Research and Development in
Intelligent Systems XXV

Max Bramer Frans Coenen Miltos Petridis
Editors

Research and Development in Intelligent Systems XXV

Proceedings of AI-2008, the Twenty-eighth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence

Springer

Max Bramer, BSc, PhD, CEng, CITP,
FBCS, FIET, FRSA, FHEA
Faculty of Technology
University of Portsmouth, Portsmouth, UK

Frans Coenen, BSc, PhD
Department of Computer Science,
University of Liverpool, Liverpool, UK

Miltos Petridis, DipEng, MBA, PhD,
MBCS, AMBA
University of Greenwich, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN 978-1-84882-170-5 e-ISBN 978-1-84882-171-2

Printed on acid-free paper

Springer Science+Business Media
springer.com

TECHNICAL PROGRAMME CHAIR'S INTRODUCTION

M.A.BRAMER
University of Portsmouth, UK

This volume comprises the refereed technical papers presented at AI-2008, the Twenty-eighth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, held in Cambridge in December 2008. The conference was organised by SGAI, the British Computer Society Specialist Group on Artificial Intelligence.

The papers in this volume present new and innovative developments in the field, divided into sections on CBR and Classification, AI Techniques, Argumentation and Negotiation, Intelligent Systems, From Machine Learning to E-Learning and Decision Making. The volume also includes the text of short papers presented as posters at the conference.

This year's prize for the best refereed technical paper was won by a paper entitled 'On the Classification Performance of TAN and General Bayesian Networks' by Michael G. Madden (College of Engineering & Informatics, National University of Ireland, Galway, Ireland). SGAI gratefully acknowledges the long-term sponsorship of Hewlett-Packard Laboratories (Bristol) for this prize, which goes back to the 1980s.

This is the twenty-fifth volume in the *Research and Development* series. The Application Stream papers are published as a companion volume under the title *Applications and Innovations in Intelligent Systems XVI*.

On behalf of the conference organising committee I should like to thank all those who contributed to the organisation of this year's technical programme, in particular the programme committee members, the executive programme committee and our administrators Rachel Browning and Bryony Bramer.

Max Bramer
Technical Programme Chair, AI-2008

ACKNOWLEDGEMENTS

AI-2008 CONFERENCE COMMITTEE

Dr. Miltos Petridis University of Greenwich	(Conference Chair and UK CBR Organiser)
Dr Frans Coenen University of Liverpool	(Deputy Conference Chair, Local Arrangements and Deputy Technical Programme Chair)
Prof. Adrian Hopgood De Montfort University	(Workshop Organiser)
Rosemary Gilligan	(Treasurer)
Dr Nirmalie Wiratunga The Robert Gordon University	(Poster Session Organiser)
Professor Max Bramer University of Portsmouth	(Technical Programme Chair)
Dr. Tony Allen Nottingham Trent University	(Application Programme Chair)
Richard Ellis Stratum Management Ltd	(Deputy Application Program Chair)
Alice Kerly University of Birmingham	(Research Student Liaison)
Dr. Kirsty Bradbrook	(Research Student Liaison)
Prof. Alun Preece University of Cardiff	(Committee Member)
Rachel Browning BCS	(Conference Administrator)
Bryony Bramer	(Paper Administrator)

TECHNICAL EXECUTIVE PROGRAMME COMMITTEE

TECHNICAL PROGRAMME COMMITTEE

CONTENTS

ARGUMENTATION AND NEGOTIATION

INTELLIGENT SYSTEMS

FROM MACHINE LEARNING TO E-LEARNING

BEST TECHNICAL PAPER

On the Classification Performance of TAN and General Bayesian Networks

Michael G. Madden[1]

Abstract. Over a decade ago, Friedman *et al.* introduced the Tree Augmented Naïve Bayes (TAN) classifier, with experiments indicating that it significantly outperformed Naïve Bayes (NB) in terms of classification accuracy, whereas general Bayesian network (GBN) classifiers performed no better than NB. This paper challenges those claims, using a careful experimental analysis to show that GBN classifiers significantly outperform NB on datasets analyzed, and are comparable to TAN performance. It is found that the poor performance reported by Friedman *et al.* are not attributable to the GBN per se, but rather to their use of simple empirical frequencies to estimate GBN parameters, whereas basic parameter smoothing (used in their TAN analyses but not their GBN analyses) improves GBN performance significantly. It is concluded that, while GBN classifiers may have some limitations, they deserve greater attention, particularly in domains where insight into classification decisions, as well as good accuracy, is required.

1 Introduction

This paper examines the performance of Bayesian networks as classifiers, comparing their performance to that of the Naïve Bayes (NB) classifier and the Tree-Augmented Naïve Bayes (TAN) classifier, both of which make strong assumptions about interactions between domain variables.

In the experiments performed for this work, described below in Section 3, standard Bayesian networks (referred to as General Bayesian Networks, GBNs, to distinguish them from NB and TAN) are compared with NB and TAN classifiers on 28 standard benchmark datasets. Our experiments indicate that the GBN classifier is substantially better than NB, with performance closer to that of TAN. This contrasts with the conclusions drawn in the landmark paper on Bayesian network classifiers by Friedman *et al.* (1997). That paper presented results on many of the same datasets, showing that GBNs constructed using the minimum description length (MDL) score tend to perform no better than NB. That result has been widely noted by other authors (e.g. Grossman & Domingos, 2004; Keogh &

[1] College of Engineering & Informatics, National University of Ireland, Galway, Ireland.
Email: michael.madden@nuigalway.ie

Pazzani, 2002); in one case the result was interpreted as indicating that NB "easily outperforms" GBN.

Our contention is that it has become 'accepted wisdom' that GBN classification performance is no better than that of NB, and significantly worse than TAN (ignoring other considerations such as computational complexity or interpretability). Our results indicate that GBN's classification performance is superior to that of NB and much closer to that of TAN, when the same parameter estimation procedure is used for all.

It turns out that Friedman *et al.* used simple frequency counts for parameter estimation in constructing GBN classifiers, whereas they used parameter smoothing in constructing TAN classifiers (see Sec. 2.3 for details). Our experiments show that if frequency counts are used for both GBN and TAN, neither is much better than NB (Sec. 3.3, Figure 5), but if parameter smoothing is used for both, they both perform similarly well (Figure 4). Furthermore, since GBN classifiers are commonly constructed through heuristic search, it is possible for improved GBN construction algorithms to lead to improved performance.

The structure of the paper is as follows. Section 2 reviews Bayesian networks and the algorithms for constructing GBN and TAN classifiers that are used in this paper. Section 3 presents experiments applying NB, TAN and two GBN algorithms to classification problems on 28 standard datasets, and identifies why the results of this paper are at odds with those of Friedman *et al.* as mentioned above. Finally, Section 4 draws general conclusions about the suitability of GBNs as classifiers.

2 Bayesian Networks and Classification

As is well known, a Bayesian network is composed of the network structure and its conditional probabilities. The structure B_S is a directed acyclic graph where the nodes correspond to domain variables x_1, …, x_n and the arcs between nodes represent direct dependencies between the variables. Likewise, the absence of an arc between two nodes x_1 and x_2 represents that x_2 is independent of x_1 given its parents in B_S. Using the notation of Cooper & Herskovits (1992), the set of parents of a node x_i in B_S is denoted π_i. The structure is annotated with a set of conditional probabilities, B_P, containing a term $P(X_i \mid \Pi_i)$ for each possible value X_i of x_i and each possible instantiation Π_i of π_i

2.1 Inductive Learning of Bayesian Networks

Several algorithms have been proposed since the late 1980s for inductive learning of general Bayesian networks. Recent developments include the global

optimization approach of Silander and Myllymäki (2006), the Greedy Equivalence Search algorithm (Chickering, 2002), and the Three-Phase Dependency Analysis algorithm (Cheng *et al.,* 2002), though this latter algorithm has subsequently been shown to be incorrect (Chickering & Meek, 2006). We evaluate two approaches to GBN construction, described in the following subsections, both of which have relatively low computational complexity:

1. The K2 search procedure (Cooper & Herskovits, 1992) in conjunction with the Bayesian BDeu scoring metric (Buntine, 1991), which is a refinement of the K2 metric
2. The approach used by Friedman *et al.* (1997), which combines hill-climbing search with the MDL score.

These are both search-and-score methods for construction of GBNs; a search heuristic is used to propose candidate networks, and a scoring function is used to assess, for any two candidates, which one is more likely given the training data.

The scoring functions and search procedures are described in greater detail in the following sub-sections. Rather can constructing general BN structures, restrictions may be placed on the structures; this is described in Section 2.2. Typically, the conditional probabilities (parameters) associated with a network are not computed from the data until after the structure has been found; parameter estimation is described in Section 2.3.

2.1.1 K2 Search with BDeu Scoring Approach

If D is a database of training cases, Z is the set of variables in each case in D, and B_{S_i} and B_{S_j} are two belief-network structures containing exactly those variables that are in Z, then the comparison amounts to calculating $P(B_{S_i}|D)/P(B_{S_j}|D)$, which in turn reduces to calculating $P(B_{S_i},D)/P(B_{S_j},D)$.

Assume that Z is a set of n discrete variables, where a variable x_i in Z has r_i possible value assignments, $(v_{i1}, \ldots, v_{ir_i})$, and that D has N cases, each with a value assignment for each variable in Z. A network structure B_S is assumed to contain just the variables in Z. Each variable x_i in B_S has zero or more parents, represented as a list π_i. Let w_{ij} denote the jth unique instantiation of π_i relative to D, and assume that there are q_i such unique instantiations of π_i. Let N_{ijk} be defined as the number of cases in D in which variable x_i has the value v_{ik} and π_i is instantiated as w_{ij}. Let N'_{ijk} denote a Dirichlet parameter. Let N_{ij} and N'_{ij} be defined as:

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk} \quad , \quad N'_{ij} \equiv \sum_{k=1}^{r_i} N'_{ijk} \tag{1}$$

With these definitions, the BD metric (Heckerman *et al.,* 1995) is defined as:

$$P(B_S, D) = P(B_S) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})} \tag{2}$$

Note that Γ is the gamma function, defined as $\Gamma(x+1) = x\Gamma(x)$, which is closely related to the factorial function but defined for real numbers, not just integers. In a practical implementation, the logs of terms in Eq. 2 are computed.

The K2 metric (Cooper & Herskovits, 1992) corresponds to Eq. 2 with all Dirichlet exponents set to 'uninformative' values of $N'_{ijk} = 1$. Alternative uninformative values are proposed by Buntine (1991):

$$N'_{ijk} = \frac{N'}{r_i q_i} \tag{3}$$

Using Buntine's values, Eq. 2 becomes what Heckerman *et al.* (1995) term the BDeu metric, which has the additional property of being structure-equivalent. This is the metric used in the current work. Assuming that all structures are equally likely *a priori,* $P(B_S)$ is constant, so to maximize $P(B_S,D)$ just requires finding the set of parents for each node that maximizes the second inner product of Eq. 2.

The K2 search procedure requires a node ordering. It operates by initially assuming that a node has no parents, and then adding incrementally that parent whose addition most increases the probability of the resulting network. Parents are added greedily to a node until the addition of no one parent can increase the structure probability. This is repeated for all nodes in the sequence specified by the node ordering.

In the experiments of Section 3, the node ordering in each dataset is arbitrarily taken to be the order of attributes in the input files, except that the class node is always placed first in the order. In addition, the maximum number of parents a node may have is limited to 4.

2.1.2 MDL Scoring Approach

In constructing GBNs, Friedman *et al.* (1997) use a scoring function based on the minimum description length (MDL) principle. The MDL score of a network B given a database of training cases D is:

$$MDL(B \mid D) = \frac{1}{2}\log N|B| - LL(B \mid D) \tag{4}$$

where $|B|$ is the number of parameters in the network and $LL(B \mid D)$ denotes the log-likelihood of B given D. To calculate $LL(B \mid D)$, let $\hat{P}_D(\cdot)$ be the empirical probability measure defined by frequencies of events in D. Then:

$$LL(B \mid D) = N\sum_i \sum_{X_i,\Pi_i} \hat{P}_D(X_i,\Pi_i)\log(\hat{P}_D(X_i|\Pi_i)) \tag{5}$$

The search procedure used by Friedman *et al.* is to start with the empty network and successively apply local operations that greedily reduce the MDL score maximally until a local minimum is found. The local operations applied are arc insertion, arc deletion and arc reversal.

2.1.3 Classification using a GBN

A Bayesian network may be used for classification as follows. Firstly, any nodes outside of the Markov blanket of the classification node x_c may be deleted. Then, assume that the value of x_c is unknown and the values of all other nodes are known. Then, for every possible instantiation of x_c, calculate the joint probability of that instantiation of all variables in the network given the database D. By the definition of a Bayesian network, the joint probability of a particular instantiation of all n variables is calculated as:

$$P(x_1 = X_1,...,x_n = X_n) = \prod_{i=1}^{n} P(x_i = X_i | \pi_i = \Pi_i) \tag{6}$$

By normalizing the resulting set of joint probabilities of all possible instantiations of x_c, an estimate of the relative probability of each is found. The vector of class probabilities may be multiplied by a misclassification cost matrix, if available. Note that the classification node is not considered 'special' when building the GBN, and in Eq. 6, x_c is just one of the variables $x_1 \ldots x_n$.

Although arbitrary inference in a GBN with discrete variables is NP-hard (Cooper, 1990), the classification procedure just described just requires Eq. 6 to be evaluated once for each possible instantiation of x_c; thus its time complexity is $O(n_m r_c)$, where n_m is the number of nodes in x_c's Markov blanket; $n_m \leq n$.

2.2 Restricted Bayesian Classifiers

Figure 1 schematically illustrates the structure of the Bayesian classifiers considered in this paper. The simplest form of Bayesian classifier is Naïve Bayes. When represented as a Bayesian network, a Naïve Bayes (NB) classifier has a simple structure whereby there is an arc from the classification node to each other node, and there are no arcs between other nodes, as illustrated in Figure 1(a). Since NB has a fixed structure, learning simply involves estimating the parameters according to one of the procedures discussed below in Section 2.3.

Several researchers have examined ways of achieving better performance than NB. Friedman *et al.* (1997) in particular consider (among other structures) *Tree Augmented Naïve Bayes* (TAN), which allows arcs between the children of the classification node x_c as shown in Figure 1(b), thereby relaxing the assumption of conditional independence. In their approach, each node has x_c and at most one other node as a parent, so that the nodes excluding x_c form a tree structure. Optimal TAN structures are constructed by finding the maximum weighted spanning tree within a complete graph connecting the nodes, where arcs are annotated by the conditional mutual information between all pairs of non-class nodes, conditioned on the class node, according to Eq. 7.

$$I(x_i, x_j|c) = \sum_{X_i, X_j, C} P(X_i, X_j, C) \log \frac{P(X_i, X_j|C)}{P(X_i|C)P(X_j|C)} \quad (7)$$

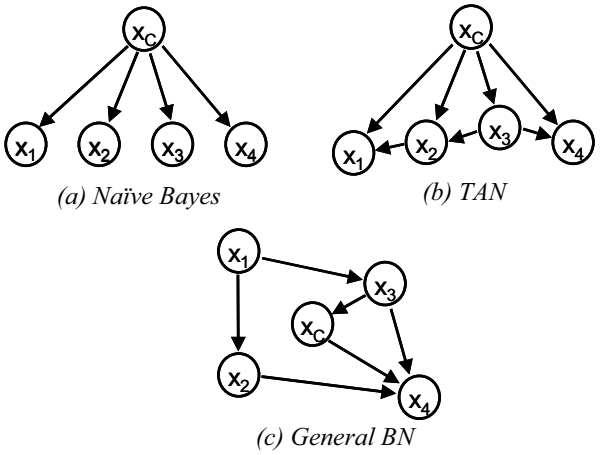

Figure 1: Illustration of Naive Bayes, TAN and General BN Structures.

2.3 Parameter Estimation

Let θ_{ijk} denote the conditional probability that a variable x_i in B_S has the value v_{ik}, for some k from 1 to r_i, given that the parents of x_i, represented by π_i, are instantiated as w_{ij}. Then $\theta_{ijk} = P(x_i=k|\pi_i=w_{ij})$ is termed a network conditional probability. The simplest form of parameter estimation is based on frequency counts (referred to as *unsmoothed* estimates by Friedman *et al*.):

$$\theta_{ijk}^{f} = \frac{N_{ijk}}{N_{ij}} \quad (8)$$

A problem with using Eq. 8 is that it can result in zero estimates for some parameters if not all combinations of variables are well represented in the training data, resulting in a probability of 0 being computed for some instantiations of all variables. One solution is to replace zero estimates by a small positive value.

As well as using unsmoothed estimates, Friedman *et al.* use technique based on Dirichlet priors that they term *parameter smoothing,* which boils down to the following calculation:

$$\theta^{s}_{ijk} = \frac{N_{ijk} + N_0 \, N_i / N}{N_{ij} + N_0} \tag{9}$$

where $N_i/N = \hat{P}(x_i)$ is the frequency of the given value of x_i observed in the dataset. (Friedman *et al.* report that, after experimentation, a value of $N_0 = 5$ was chosen.)

As part of our controlled comparisons, the same parameter smoothing is used for all classifiers in the analyses presented below in Section 3.

To avoid any ambiguity, it should be pointed out that smoothed parameter estimates are used only to estimate the conditional probabilities, B_P, after the network structure, B_S, has been determined. TAN and GBN structure learning uses simple frequency counts (Eq. 8).

3 Experiments

3.1 Methodology

For this work, the Naïve Bayes, TAN and two general BN algorithms were compared using 26 datasets from the UCI repository of Machine Learning datasets (Asuncion & Newman 2007). For consistency with previous work in this domain (Cheng & Greiner, 2001; Friedman *et al.,* 1977; Keogh & Pazzani, 2002; Madden, 2003), continuous variables were discretized using the discretization utility of MLC++ (Kohavi *et al.,* 1977) with its default entropy-based setting (*Dougherty et al.,* 1995) and any cases with missing values were removed. The two general BN algorithms are those listed earlier:

1. GBN-K2: K2 search procedure with the Bayesian BDeu scoring metric
2. GBN-HC: hill-climbing search with MDL score, following Friedman *et al*.

The GBN-HC implementation used in this work is that in WEKA (Bouckaert, 2004a). The NB, TAN and GBN-K2 algorithms were implemented for this work in Common Lisp (code available by email on request).

Previous comparisons of similar classifiers (Cheng & Greiner, 2001; Friedman *et al.,* 1977; Madden, 2003) have estimated classifier accuracy using holdout sets for the larger datasets and 5-fold cross validation for smaller datasets. However, it has been shown that such analyses may suffer from high sensitivity to the specific divisions used (Bouckaert, 2004a). Also, previous analyses have compared accuracy figures by simply considering the magnitude of the estimated accuracy without performing statistical significance tests (Cheng & Greiner, 2001; Friedman *et al.,* 1977), or using t-tests that are not corrected to account for the overlap in folds from a multi-fold cross-validation run (Madden, 2003). This latter approach has been shown to have a high Type I error (Nadeau & Bengio, 2000).

To avoid such problems, the experimental methodology used in this work follows the 10 x 10 fold sorted cross-validation approach proposed by Bouckaert (2004b), with associated t-tests to measure significance. This has been shown to have good replicablility, thereby facilitating future comparisons, and because by applying it consistently across all datasets and algorithms, coherent comparisons can be drawn.

3.2 Results

Table 1 lists the accuracy (and standard deviation of accuracy) of each of the four classification algorithms being considered, as measured from 10 runs of 10-fold cross-validation on each dataset. In each row, the best of the four classifier results are displayed in bold. Specifically, for each dataset, the classifier with the highest performance is highlighted in bold and compared with that of the other two classifiers, using a paired t-test at the 5% significance level based on the 10x10 fold sorted cross-validation results. If another's performance is not significantly different from the best, it is also highlighted, but if the differences between all four classifiers are not statistically significant, then none of them are highlighted.

As these results show, there are no statistical differences between the algorithms on 10 of the 26 datasets, at the 5% significance level. In just 2 other cases, NB is best (including joint best), in 13 cases TAN is best, in 10 cases GBN-K2 is best and in 7 cases GBN-HC is best.

Figure 2 shows two scatter-plots comparing TAN with NB and with GBN-HC. Figure 2(a) shows that TAN generally outperforms NB, as was also demonstrated in the experiments of Friedman *et al.* (1997). Figure 2(b) also shows TAN outperforming GBN-HC, though the difference in performance is not as marked as in the results of Friedman et al.

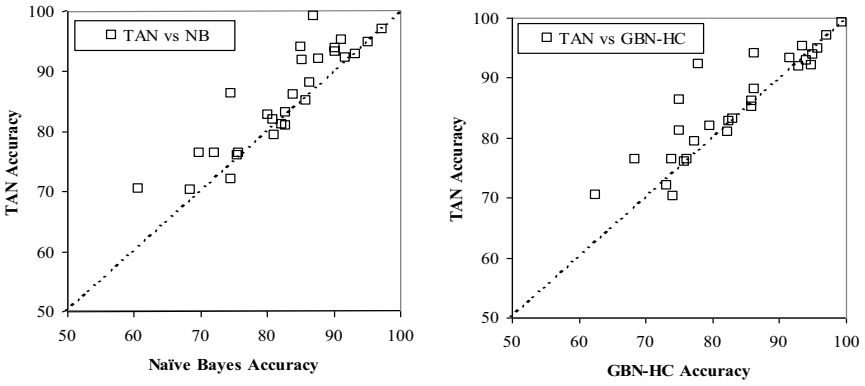

Figure 2: Relative accuracies of: (a) TAN and NB; (b) TAN and GBN-HC.

Table 1: Classification performance (accuracy ± std dev) of four algorithms as measured on 28 datasets; results in bold are best or joint best, as described in text.

No.	Dataset	Naïve Bayes	TAN	GBN-K2	GBN-HC
1	Adult	84.03 ± 0.53	**86.15 ± 0.35**	**86.16 ± 0.33**	**86.02 ± 0.48**
2	Australian	85.80 ± 4.03	85.06 ± 3.90	86.22 ± 3.83	85.93 ± 4.06
3	Breast Cancer	97.38 ± 1.84	96.99 ± 1.88	97.32 ± 1.81	97.15 ± 1.83
4	Car	85.15 ± 2.74	**93.96 ± 1.90**	89.61 ± 2.20	86.36 ± 3.15
5	Chess	87.85 ± 1.70	92.09 ± 1.39	**94.45 ± 1.41**	**94.95 ± 1.47**
6	Cleve	82.87 ± 6.20	81.04 ± 6.77	81.07 ± 6.22	82.33 ± 6.27
7	Connect-4	72.11 ± 0.63	76.43 ± 0.40	**79.08 ± 0.66**	73.88 ± 0.70
8	Corral	87.05 ± 9.46	**99.23 ± 3.19**	**99.62 ± 2.53**	**99.38 ± 2.37**
9	DNA Splice	95.26 ± 0.98	94.92 ± 1.10	**95.93 ± 1.05**	**95.81 ± 1.02**
10	Flare	80.12 ± 3.47	82.65 ± 3.47	82.24 ± 3.39	82.56 ± 3.48
11	German	74.61 ± 4.31	72.07 ± 4.04	74.20 ± 3.97	73.25 ± 4.07
12	Glass2	81.16 ± 8.68	79.37 ± 8.95	79.00 ± 9.35	77.29 ± 9.86
13	Heart	82.74 ± 6.70	83.11 ± 7.30	82.30 ± 7.49	83.04 ± 7.32
14	Hepatitis	86.38 ± 10.97	88.00 ± 11.64	87.00 ± 13.29	86.38 ± 14.22
15	Letter	74.67 ± 1.05	**86.28 ± 0.61**	81.76 ± 0.73	75.12 ± 0.72
16	Lymphography	82.16 ± 10.61	81.07 ± 9.57	77.46 ± 9.47	75.06 ± 10.98
17	Mofn-3-7-10	85.34 ± 3.43	**91.96 ± 2.63**	86.85 ± 3.56	**93.04 ± 2.86**
18	Nursery	90.29 ± 0.77	**93.30 ± 0.81**	91.18 ± 0.89	91.68 ± 0.82
19	Pima	75.69 ± 4.42	76.37 ± 3.94	76.33 ± 4.26	76.18 ± 4.27
20	Segment	91.27 ± 1.70	**95.27 ± 1.49**	**94.64 ± 1.56**	93.45 ± 1.48
21	Soybean-Large	**91.83 ± 3.50**	**92.35 ± 3.08**	89.22 ± 4.22	78.02 ± 6.45
22	Spect	68.53 ± 9.14	**70.29 ± 8.99**	**68.98 ± 8.50**	**74.19 ± 8.89**
23	Tic Tac Toe	69.76 ± 4.45	**76.32 ± 3.82**	69.26 ± 4.74	68.38 ± 4.83
24	Vehicle	60.62 ± 4.88	**70.36 ± 4.58**	**67.30 ± 5.18**	62.50 ± 5.46
25	Vote	90.27 ± 4.30	**93.84 ± 3.26**	**93.57 ± 3.53**	**95.11 ± 3.03**
26	Waveform-21	**80.90 ± 1.64**	**81.96 ± 1.70**	**81.67 ± 1.56**	79.73 ± 1.96

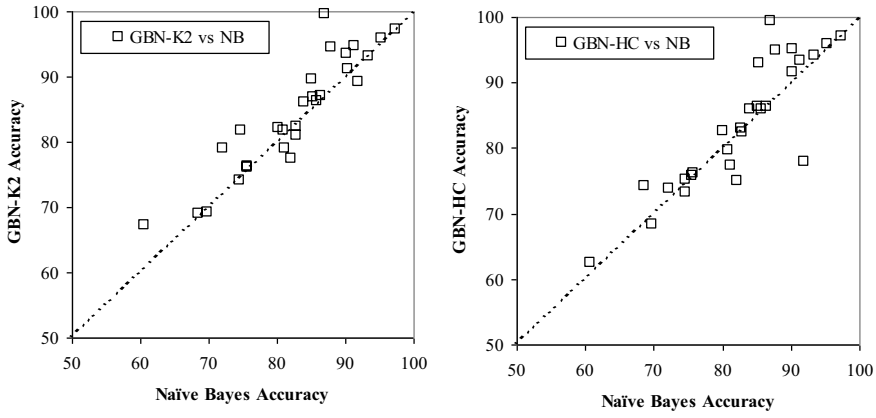

Figure 3: Relative accuracies of: (a) GBN-K2 and NB; (b) GBN-HC and NB.

But what about the claim that GBNs perform as badly as, or even worse than, NB? Figure 3 shows two scatter-plots comparing GBN-K2 and GBN-HC with NB. In this and subsequent graphs, "A vs B" indicates that A is plotted on the vertical axis and B is plotted on the horizontal axis. Visually, points above the diagonal are those where classifier A has higher accuracy. Our results do not provide evidence for that claim. They show that the classification performance of both GBN algorithms is good relative to NB, although the performance of GBN-K2 is a little better than that of GBN-HC. On the basis of paired t-tests, it is found that GBN-K2 is better than NB on 11 datasets whereas NB is better than it on just 1; likewise, GBN-HC is better than BN on 9 datasets whereas NB is better on 1.

Furthermore, when GBN-K2, rather than GBN-HC, is compared with TAN, the differences between them are not at all pronounced, as shown in Figure 4.

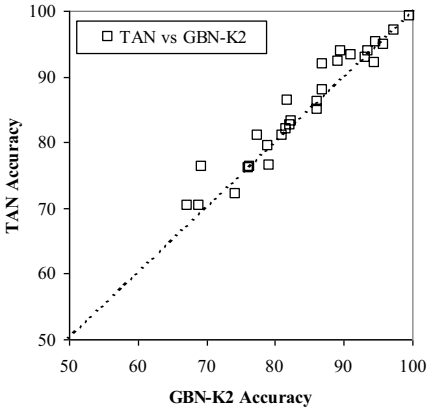

Figure 4: Relative accuracies of TAN and GBN-K2.

3.3 Discussion of Results

The results presented in Table 2 and illustrated in Figure 3 indicate that GBN outperforms NB overall. This conclusion is clearly at variance with the experimental results of Friedman et al., who compared GBN and NB on 25 datasets and reported that GBN was significantly better on 6 and significantly worse on 6. (All of those datasets are included in this study except for CRX and Glass, which are variants of the Australian and Glass2 datasets that are included.) Our GBN-HC algorithm is the same one that they used.

Differences in experimental methodology might account for some of the disparities in conclusions drawn from our work and that of Friedman et al, as their experiments may be more prone to Type I errors and have lower replicability. However, we believe that parameter estimation has a much more significant effect. For the TAN and NB algorithms, they present results using unsmoothed (Eq. 8) and smoothed (Eq. 9) parameter estimates. As would be expected,

parameter smoothing has little effect on the performance of NB, but it improves the performance of TAN since zero probability estimates are more likely to arise in more complex structures. However, Friedman et al. present results for GBN without smoothing only; they do not present corresponding smoothed GBN results, even though one would expect parameter smoothing to improve the performance of GBN also. In contrast, the results presented above in Table 2 and Figures 2-4 use parameter smoothing for all classifiers.

To explore this further, we repeated our analyses using unsmoothed parameter estimates. Figure 5(a) presents a plot comparing Unsmoothed GBN with Unsmoothed NB. These results are qualitatively similar to those of *Friedman et al.*; Unsmoothed GBN is not much better than Unsmoothed NB. However, the comparison in Figure 5(b) is also interesting, as it shows that Unsmoothed TAN is also no better than Unsmoothed NB.

In a further set of experiments, we used unsmoothed parameter estimates but replaced zero probabilities with small epsilon values. When we did so, the results were quite close to the smoothed result of Table 1. We therefore conclude that the essential cause of the poor performance of the TAN and GBN classifiers relative to NB in Figure 5 may be attributed to the zero probabilities in the computations.

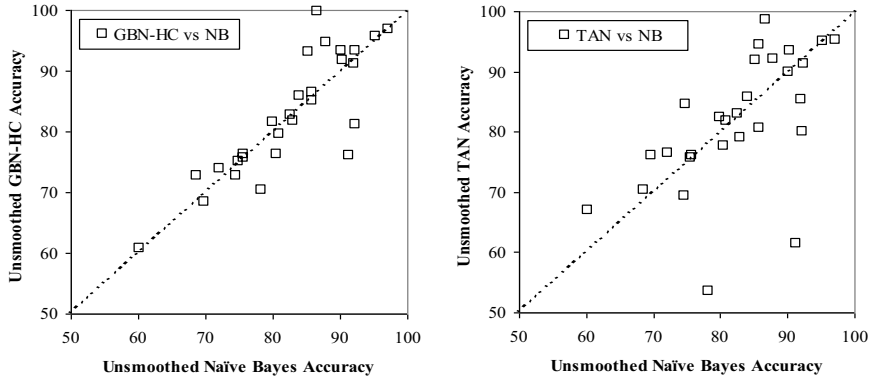

Figure 5: Relative Accuracies of: (a) Unsmoothed GBN-HC vs Unsmoothed NB; (b) Unsmoothed TAN vs Unsmoothed NB.

4 Conclusions: Suitability of GBN as a Classifier

The results of the preceding section have shown that, when TAN and GBN-K2 classifiers are compared under careful experimental procedures and using the same parameter estimation procedure for both, there is little to distinguish between them in terms of classification accuracy.

An advantage of TAN is its low computational complexity, which is $O(n^2 N)$. However, for a fixed maximum number of parents per node, to complexity of the

GBN-K2 algorithm is O(n^3 N r), which is just a factor (n r) worse. (Here, r is the maximum number of different values any node may have.)

Nonetheless, if GBN classifiers are more expensive to construct than TAN classifiers and do not offer greater classification accuracy, why use them? There are other possible drawbacks of GBNs as classifiers:

1. It is often observed in Machine Learning that we should not solve a more general problem than required, so why build a full GBN if all that is required is a classifier?
2. GBNs are in general more complex than TAN classifiers, though not necessarily; in fact, as discussed below, A GBN classifier may end up with fewer arcs than a TAN classifier on the same domain, since not all nodes might be within the Markov blanket.
3. The GBN that best describes the domain as a whole does not necessarily correspond to the one that best discriminates between classes, and the classification node might potentially be unconnected from the network.

While aware of these drawbacks, we propose three reasons for their use:

1. *Insightful analysis:* In many practical domains, particularly where it is required to convince end-users such as scientists or engineers that classification decisions are reasonable and logical, it is as important to gain insight into the problem as it is to achieve high classification accuracy. GBN classifiers support this by modelling the distribution, allowing more complex interactions between nodes to be represented than with TAN and also potentially identifying nodes that are outside the classification node's Markov blanket. They also aid in identifying conditional independencies in the data, which may also be useful for domain insight.
2. *Representational power:* Zhang & Ling (2001) have examined the representational power of discrete BNs and have concluded that, if each node has at most u parents, a BN can represent parity functions of maximum order u. This implies, as would be expected, that GBN has greater representational power than TAN which in turn has greater representational power than NB.
3. *Appropriate complexity:* As noted above, a GBN classifier may have fewer arcs than a TAN classifier for the same domain. In TAN, nodes are must have the class node as a parent, and a full tree of arcs between non-class nodes, so all but two nodes have exactly two parents each. In GBN, there are no such constraints; a node may have no parents or several. On the Adult dataset for example, the typical GBN had 13 arcs with 0-3 parents per node, which is the same number of arcs as Naïve Bayes for the dataset, which has exactly one parent per node. The TAN classifier for the Adult dataset was more complex, with 25 arcs. On the Connect 4 dataset, Naïve Bayes has 13 arcs, TAN has 83 arcs and GBN has a median of 74 arcs.

GBN approaches are not as widely used for classification tasks as TAN. Notable exceptions include the work of Cheng & Greiner (2001), the application by Baesens *et al.* (2002) of Monte Carlo Markov Chain search to constructing GBN classifiers, and Grossman & Domingos' (2004) algorithm for learning GBNs that maximize conditional likelihood.

However, a larger number of researchers have analysed TAN and proposed improvements. Examples include the work of Keogh and Pazzani (2002) who proposed the use of classification accuracy rather than conditional mutual information in building TAN-style classifiers; Zhang & Ling (2001), who extended Keogh & Pazzani's work by using AUC measures; Cerquides and de Mántaras (2005), who identified theoretical weaknesses in the TAN approach and proposed corrections for them; and Garg & Roth (2001) who addressed the question of why classifiers such as TAN that make often inaccurate assumptions tend to perform well.

Although the experiments here have shown that the GBN-K2 algorithm has quite good classification performance, it is likely that other algorithms would perform even better. Given the relative complexity of GBN construction compared to TAN construction, improving the performance of GBN classifiers would appear to be a topic with some potential for research. A limitation of GBN-K2 is that it requires an ordering on nodes. In specific applications it may be possible to determine a reasonable node ordering from domain knowledge, but it would be interesting to analyse the performance of other algorithms that do not require node ordering. That being said, GBN-HC does not require node ordering and its performance on the test datasets was slightly weaker than that of GBN-K2, but its simple hill-climbing search without restarts is quite limited.

In the future, it is hoped to analyse more sophisticated algorithms, particularly the algorithm of Silander and Myllymäki (2006), which searches for a globally optimal network. In order to address the issue noted earlier in this section that the optimal GBN for a domain is not necessarily the optimal one for classification, it would be necessary to develop an approach that constructs a Markov blanket around the classification node.

Overall, we believe that GBNs may deserve greater attention as classifiers, particularly in problem domains where data is plentiful and insight into the domain, as well as high accuracy, is required, although work remains to be done to optimize them for classification tasks.

Acknowledgements

This research has been supported by a Marie Curie Transfer of Knowledge Fellowship of the EU 6th Framework Programme, contract CT-2005-029611.

References

1. Baesens, B., Egmont-Petersen, M., Castelo, R. and Vanthienen, J. (2002) Learning Bayesian network classifiers for credit scoring using Markov Chain Monte Carlo search. *Proc. 2002 International Congress on Pattern Recognition,* IEEE Computer Society.

2. Asuncion, A. & Newman, D.J. (2007). *UCI Machine Learning Repository.* http://www.ics.uci.edu/ ~mlearn/MLRepository.html. University of California, Irvine.
3. Bouckaert, R.R. (2004): Bayesian networks in Weka. *Technical Report 14/2004.* Computer Science Department. University of Waikato.
4. Bouckaert, R.R. (2004): Estimating Replicability of Classifier Learning Experiments. *Proc. 21st International Conference on Machine Learning.*
5. Buntine, W. (1991). Theory Refinement on Bayesian Networks. *Proc. 7th International Conference on Uncertainty in Artificial Intelligence.*
6. Cerquides, J. and de Mántaras, R. (2005). TAN Classifiers Based on Decomposable Distributions. *Machine Learning* Vol. 59, pp 323-354.
7. Cheng, J. and Greiner, R. (2001). Learning Bayesian Belief Network Classifiers: Algorithms and System. *Proc. 14th Canadian Conference on Artificial Intelligence.*
8. Cheng, J., Greiner, R., Kelly, J., Bell, D. and Liu, W. (2002). Learning Belief Networks from Data: An Information Theory Based Approach. *Artificial Intelligence,* Vol. 137, pp 43-90.
9. Chickering, D.M. (2002). Optimal Structure Identification with Greedy Search. *Journal of Machine Learning Research,* Vol. 3, pp 507-554.
10. Chickering, D.M. and Meek, C. (2006). On the Incompatibility of Faithfulness and Monotone DAG Faithfulness. *Artificial Intelligence,* Vol. 170, pp 653-666.
11. Cooper, G.F. (1990). The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks. *Artificial Intelligence,* Vol. 42, pp 393-405.
12. Cooper, G.F. and Herskovits, E. (1992). A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning,* Vol. 9, pp 309-347. Kluwer Academic Publishers.
13. Domingos, P. & Pazzani, M. (1996). Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. *Proc. 13th International Conference on Machine Learning.*
14. Domingos, P. & Pazzani, M. (1997). On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning,* Vol. 29. Kluwer Academic Publishers.
15. Dougherty, J., Kohavi, R. and Sahami, M. (1995). Supervised and Unsupervised Discretization of Continuous Features. *Proc. 12th International Conference on Machine Learning*.
16. Friedman, N., Geiger, D. and Goldszmidt, M. (1997). Bayesian Network Classifiers. *Machine Learning,* Vol. 29, pp 131-163. Kluwer Academic Publishers, Boston.
17. Garg, A. and Roth, D. (2001) Understanding Probabilistic Classifiers. *Proc. 12th European Conference on Machine Learning.*
18. Grossman, D. and Domingos, P. (2004). Learning Bayesian Network Classifiers by Maximizing Conditional Likelihood. *Proc. 21st International Conference on Machine Learning.*
19. Heckerman, D., Geiger, D. and Chickering, D.M. (1995). Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning,* Vol. 20, pp 197-243.
20. Keogh, E. and Pazzani, M.J. (2002). Learning the Structure of Augmented Bayesian Classifiers. *International Journal on Artificial Intelligence Tools*, Vol. 11, No. 4, pp 587-601.
21. Kohavi, R., Sommerfield, D. and Dougherty, J. (1997). Data Mining using MLC++. *International Journal on Artificial Intelligence Tools,* Vol. 6, No. 4, pp 537-566.
22. Ling, C.X. and Zhang, H. (2002). The Representational Power of Discrete Bayesian Networks. *Journal of Machine Learning Research*, Vol. 3.
23. Madden, M.G. (2003). The Performance of Bayesian Network Classifiers Constructed using Different Techniques. *Proc. European Conference on Machine Learning,* Workshop on Probabilistic Graphical Models for Classification.
24. Nadeau, C. and Bengio, Y. (2000). Inference for the generalization error. *Advances in Neural Information Processing Systems 12,* MIT Press.
25. Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco.
26. Silander, T. and Myllymäki, P. (2006). A Simple Approach for Finding the Globally Optimal Bayesian Network Structure. *Proc. 22nd Conference on Uncertainty in Artificial Intelligence.*
27. Zhang, H. and Ling, C.X. (2001) An improved learning algorithm for augmented Naive Bayes. *Proc. Fifth Pacific-Asia Conference on Knowledge Discovery in Databases.*

CBR AND CLASSIFICATION

Code Tagging and Similarity-based Retrieval with myCBR

Thomas R. Roth-Berghofer and Daniel Bahls

Abstract This paper describes the code tagging plug-in *coTag*, which allows annotating code snippets in the integrated development environment eclipse. *coTag* offers an easy-to-use interface for tagging and searching. Using the similarity-based search engine of the open-source tool *myCBR*, the user can search not only for exactly the same tags as offered by other code tagging extensions, but also for similar tags and, thus, for similar code snippets. *coTag* provides means for context-based adding of new as well as changing of existing similarity links between tags, supported by *myCBR*'s explanation component.

1 A Programmer's Dilemma

During their professional life developers work in many projects, use many APIs and programming languages. Thereby, they re-encounter many tasks and problems that they have already solved in the past. But especially in the domain of programming, the accurate way of using a specific API, coding in a certain language or a certain algorithm can be hard to remember. One often remembers the fact that a similar situation has already been solved, but the respective piece of code cannot be found or is not even available on the file system anymore.

Not only one's own experience may contain the solution to a difficult situation. There is also the possibility to share experience with other developers. With the

Thomas R. Roth-Berghofer · Daniel Bahls
Knowledge Management Department, German Research Center for Artificial Intelligence DFKI GmbH, 67663 Kaiserslautern, Germany, and Knowledge-based Systems Group, Department of Computer Science, University of Kaiserslautern,
e-mail: thomas.roth-berghofer|daniel.bahls@dfki.de

advance of technologies, a developer has to become acquainted to more and more new programming techniques and modules in everyday life. The introduction to an unfamiliar technology requires a lot of time and money. Since the own experience does not contain the required knowledge, other resources must be used.

Many portals with tutorials, introductions, and best practices regarding API's and programming languages exist on the Web. But even though their quality is often good, finding them costs time, since it is awkward to retrieve via small text boxes on web pages that are neither meant for complex developer questions nor for code insertion. Anyway, the used search tool is a web browser in this case, which is not aware of your working context.

Yet another reason to share code references can be given by the need for documentation. New members of a large project team get lost quickly if they are not somehow familiar with the project. In order to understand the structure of the code, they have to pose many questions to others, or at least to themselves. Thereby, it may be helpful to have a system that is capable of providing guidance.

Delivering the right information at the right place at the right time is the main goal of knowledge management. As soon as there is a tool that helps formulating the problem quickly and easily, finds the desired piece of code reliably, and presents it after a few seconds right inside the developers working place, i.e., in his or her integrated development environment (IDE), then the requirements of good knowledge management are met.

With the evolution of the Web towards Web 2.0, one technique for information description called tagging has become very popular. Even though, tagging often lacks semantical foundations, its acceptance in social domains such as bookmark sharing[1] or photo exchange[2] among many others proves its practical usefulness. Easy-to-use interfaces allow annotating digital resources with concepts drawn from ones own personal information model (8) and intuitive navigation, due to their simple search and find algorithms.

In this paper, we strive for the goal to bring these capabilities also to the domain of programming, starting out with personal re-use of code by providing means for marking and finding relevant knowledge via tagging of code snippets and similarity-based search.

The plug-in *coTag*[3] allows annotating code snippets in the open source IDE eclipse[4]. Using the similarity-based search engine of the open-source tool *myCBR*[5] (9), the user can search not only for exactly the same tags as offered by other code tagging extensions, but also for similar tags and, thus, for similar code snippets. *coTag* offers an easy-to-use interface for tagging, searching, and context-based adding of new as well as changing of existing similarity links between existing tags.

[1] `http://del.icio.us`

[2] `http://flickr.com`

[3] `http://cotag.opendfki.de`

[4] `http://eclipse.org`

[5] `http://mycbr-project.net`

The rest of the paper is structured as follows: Section 2 gives an introduction to other approaches supporting code reuse. The details to our approach including case structure and similarity measures are elaborated in Section 3. All implementation issues such as the integration into eclipse, the integration of the CBR tool *myCBR*, and the design of the user interfaces are presented in Section 4. First evaluation results are presented in Section 5. The paper concludes with an outlook to further development plans in Section 6.

2 Related Work

Many approaches to support documentation, navigation, and reuse of software artefacts have been developed over the years. Each of them varies slightly regarding motivation and purpose. We want to introduce some related ideas in the following and make clear how they differ from our approach.

An interesting concept has been followed by (4) who built a suggestion system for the domain of software design. It applies a company's own development experience as a case base of a CBR system. Therefore, it has been integrated into the commercial UML tool Enterprise Architect[6] and assists every software designer of the belonging company in generating new UML diagrams. Although, it covers a bunch of questions respective software design, it cannot answer language specific ones, since it doesn't comprise coding details. Also, the practical usage of a foreign API cannot be explained, because its UML diagrams are not available ad hoc.

A similar approach was followed with CIAO-SI (6). This tool also is based on CBR techniques. At the beginning of a new development project CIAO-SI suggests software artefacts (models, documents, source code) that have been used in past projects. Therefore, the developer must formulate a query that consists of the respective application domain and additional software characteristics. With the use of CASE[7] tools, the resulting artefacts can be adapted to the requirements of the new project. CIAO-SI assists developers in the complete application design phase. It considers the outlines of the planned project in a macroscopic level of detail. In contrast to this, our intention is to show small code snippets or passages of source code documents in a much smaller context and to provide light-weight assistance for individual developers in day-to-day use.

Quite complementary to the above is the following approach (10). With TagSEA[8] Storey et al. aim at a better documentation and navigation of source code by enriching bookmarks with meta-data such as provenance and social tags.[9] TagSEA allows sharing of these among project teams. But their goal was to provide a more sophisticated use of bookmarks within source code, not to answer questions about

[6] `http://www.sparxsystems.de/`

[7] Computer Aided Software Engineering

[8] `http://tagsea.sourceforge.net`

[9] Their bookmarks are called *waypoints* following the metaphor used in navigation systems.