results when used on a different data set. In other words, the HMM approach appears to be less sensitive to the particular characteristics of the data source for which it was developed than a widely-used rule-based system.

In addition, the results indicate that although the HMMs are trained using maximum likelihood estimates, they are quite robust with respect to the source of the training data and their performance can even be improved by the addition of a small number of unrepresentative training records which represent "difficult" cases.

In those records which were not accurately standardised by the HMMs, an average of 83 per cent of all data elements present in the input record were allocated to the correct output fields – in other words, even these incorrectly standardised records would have considerable utility. In only two test records (out of 2,000) were all of the elements wrongly assigned, and both of these were foreign addresses in non-English speaking countries. The performance of **AutoStan** in this respect was similar.

Finally, the addition of a small number of deterministic post-processing rules is expected to yield even higher accuracies in future versions of the **Febrl** system.

4.2 Name Standardisation

Accuracy measurements on names were conducted using a subset of the *NSW Midwives Data Collection (MDC)*. This subset contained 962,776 records with personal information (but no medical details) of women who had given birth in New South Wales, Australia, over a ten year period (1990-2000). Most of the data was entered from hand-written forms, although some of the data for the latter years was extracted directly from a variety of computerised obstetric information systems.

A random subset of 10,000 records (around a 1% sample) with a non-empty name components were selected and split into 10 test sets each containing 1,000 records. A 10-fold cross validation study was performed, with each of the folds having a training set of 9,000 records and the remaining 1,000 records being the test set. The training records were tagged in about 10 person-hours using the bootstrapping method as explained in Section 3.1. Hidden Markov models were then trained without smoothing, and both with Laplace and absolute discount [2] smoothing, respectively.

Compared to the variation in the format of residential addresses, names in the MDC are rather homogeneous. Out of the 10,000 randomly selected names, around 85% were of the simple form *'givenname surname'*, and further 9% of either the form *'givenname givenname surname'* or *'givenname surname surname'*. Thus the trained HMMs had very few non-zero transition probabilities, and many HMM states were not linked (with non-zero transitions) to the *active* HMM part.

Table 7 shows accuracy results of the HMM and a rules-based name standardisation algorithm which is also implemented in the **Febrl** prototype software. Given the simple form of most names, the rules based approach was very accurate, achieving 97% and better. The variations in the HMM approach were much higher, with up to 17% of names wrongly standardised. There were almost no differences in the effect of the smoothing method, therefore we only report results based on the unsmoothed HMM here. In many cases the same errors in standardisation occurred for both unsmoothed and smoothed HMMs, with a maximum of less than five records standardised differently per 1,000 records.

	Min	Max	Average	StdDev
HMM	83.1	97.0	92.0	±4.7
Rules	97.1	99.7	98.2	±0.7

Table 7: Name standardisation with 10-fold cross-validation (all numbers are percentages).

Especially problematic seemed to be names with either two given or two surnames. Often the HMMs misclassified the middle name as first surname instead of second given name. This is due to the large number of names with the simple form *'givenname surname'* which results in a very high transition probability from the first given name state `'gname1'` to the first surname state `'sname1'`. A second given name therefore is often assigned as a first surname, and the real surname as a second surname. Other problems were surnames that were listed in a given name look-up table only and thus tagged with a given name tag (`'GF'` or `'GM'`), in which case the HMMs wrongly assigned the second name to the given name instead of the surname output field.

It is clear from these results that some additional rules could usefully be added to the rules-based name standardisation, as well as some rule-based post-processing to the HMM segmentation process. One simple example (both for the rules and HMM approaches) is: *'if no surname is given, but two given name are present, re-assign the second given name word to surname'* (assuming given names are generally written before surnames, otherwise the opposite way).

5. RELATED WORK

The terms *data cleaning* (or *data cleansing*), *data standardisation*, *data scrubbing*, *data pre-processing* and *ETL* (extraction, transformation and loading) are used synonymously to refer to the general tasks of transforming source data (often derived from operational, transactional information systems) into clean and consistent sets of records which are suitable for loading into databases and data warehouses, and for linking with other data sets [21]. Once the data has been standardised, the central task of data linkage is to identify records in the source data sets that represent the same real-world entity. In the computer science literature, this process is also called the object identity or merge/purge problem [9].

Fuzzy techniques and methods from information retrieval have been used to address the data linkage problem, with varying degrees of success. One approach is to represent text (or records) as document vectors and compute the *cosine distance* [3] between such vectors. Another possibility is to use an *SQL* like language [6] that allows approximate joins and cluster building of similar records, as well as decision functions that decide if two records represent the same entity. Other methods [13] include statistical outlier identification, pattern matching, clustering and association rules based approaches. Sorting data sets (to group similar records together) and comparing records within a sliding window [9] is a technique similar to blocking as applied by traditional record linkage approaches. The accuracy of the linkage can be improved by having smaller window sizes and performing several passes over the data using different (often compound) keys, rather than having a large window size but only one pass. This corresponds to applying several blocking strategies in a record linkage process.

Machine learning techniques have been applied to data linkage in recent years. The authors of [4; 24] describe a hybrid system that in a first step uses unsupervised clustering on a small sample data set to create data that can be used by a classifier in the second step to classify records into links and non-links. The authors do not directly address the problem of data cleaning and standardisation, rather they use approximate string comparison algorithms to be able to deal with variations in strings. Clustering of large and high-dimensional data sets with applications to matching is discussed in [15]. The authors propose a two step clustering algorithm that in the first step uses cheap approximate distance metrics to form overlapping canopies, which are in a second step clustered using traditional approaches. Another approach [16] learns field specific string-edit distance weights and a binary classifier based on support vector machines (SVM) to find duplicate records in text databases.

Commercial software for data cleaning and standardisation is available from various vendors (see the project web page for a non-exhaustive list of data cleaning and data integration software). Most of these products use proprietary technologies, but many are rules based. The **AutoStan / AutoMatch** [14] suite of data cleaning and linkage software for example improved on simple (or far-from-simple) regular expression rules by using an initial lexicon-based tokenisation phase followed by a re-entrant rule-based parsing and token re-writing phase. Probabilistic approaches as the one presented in this paper on the other hand allow the creation of training data in a much less time consuming fashion.

While hidden Markov models have traditionally been applied in areas like signal and speech processing [20], text recognition, and image processing, only recently they have been applied to the tasks of information extraction [22] (e.g. extracting names, titles and keywords from publication abstracts) and segmentation of addresses and bibliographic records [2], where an input string containing an address has to be segmented into well defined output fields. The approach discussed in this paper differs in that instead of using address words directly, only tags are given to the HMMs, which results in a system that can handle unseen data more robustly and is also computationally more efficient due to its smaller dictionaries of output symbols.

6. CONCLUSIONS AND OUTLOOK

In this paper we presented a probabilistic approach for the task of cleaning and standardising personal names and addresses using hidden Markov models. This process is not only an important first step before data can be loaded into databases or data warehouses, it is also necessary before data can be linked or integrated with other data. We have shown that the probabilistic approach is not only easier and less cumbersome to use compared to the traditional rule based approach, it can also result in a higher accuracy. The methods presented are part of a prototype software system, which is published under an open source license and which can be downloaded from the project web page (see `http://datamining.anu.edu.au/linkage.html`).

Currently we are developing new methods for data linkage based on the probabilistic approach as developed by Fellegi and Sunter [5] and extended by others. The two main foci are to improve the linkage performance by applying techniques from high-performance and parallel computing, as well as improving the linkage quality by exploring machine learning techniques like clustering and predictive modelling. The aim is to provide researchers and users in the biomedical and related areas with the ability to clean, standardise and link much larger data sets at reduced costs, due to the reduction in human resources needed and the free availability of the software.

Acknowledgments

This project is equally funded by the *Australian National University (ANU)* and the *NSW Department of Health* under an *AICS (ANU-Industry Collaboration Scheme) AICS #1-2001.* The authors would like to thank everybody who supported this project and helped to make it happen: Ole Nielsen, Markus Hegland, Stephen Roberts and David Bulbeck. We are specially grateful to Kim Lim (Centre for Epidemiology and Research, NSW Department of Health) for her input into the design of the prototype software and in helping to test and debug it.

7. REFERENCES

[1] G.B. Bell and A. Sethi, *Matching Records in a National Medical Patient Index*, Communications of the ACM, Vol. 44 No. 9, September 2001.

[2] V. Borkar, K. Deshmukh and S. Sarawagi, *Automatic segmentation of text into structured records*, in Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, California, 2001.

[3] W.W. Cohen, *The WHIRL Approach to Integration: An Overview*, in Proceedings of the AAAI-98 Workshop on AI and Information Integration. AAAI Press, 1998.

[4] M.G. Elfeky, V.S. Verykios and A.K. Elmagarmid, *TAILOR: A Record Linkage Toolbox*, Proceedings of the ICDE' 2002, San Jose, USA, 2002.

[5] I. Fellegi and A. Sunter, *A theory for record linkage.* In Journal of the American Statistical Society, 1969.

[6] H. Galhardas, D. Florescu, D. Shasha and E. Simon, *An Extensible Framework for Data Cleaning*, Technical Report 3742, INRIA, 1999.

[7] L. Gill, *Methods for Automatic Record Matching and Linking and their use in National Statistics*, National Statistics Methodology Series No. 25, London 2001.

[8] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2000.

[9] M.A. Hernandez and S.J. Stolfo, *The Merge/Purge Problem for Large Databases*, in Proceedings of the SIGMOD Conference, San Jose, 1995.

[10] C.W. Kelman, *Monitoring Health Care Using National Administrative Data Collections*, PhD thesis, Australian National University, Canberra, May 2000.

[11] A.J. Lait, and B. Randell, *An Assessment of Name Matching Algorithms*, Technical Report, Department of Computing Science, University of Newcastle upon Tyne, UK 1993.

[12] M. Lutz, *Python Pocket Reference, Second Edition*, O'Reilly and Associates, January 2002.

[13] J.I. Maletic and A. Marcus, *Data Cleansing: Beyond Integrity Analysis*, in Proceedings of the Conference on Information Quality (IQ2000), Boston, October 2000.

[14] *AutoStan and AutoMatch, User's Manuals*, MatchWare Technologies, Kennebunk, Maine, 1998.

[15] A. McCallum, K. Nigam and L.H. Ungar, *Efficient clustering of high-dimensional data sets with application to reference matching*, Knowledge Discovery and Data Mining, 169-178, 2000.

[16] U.Y. Nahm, M. Bilenko and R.J. Mooney, *Two Approaches to Handling Noisy Variation in Text Mining*, in Proceedings of the ICML-2002 Workshop on Text Learning (TextML'2002), pp.18-27, Sydney, Australia, July 2002.

[17] H.B. Newcombe and J.M. Kennedy, *Record Linkage: Making Maximum Use of the Discriminating Power of Identifying Information*, Communications of the ACM, Vol. 5 No. 11, 1962.

[18] L. Philips, *The Double-Metaphone Search Algorithm*, C/C++ User's Journal, Vol. 18 No. 6, June 2000.

[19] E.H. Porter and W.E. Winkler, *Approximate String Comparison and its Effect on an Advanced Record Linkage System*, Research Report RR97/02, US Bureau of the Census, 1997.

[20] L.R. Rabiner, *A tutorial on Hidden Markov Models and selected applications in speech recognition*, in Proceedings of the IEEE, vol. 77, no. 2, February 1989.

[21] E. Rahm and H.H. Do, *Data Cleaning: Problems and Current Approaches*, IEEE Bulletin of the Technical Committee on Data Engineering, Vol. 23 No. 4, December 2000.

[22] K. Seymore. A. McCallum and R. Rosenfeld, *Learning Hidden Markov Model Structure for Information Extraction*, in Proceedings of AAAI-99 Workshop on Machine Learning for Information Extraction, 1999.

[23] V.S. Verykios, A.K. Elmagarmid and E.N. Houstis, *Automating the Approximate Record-Matching Process*, Information Sciences, Vol. 126, July 2000.

[24] V.S. Verykios, A.K. Elmagarmid, M.G. Elfeky, M. Cochinwala and S. Dalal, *On the Completeness and Accuracy of the Record Matching Process*, in Proceedings of the MIT Conference on Information Quality, Boston, MA, October 2000.

[25] W.E. Winkler, *The State of Record Linkage and Current Research Problems*, U.S. Census Bureach Research Report RR99/04, 1999.

[26] W.E. Winkler, *Quality of Very Large Databases*, Research Report RR2001/04, US Bureau of the Census, 2001.

[27] W.E. Yancey, *Frequency-Dependent Probability Measures for Record Linkage*, Research Report RR00/07, Statistical Research Division, US Bureau of the Census, July 2000.

[28] W.E. Yancey, *BigMatch: A Program for Extracting Probable Matches from a Large File for Record Linkage*, Research Report RR 2000-01, Statistical Research Division, US Bureau of the Census, March 2002.

Building a Data Mining Query Optimizer

Raj P. Gopalan Tariq Nuruddin Yudho Giri Sucahyo

School of Computing
Curtin University of Technology
Bentley, Western Australia 6102

{raj, nuruddin, sucahyoy} @computing.edu.au

ABSTRACT

In this paper, we describe our research into building an optimizer for association rule queries. We present a framework for the query processor and report on the progress of our research so far. An extended SQL syntax is used for expressing association rule queries, and query trees of operators in an extended relational algebra for their internal representation. The placement of constraints in the query tree is discussed. We have developed an efficient algorithm called CT-ITL for lower level implementation of frequent item set generation which is the most critical step of association rule mining. The performance evaluations show that our algorithm compares well with the most efficient algorithms available currently. We also discuss further steps needed to reach our goal of integrating the optimizer with database systems.

Keywords

Knowledge discovery and data mining, query optimization, association rules, frequent item sets.

1. INTRODUCTION

Data mining is used to automatically extract structured knowledge from large data sets. Among the different topics of research in data mining, the efficient discovery of association rules has been one of the most active. Association rules identify relationships among sets of items in a transaction database. They have a number of applications such as increasing the effectiveness of marketing, inventory control and stock location on the shop floor. Since the introduction of association rules in [1], many researchers have developed increasingly efficient algorithms for their discovery [3], [8], [11], [19].

As the computational complexity of association rules discovery is very high, most of the research effort has focused on the design of efficient algorithms. These algorithms generally use simple flat files as input and are not integrated with database systems. Imielinski and Mannila [9] identified the need to develop data mining systems similar to DBMSs that manage business applications. They also suggested developing features such as query languages and query optimizers for these systems.

A number of tightly coupled integration schemes between data mining and database systems have been reported. Agrawal and Shim described the integration of data mining with IBM DBS/CS RDBMS using User Defined Function (UDF) [2]. Exploration of different architectural alternatives for database integration and comparisons between them on performance, storage overhead, parallelization, and inter-operability were presented in [17]. Recently, the impact of file structures and systems software support on mining algorithms was discussed in [16]. However, most of these proposals focus on enhancing existing DBMSs for Data Mining and do not deal with the problem of integrating the data mining algorithms with database technology to support data mining applications.

Several researchers have proposed query languages for discovering association rules. Imielinski et al. introduced the MINE operator as a query language primitive that can be embedded in a general programming language to provide an Application Programming Interface for database mining [10]. Han et al. proposed DMQL as another query language for data mining of relational databases [7]. Meo et al. described MINE RULE as an extension to SQL, including examples dealing with several variations to the association rule specifications [12]. However, all these proposals focus on language specifications rather than algorithms or techniques for optimizing the queries.

Chaudhuri suggested that data mining should take advantage of the primitives for data retrieval provided by database systems [4]. However, the operators used for implementing SQL are not sufficient to support data mining applications [9]. Meo et al. gave the semantics of the MINE RULE operator using a set of nested relational algebra operators [12]. Probably because their objective was only to describe the semantics of MINE RULE, the expressions of their algebra are far too complex for internal representation of queries or for performing optimization.

In this paper, we discuss our research into building a mining query optimizer that can be integrated with database systems using a common algebra for the internal representation of both data mining and database queries. We describe the general framework of the optimizer and report on our progress so far. We have focused our efforts on the critical components needed for building the optimizer.

We specify an extended SQL syntax for expressing association rule queries. The queries are represented internally as query trees of an extended relational algebra. The algebraic operators are grouped into modules to simplify the query tree. Several constraints that reduce the number of association rules generated can be integrated with the different modules. Alternative execution plans may be generated, by assigning algorithms to implement various operations in the query tree, from which an optimal plan can be chosen based on cost estimates.

We have developed an efficient algorithm called CT-ITL for lower level implementation of frequent item set generation which is the most critical step of association rule mining. The performance evaluations show that our algorithm compares well with the most efficient algorithms available currently. The algorithm and comparisons of its performance with other well-known algorithms on some typical test data sets are presented. We

also discuss further steps needed to reach our goal of integrating the optimizer with database systems.

The structure of the rest of this paper is as follows: In Section 2, we provide the framework for building an optimizer for data mining systems. In Section 3, we discuss the query language and the algebraic operators used in the query tree. The integration of constraints into the query tree is also dealt with in this Section. In Section 4, we present our algorithm for mining frequent item sets along with an evaluation of its performance and also comparison with other algorithms. Section 5 contains the conclusion and pointers for further work.

2. DEFINITION OF TERMS AND OPTIMIZER FRAMEWORK

In this Section, we define the terms used in describing association rules and then describe the framework of the optimizer.

2.1 Definition of Terms

We give the basic terms needed for describing association rules using the formalism of [1]. Let $I=\{i_1,i_2,\ldots,i_n\}$ be a set of items, and D be a set of transactions, where a transaction T is a subset of I ($T \subseteq I$). Each transaction is identified by a *TID*. An association rule is an expression of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$ and $X \cap Y = \varnothing$. Note that each of X and Y is a set of one or more items and the quantities of items are not considered. X is referred to as the *body* of the rule and Y as the *head*. An example of association rule is the statement that 80% of transactions that purchase A also purchase B and 10% of all transactions contain both of them. Here, 10% is the *support* of the item set {A, B} and 80% is the *confidence* of the rule $A \Rightarrow B$. An item set is called a *large item set* or *frequent item set* if its *support* is greater than or equal to a *support threshold* specified by the user, otherwise the item set is *small* or *not frequent*. An association rule with the *confidence* greater than or equal to a *confidence threshold* is considered as a valid association rule.

2.2 Optimizer Framework

As shown in Figure 1, the optimizer framework consists of seven stages similar to the process structure of relational query optimizers. These stages are described below.

Stage 1: Accept Query. The user submits a query written in a suitable query language. The query is parsed and checked for correct syntax. We use SQL extended with features of MINE RULE operator proposed by Meo et al. [12] as our query specification language. The language syntax is given in the next section.

Stage 2: Translation into Algebra. The parsed user queries are translated into an extended nested algebra that will be presented in Section 3. As the extended algebra is a super set of nested algebras for expressing database queries, it is suitable for representing both database and data mining queries.

Stage 3: Logical Transformation. As in traditional query optimisation, the objective of this stage is to reduce the size of intermediate results by using transformations that are independent of data values. The incorporation of mining constraints into the query tree is performed in this stage.

Stage 4: Select Algorithms to Implement Operators. There are several possible low-level algorithms to implement each operator. The main goal of this stage is to identify ways to execute the operators in the query tree.

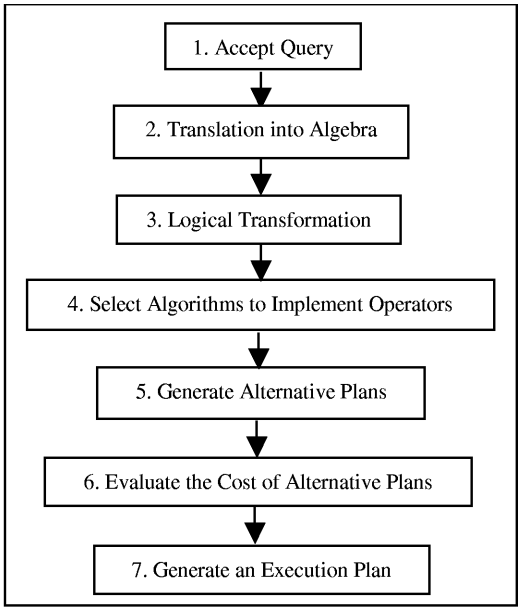


Figure 1. The optimizer framework

Stage 5: Generate Alternative Plans. Alternative plans are generated from combinations of alternative algorithms for the various operators. In a plan, every node of the query tree is associated with an algorithm to execute the corresponding operator. The plans can include existing data mining algorithms such as Apriori, OP, Eclat, and CT-ITL as alternative low-level procedures.

Stage 6: Evaluate the Cost of Alternative Plans. Suitable formulae are to be used to compute the cost of execution of each algorithm so that the cost of alternative query plans can be estimated. The cost of a plan is estimated by considering all its operations.

Stage 7: Generate an Execution Plan. The plan that has the lowest cost is chosen as the execution plan.

3. BUILDING THE QUERY TREE

In this section, we describe the construction of query trees in the optimizer. First, we present the query language and the algebraic operators. It is followed by the description of a sample query tree and the grouping of operators into modules. We also discuss the integration of constraints into the query tree.

3.1 Query Language

As mentioned in Section 2, we use SQL extended with the features of MINE RULE as our query specification language. We have developed a parser and a syntax checker for it (see Figure 2 for the syntax). The query is divided into two blocks: Data Preparation Block (DPB) and MINE RULE Specification Block (MSB).

DPB is used to specify the preparation of input data for the mining algorithm. Its syntax is similar to that of traditional SQL queries. MSB is used to specify the parameters such as *support threshold* and *confidence threshold* for data mining applications.

For example, a query to extract association rules on items having price greater than $200 from Sales table with *support threshold* 20% and *confidence threshold* 30%, can be expressed as follows:

```
MINE RULE SimpleAssociations AS
   SELECT 1..n item AS BODY,
          1..1 item AS HEAD, SUPPORT, CONFIDENCE
      WITH SUPPORT 0.2, CONFIDENCE 0.3
      FROM Sales
      WHERE price > 200
      GROUP BY tid;
```

```
          MINERULE SPECIFICATION BLOCK
  <MineRuleOp> :=
     MINE RULE <RuleListName> AS
        SELECT <BodyDescr>, <HeadDescr>
           [, SUPPORT] [, CONFIDENCE]
        WITH SUPPORT <real>,
           CONFIDENCE <real>
           [ WHERE <WhereClause> ]

             DATA PREPARATION BLOCK
        FROM <FromList>
           [ WHERE <WhereClause> ]
        GROUP BY <AttributeList>
           [ HAVING <HavingClause> ]
        [ CLUSTER BY <AttributeList>
           [ HAVING <HavingClause> ] ]

<BodyDescr> :=
    [ <CardSpec> ] <AttributeList> AS BODY
<HeadDescr> :=
    [ <CardSpec> ] <AttributeList> AS HEAD
<CardSpec>   := <Number> .. (<Number> | N)
<AttributeList> :=
   AttributeName | {, <AttributeList>}
```

Figure 2. SQL extended with features of MINE RULE

Selecting items with price greater than $200 is an example of a simple query constraint. With our syntax, users can include constraints in MSB and DPB (see Section 3.4).

3.2 Operators of the Algebra

As indicated in [9], existing operators of SQL need to be extended for data mining applications. In this paper, we use an extended nested relational algebra for expressing data mining queries. The concepts of nested relations are an integral part of current object-relational database systems, and nested algebra operations are integrated into most of the object-relational query languages. The operators needed for expressing association rule queries are shown in Table 1, where each operator is described briefly.

3.3 Query Tree for Association Rules Mining

The query tree in Figure 3 represents the example of association rules query given in Section 3.1. Grouping of operator sequences into modules simplifies the query tree as shown in Figure 3a. These modules correspond to the commonly recognized steps in association rule mining: data preparation, frequent item set mining, and association rule generation.

In Step 1 of Figure 3b, the attributes *tid* and *item* are projected from the input relation *R*, and the tuples are nested on *tid* in Step 2.

In Step 1 of Figure 3c, the powerset of *items* for each *tid* is generated. The output of this step is unnested on the *item set* attribute in Step 2 so that the number of occurrences of each item set can be counted in Step 3. In Step 4, the item sets that would fall below the minimum support, *minsup* are pruned and the support value, *sup* is computed in Step 5.

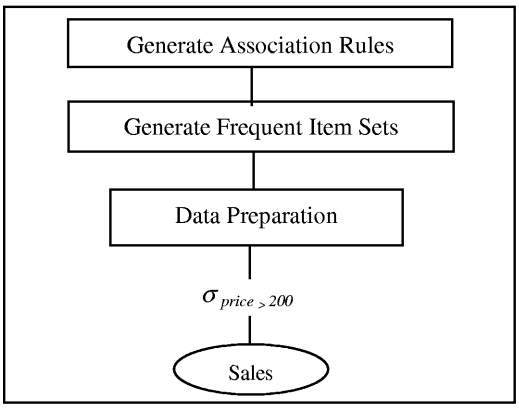
Table 1. Nested algebra operators

No	Operator	Description
1	SELECT (σ)	Returns a collection of tuples in a relation that satisfy a given condition. The condition can include set comparisons.
2	PROJECT (π)	Returns existing attribute values or new attribute values computed by functions specified as `Lambda` expressions. It can also be used to rename attributes.
3	JOIN ($\bowtie$)	Combines every pair of related tuples of two specified relations that satisfy the join condition into a single tuples of the result relation. Join condition can include set comparisons.
4	GROUPING ($\Im$)	Groups tuples based on the grouping attributes and computes aggregate functions (average, count, max, min, sum, etc.) on other specified attributes.
5	CARDINA-LITY (ς)	Counts the number of values of a set valued attribute in each tuple of a relation.
6	NEST (Γ)	Groups tuples together based on common values of specified nesting attributes. The resulting relation will contain exactly one tuple for each combination of values of the nesting attributes.
7	UNNEST (η)	Undoes the effect of `NEST`, restructures the set of tuples and flattens out the relation so that each set member (tuple) may be examined individually.
8	POWERSET ($\wp$)	Generates a powerset of values in a specified attribute. For a specified set valued attribute containing `n` values in a tuple, the `POWERSET` operator generates a tuple in the output relation, by replacing the input set value by its 2^n-1 subsets ($\varnothing$ is not included).

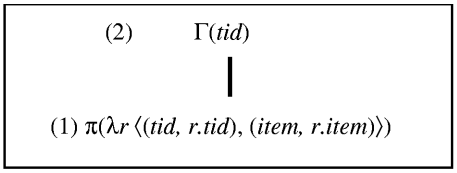
In Step 1 of Figure 3d, two copies of large item sets (*A* and *B*) are used as input. They are joined on the condition that the item set of a tuple in *A* is a proper subset of the item set of a tuple in *B*. In Step 2, the attributes are renamed to identify the smaller item set as body (*BD*) of potential rules and its corresponding larger item set as superset (*sp*). In step 3, the confidence values for candidate association rules are computed and then rules that have *confidence* greater than or equal to the *confidence threshold* are selected in Step 4.

In Step 5, the head (*HD*) of the rules are obtained by applying the set difference operator, and the result consisting of the body (*BD*), head (*HD*), support and confidence are projected for all valid

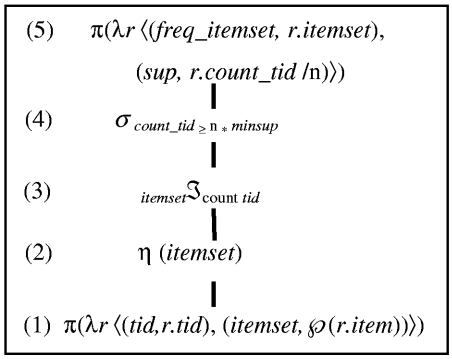
rules. More examples of query trees representing typical association rule queries can be found in [5].



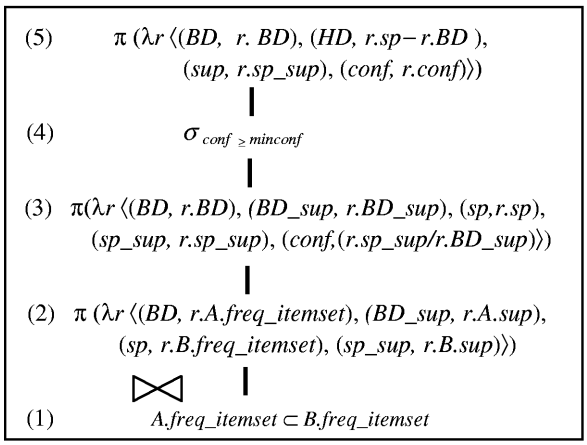
(a) Basic Query Tree



(b) Data Preparation



(c) Generate Frequent Item Sets



(d) Generate Association Rules

Figure 3. Query Tree for Association Rules Mining

A limitation of our algebra is the lack of recursion. Therefore, we have used the power set operator, although it is not suitable for efficient implementation. All the existing algorithms for frequent item set mining can be viewed as computing a partial power set of item sets present in the transactions. Instead of generating all the item sets first and then testing for support as in the query tree of Figure 3c, they try to limit the generation of item sets to those satisfying the minimum support constraint. These algorithms adopt strategies to test the support of item sets as they are generated as well as to avoid generating item sets that cannot possibly have minimum support. We have also developed an efficient algorithm that can replace the frequent item set module of the query tree in actual implementation. This algorithm will be discussed in Section 4.

3.4 Integrating Constraints into the Query Tree

Specification of constraints in mining queries can reduce the volume of uninteresting rules generated, as well as allow greater focus on the user's needs and interests at the application level. The importance of constraint-based mining has been highlighted by recent work in this area [13], [14]. A significant advantage of the algebraic query tree representation is the insight it provides for dealing with constraints in mining queries.

A categorization of constraints at the application level was proposed in [13] as follows:

1. *Item Constraint*: Specifies what are the particular individual or groups of items that should or should not be present in the pattern. E.g., mine only beverage products.
2. *Length Constraint*: Specifies restrictions on the length of patterns. E.g., mine only transactions with at least five items.
3. *Model-based Constraint*: Looks for patterns that are sub- or super patterns of some given patterns (models). For example, a shop manager may be interested in what other goods a customer is likely to buy if she buys 'Pepsi' and 'Hot Dog'.
4. *Aggregate Constraint*: For items in a pattern, find aggregates such as MAX, MIN, AVG, and SUM. For example, a shop manager wants to find frequent item sets where the average price of items is $50.

The main idea for efficient processing of constrained queries is to push the constraints down the query tree, so that the size of intermediate results is reduced. The types of constraints that can be processed in each module of the query tree are discussed below.

1. Data Preparation: Most of the current algorithms for mining association rules just use a single flat file containing a list of transactions with items. In a real application, the data input for mining could be made up from several tables requiring join, grouping, select, project and other SQL operators. Types of constraints that can be applied in this module are as follows:
   - *Item Constraint*: Can be expressed as predicate of select operation. E.g., mine only beverages products.
   - *Aggregate Constraint*: Constraints using aggregate functions MIN or MAX on attributes of items. E.g., mine only items with a MIN/MAX price of $100.
   - *Length Constraint*: Transactions with number of items less than a given value can be filtered out. E.g., to mine all frequent item sets with length at least *x*, all

transactions with number of items less than *x* can be deleted. However, this constraint will need to be applied again in the next step.

- *Model-based Constraint*: For mining frequent item sets that contain certain items. E.g., frequent item sets with 'Pepsi' and 'Hot Dog'. In this case, all transactions that do not contain 'Pepsi' and 'Hot Dog' can be deleted.

2. Generate Frequent Item Sets: In the query tree, all item sets are generated before they are checked for minimum support. Although actual implementation will be different as explained earlier, the processing of constraints in this module provides the basis for placing the constraints in real algorithms. Constraints that can be applied in this module are as follows:
   - *Aggregate Constraint*: Constraints using aggregate function AVG can only be applied in this step. Prune item sets that fail the constraint.
   - *Length Constraint*: Prune all item sets of less than *x* number of items.
   - *Model-based Constraint*: Prune item sets that do not contain certain items. E.g., to mine only frequent item sets with 'Pepsi' and 'Hot Dog', remove all item sets that do not contain these.
   - *Support Constraint*: Any item set with support less than *support threshold* will not be included in frequent item sets.
3. Generate Association Rules: Constraints that cannot be applied earlier are processed in this module.
   - *Confidence Constraint*: Any rule with confidence less than *confidence threshold* will not included in the output.
   - *Item Constraint*: User may require specific items in the left hand side (LHS) or specific items in the right hand side (RHS) of the rules.
   - *Length Constraint*: User may require a specified number of items in the LHS or in the RHS.

4. ALGORITHM FOR GENERATING FREQUENT ITEM SETS

We focus on mining frequent item sets since it is computationally the most expensive module. In this section, we describe our most recent low-level algorithm for generating frequent item sets. A performance comparison of our algorithm with other similar algorithms is also presented.

4.1 Transaction Tree and ITL Data Structure

Researchers have proposed various data representation schemes for association rules mining. They can be classified as horizontal, vertical, or a combination of the two. Algorithms like Apriori and its variants use horizontal data layout, Eclat uses vertical data layout, and algorithms like FP-Growth and its extensions use a combination of vertical and horizontal data layouts.

The optimizer will choose the best algorithm based on the dataset characteristics and other important factors (index availability etc.). We have developed a generic data structure Item-Trans Link (ITL) that can be used for most algorithms. The data representation in ITL is based on the following observations: 1) Item identifiers may be mapped to a range of integers; 2) Transaction identifiers can be ignored provided the items of each transaction are linked together.

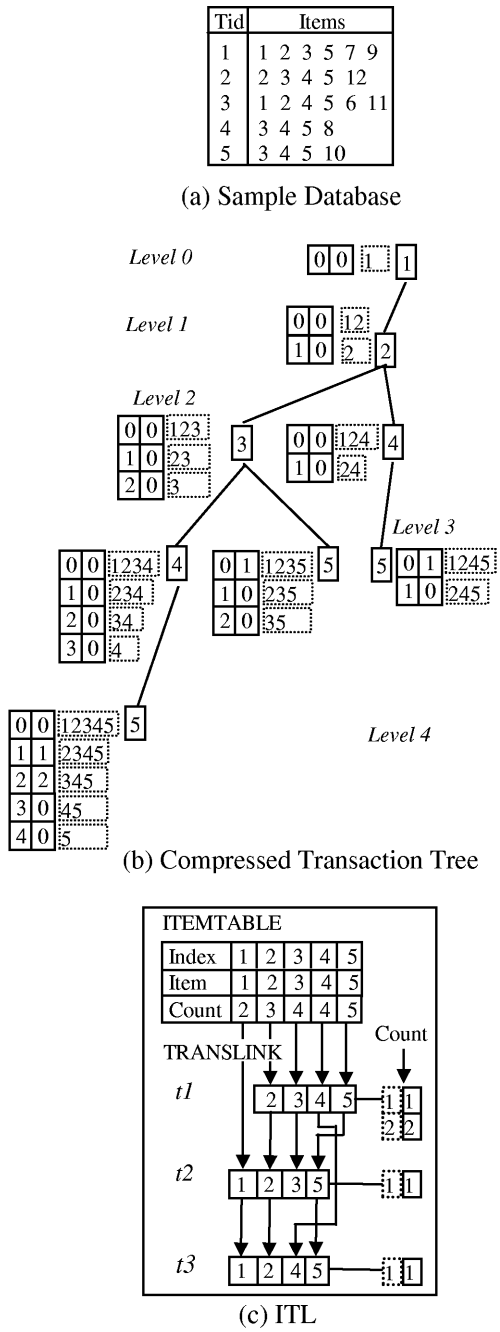


Figure 4. Compressed Transaction Tree and ITL Data Structure

ITL consists of an item table (named ItemTable) and the transactions linked to it (TransLink). ItemTable contains all individually frequent items where each item is stored with its support count and a link to the first occurrence of that item in TransLink. TransLink represents each transaction of the database that contains items of ItemTable. The items of a transaction are

arranged in sorted order and for each item, there is a link to the next occurrence of that item in another transaction.

We reduce the number of transactions that has to be traversed during mining, by replacing, a set of transactions containing identical item sets by a single transaction and a count. We found that this can be done efficiently using a modified prefix tree to identify transaction sets. We also developed a scheme to compress the prefix tree by storing information of identical subtrees together.

For example, using the sample database shown in Figure 4a, and minimum support of 2 transactions, we create a compressed transaction tree of 1-frequent items as in Figure 4b, before mapping it to ITL in Figure 4c.

Each node in the compressed transaction tree has additional entries to keep the count of transactions represented by the node. For example, the entry (0,0) at the leaf node of the leftmost branch of the tree represents the item set 12345 that incidentally does not occur in the sample database. The first 0 in the entry indicates that the item set starts at level 0 in the tree which makes 1 the first item. The second 0 indicates that no transaction in the database has this item set. Similarly, the entry (1,1) means there is one transaction with item set 2345 and (2,2) means there are two transactions with item set 345. In the implementation of the tree, the counts at each node are stored in an array so that the level for an entry is the array index that is not stored explicitly. The doted rectangles in Figure 4b, that show the item sets corresponding to the nodes in the compressed tree are not part of the data structure.

When the compressed transaction tree is mapped to ITL, information is added to each row of TransLink, indicating the count of transactions that have different subsets of items. Each node in the tree that has transaction count greater than zero is mapped to a row in TransLink, and the nonzero entries at the tree node attached to the corresponding row.

4.2 CT-ITL

We have developed several algorithms for mining frequent item sets based on ITL [6], [18]. CT-ITL is the most efficient of these algorithms. The number of transactions to be traversed during mining is reduced by grouping transactions as discussed above. The number of item traversals within transactions is minimized by using tid-count intersection. Because of transaction grouping, a tid represents a group of transactions rather than a single transaction. Therefore, tid-count lists are shorter than simple tid lists, and the intersections are performed faster.

There are four steps in CT-ITL algorithm as follows:

1. Identify the 1-frequent item sets and initialise the ItemTable: The transaction database is scanned to identify all 1-freq items, which are then recorded in ItemTable. To support the compression scheme used in the second step, all entries in the ItemTable are sorted in ascending order of their frequency and the items are mapped to new identifiers that are ascending sequence of integers. (In Figure 4c, the change of identifiers is not reflected due to our choice of sample data.)
2. Construct the compressed transaction tree: Using the output of Step 1, only 1-freq items are read from the transaction database. They are mapped to the new item identifiers and the transactions inserted into the compressed transaction tree. Each node of the tree will contain a 1-freq item and a set of counts indicating the number of transactions that contain subsets of items in the path from the root as shown in Figure 4c.
3. Construct compressed ITL: The compressed transaction tree is traversed to construct the ItemTable and TransLink.
4. Mine Frequent Item Sets: All the frequent item sets of two or more items are mined by a recursive function following the pattern growth approach and using tid-count-intersection method.

A detailed description of CT-ITL is available in [18].

4.3 Performance Study

We have compared the performance of CT-ITL with other well-known algorithms including Apriori, Eclat [19] and OP [11]. Apriori version 4.03 is generally acknowledged as the fastest Apriori implementation available. In this new version, it uses a prefix tree to store the transactions and the frequent item sets. Eclat uses the tid-intersection in its mining process. Implementations of Eclat and Apriori were downloaded from [20]. OP is currently the fastest available pattern growth algorithm.

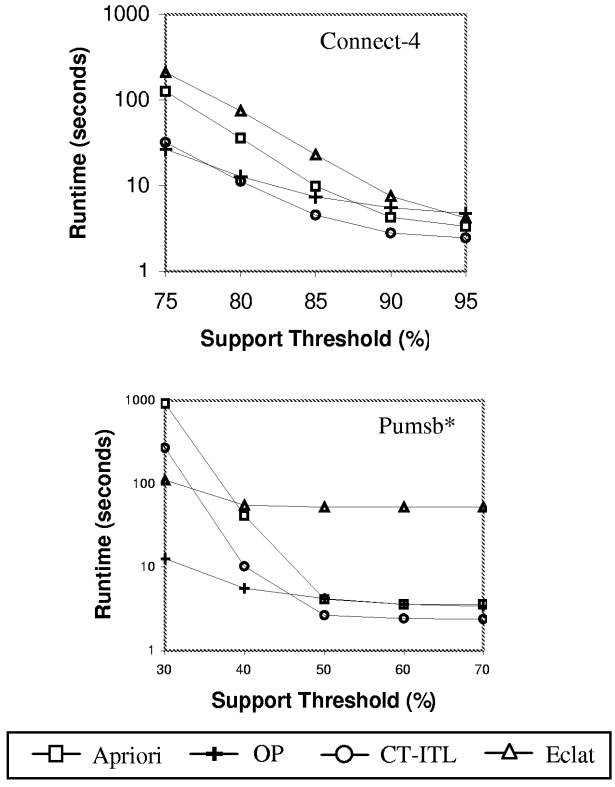


Figure 5. Performance comparisons of CT-ITL, Apriori, Eclat and OP on Connect-4 and Pumsb*

The results of our experiments are shown in Figure 5. All programs are written in Microsoft Visual C++ 6.0. All the testing was performed on an 866MHz Pentium III PC, 512 MB RAM, 30 GB HD running Microsoft Windows 2000. In this paper, the runtime includes both CPU time and I/O time.

Several datasets were used to test the performance including Mushroom, Chess, Connect-4, Pumsb* and BMS-Web-View1. We present the result of testing on Connect-4 (67,557 transactions, 129 items, 43 is the average transactions size) and Pumsb* (49,046 transactions, 2,087 items, 50 is the average transactions size), both being dense data sets that produce long patterns even at high support levels. Connect-4 was downloaded from Irvine Machine Learning Database Repository [21]. Pumsb* contains census data from PUMS (Public Use Microdata Samples). Each transaction in it represents the answers to a census questionnaire, including the age, tax-filing status, marital status, income, sex, veteran status, and location of residence of the respondent. In Pumsb* all items of 80% or more support in the original PUMS data set have been deleted. This dataset was downloaded from [22]. Performance comparisons of CT-ITL, Apriori, Eclat and OP on these datasets are shown in Figure 5. Note that we have used logarithmic scale along the *y*-axis.

CT-ITL outperforms all other algorithms on Connect-4 (support 80-95) and Pumsb* (support 50-70). It always performs better than other algorithms at higher support levels. As the support threshold gets lower and number of frequent item sets generated is significantly larger, the relative performance of OP improves. A detailed discussion of the results on these and other data sets can be found in [18].

4.4 Comparison with Other Algorithms

We compare CT-ITL with a few well-known algorithms for frequent item set mining. In the following, we highlight the significant differences between our algorithm and others:

Apriori. The Apriori algorithm combines the candidate generation and test approach with the anti-monotone or Apriori property that every subset of a frequent item set must also be a frequent item set [1]. A set of candidate item sets of length $n + 1$ is generated from item sets of length n and then each candidate item set is checked to see if it is frequent. Apriori suffers from poor performance since it has to traverse the database many times to test the support of candidate item sets. CT-ITL follows the pattern growth approach where support is counted while extending the frequent patterns, using a compact representation of the transaction database. As mentioned earlier, CT-ITL consistently performs better than Apriori.

FP-Growth. FP-Growth algorithm builds an FP-Tree based on the prefix tree concept and uses it during the whole mining process [8]. We have used the prefix tree for grouping transactions, by compressing it significantly as described in Section 4.1. We map the tree to the ITL data structure in order to reduce the cost of traversals in the mining step using tid-intersection. The cost of mapping the tree to ITL is justified by the performance gain obtained in the mining process.

FP-Growth uses item frequency descending order in building the FP-Tree. For most data sets, descending order usually creates fewer nodes compared to ascending order. In CT-ITL, a tree with fewer nodes but more paths will run slower compared to a tree with more nodes but fewer paths. From our experiments, it was found that CT-ITL is faster using ascending order than with descending order although the number of nodes is usually fewer with descending order.

Eclat. Tid-intersection used by Zaki in [19] creates a tid-list of all transactions in which an item occurs. In our algorithm, each tid in the tid-list represents a group of transactions and we need to note the count of each group. The tid and count are used together in tid-count intersection. The tid-count lists are shorter because of transaction grouping and therefore perform faster intersections.

H-Mine. In the mining of frequent item sets after constructing the ITL, our algorithm may appear similar to H-Mine [15] but there are significant differences between the two as given below.

In the ITL data structure of CT-ITL, each row is a group of transactions while in H-struct, each row represents a single transaction in the database. Grouping the transactions significantly reduces the number of rows in ITL compared to H-struct.

After the ITL data structure is constructed, it remains unchanged while mining all of the frequent patterns. In H-Mine, the pointers in the H-struct need to be continually re-adjusted during the extraction of frequent patterns and so needs additional computation.

CT-ITL uses a simple temporary table called TempList during the recursive extraction of frequent patterns. CT-ITL need to store in the TempList only the information for the current recursive call which will be deleted from the memory if the recursive call backtracks to the upper level. H-Mine builds a series of header tables linked to the H-struct and it needs to change pointers to create or re-arrange queues for each recursive call. The additional memory space and the computation required by H-mine to extract the frequent item sets from H-struct are significantly more than for CT-ITL.

OpportuneProject (OP). OP is currently the fastest available program based on pattern growth approach. OP is an adaptive algorithm that can choose between an array or a tree to represent the transactions in the memory. It can also use different methods of projecting the database while generating the frequent item sets. However, OP is essentially a combination of FP-Growth and H-Mine. H-Mine does not compress the transactions as in our algorithm and the FP-Tree is significantly larger than our compressed transaction tree. As discussed in Section 4, our compression method makes CT-ITL perform better than OP on several typical datasets and support levels.

5. CONCLUSION

In this paper, we have reported on our progress in building the components of a data mining query optimizer. A framework for the optimizer was presented in Section 2. It consists of seven stages. The research has progressed to Stage 4 of this framework, with our efforts focused on constructing the critical components.

We have developed a parser and syntax checker for the query language, as well as an extended algebra and query tree for internal representation of mining queries. Constraints are integrated into the query tree, as briefly discussed in Section 3. Further, we have developed an efficient algorithm for implementing the most critical module of the query tree. Work is in progress to integrate the processing of constraints into this algorithm.

Further work is needed to generate alternative plans by assigning algorithms to all the modules of the query tree. We also have to develop cost models for estimating the cost of different algorithms and of potential query plans.

6. ACKNOWLEDGMENTS

We are thankful to Junqiang Liu for providing us the OpportuneProject program and to Christian Borgelt for the Apriori and Eclat programs.

7. REFERENCES

[1] Agrawal, R., Imielinski, T. and Swami, A., Mining Association Rules between Sets of Items in Large Databases. In Proceedings of ACM SIGMOD, (Washington DC, 1993), ACM Press, 207-216.

[2] Agrawal, R. and Shim, K., Developing Tightly-Coupled Data Mining Applications on a Relational Database System. In Proceedings of the 2nd Int. Conf. on Knowledge Discovery in Databases and Data Mining, (Portland, Oregon, 1996).

[3] Agrawal, R. and Srikant, R., Fast Algorithms for Mining Association Rules. In Proceedings of the 20th International Conference on Very Large Data Bases, (Santiago, Chile, 1994), 487-499.

[4] Chaudhuri, S. Data Mining and Database Systems: Where is the intersection? IEEE Data Engineering Bulletin, 1998.

[5] Gopalan, R.P., Nuruddin, T. and Sucahyo, Y.G., Algebraic Specification of Association Rules Queries. In Proceedings of Fourth SPIE Data Mining Conference, (Orlando, FL, USA, 2002).

[6] Gopalan, R.P. and Sucahyo, Y.G., TreeITL-Mine: Mining Frequent Itemsets Using Pattern Growth, Tid Intersection and Prefix Tree. In Proceedings of 15th Australian Joint Conference on Artificial Intelligence, (Canberra, Australia., 2002).

[7] Han, J., Fu, Y., Wang, W., Koperski, K. and Zaiane, O., DMQL: A Data Mining Query Language for Relational Databases. In Proceedings of SIGMOD DMKD workshop, (Montreal, Canada, 1996).

[8] Han, J., Pei, J. and Yin, Y., Mining Frequent Patterns without Candidate Generation. In Proceedings of ACM SIGMOD, (Dallas, TX, 2000).

[9] Imielinski, T. and Mannila, H. A Database Perspective on Knowledge Discovery Communications of the ACM, 1996, 58-64.

[10] Imielinski, T., Virmani, A. and Abdulghani, A., DataMine: Application Programming Interface and Query Language for Database Mining. In Proceedings of the 2nd Int. Conf.on Knowledge Discovery and Data Mining, (1996).

[11] Liu, J., Pan, Y., Wang, K. and Han, J., Mining Frequent Item Sets by Opportunistic Projection. In Proceedings of ACM SIGKDD, (Edmonton, Alberta, Canada, 2002).

[12] Meo, R., Psaila, G. and Ceri, S. An Extension to SQL for Mining Association Rules. Data Mining and Knowledge Discovery, 2 (2). 195-224.

[13] Pei, J. and Han, J. Constrained Frequent Pattern Mining: A Pattern-Growth View. ACM SIGKDD Explorations, 4 (1).

[14] Pei, J., Han, J. and Lakshmanan, L.V.S., Mining Frequent Itemsets with Convertible Constraints. In Proceedings of 17th International Conference on Data Engineering, (Heidelberg, Germany, 2001).

[15] Pei, J., Han, J., Lu, H., Nishio, S., Tang, S. and Yang, D., H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. In Proceedings of IEEE ICDM, (San Jose, California, 2001).

[16] Ramesh, G., Maniatty, W.A. and Zaki, M.J., Indexing and Data Access Methods for Database Mining. In ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, (2002).

[17] Sarawagi, S., Thomas, S. and Agrawal, R., Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications. In Proceedings of ACM SIGMOD, (1998).

[18] Sucahyo, Y.G. and Gopalan, R.P. CT-ITL: Efficient Frequent Item Set Mining Using a Compressed Prefix Tree with Pattern Growth. To appear in Proceedings of 14th Australasian Database Conference, (Adelaide, 2003).

[19] Zaki, M.J. Scalable Algorithms for Association Mining. IEEE Transactions on Knowledge and Data Engineering, 12 (3). 372-390.

[20] http://fuzzy.cs.uni-magdeburg.de/~borgelt

[21] http://www.ics.uci.edu/~mlearn/MLRepository.html

[22] http://augustus.csscr. washington.edu/census/

About the authors:

Raj P. Gopalan completed a MSc in Computer Science at the National University of Singapore and a PhD in Computer Science at the University of Western Australia. He is currently a lecturer in the School of Computing at Curtin University of Technology. His research interests include, Data Mining, Data Warehousing and OLAP, and Multi-media databases.

Tariq Nuruddin completed a MSc in Mathematics and Statistics at the Louisiana Tech University. He is currently a graduate student at the School of Computing, Curtin University of Technology. His research interests include data mining, algebraic methods, and category theory.

Yudho Giri Sucahyo completed a MSc at the Faculty of Computer Science, University of Indonesia and is currently a PhD candidate at the School of Computing, Curtin University of Technology. He is supported by an AusAID scholarship. He is also a lecturer in the Faculty of Computer Science, University of Indonesia. His research interests include, Data Mining, Database Systems and Software Engineering.

How Fast is '-fast'? Performance Analysis of KDD Applications using Hardware Performance Counters on UltraSPARC-III

Adam Czezowski
CAP Research Group
Department of Computer Science
Australian National University
Canberra ACT 0200, Australia
Adam.Czezowski@anu.edu.au

Peter Christen
CAP Research Group /
ANU Data Mining Group
Department of Computer Science
Australian National University
Canberra ACT 0200, Australia
Peter.Christen@anu.edu.au

ABSTRACT

Modern processors and computer systems are designed to be efficient and achieve high performance with applications that have regular memory access patterns. For example, dense linear algebra routines can be implemented to achieve near peak performance. While such routines have traditionally formed the core of many scientific and engineering applications, commercial workloads like database and web servers, or decision support systems (data warehouses and data mining) are one of the fastest growing market segments on high-performance computing platforms. Many of these commercial applications are characterised by more complex codes and irregular memory access patterns, which often result in a decrease of performance that is achieved. Due to their complexity and the lack of source code, performance analysis of commercial applications is not an easy task. Hardware performance counters allow detailed analysis of program behaviour, like number of instructions of various types, memory and cache access, hit and miss rates, or branch mispredictions. In this paper we describe experiments and present results conducted with various KDD applications on an UltraSPARC-III platform, and we compare these applications with an optimised dense matrix-matrix multiplication. We focus on compiler optimisations using the `-fast` flag and discuss differences in un-optimised and optimised codes.

Keywords

Data mining, performance analysis, compiler optimisation, UltraSPARC-III, C4.5, Apriori.

1. INTRODUCTION

Commercial applications like database and web servers, or decision support systems (data warehouses and data mining) represent one of the most rapidly growing segments in the high-performance computing market. Modern processors and memory systems are designed to be efficient and achieve high performance with applications that have regular memory access patterns (like dense linear algebra software). They often perform purely when running commercial applications. Complex codes, irregular memory access patterns, dynamic data structures and dynamic memory allocation schemes characterise such applications. This paper aims to analyse the characteristics of such commercial applications, especially to give an insight into their cache and memory access behaviour.

A rapidly growing segment of commercial applications is data mining or KDD (Knowledge Discovery in Databases) [14], which deals with the analysis of large and complex data sets. KDD combines techniques from machine learning, statistics, databases and high-performance computing. Tasks involved are data cleaning and pre-processing, data exploration, clustering, predictive modelling, association rules generation, decision tree induction, and others. Three common characteristics of these applications are (1) they operate on large data sets, (2) they are compute and memory intensive, and (3) they involve irregular memory access patterns which is due to their dynamic and often recursive data structures (like hash tables and trees, index or linked lists). The first two characteristics make them attractive for implementation on high-performance platforms, specially for shared memory multiprocessors (SMPs), where all CPUs have access to the same memory system, while the last characteristic is an obstacle for efficient system utilisation and high performance. Traditional compiler optimisation techniques (based on arrays and data locality), which proved to be successful for many scientific and engineering applications, can only be applied with limited success to such data structures.

Hardware performance counters are an easy to use instrument and they can provide detailed analysis of application performance behaviour at instruction level. Such counters are available on most modern microprocessors, including *UltraSPARC*, *Pentium* and *Alpha*. They can count various events, including different types of instructions (loads, stores, branches or floating-point operations), cache hits and misses, TLB[1] misses, branch mispredictions, cycles and instructions completed, and others. Machine and operating-system dependent libraries (like the *Solaris* `libcpc` [12]) provide access to hardware counters on a specific platform and operating system. Platform independent counter libraries are currently under development in various research

[1]Translation Lookaside Buffer

projects. Two such libraries are the *Performance Counter Library (PCL)* [4] and the *Performance Application Programming Interface (PAPI)* [7]. Their aim is to provide a set of platform independent counters, that allow easy portability of programs instrumented with these libraries, and to allow inter-platform performance comparisons. As we currently restrict our research to a *Solaris/UltraSPARC* platform, we use the `libcpc` [12]. Although we have made initial experiments with *PAPI* we have turned to the `libcpc` as it was the only library available to support *UltraSPARC-III* counters at the time.

In the next section we present the four applications and the data sets chosen for our experiments, and in Section 3 we discuss our experimental setup, the *Solaris/UltraSPARC* platform used and give an example of how source code can be instrumented with calls to hardware performance counter libraries. Results are then presented in Section 4 and conclusions are given in Section 5. Related work in the area of performance analysis of KDD and other commercial applications is presented in Section 6, and finally we give an outlook on future plans in Section 7.

2. APPLICATIONS AND DATA SETS

We choose three KDD applications and one vendor optimised linear algebra code for hardware counter performance analysis using the `libcpc` [12] library on a *Solaris/UltraSPARC* platform. We only counted events in the core computation routines, i.e. without file in- or output. The KDD programs we analysed use mainly input from text files (which is generally slow). Commercial versions of such programs would either read data from binary files or access them directly from a database server. We now present the analysed programs and subsequently discuss the data set we used.

2.1 Decision Tree Induction – C4.5

The freely available popular decision tree induction program *C4.5*[2] [19] was chosen as a typical KDD application. This program has already been analysed in its memory access behaviour using a machine simulator [5; 6].

C4.5 is written in ANSI C, it reads data from text files and then builds a decision tree. Only the tree building routine (i.e. the function `BestTree()`) was analysed, with the complete primary data set (the table loaded from disk) stored in main memory. This data is stored in an array with pointers to vectors, with each vector (one data record) having a length corresponding to the number of attributes. Every element in this vector consists of a `short` and a `float` variable. In the case of a categorical type attribute, the category number is stored as a `short`, while a continuous attribute (a real number) is stored as a `float`. Thus one of the two variables is always unused. The decision tree is a complex recursive data structure that is built dynamically in the tree building routine.

2.2 Association Rule Induction – APRIORI

Mining association rules is a popular data mining algorithm. It is for example used to analyse market basket data to find frequent item sets and extract rules like *'if a customer buys milk then she will most likely also buy cheese.'* For our performance analysis we use a freely available implementation of the *APRIORI*[3] algorithm [2]. An older version of this program is incorporated in the data mining tool *Clementine 5.0*. The program is written in C, it reads a text file with transactional data and writes the resulting rules either into a text output file or displays them. The items in the input transactions are stored in a vector data structure as integer numbers. Once all data is loaded, the items are sorted with descending frequency, and a prefix tree is built, which is then modified and updated for item sets of increasing length. Besides pointers to parent and (variable number of) child nodes, the prefix tree contains a counter vector, stored as integer numbers. Once all frequent item sets are found, they are sorted and the extracted rules are displayed or saved. The code analysed includes the sorting and recoding of the items, the creation of the item set tree, the checking of the subset size, and finally the sorting of the transactions (in order to find the frequent item sets).

2.3 Additive Models – ADDFIT

The *ADDFIT* algorithm [9] was developed by the ANU Data Mining group and implemented sequentially and on distributed memory platforms (using C/MPI) [10]. This algorithm builds an additive model of the data by assembling a dense symmetric linear system in a first step, which is then solved in a second step using either a sequential or parallel solver [10; 21]. For the performance analysis we are only interested in the first step, as it involves reading the data from (binary) files once and assembling each data record into a matrix and vector at data dependent locations. The primary data structure (the table from disk) is loaded first and then the assembly is started. Only the assembly routine is analysed using hardware performance counters. The data dependent assembly results in irregular memory access patterns. For each continuous attribute in a data record four non-zero values are added into the matrix, while for a categorical attribute only one value is added. The locations where these values are added is data dependent and can be anywhere in the matrix. As the assembled linear system is symmetric, only a dense upper triangular matrix with a corresponding vector is allocated, whereby each entry is a `double` sized floating-point value. The size of this linear system is determined by the number of categories for categorical attributes, and the resolution of the model for continuous attributes, but it is completely independent from the size (i.e. number of records) in the primary input data set.

2.4 Dense Matrix-Matrix Multiplication – BLAS (SUNPERF)

To allow a comparison with a platform optimised application with regular memory access patterns we also instrumented a dense matrix-matrix multiplication (the *BLAS* routine `dgemm()` as implemented in Sun's *SUNPERF* library) with calls to the `libcpc` hardware counter library. The `dgemm()` routine is typically used in the core of various scientific and engineering applications. For the performance analysis two *Hilbert* matrices were created and dense matrix-matrix multiplications of two such matrices where performed and analysed.

[2] `http://www.cse.unsw.edu.au/~quinlan/`

[3] `http://fuzzy.cs.uni-magdeburg.de/~borgelt/`

Characteristic	Level-1 Instruction	Level-1 Data	Level-2 Unified	D-TLB	I-TLB
Size	32 KB	64 KB	8192 KB	512x8 KB	128x8 KB
Associativity	4-way	4-way	1-way	2-way	2-way
Line length	32 Bytes	32 Bytes	512 Bytes	–	–
Latency (cycles)	2	2	15	–	–
Miss cost (cycles)	15	15	75	–	–
Write policy	Write-invalidate	Write-through	Write-back	–	–

Table 1: UltraSPARC-III cache, D-TLB and I-TLB characteristics.

2.5 Data Sets and Data Structures

For *C4.5* and *ADDFIT* we used the *Census-Income* data set which is freely available from the *UCI KDD Archive*[4]. This data consists of a training file which contains 199,523 records and a test set with 99,762 records. For our purpose we concatenated both files into one to get a large enough test data. The *Census-Income* data set contains 5 continuous and 37 categorical attributes.

For *APRIORI* we created synthetic data sets of various size and complexity using a data set generator as described in [2]. For the tests we then choose a smaller data set with 10,000 records and a larger one with one million records.

The *primary* data structures used by the KDD applications hold the input data. They are mostly of arrays or vectors, and their size and dimension usually increases linearly with the size of the input data set. In the case of *ADDFIT*, this data is only used once (i.e. each data record is accessed once), but for *C4.5* and *APRIORI* usually several iterations are needed each accessing the primary data structure.

The size of the *secondary* data structures built by the KDD applications, i.e. the decision tree in *C4.5*, the prefix tree in *APRIORI*, and the dense matrix used by *ADDFIT*, are not directly proportional to the size of the input data. Rather, they are data dependent (e.g. a *C4.5* decision tree) or their size is determined by some parameters (e.g. model resolution in *ADDFIT*). The size of the prefix tree for *APRIORI* depends both on the data as well as on parameters like *support* and *confidence*, which have to be set by the user. It is therefore often very hard to specify the amount of memory some KDD applications will use.

3. EXPERIMENTAL SETUP AND PROCEDURES

A common way to obtain detailed, performance related data at the level of architectural units, i.e. at the level of cache, memory, integer and floating point units, is to use full machine simulators. Although indispensable to evaluate new alternative architectural designs, full machine simulators are slow and often provide a simplified view of a real architecture [11]. Another, more convenient way to gather such data is to use specialised registers called *hardware performance counters* or simply *hardware counters*. Today, nearly all general purpose microprocessors on the market have such counters and provide user level interfaces via specialised libraries. In the next few sections we describe our hardware platform and give some more insight into how to use hardware counters.

3.1 Hardware Platform

The presented experiments throughout this report were conducted on a *Sun-Blade-1000* workstation with two *UltraSPARC-III* (US-III) processors running at 750 MHz, and having in total 2 GBytes of main memory. The summary of characteristics for both Level-1 Data and Instruction caches as well as the unified Level-2 cache are presented in Table 1. Cache latencies where established experimentally using the *Calibrator*[5] tool and averaged as their exact value depends on the SDRAM model [22], prefetch cache operation and the instruction mix. We do not provide TLB latencies in Table 1 as in the US-III processor the D-TLB is an integral part of the data cache unit (DCU) and the I-TLB and the instruction cache are part of the instruction issue unit (IIU). Thus, both TLBs latencies are hidden by Level-1 cache latencies. Because of the complexity of *software-managed* TLBs only a range of the miss costs can be given. It lies somewhere between one hundred and a few hundred cycles. At this stage we will not go into the analysis of the *UltraSPARC-III* prefetch and write caches behaviour. However, we acknowledge that such an analysis will assist in formulating a more accurate representation of the quite sophisticated *UltraSPARC-III* on-chip memory system.

The test workstation is running *SunOS Version 5.8* with *Forte Developer 7 Compiler* and *Development Tools Collection*. We have chosen a *Sun-Blade-1000* workstation as test platform because of its modern *UltraSPARC-III* microprocessor and wide range of events that can be monitored using performance counters. In the future we would like to undertake similar study on *Intel Pentium* and other processors.

3.2 Performance Counters

All run time measurements of the various hardware events were obtained by using *UltraSPARC-III* hardware performance counters. Two specialised processor registers are used for this purpose. The *Performance Control Register (PCR)* controls which event and which mode (user, system or both) of operation is selected for a pair of 32-bit *Performance Instrumentation Counters (PICs)*. There are over seventy possible events that can be counted, logically gathered into the following groups: instruction execution rates, integer unit statistics and stall counts, pipeline R-stage stall counts, recirculate counts, memory access statistics, system interface, floating point operation and memory controller statistics. To name just a few typical events: both Level-1 and Level-2 cache references and misses, D-TLB (Data Translation Lookaside Buffer) and I-TLB (Instruction Translation Lookaside Buffer) misses, pipeline stalls, branch mispredictions, floating point addition and multiplication pipe com-

[4] `http://kdd.ics.uci.edu/`

[5] `http://www.cwi.nl/~manegold/Calibrator/`

```
/*
  The example code measures instruction and cycle counts while transposing a matrix.

  Compilation and execution:

  sun% cc -xO2 -o simple_cpc simple_cpc.c -lcpc
  sun% simple_cpc
  pic0=Cycle_cnt,pic1=Instr_cnt,sys,nouser: 6114336263, 3157421639 cpi=1.94
  sun%
*/

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <inttypes.h>
#include <libcpc.h>

void transpose512(){
    static int  x[512][512];
    static int  y[512][512];

    ...
}

void main() {
    char*               eventnames = "pic0=Cycle_cnt,pic1=Instr_cnt,sys,nouser";
    int                 cpuver;
    cpc_event_t         event, before, after;
    unsigned long long  pic0, pic1;

    /* Get processor version and check if hardware counters are available */
    if ((cpuver = cpc_getcpuver()) == -1) exit(EXIT_FAILURE);

    /* Translate event string into data structure */
    if (cpc_strtoevent(cpuver, eventnames, &event) != 0) exit(EXIT_FAILURE);

    /* Bind events to process */
    if (cpc_bind_event(&event, 0) == -1) exit(EXIT_FAILURE);

    (void) cpc_count_usr_events(1);  /* Enable user mode counting */
    (void) cpc_count_sys_events(1);  /* Enable system mode counting */

    /* Sample counters to get start values */
    if (cpc_take_sample(&before) == -1) exit(EXIT_FAILURE);

    transpose512();  /* Do some computations... */

    /* Sample counters to get stop values */
    if (cpc_take_sample(&after) == -1) exit(EXIT_FAILURE);

    (void) cpc_count_usr_events(0);  /* Disable user mode counting */
    (void) cpc_count_sys_events(0);  /* Disable system mode counting */

    pic0 = after.ce_pic[0] - before.ce_pic[0];  /* Get user count value */
    pic1 = after.ce_pic[1] - before.ce_pic[1];  /* Get system count value */

    printf("%s: %lld, %lld cpi=%f\n", eventnames, pic0, pic1,(float)pic0/(float)pic1);
}
```

Figure 1: Simplified example code augmented with calls to Sun Performance Library `libcpc`.

Option	Comment
-fns	Disable gradual underflow for improved performance.
-fsimple=2	Enable non-IEEE 754 standard, aggressive floating-point operations that may lead to different numeric results.
-fsingle	Use single-precision arithmetic.
-ftrap=%none	Disable the floating-point traps.
-xalias_level=basic	Pointers to different C basic data types do not alias each other. References using char pointers may alias any other type.
-xarch=v8plusb	Generate 32-bit subset of SPARC-V9 ISA including the VIS 1.0 and US III extensions. Runs only on US III.
-xbuiltin=%all	Inline or substitute intrinsic functions for system functions.
-xcache=64/32/4:8192/512/1	Definition of cache properties passed to the compiler.
-xchip=ultra3	Specifies target processor to ultra3.
-xdepend	Loop dependency analysis and optimisation.
-xlibmil	Inline selected libm math routines for optimization.
-xmemalign=8s	Assume 8-byte data alignment.
-xO5	Set optimisation to the highest possible level.
-xprefetch=auto,explicit	Enable automatic generation of prefetch instructions.
-xvector=no	Disable vectorised mathematical library functions.
-depend	Perform dependency analysis to optimise loops.

Table 2: Expansion of `-fast` macro for *Forte Developer 7 C 5.4 Sun* compiler release.

pletions and many others. Events can be counted in user and/or system mode.

Access to these counters is obtained via function calls to the `libcpc` and `libpctx` [12] libraries which have to be embedded into the application's source code. The main feature of these libraries is the very low overhead. For example the counter start and stop calls have overheads of less than 3,000 cycles as measured on a *Sun-Blade-1000*. In parallel with performance counter libraries, *SunOS 5.8* provides two monitoring tools: *cputrack* and *cpustat* that display counter statistics. While *cputrack* gathers such statistics for a particular process, *cpustat* provides system-wide statistics hence requires super user privileges.

An example of a simple code which was augmented by calls to the performance counter library is presented in Figure 1. In this code there are the following function calls to the `libcpc` library: `cpc_getcpuver()` is used to determine if the processor provides any performance counters. It returns an abstract description of the processor which is used by every other function in `libcpc`. The function `cpc_strtoevent()` translates the event specification from a string representation into the appropriate set of control bits for the processor and stores them in a `cpc_event_t` structure. Calls to `cpc_bind_event()` bind the selected two events to the current *light-weight process (LWP)*. The counters can then be sampled at any time by using `cpc_take_sample()` and reading the `ce_pic[]` fields of the `cpc_event_t` structure.

3.3 Methods

Each experiment was performed on a dedicated CPU wholly reserved to the test application. This isolation of the code being measured from the activity of the other processes being run on the system ensured good reproducibility of the results particularly in the system mode.

The `-fast` option in the *C Forte Developer 7 Compiler Suite* is most likely to be used by many programmers as a starting point for compiler optimisations. We wanted to see and measure the effects of this option on the four applications to be evaluated. This option is in fact a macro which combines many different optimisations that benefit in a wide range of programs [12]. We can not explain in detail every option used but we have expanded the `-fast` macro in Table 2 with comments. It is worth to note that the content of the `-fast` macro depends on the release of the *Sun* compiler. The one shown here is for the *Forte Developer 7 C 5.4* release. In the un-optimised or default mode the compiler assumed a generic *UltraSPARC* architecture and 4-byte data alignment.

We need to stress that our goal is not to perform optimisations of the selected applications, merely to comment on their performance in two situations, i.e. when compiled with and without the `-fast` optimisation flag.

All tests were run over weekends or night times when the machine was otherwise idle. Table 3 shows the characteristics of the test programs and data sets we used. For the three KDD applications we choose a smaller and a larger data set each that resulted in a heap size (i.e. size of the data structure in main memory) between 4 and 19 MBytes and between 62 and 100 MBytes, respectively. For our comparison application *BLAS* we added a small problem size which resulted in a 1 MByte heap size. All results for a given program/data set pair in Table 3 are averaged over the number of iterations listed, and both run times and user code percentages are given for the codes compiled without optimisation.

4. RESULTS

We performed experiments with the four selected applications *ADDFIT*, *APRIORI*, *C4.5* and *BLAS* both with the `-fast` optimisation turned on and off, and we report them here as *optimised* and *un-optimised*. As can be seen from Table 3, all of the applications – with the exception of *C4.5* with the large data set – do spend more than 90% of their time in user space. We therefore only report results in user space. The measurements in the system (or kernel) space were performed and will be reported in the near future. The

Program	BLAS (SUNPERF)			ADDFIT	
	small	medium	large	small	large
Data	209 × 209 matrices	660 × 660 matrices	2090 × 2090 matrices	Census with 10,4858 records	Census with 209,715 records
Run time	0.03 sec	1.10 sec	44.03 sec	1.09 sec	5.89 sec
Iterations	100	10	1	10	10
Heap size	1 MB	10 MB	100 MB	10,024 KB	90,408 KB
User code	99.46%	97.09%	93.03%	99.64%	96.36%

Program	APRIORI		C4.5	
	small	large	small	large
Data	T5I4D10K with 10,000 records	T10I8D1000K with 1,000,000 records	Census with 8,322 records	Census with 266,305 records
Run time	3.36 sec	31.78 sec	2.35 sec	421.04 sec
Iterations	10	1	5	1
Heap size	19,776 KB	70,512 KB	3,960 KB	62,152 KB
User code	89.37%	94.30%	98.43%	75.93%

Table 3: Program and test characteristics.

Program		BLAS (SUNPERF)			ADDFIT		APRIORI		C4.5	
		small	medium	large	small	large	small	large	small	large
Loads	Un-optimised	23.1	23.1	23.2	38.2	38.7	28.1	27.0	34.5	33.7
	Optimised	23.1	23.1	23.2	30.2	31.9	21.5	21.3	30.0	24.0
	Change	**0**	**0**	**0**	**-21**	**-18**	**-24**	**-21**	**-13**	**-29**
Stores	Un-optimised	1.3	1.0	1.0	12.5	13.4	9.5	14.0	8.4	8.5
	Optimised	1.3	1.0	1.0	11.2	11.6	6.7	9.3	9.7	5.1
	Change	**0**	**0**	**0**	**-11**	**-13**	**-29**	**-34**	**+16**	**-39**
Branches	Un-optimised	4.2	4.1	4.1	6.7	5.6	15.2	13.1	7.2	10.2
	Optimised	4.2	4.1	4.1	9.8	8.4	19.7	22.8	11.6	20.7
	Change	**0**	**0**	**0**	**+46**	**+49**	**+30**	**+74**	**+60**	**+103**
FP ops	Un-optimised	58.7	59.1	59.1	3.5	3.8	0.02	0.0	8.4	7.7
	Optimised	58.7	59.1	59.1	5.6	7.7	0.04	0.0	14.7	16.1
	Change	**0**	**0**	**0**	**+62**	**+104**	**+93**	**+91**	**+76**	**+110**
Others	Un-optimised	12.7	12.7	12.6	39.1	38.5	47.2	45.9	41.5	39.9
	Optimised	12.7	12.7	12.6	43.2	40.4	52.1	46.6	34.0	34.1
	Change	**0**	**0**	**0**	**+10**	**+5**	**+10**	**+2**	**-18**	**-15**

Table 4: Percentile values of loads, stores, branches, floating point operations and other instructions present in optimised and unoptimised codes. The relative change of instruction mix between optimised and unoptimised codes, also expressed in percentages, is given in bold fonts.

use of the `-fast` optimisation moderately increased the percentage of time spent in system space.

4.1 Instruction Mix

Ideally, an execution time breakdown into computation, memory stalls, branch misprediction and resource stalls would be desirable. Due to interdependencies in the measured hardware events, a reliable and meaningful breakdown into these components is impossible using the data we have collected (even assuming average latencies per counted event would be grossly inaccurate). However, we were able to count the number of different instructions types in an application, and Table 4 shows the percentage values of loads, stores, branches, floating-point operations and all other (mainly integer operations) instructions in user mode. The numbers are given for both the un-optimised and the optimised compilations, and the corresponding percentage change (positive means an increase, negative a decrease).

No changes can be seen for the *BLAS* test runs because they are basically a simple call to a *SUNPERF* library routine (which is part of the operating system) and thus not affected by compilation options. For the three KDD applications we can clearly see that the ratio of loads and stores is generally reduced (one exception is *C4.5* with the small data set which has a noticeable increase in store instructions), while the percentages of all other instruction types are mostly increased. This can be explained due to compiler optimisations, like using register variables for loop counters and other frequently used variables, which reduces the number of loads and stores to access these variables.

The three main differences between the *BLAS* application and the KDD applications is the larger percentage of stores in KDD applications, their very small number of floating-point operations compared to the dense matrix-matrix multiplication, and finally the much larger percentage of all other instructions, which are mainly integer computations,

Program	BLAS (SUNPERF)			ADDFIT		APRIORI		C4.5	
	small	medium	large	small	large	small	large	small	large
Un-optimised	30.15	9.39	2.44	9.05	15.56	6.45	2.29	1.67	0.19
Optimised	30.13	9.38	2.44	19.03	35.09	10.26	3.76	2.51	0.22
Improvement	0%	0%	0%	53%	56%	37%	39%	33%	14%

Table 5: Ratio of allocated memory per execution time (MB/sec).

for KDD programs. This larger amount of input and output results in a higher load on the cache and memory system, but also in almost unused floating-point units. One possible improvement for KDD applications might be a re-design of algorithms to increase their use of floating-point operations.

4.2 Memory Allocation

As the memory imprint has a significant impact on application performance we have tested each of the applications with different data sets as indicated in Section 2.5.

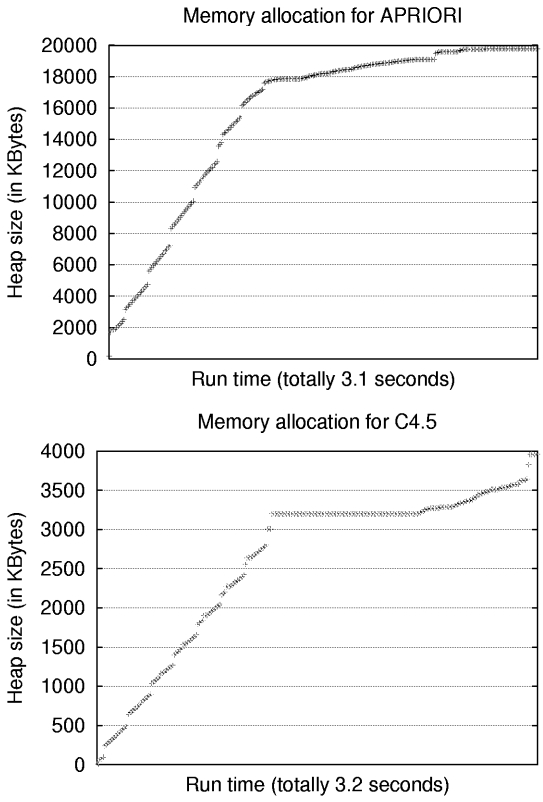


Figure 2: Dynamic memory allocation for APRIORI and C4.5

The maximal allocated memory for data (the heap size) is listed in Table 3. While *BLAS* and *ADDFIT* allocate all the memory they need (basically the dense matrices plus some buffers for input and temporary data) at the beginning, both *APRIORI* and *C4.5* dynamically allocate smaller memory blocks at run time. Figure 2 shows the heap sizes as measured with the Unix command `pmap` (and a small Python script filtering the output) with the smaller data sets for both *C4.5* and *APRIORI*. Two phases are clearly distinguishable, the first being the loading of the input data (steeper slope of the graphs), while the second being the computation of the decision tree or the frequent item sets (prefix tree), respectively. Such dynamic memory allocation and re-allocation results in many system calls and also prevents good data locality.

It is interesting to see the ratio of number of megabytes allocated by each application per execution time measured in seconds, as displayed in Table 5. This indicates the overall capacity of a particular application to process the given amount of data. Of course this "megabyte throughput" depends on the complexity of the operations performed on a given data set but it gives an idea on relative performance of the tested applications.

The improvement factors between the un-optimised and optimised codes are similar to those presented in Table 6, which were based on execution times only. We see rather poor performance of *C4.5* and most of the applications with larger data sets. The only exception from this is *ADDFIT*. For the large data set it achieves 35 MB/sec "megabyte throughput", larger than for the smaller data set which was 19 MB/sec. This is explained by the fact that the size of the linear system being assembled by *ADDFIT* is determined by the number of categories for categorical and the resolution of the model for continuous attributes. Since these attributes didn't change as we went from the small to the large data set the overall throughput has increased.

4.3 Overall Performance

The reduction in run times between un-optimised compilation and using the `-fast` compiler optimisation is listed in Table 6. As can be seen all KDD applications gain between around 10% and up to 50% with *ADDFIT*'s run time almost reduced to half. Optimised compilation does not affect the run time of the *BLAS* dense linear algebra code, because the library used for this application is already compiled and is unaffected by compiler optimisations.

MIPS and MFLOPS are popular measures to show the overall performance mainly for scientific and engineering codes. In Figure 3 one can clearly see that only the *BLAS* dense matrix-matrix multiplication is actually dominated by floating-point operations. The three KDD programs perform mainly integer and other operations (which is consistent with the results from Table 4), which is what one can expect from applications working on strings and integer numbers. It is also interesting to see that for all four applications the MIPS numbers decrease with larger data sizes (except *BLAS* which has a peak performance with a medium sized matrix). For all tested applications, most of the time is spent in user mode, with around 90% (and more) in most cases. The only exception is *C4.5* with the large data set, in which case the user proportion is reduced to around three quarters of the total run time (see Table 3).

Program	BLAS (SUNPERF)			ADDFIT		APRIORI		C4.5	
	small	medium	large	small	large	small	large	small	large
Un-optimised	0.03	1.10	44.03	1.09	5.89	3.36	31.78	2.35	421.04
Optimised	0.03	1.10	44.03	0.52	2.79	2.23	19.93	1.56	375.50
Improvement	0%	0%	0%	52%	53%	34%	37%	34%	11%

Table 6: Run times (in seconds).

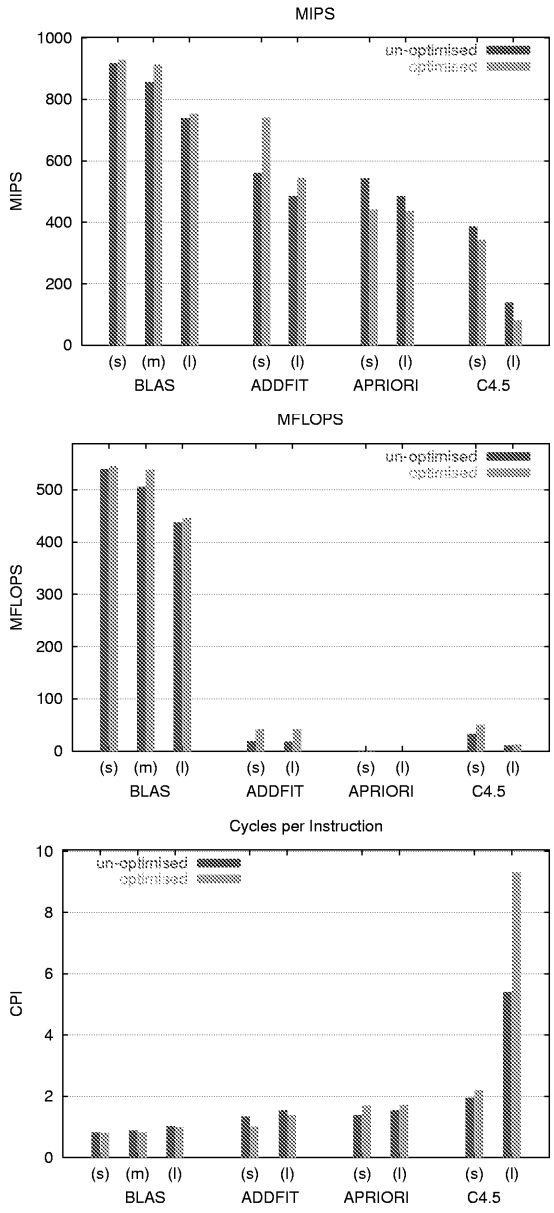


Figure 3: Overall performance indicators: MIPS, MFLOPS and CPI.

While compiler optimisations clearly help in reducing the run times of applications (the most important performance aspect from a user's point of view), for some applications this results in actually a poorer performance. The MIPS rate for both *APRIORI* and *C4.5* is smaller for the optimised code, and their CPI (cycles per instruction) rate is higher. This means that even though the program runs faster, the system utilisation is less and more time is wasted on resource stalls. The question now is where? Because the reduction in run time is larger than the reduction in e.g. load, store or branch instructions, the "density" of these instructions is higher (i.e. the ratio of loads, stores and branches per instruction is higher). This results in higher miss and stall rates in most cases, as can be seen in the figures in the following sections.

4.4 Cache Performance

The cache concept makes sense for applications which exhibit temporal and spatial locality. What about applications with irregular memory access patterns where traditional cache optimisation techniques simply will not work? In this section we describe performance of all four test applications at both the Level-1 and the Level-2 cache as well as the TLB.

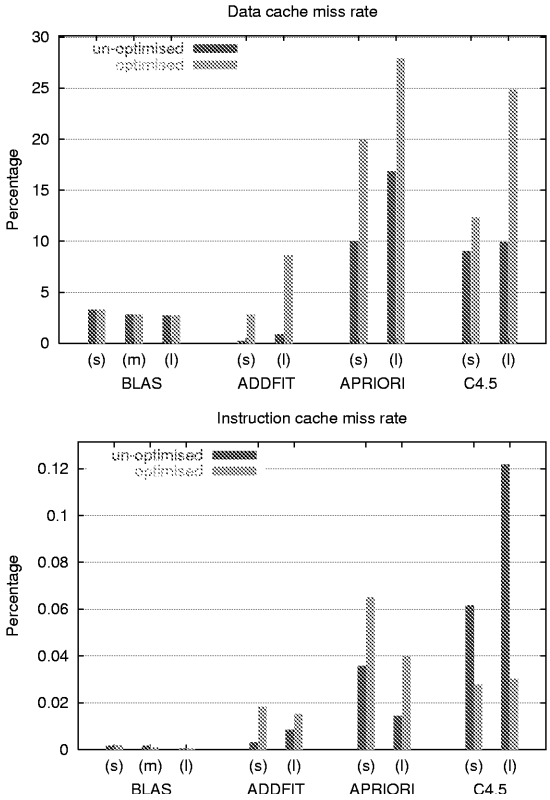


Figure 4: Data and instruction cache miss rates.

4.4.1 Level-1

Figure 4 shows Level-1 cache performance for the data and instruction cache. *BLAS* and *ADDFIT* exhibit small Level-1 cache misses of 3% and around 1% for the small data sets. Both *APRIORI* and *C4.5* have cache misses of 10% and more for the un-optimised versions. This behaviour is typical for irregular memory access patterns. Although, after optimisations, both applications' execution times were reduced the Level-1 cache misses were significantly increased. We are not surprised by good cache performance of *BLAS* for which the `dgemm()` matrix-matrix multiplication routine provided by the *SUNPERF* library is "cache friendly". However, it is unexpected that *ADDFIT* exhibits such good cache utilisation for un-optimised runs. The plausible explanation is that during the assembly stage (which performance we are measuring) the new data added to the dense symmetric matrix mostly are no further than a Level-1 cache size apart so the cache updates are rare. The instruction cache miss ratios are very small which reflect the fact of relative small number of branches and a lack of procedure calls. The increase in instruction cache miss ratios for *APRIORI* and *C4.5* is well correlated with their higher percentage of branch instructions as seen in Table 4.

We know that the main improvement in execution times after the introduction of the `-fast` compiler flag comes from the significant reduction in number of instructions. For *ADDFIT* this amounted to 39% reduction for the small data size (51% for large), for *APRIORI* 48% (46%) and for *C4.5* 42% (52%). On the same Figure 4 we see a dramatic increase in Level-1 cache misses for all applications except *BLAS* for which `-fast` option had no effect. Aggressive optimisations, and as such we should regard those introduced by using the `-fast` option, will lead to complete alteration of application characteristics such as data and instruction flow. It will be incorrect to try to relate new memory performance metrics to the old ones as the optimised (read new) application have nothing to do with the un-optimised (read old) except that both produce the same results.

Having said this, we still should try to understand why optimisations bring such drastic cache performance penalty. This is particularly visible in *ADDFIT* where Level-1 data cache miss ratio rose 8.5 fold for the small and 7.3 fold for the large data set. For *ADDFIT* we see particularly high number of counts for the *instruction queue being empty due to a refetch of a second branch within a fetch group* type of event and also very high number of counts for the *stalls in the store queue due to the store instruction being first in the group*. Although significant for overall performance, these type of events can not cause such Level-1 data cache miss increase. In order to bring light into this we had to resort to the inspection of the disassembled codes of both the un-optimised and optimised *ADDFIT*. Despite of the overwhelming complexity of the code a large number of consecutive double loads greater than the cache line size are prevailing in the optimised code and they are most likely to cause such high incidents of Level-1 cache misses. At this point immediately another question can be raised – why does the compiler introduce such *inefficient* loads distribution? We can only presume that the optimisations based on data flow analysis combined with interprocedural analysis yield higher performance gains than some "on the way inefficiencies" introduced by the compiler.

4.4.2 Level-2

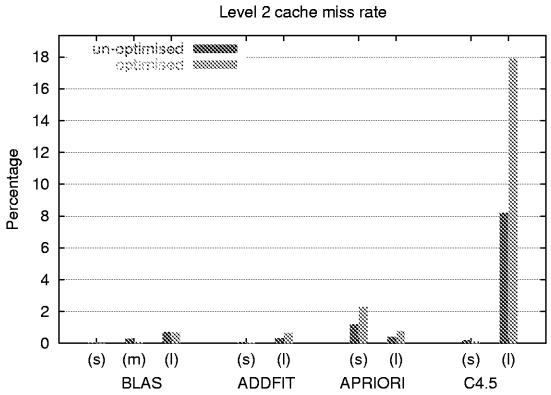


Figure 5: Level-2 cache miss rates.

The results for Level-2 (external) unified cache are presented in Figure 5. The Level-2 cache is large (8 MB), hence the number of misses is relatively low for most of the applications except *C4.5* with the large data set for which the miss rate is 8% for the un-optimised code and extraordinarily high 18% for the optimised version. Inspection of D-TLB misses in Figure 6 also reveal very high miss ratio for *C4.5* with the large data set. It is unlikely that such high miss ratio will be caused by inefficient instruction placement. Most likely this has to be due to the sorting of entire categorical attributes (using recursive quicksort) in *C4.5*, which results in almost no locality for data access. The problem propagates throughout the whole memory hierarchy. In this case we have observed high rate of stalls to memory banks which can possibly be reduced by exploring an alternative sorting method to deal with particularly large data sets.

One of the possible solution for large data sets will be to resign from quicksort and try another specifically designed sorting algorithm, or alternatively use quicksort on a smaller subsets of the categorical attributes. The guide to possible success of such an approach is the performance of *C4.5* for the small data set.

4.4.3 D-TLB

Overall D-TLB (Figure 6) and I-TLB (not shown) miss rates are very small. A problematic D-TLB miss rate only appears with *C4.5* executed with the large data set. Apart from earlier proposed remedy we could try to increase the page size from 8KB to 4MB and see how this single approach will work.

4.4.4 Data Stall

One of the more indicative metrics is data stall rate. It is a measure of the fraction of a CPU cycle which was wasted in waiting for data. This wait fraction is measured for both arriving and departing data from the CPU core. Most often the high percentages in this measure indicate cache misses or TLB misses but may also depend on instruction grouping and scheduling [12].

In Figure 7 we present data stall rates for all four test applications. Both *APRIORI* and *C4.5* show that over 50% of the CPU cycles were used for data coming or departing from the CPU core. This fact well correlates with the high Level-1 data cache misses for both applications.

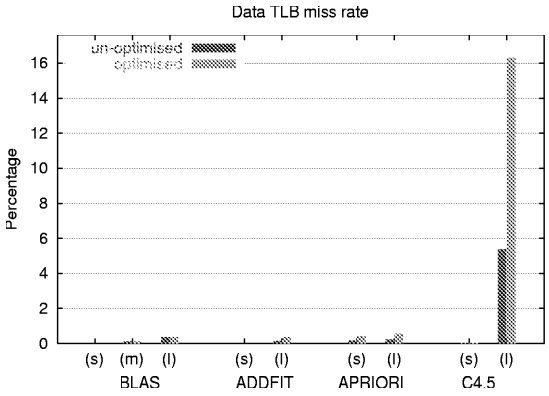


Figure 6: Data TLB miss rates.

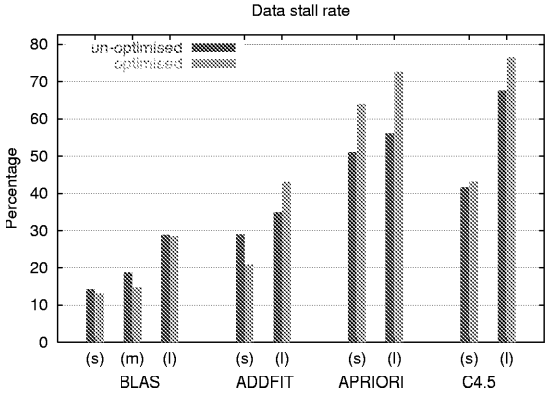


Figure 7: Data Stall rates.

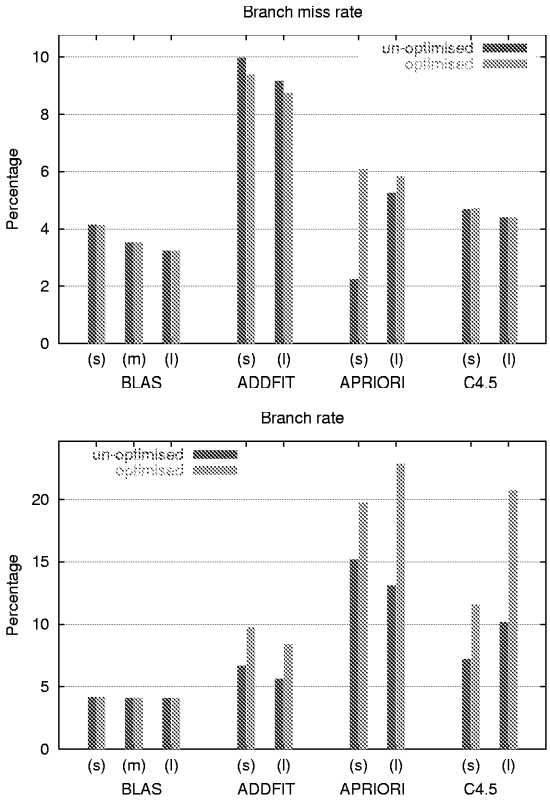


Figure 8: Branch miss rate and branch rate.

It is interesting to point out that the biggest contribution to the data stall rates comes from the `Rstall_storeQ` event which means that the store queue is full and that the store instruction is the first instruction in a group. This is the case mostly for large data sets.

4.5 Branches

In Section 4.1 about instruction mix (Table 4) we saw that except for *BLAS* the optimised codes displayed increased number of branches as compared with un-optimised codes. The highest figures were for *APRIORI* and *C4.5* with large data sets. The increased frequency of branches can clearly be seen in the lower plot of Figure 8 (branch rate).

It is surprising or rather revealing how well branch prediction works in modern CPUs. Despite the relatively large increases in number of branches the branch miss rates remained almost unchanged for all tested applications but *APRIORI* for small data set. At this moment it is unclear to us why there is such a discrepancy in branch miss rate for *APRIORI* only.

5. CONCLUSIONS

In general we conclude that the performance counters proved to be a reliable and invaluable source of information about many aspects of applications performance. However, we see that the deep understanding of hardware and the measured code is essential to correctly interpret the results. We also see that smaller sections of code are better candidates for performance evaluation using hardware counters as their impact on hardware components can be easier understood and hence altered to remove performance bottlenecks.

Ways to improve the performance of KDD and similar applications is to try to use algorithms and data structures which follow as close as possible current computer architectural designs, and – if possible – redesign algorithms to better utilise the floating-point capabilities of modern processors. Designers of processors, computer and operating systems, and compilers on the other hand should be aware of the emerging class (and market) of commercial applications that do not have regular memory and I/O access patterns and that are not mainly performing floating-point operations. As such applications become more and more important in the future, there is a need for computing platforms that are optimised for them.

Another, quite separate issue, is the use of compiler flags. With the advancements of optimising compilers various compiler flags, in our opinion, often will have more bearing on performance than hand placed changes in the code. Is is not to say that we should not search for better algorithms. It merely points out the fact that todays compilers "are more aware" of underlaying hardware than the average user and are able to do optimisations on larger and larger sections of source code.

While many people simply compile their programs with a `-O`, `-O2`, `-fast` or a similar option, one question that arises is how well such compiler optimisations work on commercial applications that have dynamic and irregular data struc-

tures. Let us quote words from the ultimate reference on Sun compilers [12]:

> "The `-fast` option is a good starting point for compiler optimization of well-behaved programs."

Most compiler options were designed with regular applications in mind. Modern compiler have tens up to hundreds of different flags, that can change the run time of a given application tremendously. Many options are designed to improve floating-point intensive applications, but the question arises how good they help KDD and other commercial applications.

6. RELATED RESEARCH

There is much ongoing research in dedicated KDD and data mining algorithms and in their parallel implementations. In contrast, we are only aware of a small number of publications dealing with the performance analysis of KDD applications [5; 6; 16; 18]. More work has been done on analysing database servers and related commercial applications [1; 3; 15; 23]. To our knowledge, no performance analysis of KDD applications has previously been done using hardware counters.

The memory behaviour of a parallel association rule algorithm is discussed in [18]. The authors looked at custom memory placement scheme and found that simple schemes (like the different hash tree building blocks being allocated in a single memory region) can be quite efficient – improving the execution time for some data sets up to a factor of two. They state that the data structures used by association rules algorithms (hash trees and lists) exhibit poor locality, and the arbitrary allocation of memory makes it difficult to detect and eliminate false sharing. A run time memory allocation library based on the Unix `malloc()` library is presented, which allows customised memory allocation.

Memory characteristics of a parallel implementation of the self-organising map (SOM) neural network model is discussed in [16]. Four characteristics were examined and compared. First, the working set size (temporal locality), second the spatial locality and memory block utilisation, third the communication characteristics and scalability, and fourth the TLB performance. The authors use a simulation tool adapted from the *Augmint* toolkit. They conclude that the size of the working set is not sensitive to the number of input records.

In [5; 6] the popular decision tree induction algorithm *C4.5* is analysed in its memory and parallelisation characteristics. The authors are using *RSIM* [17] simulating three different instruction level parallelism (ILP) processors. One of their conclusions is that such an algorithm is limited by the memory latency and bandwidth, and cache size has a significant effect on performance as well. In [6] a parallel version of *C4.5* optimised for a *ccNUMA* is presented and analysed. This parallel version puts significantly less pressure on the memory hierarchy, and has a larger working set.

The memory system characteristics of some commercial workloads is studied in [3]. The authors present detailed performance studies of three different important classes of workloads: Online transaction processing (OLTP), decision support systems (DSS) and Web index search. They use monitoring experiments and *SimOS* [20] to study the effects of architectural variations. One of their findings is that operating system activity and I/O latencies do not dominate the behaviour of well-tuned database workloads. For OLTP a large off-chip cache is in favour, while DSS and the Web index search are primarily sensitive to the size and latency of on-chip caches.

A performance analysis of the *TPC-C* benchmark on a four-processor Pentium based SMP is presented in [23]. The authors analytically model the performance and then validate their results with simulations (using *SimOS*) and hardware counter experiments. They conclude that experimentally based evaluation of complex commercial applications is time consuming, and that analytical modelling is a feasible alternative.

The authors of [8] discuss methods to make pointer-based data structures cache conscious. They present three different methods. First, clustering (packing data structures that a program is likely to access at the same time into a cache block), secondly coloring (place elements in memory such that elements accessed at the same time map to non-conflicting cache regions) and thirdly compression (compressing data structures so that more elements fit into a cache block). They also propose structure splitting into hot and cold parts (with the hot parts being accessed more frequently than cold parts), and show the suitability of these approaches and the resulting performance improvements.

Four commercial database systems are compared in a study [1] on an *Intel Xeon* processor using hardware counters. The authors used memory resident databases and basic operations (simple queries) to identify common trends in the performance and memory access behaviour. One conclusion is that almost half of the time is spent on stalls. Detailed analysis presented show that 90% of the memory stalls are due to second-level cache data misses (while first-level data stalls are not important) and first-level instruction cache misses (while second-level instruction cache misses are not important). These results therefore suggest that database developers should pay more attention to the data placement (layout) in the second-level cache, and also focus on optimising the critical path for the instruction cache.

An earlier study [13] uses traces and simulations from an *IBM Power* architecture to contrast the differences between technical and commercial workloads. Six commercial applications (including TPC benchmarks, file servers, etc.) are compared to eight technical and scientific applications (which included computational chemistry codes, various simulations, etc.). The findings include that commercial applications are often multi-user and contain many processes, and thus have more operating system calls. The branch behaviour is much less predictable (no long loops, but more decision type branches), they also contain less floating-point operations and have different I/O characteristics. Commercial applications also have larger instruction foot-prints compared to technical applications, which means they can profit more from larger (instruction) cache sizes than technical applications. The authors also state that commercial data is not, and cannot be, very cache efficient because data is often private to processes. Process switching also increases the likelihood that cache contents are overwritten by the time a process is re-scheduled after context switching.

A simulator study using SimICS using two database engines is presented in [15]. The authors analyse and then model the size of working-sets for various database queries for decision

support systems (thee queries from TPC-D). Their results show that the most performance critical working-sets are small even for large databases and they do not grow with the size of a database. These working-sets are caused by the instructions and private data that are needed to access a single tuple.

7. OUTLOOK

In this paper we presented experiments of analysing KDD applications with hardware counters on an *UltraSPARC-III* platform. To better understand the memory access characteristics of KDD applications further experiments with various data sets and other hardware event counters are needed. Running KDD applications on a machine simulator (e.g. *SPARC Sulima* [11]) will allow us to change machine parameters like cache size and access times, and thus help to find bottlenecks in such applications. We are also planning to port our experimental setup to different processors (e.g. *Fujitsu Primepower SPARC server* or *Intel Pentium*).
The vast number of options and switches present in optimising compilers is overwhelming. Most of them were designed to assist scientific and engineering applications but not KDD and similar commercial applications. Changing a particular option and tracing the effects of such change on the behaviour of the application (using performance counters) will lead to better understanding of the compiler options/application performance relation. It can also bring more light on which non-trivial changes in architecture could benefit KDD type applications. In future research we are therefore planning to conduct systematic tests with KDD applications using various possible compiler optimisations, to see which ones are favourable for such applications.

Acknowledgment

This research is funded by the ANU/Fujitsu CAP program. The authors would like to thank Peter Strazdins for helpful discussions on SPARC architecture and Solaris operating system issues, and Alistair Rendell for his support.

8. REFERENCES

[1] A.G. Ailamaki, D.J. DeWitt, M.D. Hill and D.A. Wood, *DBMSs on modern processors: Where does the time go?* Technical Report 1394, University of Wisconsin, Department of Computer Science, 1999.

[2] R. Agrawal and R. Srikant, *Fast Algorithms for Mining Association Rules*, in Proceedings of the 20th VLDB Conference, Santiago, Chile, 1994.

[3] L. Barroso, K. Gharachorloo and F. Bugnion, *Memory System Characterization of Commercial Workloads*, Proceedings of the 25th Annual International Symposium on Computer Architecture (ISCA-98), 1998.

[4] R. Berrendorf and B. Mohr, *PCL – The Performance Counter Library: A Common Interface to Access Hardware Performance Counters on Microprocessors (Version 2.0)*, Research Centre Juelich, Central Institute for Applied Mathematics, September 2000.
`http://www.kfa-juelich.de/zam/PCL/`

[5] J.P. Bradford and J. Fortes, *Performance and Memory-Access Characterization of Data Mining Applications*, Workshop on Workload Characterization, 1998. Workshop held in conjunction with the 31st Annual International Symposium on Microarchitecture.

[6] J.P. Bradford and J. Fortes, *Characterization and Parallelization of Decision Tree Induction*, School of Electrical and Computer Engineering, Purdue University, 1999.

[7] S. Browne, J. Dongarra, N. Garner, K. London and P. Mucci, *A Scalable Cross-Platform Infrastructure for Application Performance Tuning Using Hardware Counters*, Proceedings SC'2000, November 2000.
`http://icl.cs.utk.edu/projects/papi/`

[8] T.M. Chilimbi, M.D. Hill and J.R. Larus, *Making Pointer-Based Data Structures Cache Conscious*, IEEE Computer, December 2000.

[9] P. Christen, M. Hegland, O.M. Nielsen, S. Roberts, P.E. Strazdins and I. Altas, *Scalable Parallel Algorithms for Surface Fitting and Data Mining*, Elsevier Journal of Parallel Computing, special issue on Aspects of Parallel Computing for Linear Systems and Associated Problems, September 2001.

[10] P. Christen, O.M. Nielsen, M. Hegland and P.E. Strazdins, *Parallel Data Mining on a Beowulf Cluster*, Accepted by the HPC Asia 2001 Conference, Gold Coast, Queensland, Australia, September 2001.

[11] B. Clarke, A. Czezowski and P. Strazdins, *Implementation Aspects of Sparc V9 Complete Machine Simulator* , to appear in ACSAC-2002, the Australasian Computer Systems Architecture Conference, Melbourne, Australia, January 2002.

[12] R. Garg and I. Sharapov, *Techniques for Optimizing Applications*, SUN Blueprints, Sun Microsystems Press, 2002.

[13] A.M. Grizzaffi Maynard, C.M. Donnelly and B.R. Olszewski, *Contrasting characteristics and cache performance of technical and multi-user commercial workloads*, in Proceedings of the sixth international conference on Architectural support for programming languages and operating systems, San Jose, California, 1994.

[14] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2000.

[15] M. Karlsson and P. Stenström, *An analytical model of the working-set sizes in decision-support systems*, in Proceedings of the international conference on Measurements and modeling of computer systems, Santa Clara, California, 2000.

[16] J.S. Kim, X. Qin and Y. Hsu, *Memory characterization of a parallel data mining workload*, in Workload Characterization: Methodology and Case Studies. Based on the First Workshop on Workload Characterization. IEEE Comput. Soc, Los Alamitos, CA, USA, 1999.

[17] V. Pai. P. Ranganathan and S. Adve, *RSIM: An Execution-Driven Simulator for ILP-Based Shared-Memory Multiprocessors and Uniprocessors*, In Proceedings of the Third Workshop on Computer Architecture Education, February 1997.

[18] S. Parthasarathy, M.J. Zaki and W. Li, *Custom Memory Placement for Parallel Data Mining*, Technical Report 653, University of Rochester, Computer Science Department, 1997.

[19] J.R. Quinlan, *Programs for Machine Learning*, Morgan Kaufmann, 1993.

[20] M. Rosenblum, S.A. Herrod, E. Witchel and A. Gupta, *Complete Computer System Simulation: The SimOS Approach*, IEEE parallel and distributed technology: Systems and applications, vol. 3, no. 4, 1995.

[21] P.E. Strazdins and J.G. Lewis, *An Efficient and Stable Method for Parallel Factorization of Dense Symmetric Indefinite Matrices*, in Proceedings of the 5th International Conference and Exhibition on High-Performance Computing in the Asia-Pacific Region (HPC Asia 2001), Gold Coast, September 2001.

[22] *SPARC JPS1. Implementation Supplement:Sun UltraSPARC III*, May 2001.

[23] X. Zhang, Z. Zhu and X. Du, *Analysis of Commercial Workload on SMP Multiprocessors*, Proceedings of Performance'99, August, 1999.

Author Index