Efficient discriminative learning of Bayesian network classifier via Boosted Augmented Naive Bayes

Yushi Jing YJING@CC.GATECH.EDU
College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA

Vladimir Pavlović VLADIMIR@CS.RUTGERS.EDU
Department of Computer Science, Rutgers University, Piscataway, NJ 08854 USA

James M. Rehg REHG@CC.GATECH.EDU
College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA

Abstract

The use of Bayesian networks for classification problems has received significant recent attention. Although computationally efficient, the standard maximum likelihood learning method tends to be suboptimal due to the mismatch between its optimization criteria (data likelihood) and the actual goal for classification (label prediction). Recent approaches to optimizing the classification performance during parameter or structure learning show promise, but lack the favorable computational properties of maximum likelihood learning. In this paper we present the Boosted Augmented Naive Bayes (BAN) classifier. We show that a combination of discriminative data-weighting with generative training of intermediate models can yield a computationally efficient method for discriminative parameter learning and structure selection.

1. Introduction

A Bayesian network is an annotated directed graph that encodes the probabilistic relationships among variables of interest. Its modularity and intuitive graphical representation make it an attractive model for real world problems. A Bayesian network classifier (Friedman et al., 1997) computes the posterior of class given an instance of the attributes and predicts the class with the highest posterior probability.

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

As a special case of Bayesian networks, the naive Bayes classifier has received significant amount of attention for its simplicity and surprisingly good performance. However, the standard maximum likelihood parameter learning for naive Bayes classifiers tends to be suboptimal (Friedman et al., 1997). It optimizes the joint likelihood, rather than the conditional likelihood, a score more closely related to the classification task. The ELR algorithm (Greiner & Zhou, 2002) discriminatively trains the parameters of Bayesian network using gradient descent and line search. Although ELR algorithm tends to outperform its generative counterparts, it is computationally demanding and not practical in the presence of a large attribute space.

Under the correct model structure, the parameters that maximize the likelihood also maximize the conditional likelihood. For this reason, structure learning can potentially be used to improve the classification accuracy. However, learning an unrestricted Bayesian network fails to outperform naive Bayes on a large sample of benchmark data (Friedman et al., 1997) due to the generative structure evaluation function. Instead, Friedman et al. proposed Tree Augmented Naive Bayes (TAN), a structure learning algorithm that learns a maximum spanning tree from the attributes, but retains naive Bayes model as a part of its structure to bias towards the estimation of conditional distribution. BNC-2P (Grossman & Domingos, 2004), on the other hand, is a heuristic structure learning method with a discriminative scoring function. Although the structures in TAN and BNC-2P are selected discriminatively, the parameters are trained via ML training for computational efficiency.

Grossman et al. (Grossman & Domingos, 2004) also explored the idea of combining parameter optimiza-

tion with structure optimization. However, their experiments showed that, at least in small to medium sample size cases, combined optimization did not significantly improve the classification accuracy to justify the additional computational cost of gradient descent parameter learning.

In this work we propose the Boosted Augmented Naive Bayes classifier, and demonstrate that it is possible to efficiently combine discriminative structure learning with parameter learning to improve the classification accuracy. By constructing an ensemble Bayesian network classifier that greedily maximizes the conditional likelihood, discriminative parameter learning takes time linear in the number of attributes. Combined with an efficient TAN-like structure search algorithm to discriminatively select additional edges to naive Bayes, BAN outperforms naive Bayes, TAN, BNC-MDL, BNC-2P on a large suite of benchmark datasets. We also demonstrate that BAN is less expensive computationally than many alternative methods with competitive classification performance.

2. Bayesian Network Classifier

A Bayesian network B is a directed acyclic graph that encodes a joint probability distribution over a set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_N\}$. It is defined by the pair $B = \{G, \theta\}$. G is the structure of the Bayesian network. θ is the vector of parameters that quantifies the probabilistic model. B represents a joint distribution $P_B(X)$, factored over the structure of the network where $P_B(X) = \prod_{i=1}^{N} P_B(X_i|Pa(X_i)) = \prod_{i=1}^{N} \theta_{X_i|Pa(X_i)}$. We set $\theta_{x_i|Pa(x_i)}$ equal to $P_B(x_i|Pa(x_i))$ for each possible value of X_i and $Pa(X_i)$[1]. For notational simplicity, we define a one-to-one relationship between the parameter θ and the entries in the local Conditional Probability Table. Given a set of i.i.d. training data $D = \{x^1, x^2, x^3, \ldots, x^M\}$, the goal of learning a Bayesian network B is to find a $\{G, \theta\}$ that accurately models the distribution of the data. The selection of θ is known as parameter learning and the selection of G is known as structure learning.

The goal of a Bayesian network classifier is to correctly predict the label for class variable $X_c \in \mathbf{X}$ given a vector of attributes $X_a = \mathbf{X} \backslash X_c$. A Bayesian network classifier represents a model of the joint distribution $P(X_c, X_a)$ and converts it to conditional distribution $P(X_c|X_a)$. Class label predictions can be obtained by applying an estimator such as MAP to the conditional distribution.

[1]We use capital letters to represent random variable(s) and lowercase letters to represent their corresponding instantiations. Subscripts are used as variable indices and superscripts are used to index the training data. $Pa(X_i)$ represents the parent node of X_i and $Pa^j(X_i)$ is the jth instantiation of $Pa(X_i)$ in the training data. In this paper, we assume all of the variables are discrete and fully observed in the training data.

3. Parameter Learning

The Maximum Likelihood (ML) method is one of the most commonly used parameter learning techniques. It chooses the parameter values that maximize the Log Likelihood (LL) score, a measure of how well the model represents the data. Given a set of training data D with M samples and a Bayesian Network structure G with N nodes, the LL score is decomposed as:

$$\mathrm{LL_G}(\theta|D) = \sum_{i=1}^{M} \log P_\theta(D^i) = \sum_{i=1}^{M} \sum_{j=1}^{N} \log \theta_{x_j^i|Pa(x_j)^i}. \quad (1)$$

$\mathrm{LL}_G(\theta|D)$ is maximized by simply setting each parameter $\theta_{x_j|Pa(x_j)}$ to $\widehat{P}_D(x_j|Pa(x_j))$, the empirical distribution of the data D. For this reason, ML parameter learning is computationally efficient and very fast in practice.

However, the goal of a classifier is to accurately predict the label given the attributes, a function that is directly tied to the estimation of the conditional likelihood. Instead of maximizing the LL score, we would prefer to maximize the Conditional Log Likelihood (CLL) score. As pointed out in (Friedman et al., 1997), the LL score factors as $\mathrm{LL_G}(\theta|D) = \mathrm{CLL_G}(\theta|D) + \sum_{i=1}^{M} \log P_\theta(x_a^i)$, where

$$\mathrm{CLL_G}(\theta|D) = \sum_{i=1}^{M} \log P_\theta(x_c^i|x_a^i) \quad (2)$$

$$= M \sum_{\substack{x_a \in X_a \\ x_c \in X_c}} \widehat{P}_D(x_c x_a) \log P_\theta(x_c|x_a). \quad (3)$$

Equation 3 is maximized when $P_\theta(x_c|x_a) = \widehat{P}_D(x_c|x_a)$. However, for generative model such as Bayesian network, $P_\theta(x_c|x_a)$ can not be expressed as a function of model parameters in log linear form, thus it does not have closed a form solution. A direct optimization approach requires computationally expensive numerical techniques. For example, the ELR method of (Greiner & Zhou, 2002) uses gradient descent and line search to directly maximize the CLL score. However, this approach is unattractive in the presence of a large feature space, especially when used in conjunction with structure learning.

Table 1. Boosted Parameter learning algorithm.

1. Given a base structure G and the training data D, where M is the number of training cases. $D = \{x_c^1 x_a^1, x_c^2 x_a^2, \ldots, x_c^M x_a^M\}$ and $x_c \in \{-1, 1\}$.
2. Initialize training data weights with $w_i = 1/M, i = 1, 2, \ldots, M$
3. Repeat for $k = 1, 2, \ldots$
   - Given G, θ_k is learned through ML parameter learning on the weighted data D_k.
   - Compute the weighted error, $err_k = E_w[1_{x_c \neq f_{\theta_k}(x_a)}]$, $\beta_k = 0.5 \log \frac{1-err_k}{err_k}$
   - Update weights $w_i = w_i \exp\{-\beta_k x_c^i f_{\theta_k}(x_a^i)\}$ and normalize.
4. Ensemble output: sign $\sum_k \beta_k f_{\theta_k}(x_a)$

4. Boosted Parameter Learning

4.1. Ensemble Model

Instead of maximizing the CLL score for a single Bayesian network model, we are going to take the ensemble approach and maximize the classification performance of the ensemble Bayesian network classifier.

Given the class x_c and the attributes x_a, an ensemble model has the general form: $F_{x_c}(x_a) = \sum_{k=1}^{K} \beta_k f_{k,x_c}(x_a)$, where $f_{k,x_c}(x_a)$ is the classifier confidence on selecting label x_c given x_a, and β_k is its corresponding weight. In the case where $x_c \in \{-1, 1\}$, $f_{k,x_c}(x_a)$ is typically defined as the following: $f_{k,x_c}(x_a) = x_c f_k(x_a)$, where $f_k(x_a)$ is the output of each classifier given x_a. The conditional probability distribution given the additive model F can be expressed as:

$$P_F(x_c|x_a) = \frac{\exp\{F_{x_c}(x_a)\}}{\sum_{x_c' \in X_c} \exp\{F_{x_c'}(x_a)\}}. \quad (4)$$

In binary classification, Equation 4 is then updated as:

$$\begin{aligned} P_F(x_c|x_a) &= \frac{\exp\{x_c F(x_a)\}}{\exp\{F(x_a)\} + \exp\{-F(x_a)\}} \\ &= \frac{1}{1 + \exp\{-2x_c F(x_a)\}}. \end{aligned} \quad (5)$$

Similar to Equation 3, the negative CLL score for the ensemble Bayesian network classifier can be defined as:

$$\begin{aligned} -\mathrm{CLL}_F(F|D) &= \sum_{i=1}^{M} \log \frac{1}{P_F(x_c^i|x_a^i)} & (6) \\ &= M \sum_{\substack{x_a \in X_a \\ x_c \in X_c}} \widehat{P}_D(x_c x_a) \log \frac{1}{P_F(x_c|x_a)} & (7) \end{aligned}$$

4.2. Exponential Loss Function as an upper bound on the negative CLL score

As an alternative to the CLL score, we are proposing to minimize the classification error for binary ensemble classifier via the following loss function.

$$Loss_F = \sum_{i=1}^{M} \Theta(-x_c^i F(x_a^i)), \Theta(z) = \begin{cases} 0 & \text{for } z < 0 \\ 1 & \text{otherwise} \end{cases} \quad (8)$$

$Loss_F$ is simply the number of incorrectly predicted class labels in the training data. An upper bound on Equation 8 is given by the following exponential loss function (Friedman et al., 2000):

$$ELF_F = \sum_{i=1}^{M} \exp\{-x_c^i F(x_a^i)\} \quad (9)$$

Solving for $x_c F(x_a)$ in Equation 5 and combining with Equation 9, we have

$$\begin{aligned} \mathrm{ELF}_F &= \sum_{i=1}^{M} \exp\left\{\frac{1}{2} \log \frac{1 - P_F(x_c^i|x_a^i)}{P_F(x_c^i|x_a^i)}\right\} & (10) \\ &= \sum_{i=1}^{M} \sqrt{\frac{1 - P_F(x_c^i|x_a^i)}{P_F(x_c^i|x_a^i)}} \\ &= M \sum_{\substack{x_a \in X_a \\ x_c \in X_c}} \widehat{P}_D(x_c x_a) \sqrt{\frac{1}{P_F(x_c|x_a)} - 1} & (11) \end{aligned}$$

Equation 11 simply leads to a loss function that uses the square root of the inverse conditional distribution of the true training sequence, which can be readily proven as an upper bound for negative CLL score in Equation 7.

4.3. Boosted Parameter Learning

An ensemble Bayesian network classifier takes the form $F_{\theta,\beta}$ where θ is a collection of parameters in the Bayesian network model and β is the vector of hypothesis weights. We want to minimize $\mathrm{ELF}_{\theta,\beta}$ of the ensemble Bayesian network classifier as an alternative way to maximize the CLL score. We used Discrete AdaBoost algorithm, which is proven to greedily and approximately minimize the exponential loss function in Equation 9 (Friedman et al., 2000).

At each iteration of boosting, the weighted data uniquely determines the parameters for each Bayesian network classifier θ_k and hypothesis weights β_k via efficient ML parameter learning. The algorithm is shown in Table 1.

There is no guarantee that AdaBoost will find the global minimum of the ELF. In spite of these issues, boosted classifiers tend to produce excellent results in practice. Boosted Naive Bayes (BNB) has been previous shown to improve the classification accuracy of naive Bayes (Elkan, 1997) (Ridgeway et al., 1998). In the next section, we demonstrate that BNB outperforms naive Bayes and TAN on a large set of benchmark data.

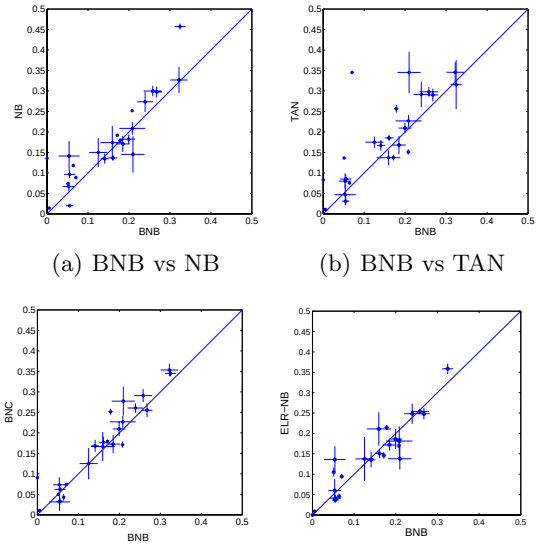
4.4. Computational Complexity of BNB

Given a naive Bayesian network with N attributes and training data with M samples, the ML training complexity is $O(NM)$, optimal when every attribute is observed and used for classification. Parameter boosting on naive Bayes takes $O(NMT)$ where T is the number of iterations of boosting. In our experiments, boosting seems to give a good performance with a constant number (10-30) of iterations regardless of the number of attributes. Therefore, the training complexity for BNB is essentially $O(NM)$. This is consistent with the finding of Elkan (Elkan, 1997). In comparison to the ELR algorithm which uses gradient descent to maximize CLL score, BNB is efficient to train and simple to implement.

4.5. Experiments

We evaluated the performance of BNB on 23 datasets from the UCI repository (Blake & Merz, 1998) and two artificial data sets, Corral and Mofn, designed by John and Kohavi. The code is written on top of the BNT toolkit by Kevin Murphy. For binary classification, we used Discrete AdaBoost for parameter boosting, for multi-class problem, we used AdaBoost.MH. Table 3 in Page 7 lists the average testing error for BNB and other Bayesian network classifiers including our novel algorithm BAN, which is introduced in Section 5. Figure 1(a) to 1(d) presents the average testing errors and their corresponding one-standard-deviation bars for competing Bayesian network classifiers. In Figure 1, points above the line $y = x$ correspond to data sets for which BNB outperforms the competing algorithm. The average testing error is shown in the caption. We applied pairwise t-test on the 25 pairs of average testing errors for competing algorithms to obtain confidence scores.

Figure 1(a) and 1(b) show that BNB has lower average



(a) BNB vs NB (b) BNB vs TAN

(c) BNB vs BNC-2P (d) BNB vs ELR-NB

Figure 1. Scatter plots for experiments on 25 sets of UCI and artificial benchmark data. The average testing error for each methods are: BNB (0.151), NB (0.173), TAN (0.184), BNC-2P (0.164), ELR-NB(0.161)

testing error than NB ($p < 0.02$) and TAN ($p < 0.02$). Also, we find BNB to slightly outperform the BNC-2P discriminative structure learning algorithm on average testing error ($p < 0.04$).

We also compared BNB to ELR-NB, a naive Bayes trained using ELR algorithm. The performance scores for ELR-NB were taken from the auxiliary material published online with (Greiner & Zhou, 2002). From the graph, it is reasonable to conclude that BNB is comparable with ELR-NB on this set of benchmark data. However, BNB has computational complexity asymptotically equivalent to naive Bayes, making it an efficient and simple alternative to ELR-NB.

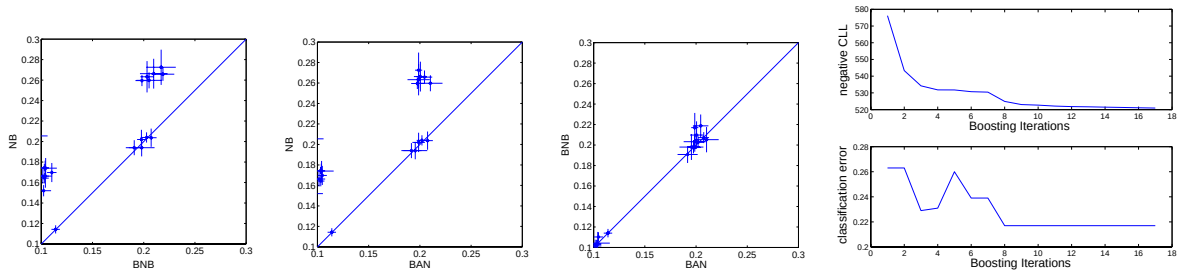
Given the excellent performance of BNB, it is natural to ask whether it could be combined with structure learning to further improve the classification performance. In the next section, we introduce BAN, a novel and efficient discriminative structure learning algorithm.

5. Boosted Augmented Naive Bayes

Though the training complexity of parameter boosting is within a constant factor of ML learning, combining parameter boosting with exhaustive structure search is still impractical. Even with constrained search space, hill-climbing (Heckerman, 1999) search and K-2 (Cooper & Herskovits, 1992) algorithm could

Table 2. Boosted Augmented Naive Bayes.

1. Given the training data D with N attributes, construct a TAN structure, G_{TAN}.
2. Initially set G_{BAN} = naive Bayes, $\text{CLL}_{best} = -\inf$.
3. For $k = 1$ to $N - 1$
   - Parameter boosting using G_{BAN} as base structure.
   - Evaluate the CLL score for the current G_{BAN}, terminate if the new CLL score is less than CLL_{best}.
   - else, update CLL_{best}. Remove the edge $\{X_{a_i} X_{a_j}\}$ containing the largest conditional mutual information $I_p(X_{a_i}; X_{a_j} | X_c)$ from G_{TAN} and add it to G_{BAN}.



(a) BNB vs NB (b) BAN vs NB (c) BAN vs BNB (d) Decreasing negative CLL score

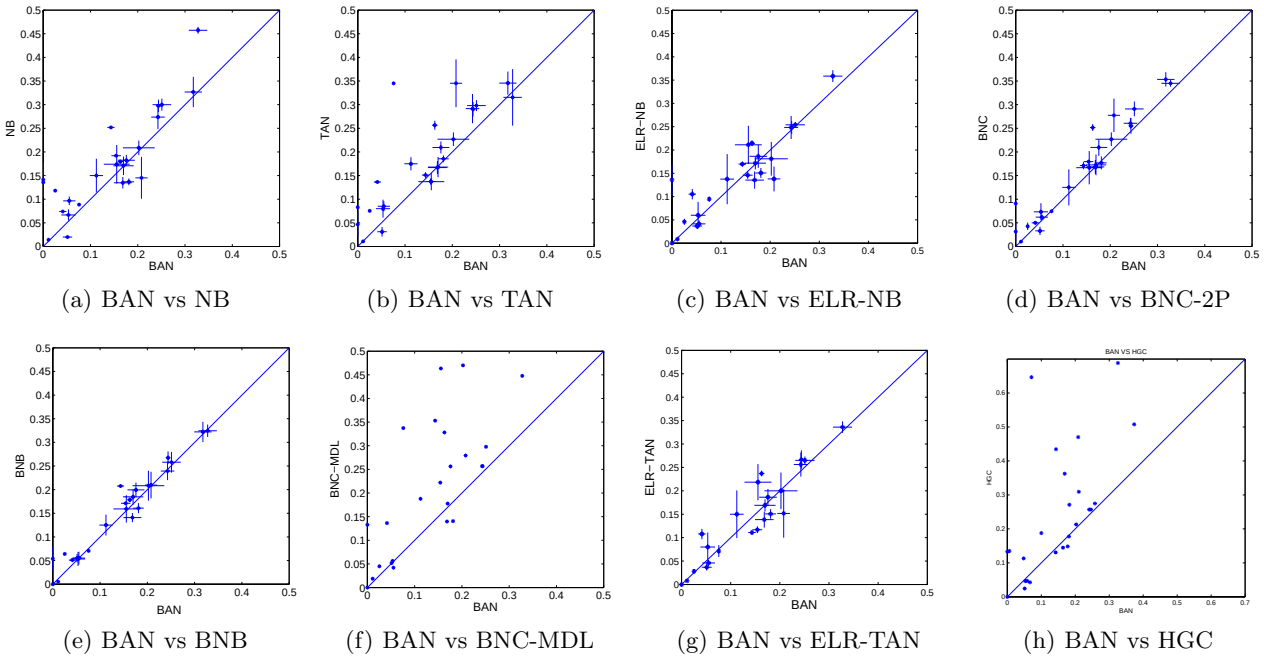
Figure 2. Test error on simulated experiment. We varied the number of nodes in the chain-structure Bayesian network and their parameter values to generate different distributions (25 sets). Each point in the graph represents the classification accuracy for one particular model distribution. BNB and BAN outperforms NB in 19 out of the 25 simulated datasets. In the remaining 6 datasets, the suboptimal posterior estimation by naive Bayes did not result in label prediction error.

still search through a high order polynomial number of structures. On the other hand, Tree Augmented Bayesian Network (TAN) (Friedman et al., 1997), an extension of work by (Chow & Liu, 1968), supports efficient learning by limiting the number of parents per attribute to two. TAN augments a standard NB classifier by adding up to $N - 1$ additional edges between attributes. The additional edges are constructed from a maximal weighted spanning tree with attributes as vertices. The weights are defined as the conditional mutual information between two attributes X_{a_i}, X_{a_j} given the class node X_c. TAN learning algorithm constructs the optimal tree-augmented network B_T that maximizes $\text{LL}(B_T|D)$. However, the TAN model adds a fixed number of edges regardless of the distribution of the training data. If we can find a simpler model to describe the underlying conditional distribution, then there is usually less chance of over-fitting.

Our BAN learning algorithm extends the TAN approach using parameter boosting. Starting from a naive Bayes, at iteration k, BAN greedily augments the naive Bayes with k edges with the highest conditional mutual information from TAN. We call the resulting structure BAN^k. We then maximize the CLL score of ensemble BAN^k classifier with parameter boosting. BAN terminates when the added edge does not improve the CLL score. Since TAN contains $N - 1$ augmenting edges, BAN in worst case evaluates $N - 1$ structures. However, BAN usually terminates after adding 0 to 5 edges into naive Bayes. Therefore, BAN learning algorithm is very efficient.

The algorithm is shown in Table 2. The calculation of CLL score at each iteration can be also done on hold-out set when the number of training samples is sufficient. The step 1 in BAN algorithm has computational complexity of $O(N^2 M)$, where N is the number of attributes and M is the amount of training data (Friedman et al., 1997). Since we only add a maximum of $N - 1$ edges into the network, step 2-4 has worst case complexity of $O(N^2 M)$. Thus BAN has $O(N^2 M)$ complexity.

BAN learning algorithm searches and evaluates only a small number of structures, much less than competing algorithms like BNC-2P and BNC-MDL. As a result, the base Bayesian network structure constructed from BAN contains fewer edges than other competing structure learning algorithms.



(a) BAN vs NB (b) BAN vs TAN (c) BAN vs ELR-NB (d) BAN vs BNC-2P

(e) BAN vs BNB (f) BAN vs BNC-MDL (g) BAN vs ELR-TAN (h) BAN vs HGC

Figure 3. Scatter plots for experiments on 25 benchmark UCI and artificial datasets.

6. Experiments

6.1. Experiment on simulated data

We show that when the structure is incorrect, BNB and BAN algorithm can significantly outperform their generative counterparts. We generated a collection of data from binary chain-structured Bayesian network where $Pa(X_i) = \{X_{i-1}\}$. The class node X_1 is the root of the chain. We varied the number of attributes and conditional probability tables to generate datasets with different posterior distribution. Since the attributes are highly correlated, naive Bayes can sometimes give a suboptimal classification boundary.

We present the average testing error with its one-standard-deviation bar in Figure 2. Figure 2(a) and 2(b) show that BNB and BAN outperforms NB with ($p < 0.005$). We want to point out that in 6 out of the 25 datasets, the suboptimal posterior estimation by naive Bayes did not result in label prediction error. In those datasets, NB, BNB and BAN have similar testing error.

As shown in Figure 2(c), the average testing error for BNB is only slightly higher than that of BAN. This is largely because BNB achieved optimal Bayes error in 20 out of the 25 datasets due to the simplicity of our true model. BAN has comparable testing accuracy with BNB in those 20 datasets and has lower average testing error (difference of 2%) than BNB in the remaining 5 datasets. In those datasets, BAN selected edges that corresponds to the true model structure (between adjacent attributes). Next section will show that in real-world datasets, where attributes often have complex and strong dependence relationship, BAN outperforms BNB by exploring the structures in the problem domain.

Figure 2(d) shows the decrease in negative CLL score and testing error in each iteration of parameter boosting. In this dataset, BNB achieves the optimal Bayes error after 8 iterations but the negative CLL score continues to decrease.

6.2. Experiments on UCI datasets

We used the same UCI datasets and evaluation procedures as in Section 4.5 to compare the accuracy of BAN with competing algorithms. We did our own experiments on BAN, BNB, BNC-2P and TAN, and used the performance results for BNC-MDL, ELR, C4.5 and HGC from (Grossman & Domingos, 2004) (Greiner & Zhou, 2002). HGC (Heckerman, 1999) is a generatively trained unrestricted Bayesian network. ELR-NB and ELR-TAN are Bayesian network learned using the ELR algorithm. The scatter plots are shown in Figure 3 and the average testing error is shown in Table 3.

Figure 3(a) and 3(b) show that the average testing error for BAN algorithm is significantly lower than naive

Table 3. Testing error for 25 UCI datasets

Name	**BAN**	**BNB**	**TAN**	**NB**	**BNC$_{2P}$**	**BNC$_{MDL}$**	**ELR$_{NB}$**	**ELR$_{TAN}$**	**C4.5**	**HGC**
australian	.1812	.1609	.1855	.1368	.1768	.1405	.1488	.1723	.1510	.1445
breast	.0513	.0549	.0312	.0200	.0330	.0518	.0339	.0351	.0610	.0245
chess	.0253	.0640	.0753	.1180	.0428	.0450	.0600	.0375	.0050	.0469
cleve	.1758	.1995	.2095	.1825	.2095	.2563	.1660	.0375	.2060	.2129
corral	.0000	.0538	.0468	.1412	.0314	.0000	.1273	.0771	.0150	.0000
crx	.1684	.1408	.1669	.1347	.1684	.1397	.1505	.1603	.1390	.1308
diabetes	.2438	.2675	.2903	.2974	.2553	.2569	.2419	.2384	.2590	.2569
flare	.1698	.1848	.1679	.1707	.1726	.1776	.1803	.1780	.1730	.1776
german	.2510	.2580	.2980	.3000	.2910	.2977	.2456	.2409	.2710	.2748
glass	.3175	.3221	.3456	.3268	.3535	.6884	.4220	.5018	.4070	.6884
glasss2	.2023	.2083	.2269	.2087	.2269	.4701	.1938	.2249	.2390	.4701
heart	.1556	.1593	.1371	.1741	.1667	.4635	.1550	.1847	.2180	.1484
hepatitis	.1125	.1250	.1750	.1500	.1250	.1877	.1294	.1302	.1750	.1877
iris	.0533	.0533	.0800	.0667	.0733	.0563	.0485	.0763	.0400	.0427
letter	.1433	.2076	.1511	.2520	.1712	.3530	.3068	.1752	.1220	.3092
lymphography	.2078	.2097	.3453	.1452	.2775	.2794	.1470	.1784	.2160	.3624
mofn-3-7-10	.0000	.0000	.0830	.1357	.0908	.1328	.1367	.0000	.1600	.1328
pima	.2427	.2394	.2916	.2737	.2606	.2569	.2505	.2384	.2590	.2569
satimage	.1543	.1712	.1374	.1920	.1795	.2220	.1730	.1420	.1770	.2710
segment	.0415	.0510	.1364	.0740	.0500	.1364	.0701	.0571	.0820	.1130
shuttle-small	.0113	.0052	.0108	.0142	.0102	.0186	.0083	.0052	.0060	.1349
soybean-large	.0758	.0704	.3451	.0885	.0746	.3373	.0920	.0663	.0890	.6466
vehicle	.3276	.3246	.3154	.4573	.3452	.4478	.3453	.2727	.3170	.5077
vote	.0552	.0552	.0851	.0966	.0621	.0420	.0370	.0487	.0530	.0463
waveform	.1630	.1785	.2566	.1795	.2516	.3281	.1772	.2534	.3490	.4345
Average	.1412	.1506	.1837	.1734	.1640	.2314	.1613	.1554	.1676	.2409

Bayes ($p < 0.01$) and TAN ($p < 0.01$). BAN also outperforms BNC-2P ($p < 0.005$) in Figure 3(d). We did not have access to variance data for BNC-MDL, HGC and C4.5. As shown in Figure 3(c) 3(g) and Table 3, BAN has comparable classification accuracy as ELR-NB and ELR-TAN.

As shown in Figure 3(e), the average testing errors for BAN and BNB are 0.141 and 0.151 respectively. This difference is significant with confidence $p < 0.029$. BAN has lower average testing error (difference of 0.5% - 5%) than BNB in 16 out of the 25 datasets. BNB is better in 6 (difference of 0.5% - 2%) and they tie in 3. Since BAN generalizes BNB, in several datasets (MOFN, IRIS), the structure chosen by BAN is very similar to BNB (with 0 and 1 augmented edges). BAN is more beneficial in datasets where the conditional dependencies among attributes are strong and complex (CORRAL). This is an interesting result since it shows that combining discriminative structure learning with parameter optimization seems to improve classification accuracy.

7. Discussion and Related Work

The above experiments demonstrated that boosted parameter optimization in conjunction with greedy structure optimization can improve the classification performance. It is interesting to note that unlike the experimental results in combining ELR with structure learning (Grossman & Domingos, 2004), we find significant benefit in combining parameter boosting with structure learning. We would like to investigate the exact explanation for the benefit in our future research.

We attribute the success of our approach to the following reasons. First, BAN takes advantage of AdaBoost's resistance to over-fitting (Schapire & Singer, 1998) and the variance reduction and bias reduction property of the ensemble classifiers (Webb & Zheng, 2004). Also, as a result of the parameter boosting, the base Bayesian network classifier constructed by BAN is simpler than BNC-2P and TAN. In our experiments, BAN adds 0 to 5 edges to the naive Bayes while BNC-2P typically adds 4 to 16 edges. If both Bayesian networks model the underlying conditional distribution equally well, a simpler structure is usually preferred over a more densely connected one. We believe that the primary advantage of boosted parameter learning is its simplicity and computational efficiency, coupled with its good performance in practice. Its use of weighted maximum likelihood parameter learning uniquely determines the parameters of the Bayesian network, providing an efficient mechanism for discriminative training.

Elkan (Elkan, 1997) demonstrated the excellent classification performance of boosted Naive Bayes and its

efficient training mechanism. We build on this work by extending the use of boosting to structure learning. We also include a more thorough comparison between BNB and competing methods. Greiner and Zhou (Greiner & Zhou, 2002) proposed ELR algorithm to directly maximize the CLL score of Bayesian network via gradient descent and line search. Therefore ELR-NB is essentially a logistic regression model. Ng and Jordan (Ng & Jordan, 2002) compared the classification performance of naive Bayes and logistic regression and demonstrated that naive Bayes usually performs better in very small sample data. However for the size of dataset typically collected from real world problems, their results and those of (Grossman & Domingos, 2004) all support our observation that discriminative training methods are preferred. (Webb et al., 2005) uses ensemble techniques (AODE model) to discriminatively improve naive Bayes by averaging a class of 1-dependence models. BAN differs from AODE work by using a fixed structure for each classifier and uses AdaBoost to train the ensemble.

8. Conclusion and Future work

This paper showed that an effective Bayesian network classifier can be constructed by parameter boosting coupled with discriminative structure learning. We demonstrated that the resulting classifier BAN is easy to implement, efficient to train and in our experiments outperforms naive Bayes, TAN, HGC, BNC-2P and BNC-MDL. In future work, we plan to apply BAN to a wider range of classification problems, investigate further the reasons behind BAN's excellent classification performance and extend this work to other generative models, including Dynamic Bayesian network and Markov models.

9. Acknowledgement

We thank Matt Mullin for the fruitful discussion and suggestion to use chain-structured model to illustrate the benefit of BNB and BAN. We also thank the reviewers for their comments. This material is based upon work supported in part by the National Science Foundation under NSF Grant IIS-0205507.

References

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Chow, C. K., & Liu, C. N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, *IT-14*, 462–467.

Cooper, G. F., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, *9*, 309–347.

Elkan, C. (1997). *Boosting and naive bayesian learning* (Technical Report). Department of Computer Science and Engineering, University of California, San Diego.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, *38*, 337–374.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*, 131–163.

Greiner, R., & Zhou, W. (2002). Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Proceedings of annual meeting of the American Association for Artificial Intelligence* (pp. 167–173).

Grossman, D., & Domingos, P. (2004). Learning bayesian network classifiers by maximizing conditional likelihood. *Proc. 21st International Conference on Machine Learning* (pp. 361–368). Banff, Canada: ACM Press.

Heckerman, D. (1999). A tutorial on learning with bayesian networks. 301–354.

Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Proc. 14th Conference on Advances in Neural Information Processing Systems* (pp. 841–848). Cambridge, MA: MIT Press.

Ridgeway, G., Madigan, D., Richardson, T., & O'Kane, J. (1998). Interpretable boosted naive bayes classification. *Proceedings Fourth International Conference on Knowledge Discovery and Data Mining.*

Schapire, R. E., & Singer, Y. (1998). Improved boosting algorithms using confidence-rated predictions. *Proc. 11th Annual Conference on Computational Learning Theory.*

Webb, G., & Zheng, Z. (2004). Multistrategy ensemble learning: Reducing error by combining ensemble learning techniques. *IEEE Transactions on Knowledge and Data Engineering*, *16*, 980–991.

Webb, G. I., Boughton, J., & Wang, Z. (2005). Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, *58*, 5–24.