Mach Learn
DOI 10.1007/s10994-008-5065-7

Boosted Bayesian network classifiers

Yushi Jing · Vladimir Pavlović · James M. Rehg

Received: 16 December 2005 / Revised: 21 November 2007 / Accepted: 5 May 2008

Abstract The use of Bayesian networks for classification problems has received a significant amount of 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 of classification (label prediction accuracy). Recent approaches to optimizing classification performance during parameter or structure learning show promise, but lack the favorable computational properties of maximum likelihood learning. In this paper we present boosted Bayesian network classifiers, a framework to combine discriminative data-weighting with generative training of intermediate models. We show that boosted Bayesian network classifiers encompass the basic generative models in isolation, but improve their classification performance when the model structure is suboptimal. We also demonstrate that structure learning is beneficial in the construction of boosted Bayesian network classifiers. On a large suite of benchmark data-sets, this approach outperforms generative graphical models such as naive Bayes and TAN in classification accuracy. Boosted Bayesian network classifiers have comparable or better performance in comparison to other discriminatively trained graphical models including ELR and BNC. Furthermore, boosted Bayesian networks require significantly less training time than the ELR and BNC algorithms.

Keywords Bayesian network classifiers · AdaBoost · Ensemble models · Structure learning

Editor: Zoubin Ghahramani.

Y. Jing (✉) · J.M. Rehg
College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA
e-mail: yjing@cc.gatech.edu

J.M. Rehg
e-mail: rehg@cc.gatech.edu

V. Pavlović
Department of Computer Science, Rutgers University, Piscataway, NJ 08854, USA
e-mail: vladimir@cs.rutgers.edu

1 Introduction

A Bayesian network is an annotated directed graph that encodes the probabilistic relationships among variables of interest (Pearl 1988). The explicit representation of probabilistic relations can exploit the structure of the problem domain, making it easier to incorporate domain knowledge in the model design. In addition, a Bayesian network has a modular and intuitive graphical representation which is very beneficial in decomposing a large and complex problem representation into several smaller, self-contained models for tractability and efficiency. Furthermore, the probabilistic representation combines naturally with the EM algorithm to address problems with missing data. These advantages of Bayesian networks and generative models as a whole make them an attractive modeling choice.

In many problem domains where a Bayesian network is applicable and desirable, we may want to infer the label(s) for a subset of the variables (class variables) given an instantiation of the rest (attributes). Bayesian network classifiers (Friedman et al. 1997) model the conditional distribution of the class variables given the attributes and predict the class with the highest conditional probability. Bayesian network classifiers have been applied successfully in many application areas including computational molecular biology (Segal et al. 2003; Pavlović et al. 2002; Jojic et al. 2004), computer vision (Torralba et al. 2004; Rehg et al. 2003; Schneiderman 2004), relational databases (Friedman et al. 1999), text processing (Cutting et al. 1992; McCallum et al. 2000; Lafferty et al. 2001), audio processing (Rabiner and Juang 1993) and sensor fusion (Pavlović et al. 2000). Its simplest form, the naive Bayes classifier, has received a significant amount of attention (Langley et al. 1992; Duda and Hart 1973; Ng and Jordan 2002).

However, standard Maximum Likelihood (ML) parameter learning in Bayesian network 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. Unlike the joint likelihood, however, the conditional likelihood cannot be expressed in a log linear form, therefore no closed form solution is available to compute the optimal parameters. Recently there has been substantial interest in discriminative training of generative models coupled with advances in discriminative optimization methods for complex probabilistic graphical models (Greiner and Zhou 2002; McCallum et al. 2000; Lafferty et al. 2001; Chelba and Acero 2004; Altun et al. 2003; Taskar et al. 2004).

If the selected model structure contains the structure from which the data is generated, the parameters that maximize the likelihood also maximize the conditional likelihood (see p. 5). For this reason, structure learning (Cooper and Herskovits 1992; Heckerman 1995; Friedman and Koller 2000; Chickering and Heckerman 1997; Heckerman et al. 1995; Lam and Bacchus 1992) can potentially be used to improve the classification accuracy. However, experiments show that learning an unrestricted Bayesian network fails to outperform naive Bayes in classification accuracy on a large sample of benchmark data (Friedman et al. 1997; Grossman and Domingos 2004). Friedman et al. (1997) attribute this to the mismatch between the structure selection criteria (data likelihood) and the actual goal for classification (label prediction accuracy). They proposed Tree Augmented Naive Bayes (TAN), a structure learning algorithm that learns a maximum spanning tree from the attributes, but incorporates a naive Bayes model as part of its structure to bias the model towards the estimation of the conditional distribution. On the other hand, Keogh and Pazzani (1999) proposed Augmented Naive Bayes (ANC) algorithm that uses the 0-1 loss function as the optimization criteria in its heuristic search for the best k-tree Bayesian network classifiers. Recently proposed BNC-2P (Grossman and Domingos 2004) uses the conditional likelihood as optimization criteria in its structure search and is shown to outperform naive Bayes, TAN, and

generatively trained unrestricted networks. Although the structures in TAN and BNC-2P are selected discriminatively, the parameters are trained via ML training for computational efficiency.

In this work, we propose a new approach to the discriminative training of Bayesian networks. Similar to a standard boosting approach, we recursively form an ensemble of classifiers. However in contrast to situations where the weak classifiers are trained discriminatively, the "weak classifiers" in our method are trained generatively to maximize the likelihood of weighted data. Our approach has two benefits. First, ML training of generative models is dramatically more efficient computationally than discriminative training. By combining maximum likelihood training with discriminative weighting of data, we obtain a computationally efficient method for the discriminative training of a general Bayesian network. Second, our classifiers are constructed from generative models. This is important in many practical problems where domain knowledge can be readily encoded in the structure of the generative model.

This paper makes five contributions:

1. We introduce Boosted Augmented Naive Bayes (bAN), an approach to improving the classification accuracy of generative models. We demonstrate that bAN's classification accuracy on a large suite of benchmark datasets is comparable or superior to competing methods such as naive Bayes, TAN and ELR.
2. We investigate alternative training strategies for constructing boosted Bayesian network classifiers and introduce a novel algorithm $b\text{AN}_{mix}$ as a more efficient alternative to bAN due to its heterogeneous assembly of structures.
3. We study the manner in which the complexity of the Bayesian network structure affects the classification accuracy of the final boosted ensemble by comparing bAN against bBNC-2P, bTAN, and other alternative boosted Bayesian network classifiers. Our experiments highlight the importance of structure learning.
4. We demonstrate that bAN and $b\text{AN}_{mix}$ are significantly faster to train than competing algorithms including ELR and BNC-2P. To ensure fairness, we implemented bNB, bAN, TAN, BNC-2P, bTAN, bBNC-2P and ELR algorithms such that they share a common code base in C++. We optimized the common code base for efficiency. In addition, we optimized the published BNC-2P method to improve its training speed.
5. We describe a multiclass extension of the bAN algorithm based on AdaBoost.MH and AdaBoost.M2. The results further demonstrate the advantages of structure selection.

A preliminary version of a portion of the results in this paper appeared in (Jing et al. 2005). This paper differs from (Jing et al. 2005) in the following areas. We provide a detailed comparison against five alternative boosted Bayesian network learning algorithms to demonstrate that structure learning is crucial to the ensemble classification accuracy. We also investigate the effect of combining heterogeneous mixture of Bayesian network structures into one ensemble and propose $b\text{AN}_{mix}$ as a more efficient alternative to bAN. We provide additional experiments with simulated datasets (generated from more complex structures) to highlight the capability of bAN to explore the structure of the data distribution. We supplement the analysis of training cost in (Jing et al. 2005) with a thorough empirical evaluations and comparisons (Sect. 7, Fig. 9 to Fig. 11). We include both AdaBoost.MH and AdaBoost.M2 variations in our analysis of multi-class bAN algorithm. Finally, we provide a graphical interpretation of boosted Bayesian network classifier in Fig. 12 in the appendix.

This paper is divided into 8 sections. Sections 1 through 3 review the formal notations of Bayesian networks and parameter learning methodologies. Section 4 introduces AdaBoost as an effective way to improve the classification accuracy of naive Bayes. Section 5 extends this work to structure learning and proposes the bAN and $b\text{AN}_{mix}$ structure learning

algorithm. Sections 6 and 7 contain the experiments and analysis for boosted naive Bayes, bAN and $b\text{AN}_{mix}$ structure learning algorithm. The last three sections contain related works, conclusions and acknowledgments.

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 $\mathbf{X} = \{X_1, X_2, \ldots, X_N\}$ (Pearl 1988). 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 each possible instantiation of the parent of X_i, $Pa(X_i)$.[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 $X_c \in \mathbf{X}$ given a vector of attributes $X_a = \mathbf{X} \backslash X_c$. A Bayesian network classifier models the joint distribution $P(X_c, X_a)$ and converts it to conditional distribution $P(X_c|X_a)$. Prediction for X_c can be obtained by applying an estimator to the conditional distribution. For example, Maximum a Posteriori estimator (MAP) can be used to select the label associated with the highest conditional probability.

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:

$$\text{LL}_\text{G}(\theta|D) = \sum_{j=1}^{M} \log P_\theta(D^j) = \sum_{j=1}^{M} \sum_{i=1}^{N} \log \theta_{x_i^j|pa(x_i)^j}$$

[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, $pa(X_i)$ represents an instantiation of $Pa(X_i)$ and $pa(X_i)^j$ is the instantiated value of $Pa(X_i)$ in the j-th training data. In this paper, we assume all of the variables are discrete and fully observed in the training data.

$$= M \sum_{i=1}^{N} \sum_{\substack{x_i \in X_i \\ pa(x_i) \in Pa(x_i)}} \widehat{P}_D(x_i, pa(x_i)) \log \theta_{x_i|pa(x_i)}. \quad (1)$$

$\mathrm{LL}_G(\theta|D)$ is maximized by simply setting each parameter $\theta_{x_i|Pa(x_i)}$ to $\widehat{P}_D(x_i|Pa(x_i))$, 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

$$\mathrm{LL_G}(\theta|D) = \mathrm{CLL_G}(\theta|D) + \sum_{j=1}^{M} \log P_\theta(x_a^j),$$

where

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

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

Given a network structure G that encapsulates the true structure of the data, parameters that maximize LL_G also maximize CLL_G.[2] However, in practice the structure may be incorrect and ML learning will not optimize the CLL score, which can result in a suboptimal classification decision.[3]

An alternative approach is to directly optimize Eq. 3, which is maximized when

$$P_\theta(x_c|x_a) = \frac{\theta_{x_c} \prod \theta_{x_a|Pa(x_a)}}{\sum_{x_c} \theta_{x_c} \prod \theta_{x_a|Pa(x_a)}} = \widehat{P}_D(x_c|x_a). \quad (4)$$

For a generative model such as a Bayesian network, Eq. 4 cannot be expressed in log-linear form and has no closed form solution. A direct optimization approach requires computationally expensive numerical techniques. For example, the ELR method of (Greiner and 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.

[2]The asymptotic properties of likelihood and conditional likelihood learning is discussed in greater detail in (Nadas 1983).

[3]If G is incorrect, maximizing CLL_G may also lead to poor performance. However, LL score is more sensitive than CLL since it requires correct knowledge of the features (Devroye et al. 1996).

4 Boosted parameter learning

4.1 Ensemble model

Instead of maximizing the CLL score for a single Bayesian network model, we take the ensemble approach and maximize the classification performance of the ensemble Bayesian network classifier. For binary classification, given a class x_c and the attributes x_a, an ensemble model has the general form:

$$F_{x_c}(x_a) = \sum_{k=1}^{K} \beta_k f_{k,x_c}(x_a). \tag{4.1}$$

and $f_{k,x_c}(x_a)$ is the classifier confidence on selecting label x_c given x_a during boosting iteration k, 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 $f_k(x_a) \in \{-1, 1\}$ is the output of each classifier given x_a, and $F(x_a)$ is the ensemble of $f(x_a)$. Equation 4.1 can be expressed as a conditional probability distribution over X_c given the additive model F with logistic transformation:

$$P_F(x_c|x_a) = \frac{\exp\{F_{x_c}(x_a)\}}{\sum_{x'_c \in X_c} \exp\{F_{x'_c}(x_a)\}}. \tag{5}$$

For binary classification, Eq. 5 is then updated as:

$$P_F(x_c|x_a) = \frac{1}{1 + \exp\{-2x_c F(x_a)\}}. \tag{6}$$

Instead of directly optimizing CLL, we minimize the often used Exponential Loss Function (ELF) of the ensemble Bayesian network classifier. ELF is defined as:

$$\mathrm{ELF}_F = \sum_{i=1}^{M} \exp\{-x_c^i F(x_a^i)\}. \tag{7}$$

Solving for $x_c F(x_a)$ in Eq. 6 and combining with Eq. 7, we have

$$\mathrm{ELF}_F = \sum_{i=1}^{M} \exp\left\{\frac{1}{2} \log \frac{1 - P_F(x_c^i|x_a^i)}{P_F(x_c^i|x_a^i)}\right\} \tag{8}$$

$$= M \sum_{\substack{x_a \in X_a \\ x_c \in X_c}} \widehat{P}_D(x_c, x_a) \sqrt{\frac{1}{P_F(x_c, x_a)} - 1}. \tag{9}$$

Equation 9 is a differentiable upper bound on the classification error, and an approximation to the negative CLL score (Friedman et al. 2000).[4]

[4]The ELF criteria has been demonstrated to be an effective classification error minimization criteria, and to perform well on real data. A detailed comparison of ELR and CLL can be found in (Friedman et al. 2000).

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 $w_i = w_i \exp\{-\beta_k x_c^i f(x_a^i \mid \theta_k, G_k)\}$ and normalize.
4.	Ensemble output: sign $\sum_k \beta_k f(x_a \mid \theta_k, G_k)$

4.2 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 Eq. 9 (Friedman et al. 2000).

At k-th iteration of boosting, the weighted data uniquely determines the parameters for each Bayesian network classifier θ_k and the hypothesis weights β_k via efficient ML parameter learning, where

$$\beta_k = 0.5 \log \frac{1 - Err_k}{Err_k}, \theta_k(x_a|x_c) = \widehat{P}_{D_k}(x_a|x_c) \tag{10}$$

where Err_k is the classification error on the weighted data. The full algorithm is shown in Table 1. There is no guarantee that AdaBoost will find the global minimum of the ELF. Also, AdaBoost has been shown to be susceptible to label noise (Dietterich 2000; Bauer and Kohavi 1999). In spite of these issues, boosted classifiers tend to produce excellent results in practice (Schapire and Singer 2000; Drucker and Cortes 1996). Boosted Naive Bayes (bNB) has been previously shown to improve the classification accuracy of naive Bayes (Elkan 1997; Ridgeway et al. 1998).

5 Structure learning

Given training data D, structure learning is the task of finding a set of directed edges G that best model the true density of the data. In order to avoid overfitting, Bayesian Scoring Function (Cooper and Herskovits 1992; Heckerman et al. 1995) and Minimal Description Length (MDL) (Lam and Bacchus 1992) are commonly used to evaluate candidate structures. The MDL score is asymptotically equivalent to the Bayesian scoring function in the large sample case and this paper will concentrate on the MDL score. MDL score is defined as

$$\mathrm{MDL}(B|D) = \frac{\log |D|}{2}|B| - \mathrm{LL}(B|D) \tag{11}$$

where $|B|$ is the total number of parameters in model B, and $|D|$ is the total number of training samples.

Table 2 Boosted augmented naive Bayes (*b*AN)

1. Given training data D, construct a complete graph G_{full} with attributes X_a as vertices. Calculate $I_p(X_{a_i}; X_{a_j}|X_c)$ for each pair of attributes X_a, $i \neq j$, where

$$I_p(X_{a_i}; X_{a_j}|X_c) = \sum_{\substack{x_{a_i} \in X_{a_i} \\ x_{a_j} \in X_{a_j}, x_c \in X_c}} P(x_{a_i}, x_{a_j}, x_c) \log \frac{P(x_{a_i}, x_{a_j}|x_c)}{P(x_{a_i}|x_c)P(x_{a_j}|x_c)} \tag{12}$$

2. Construct G_{TAN} from G_{full} with I_p as weights, set $G_1 =$ naive Bayes.
3. For $s = 1$ to $N - 1$
   - Parameter boosting using G_s as base structure.
   - Evaluate the classification accuracy for the current $G_{b\text{AN}}$ on training data D, and terminate if it stops decreasing.
   - Else, 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_s.
   - $G_{s+1} = G_s$.
4. Ensemble output: sign $\sum_{k=1}^{K} \beta_k f(x_a|\theta_k, G_s)$, where K is the number of classifiers in the ensemble with G_s as base structure.

An exhaustive search over all structures against an evaluation function can in principle find the best Bayesian network model, but in practice, since the structure space is super exponential in the number of variables in the graph, it is not feasible in nontrivial networks. Several tractable heuristic approaches have been proposed to limit the search space. The algorithms K2 (Cooper and Herskovits 1992) and MCMC-K2 (Friedman and Koller 2000) define a node ordering such that a directed edge can only be added from a high ranking node to a low ranking node. Heckerman et al. (1995) proposed a hill-climbing local search algorithm to incrementally add, remove or reverse an edge until a local optimum is reached.

An additional structure penalty is to simply limit the number of parents an attribute can have. Friedman et al. (1997), based on the approach in (Chow and Liu 1968), proposed an efficient algorithm to construct an optimal Tree Augmented Naive Bayes (TAN) based on the conditional mutual information among the features. In comparison to TAN, Augmented Bayesian Network Classifier (ANC) (Keogh and Pazzani 1999), has the flexibility to add fewer edges to Naive Bayes. An example of ANC classifier is shown in Fig. 1. TAN can be considered as a special case of ANC. All of the methods (K2, Heckerman's method, ANC and TAN) utilize standard ML parameter learning for simplicity and efficiency.

5.1 Boosted augmented naive Bayes (*b*AN)

Although the training complexity of parameter boosting is within a constant factor (Elkan 1997) of ML learning, combining parameter boosting with exhaustive structure search is still impractical. Even with a constrained search space, hill-climbing search and K2 algorithm must consider a large number of structures.

On the other hand, ANC and TAN support efficient learning by limiting the number of parents per attribute to two. TAN augments a standard naive Bayes 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 $I_p(X_{a_i}; X_{a_j}|X_c)$ between two attributes X_{a_i}, X_{a_j} given the

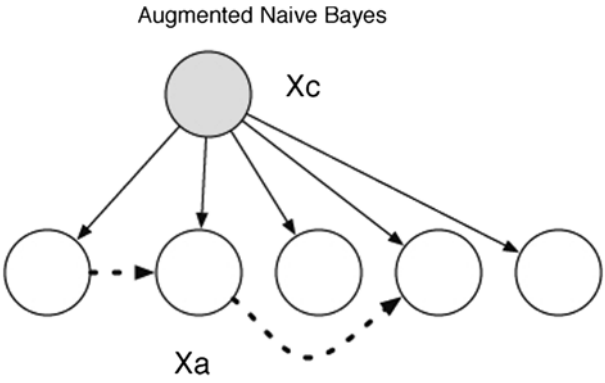


Fig. 1 An example of ANC, the dotted edges are structural extensions to naive Bayes

class node X_c where

$$I_p(X_{a_i}; X_{a_j}|X_c) = \sum_{\substack{x_c \in X_c \\ x_{a_i} \in X_{a_i} x_{a_j} \in X_{a_j}}} P(x_{a_i}, x_{a_j}, x_c) \log \frac{P(x_{a_i}, x_{a_j}|x_c)}{P(x_{a_i}|x_c)P(x_{a_j}|x_c)}.$$

TAN learning algorithm constructs the optimal tree-augmented network B_T that maximizes $\mathrm{LL}(B_T|D)$. 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 *b*AN learning algorithm extends the TAN approach using parameter boosting. Starting from a naive Bayes model, at iteration s, *b*AN greedily augments the current structure with the s-th edge having the highest conditional mutual information. We call the resulting structure $b\mathrm{AN}^s$. We then minimize the ELF score of the $b\mathrm{AN}^s$ classifier with parameter boosting. *b*AN terminates when the added edge does not improve the classification accuracy of the training data. Since TAN contains $N-1$ augmenting edges, *b*AN in worst case evaluates $N-1$ structures. This complexity is linear in N, in comparison to the polynomial number of structures examined by K2 or Hill-Climbing algorithms. Moreover, we find that in practice (as shown in Table 4) that *b*AN usually only adds a small number of edges to naive Bayes, significantly fewer than the competing algorithms like TAN, BNC-2P and BNC-MDL. As a result, the base Bayesian network structure constructed from *b*AN usually contains fewer edges than other competing structure learning algorithms, making it computationally very efficient. The algorithm is shown in Table 2.

5.2 Boosted augmented naive Bayes with heterogeneous mixture of structures ($b\mathrm{AN}_{mix}$)

The *b*AN learning algorithm constrains all weak classifiers to share a homogeneous structure. In our previous work (Jing et al. 2005), this constraint was chosen partially to facilitate comparison with other Bayesian network classifiers. As a result, structure selection can be seen as a wrapper function to the parameter boosting.

This section explores techniques in learning an ensemble Bayesian network classifier with heterogeneous structures. Choudhury et al. (2002) explored the idea of combining the re-weighting of training data with structure selection during each iteration of boosting for dynamic Bayesian networks.

In our work, we employ the greedy strategy of incrementally adding edges to the base structure during iterations of boosting. The greedy structure learning algorithm, named

$b\text{AN}_{mix}$, starts by constructing a TAN structure similar to the one proposed in bAN. We sort the edges in TAN based on their conditional mutual information in descending order. Starting with naive Bayes, we apply parameter boosting on the initial structure until the weighted error is higher than ϵ.[5] This step is analogous to the first step of bAN (up to $S = 1$). Next, we augment the current structure with the next "best" edge from TAN. However, instead of restarting a new round of parameter boosting with a new structure, we retain the ensemble generated so far, but train the new and more complex structure on the re-weighted data. The addition of a more complex structure, by better capturing the underlying distribution of the data, usually reduces the weighted error and allows boosting to continue. We iterate this process until the new structure does not improve the classification accuracy. In effect, we are incrementally generating an ensemble of sparsely connected, heterogeneous mixture of Bayesian network classifiers. The psudocode for $b\text{AN}_{mix}$ is shown in Table 3.

There are two basic strategies that underly the $b\text{AN}_{mix}$ algorithm. First, in order to avoid overfitting, we select edges parsimoniously to maintain the sparsity of our generatively trained Bayesian network classifiers. An edge is added to the structure only when the existing weak classifier can no longer effectively classify the weighted samples. Our second strategy is to add edges with the highest conditional mutual information from TAN, under the assumption that those edges, by capturing the most important relationship information in the data, have the most potential to contribute to the classification accuracy.

To assess the effectiveness of these two strategies, we formulated two alternative training algorithms. The first algorithm, $b\text{AN}_{mix(1)}$, tests the first strategy. Rather than waiting for boosting to converge, $b\text{AN}_{mix(1)}$ adds a new edge to the existing structure at each iteration of boosting. Therefore, each structure is used exactly once (hence the name $b\text{AN}_{mix(1)}$). $b\text{AN}_{mix(1)}$ terminates when the classification accuracy does not improve, or when the maximum number of edges have been added. The second algorithm, $b\text{AN}_{mix(r)}$, tests a random structure selection strategy. Instead of selecting the edge with the best conditional mutual information from TAN, it randomly selects an edge from TAN to add to the existing Bayesian network. The next two sections provide a detailed analysis of these three algorithms and demonstrate that $b\text{AN}_{mix}$ outperforms $b\text{AN}_{mix(1)}$, $b\text{AN}_{mix(r)}$ and has comparable performance to bAN. However, $b\text{AN}_{mix}$ is more efficient to train than bAN.

We also implemented the MCMC variation of K2 from Friedman and Koller (2000) to study the effect of combining structure optimization with parameter boosting. K2 is a greedy search algorithm which uses a known ordering of the nodes and a maximum limit on the number of parents for any node to constrain the search, and the MCMC variant samples from the space of node orderings. As we show in Sect. 6, in spite of its tremendous computational cost, MCMC K2 did not yield superior classification accuracy than $b\text{AN}_{mix}$, further confirming the observation by Grossman and Domingos (2004) that exhaustive structure optimization offers little classification improvement when combined with discriminative parameter learning.

6 Experiments

We evaluated the performance of bNB and variations of bAN on 23 datasets from the UCI repository (Blake and Merz 1998) and two artificial data sets, Corral and Mofn, designed by Kohavi and John (1997). Friedman et al. (1997), Greiner and Zhou (2002), Grossman

[5] We use $\epsilon = 0.45$ in our experiments as the stopping criterion to reduce the number of model boosting rounds. No significant change in accuracy was observed compared to $\epsilon = 0.50$.

Table 3 Boosted augmented naive Bayes with mixed structure

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. construct a complete graph G_{full} with attributes X_a as vertices. Calculate $I_p(X_{a_i}; X_{a_j}|X_c)$ for each pair of attributes X_a, $i \neq j$, where

$$I_p(X_{a_i}; X_{a_j}|X_c) = \sum_{\substack{x_{a_i} \in X_{a_i} \\ x_{a_j} \in X_{a_j}, x_c \in X_c}} P(x_{a_i}, x_{a_j}, x_c) \log \frac{P(x_{a_i}, x_{a_j}|x_c)}{P(x_{a_i}|x_c)P(x_{a_j}|x_c)} \tag{13}$$

3. Construct G_{TAN} from G_{full} with I_p as weights.
4. Initialize the training data weights with $w_i = 1/M, i = 1, 2, \ldots, M$
5. Set the initial Bayesian network structure as $G_1 =$ naive Bayes, and $G_{b\text{AN}_{mix}} = \{\}$.
6. Repeat for $k = 1, 2, \ldots$
   - Given G_k, θ_k is learned through ML parameter learning on the weighted data D at iteration k.
   - Compute the weighted error, $Err_w(G_k) = E_w[1_{x_c \neq f_{\theta_k}(x_a)}]$, $\beta_k = 0.5 \log \frac{1-Err_w(G_k)}{Err_w(G_k)}$.
   - $G_{b\text{AN}_{mix}} = \{G_{b\text{AN}_{mix}}, G_k\}$
   - If $Err_w(G_k) \leq \epsilon$, update weights $w_i = w_i \exp\{-\beta_k x_c^i f_{\theta_k}(x_a^i)\}$ and normalize.
   - Else, if $Err(G_{b\text{AN}_{mix}})$ calculated from the training data D remains the same, terminate the loop; or remove the edge with the highest $I_p(X_{a_i}; X_{a_j}|X_c)$ from G_{TAN} and add to G_k.
   - Set $G_{k+1} = G_k$.
7. Ensemble output: sign $\sum_k \beta_k f(x_a|\theta_k, G_k)$

and Domingos (2004) and Pernkopf and Bilmes (2005) used this group of data sets as benchmarks for Bayesian network classifiers. We used hold-out test for larger data sets and 5 fold cross validation for smaller sets.[6] To ensure fairness, we used Dirichlet prior Bayesian smoothing, with parameters identical to the ones from page 18 of Friedman et al. (1997) for all classifiers when appropriate. The data preparation, experimental set-up as well as Bayesian smoothing are the same as those used by Friedman et al. for the evaluation of TAN (Friedman 1997). Since the Wilcoxon Signed-Ranks Test is demonstrated to provide the most unbiased evaluations for comparing methods across multiple datasets (Demsar 2006), we also used it to generate confidence scores. All algorithms are implemented in C++.[7]

The abbreviations for competing algorithms are described below:

- ***b*AN$_{MH}$, *b*AN$_{M2}$:** Boosted Augmented Naive Bayes trained via AdaBoost.MH and AdaBoost.M2 algorithm respectively.
- **NB:** Naive Bayes.
- **TAN:** Tree Augmented naive Bayes (Friedman et al. 1997).
- **BNC-2P:** Discriminative structure selection via CLL score (Grossman and Domingos 2004).
- **ELR$_{NB}$, ELR$_{TAN}$:** NB and TAN with parameters optimized for conditional log likelihood as in Greiner and Zhou (2002).

[6] We used hold-out test for "chess", "letter", "mofn", "satimage", "segment", "shuttle", "waveform," and 5 folds cross validation for the rest. We removed the data points with missing values and used pre-discretization step in manner described by Dougherty et al. (1995).

[7] The C++ code can be found at www.cc.gatech.edu/cpl/projects/boosted_bnc.html.

6.1 *b*NB on UCI datasets

Table 4 on lists the average testing error and confidence score for each algorithm. Figure 2 contains the scatter plots which compare *b*NB against NB, TAN, ELR_{NB} and BNC-2P. In each plot, points above the diagonal line $y = x$ correspond to data sets for which *b*NB outperforms the competing algorithm. The average testing error is shown next to the name of the method.

As shown in Fig. 2(a), *b*NB has lower average testing error than NB. Figures 2(b) and 2(d) show that *b*NB has comparable classification accuracy to TAN and ELR_{NB}. Also, we find BNC-2P, a discriminative structure learning algorithm, to slightly outperform *b*NB. However, we will demonstrate in Sect. 7 that *b*NB is significantly faster to train than ELR and BNC-2P.

6.2 *b*AN on simulated dataset

We demonstrate that when the model structure is incorrect, *b*NB and *b*AN algorithm can significantly outperform their generative counterparts. In datasets where there are strong correlation among attributes, *b*AN tends to produce more accurate classification performance than *b*NB by having the ability to capture the relationships in the model structure.

We generated two collections of data from the two binary Bayesian network models depicted in Fig. 3. In Model A, the variables form a Markov chain. Model B is similar to model A except it introduces an additional complexity by adding an edge between the class variable and the last feature variable. We varied the number of attributes and their parameter values to generate 25 datasets with different distributions. Since the attributes are correlated, naive Bayes can sometimes give a suboptimal classification boundary.

Figures 4 and 5 contain the one-standard-deviation error bars produced from different classifiers evaluated on data generated from structures A and B respectively. Figures 4(a), 4(b), 5(a) and 5(b) show that *b*NB and *b*AN have lower average testing errors than NB. 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, *b*NB and *b*AN have similar testing error.

As shown in Fig. 4(c), with data generated from structure A, the average testing error for *b*AN is only slightly higher than that of *b*NB. This is largely due to the simplicity of the Markov-chain model: *b*AN selected *b*NB as the best model (adding 0 edges) in 20 out of the 25 datasets. On the other hand, when data is generated from more complex structure B, *b*AN significantly outperforms *b*NB shown in Fig. 5(c) by adding additional model structures.

6.3 *b*AN on UCI dataset

Table 4 and Fig. 6 contain the testing error and confidence scores comparing *b*AN against other Bayesian network classifiers on the UCI data described in the beginning of Sect. 6. Table 4 also contains the average number of edges added for each of the classifiers given a dataset. We implemented two variations of *b*AN for multi-class classification tasks. $b\text{AN}_{MH}$ breaks a D class problem into D sets of binary classification tasks. $b\text{AN}_{M2}$ directly applies the AdaBoost.M2 algorithm to build a single ensemble for the multi-class problem.[8] $b\text{AN}_{M2}$ and $b\text{AN}_{MH}$ have comparable classification performance.

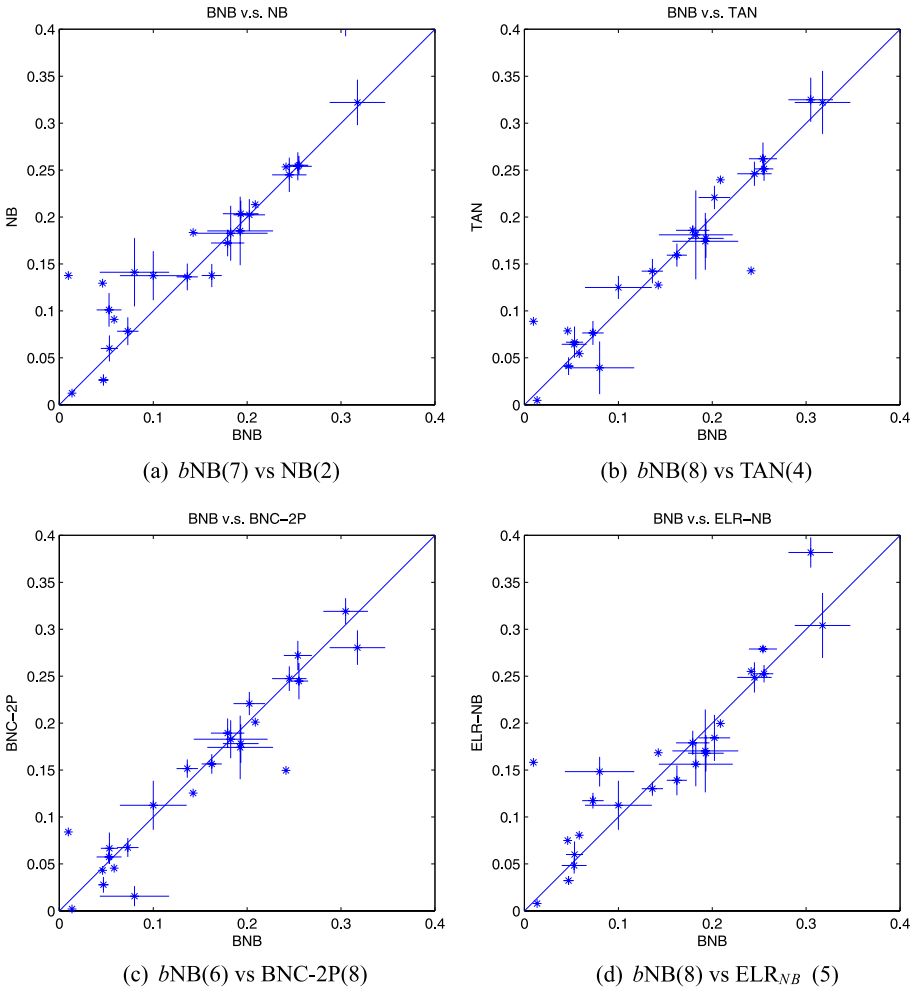
[8] $b\text{AN}_{M2}$ and $b\text{AN}_{MH}$ are essentially the same for binary classification.

Table 4 Testing errors and its associated confidence scores comparing two classifiers on multiple datasets. We also included the total number of edges added to final AN structure, averaged over different evaluation folds. Wilcoxon Signed-Ranks Test and Friedman Test with post-hoc Bonferroni test are used to generate confidence scores. "*" indicates datasets with binary class variables, where $b\text{AN}_{MH}$ and $b\text{AN}_{M2}$ produce the same classification accuracy. "⇐" indicates the cases where $b\text{AN}_{M2}$ statistically significantly outperforms the competing model (according to the corresponding test). We also included the number of augmented edges for each Bayesian network classifier. (By definition, naive Bayes and ELR_{NB} contains 0 augmented edges. TAN and ELR_{TAN} contains $N-1$ number of augmented edges, where N is the number of features in the datasets. When 5-fold cross validation is used, we averaged the number of augmented edges)

Name	$b\text{AN}_{M2}$ / #edges	$b\text{AN}_{MH}$ / #edges	bNB	NB	TAN / #edges	BNC-2P / #edges	ELR_{NB}	ELR_{TAN}
australian	0.1725 / 0.4	*	0.1623	0.1377	0.1594 / 13	0.1565 / 11.5	0.1391	0.1435
breast	0.0483 / 1.0	*	0.0469	0.0264	0.0412 / 9.0	0.0278 / 7.0	0.0322	0.0322
chess	0.0460 / 0.0	*	0.0460	0.1295	0.0788 / 35	0.0432 / 32	0.0750	0.0686
cleve	0.1825 / 0.8	*	0.1792	0.1724	0.1858 / 12	0.1893 / 9.0	0.1710	0.1689
corral	0.0000 / 1.0	*	0.1240	0.1412	0.0394 / 5.0	0.0157 / 3.5	0.1483	0.0825
crx	0.1378 / 0.4	*	0.1362	0.1362	0.1424 / 14	0.1516 / 11	0.1381	0.1454
diabetes	0.2487 / 0.8	*	0.2552	0.2552	0.2513 / 7.0	0.2448 / 5.0	0.2526	0.2760
flare	0.1914 / 2.0	*	0.1932	0.2035	0.1773 / 9.0	0.1782 / 9.0	0.1679	0.1838
german	0.2540 / 1.6	*	0.2540	0.2540	0.2620 / 19	0.2720 / 16	0.2790	0.2670
glass	0.2990 / 1.2	0.2989 / 0.6	0.3130	0.3221	0.3221 / 8.0	0.2804 / 4.0	0.3040	0.3082
glass2	0.2023 / 0.0	*	0.2023	0.2023	0.2208 / 8.0	0.2208 / 7.0	0.1843	0.2025
heart	0.1778 / 1.0	*	0.1926	0.1852	0.1741 / 12	0.1741 / 10	0.1764	0.1741
hepatitis	0.1000 / 0.0	*	0.1000	0.1375	0.1250 / 18	0.1125 / 16	0.1125	0.1250
iris	0.0467 / 0.2	0.0533 / 0.0	0.0467	0.0600	0.0667 / 3.0	0.0667 / 3.0	0.0600	0.0600
letter	0.1316 / 13	0.1858 / 3.0	0.2194	0.2534	0.1428 / 15	0.1496 / 11	0.2550	0.2550
lymphography	0.1765 / 1.4	0.1830 / 1.0	0.1897	0.1827	0.1810 / 17	0.1828 / 14	0.1563	0.1834
mofn-3-7-10	0.0000 / 1.0	*	0.0096	0.1377	0.0889 / 9.0	0.0840 / 8.5	0.1542	0.0898
pima	0.2449 / 1.0	*	0.2449	0.2449	0.2461 / 7.0	0.2474 / 6.5	0.2487	0.2761
satimage	0.1465 / 15	0.1425 / 1.0	0.1750	0.1835	0.1275 / 35	0.1255 / 32	0.1685	0.1830
segment	0.0714 / 1.0	0.0584 / 0.0	0.0714	0.0909	0.0545 / 18	0.0455 / 17	0.0805	0.0805
shuttle-small	0.0036 / 2.0	0.0062 / 2.0	0.0124	0.0124	0.0047 / 8.0	0.0020 / 8.0	0.0078	0.0626

Table 4 (*Continued*)

Name	$b\text{AN}_{M2}$ / #edges	$b\text{AN}_{MH}$/ #edges	bNB	NB	TAN / #edges	BNC-2P / #edges	ELR_{NB}	ELR_{TAN}
soybean-large	0.0676 / 1.6	0.0800 / 1.6	0.0747	0.0783	0.0765 / 34	0.0675 / 30	0.1174	0.1328
vehicle	0.3341 / 3.8	0.3120 / 3.8	0.3675	0.4101	0.3249 / 17	0.3191 / 15	0.3817	0.2894
vote	0.0506 / 0.6	*	0.0529	0.1011	0.0644 / 15	0.0575 / 14	0.0483	0.1126
waveform	0.2087 / 1.0	0.2121 / 1.0	0.2087	0.2132	0.2396 / 20	0.2011 / 18	0.1996	0.2274
Average	0.1417 / (2.04)	0.1436 / (1.03)	0.1551	0.1709	0.1519 / (14.68)	0.1446 / (12.72)	0.1623	0.1652
		$b\text{AN}_{MH}$	bNB	NB	TAN	BNC-2P	ELR_{NB}	ELR_{TAN}
Wilcoxon SR test	$b\text{AN}_{M2}$		$\Leftarrow -2.771$	$\Leftarrow -3.861$	$\Leftarrow -3.619$	$\Leftarrow -1.924$	$\Leftarrow -2.893$	$\Leftarrow -3.807$
Friedman/Bonferroni test	$b\text{AN}_{M2}$		$\Leftarrow -2.13$	$\Leftarrow -2.91$	$\Leftarrow -1.96$	-0.28	$\Leftarrow -1.66$	$\Leftarrow -3.20$



(a) bNB(7) vs NB(2)

(b) bNB(8) vs TAN(4)

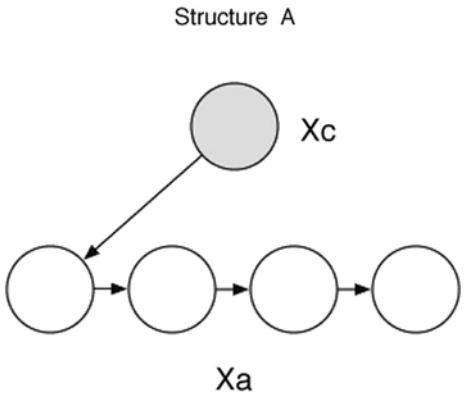
(c) bNB(6) vs BNC-2P(8)

(d) bNB(8) vs ELR$_{NB}$ (5)

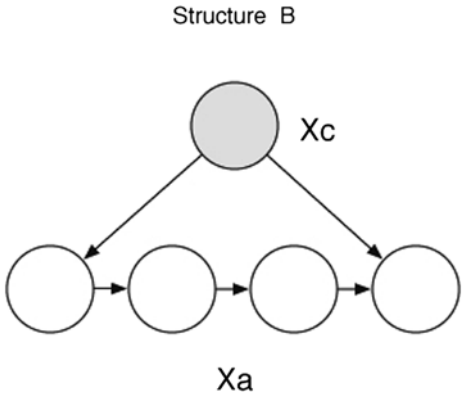
Fig. 2 Scatter plots for experiments on 25 sets of UCI and artificial benchmark data. We also measure the significance of the error difference in a specific datasets between two competing algorithms. The number of statistically significant "winners" are accumulated and placed in parenthesis next to the name of the algorithm

bAN$_{M2}$ and bAN$_{MH}$ chose very different base structures in our empirical study. On average, bAN$_{M2}$ selects 1 to 15 more augmenting edges than bAN$_{MH}$. This result is not surprising since AdaBoost.MH algorithm divides a potentially difficult multi-class problem into several relatively easier binary classification problems, each of which can be sufficiently classified with a sparser boosted Bayesian network. On the other hand, bAN$_{M2}$ fits a single ensemble to the multi-class problem, requiring more descriptive base structure. This further validates the need for structure selection in the construction of boosted Bayesian network classifiers.

From Figs. 6(b) and 6(c), we can see that bAN algorithm significantly outperforms naive Bayes and slightly outperforms TAN. Also, bAN algorithm outperforms ELR$_{NB}$ and ELR$_{TAN}$. bAN has comparable classification accuracy with BNC-2P algorithm.

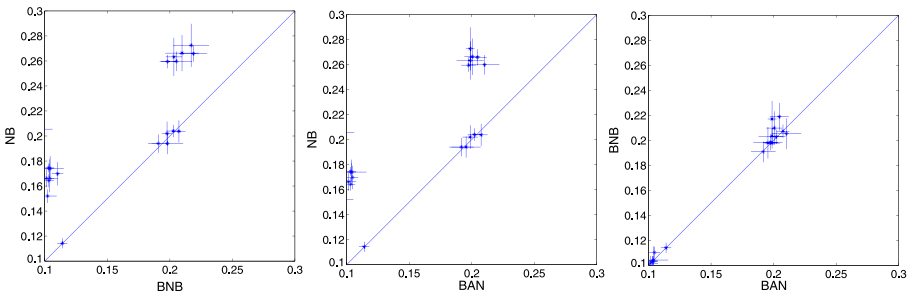


(a) Structure A



(b) Structure B

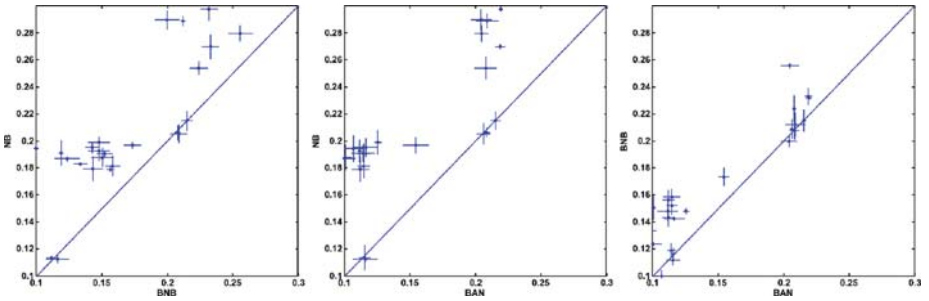
Fig. 3 We generated two collections of simulated datasets from structures A and B. Since feature variables are correlated, Naive Bayes can often produce sub-optimal classification accuracies



(a) *b*NB vs NB (b) *b*AN vs NB (c) *b*AN vs *b*NB

Fig. 4 Scatter plots for data generated from Structure A shown in Fig. 3. Each point in the graph represents the classification error for one particular model setting. *b*NB and *b*AN outperform 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

As shown in Table 4, the average testing errors for *b*AN and *b*NB are 0.142 and 0.155 respectively. The differences is significant under Wilcoxon Signed-rank test. *b*AN has lower average testing error (difference of 0.5%–5%) than *b*NB in 16 out of the 25 datasets. Fig-



(a) *b*NB vs NB (b) *b*AN vs NB (c) *b*AN vs *b*NB

Fig. 5 Scatter plots for data generated from Structure B shown in Fig. 3. As shown in (**c**), *b*AN outperforms *b*NB outperform NB in 17 out of the 25 simulated datasets. This demonstrates the importance of structure selection in the construction of boosted Bayesian network classifiers. Both *b*AN and *b*NB outperform naive Bayes

ure 6(a) shows that *b*AN outperforms *b*NB in 5 out of the 5 datasets with significant error differences. Since *b*AN generalizes *b*NB, in several datasets (Mofn, Iris), the structure chosen by *b*AN is very similar to *b*NB (with 0 to 1 augmented edges). *b*AN is more beneficial in datasets where the conditional dependencies among attributes are complex (Letter). This is an interesting result since it shows that combining discriminative structure learning with parameter optimization seems to improve classification accuracy.

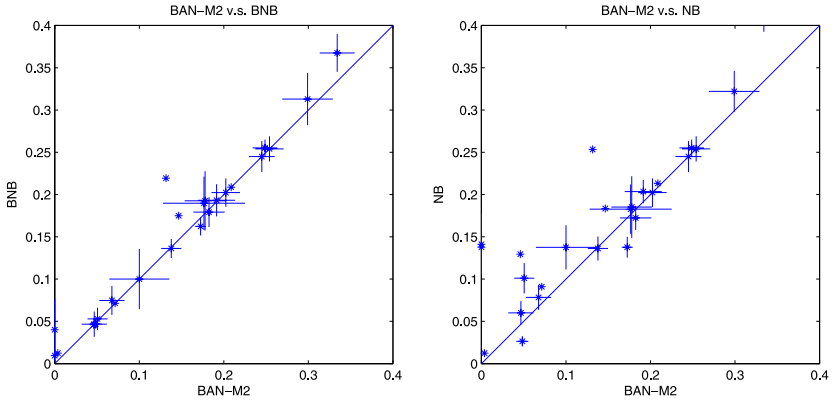
We would like to mention that *b*AN tends to produce poorly calibrated conditional distributions (CLL score) on the testing data. This supports the observation by Niculescu-Mizil and Caruana (2005) that boosting tends to do poorly on the calibration task. BNC-2P, on the other hand, has been demonstrated to produce an accurate calibration of the conditional distribution of the data.

6.4 Alternative boosted Bayesian network classifier on UCI dataset

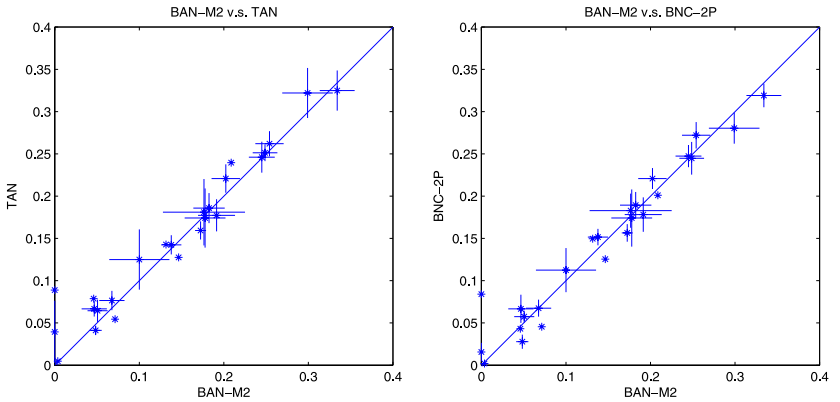
In this section, we study the performance of different boosted Bayesian network classifiers on the UCI datasets. We also measured the level of variance (data noise) in each of the 25 UCI datasets with method described in (Kohavi and Wolpert 1996), shown next to the name of the datasets. The abbreviations for competing algorithms are described below:

- ***b*AN:** Boosted Augmented naive Bayes. Each of the Bayesian network classifier in the ensemble shares a common structure.
- ***b*AN$_{mix}$:** Boosted Augmented naive Bayes with heterogeneous structures in the ensemble.
- ***b*NB**, ***b*TAN** and ***b*BNC-2P:** Apply AdaBoost to Naive Bayes and structure selected by TAN and BNC-2P algorithm.
- ***b*AN$_{mix(1)}$** and ***b*AN$_{mix(r)}$** are two alternative training algorithm for *b*AN$_{mix}$ proposed in Section 5.2.
- ***b*AN$_{MCMC(K2)}$** combines parameter boosting with the MCMC variation of K2 structure learning algorithm.

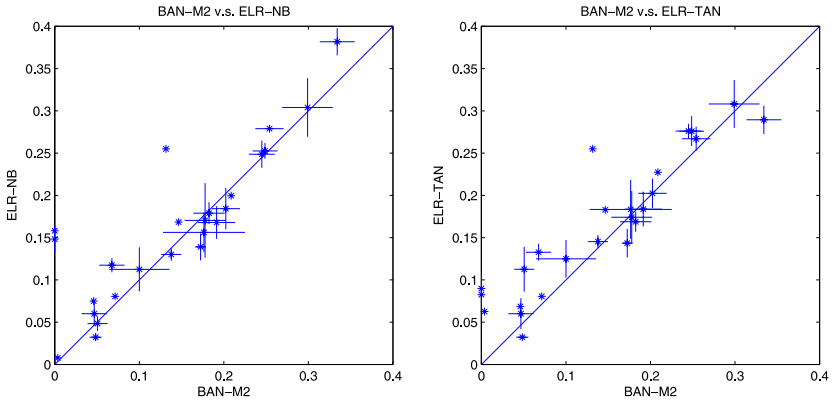
As shown in Table 5 and Fig. 7, *b*AN and *b*AN$_{mix}$ significantly outperformed other boosted Bayesian network classifiers including *b*NB, *b*TAN, *b*BNC-2P, *b*AN$_{mix(1)}$ and *b*AN$_{mix(r)}$. We believe that the experiment results provide strong evidence to validate



(a) $b\text{AN}_{M\,2}(5)$ vs bNB(0) (b) $b\text{AN}_{M\,2}(9)$ vs NB(2)



(c) $b\text{AN}_{M\,2}(9)$ vs TAN(3) (d) $b\text{AN}_{M\,2}(6)$ vs BNC-2P(5)



(e) $b\text{AN}_{M\,2}(7)$ vs ELR_{NB} (3) (f) $b\text{AN}_{M\,2}$ (9) vs $\text{ELR}_{TA\ N}$ (5)

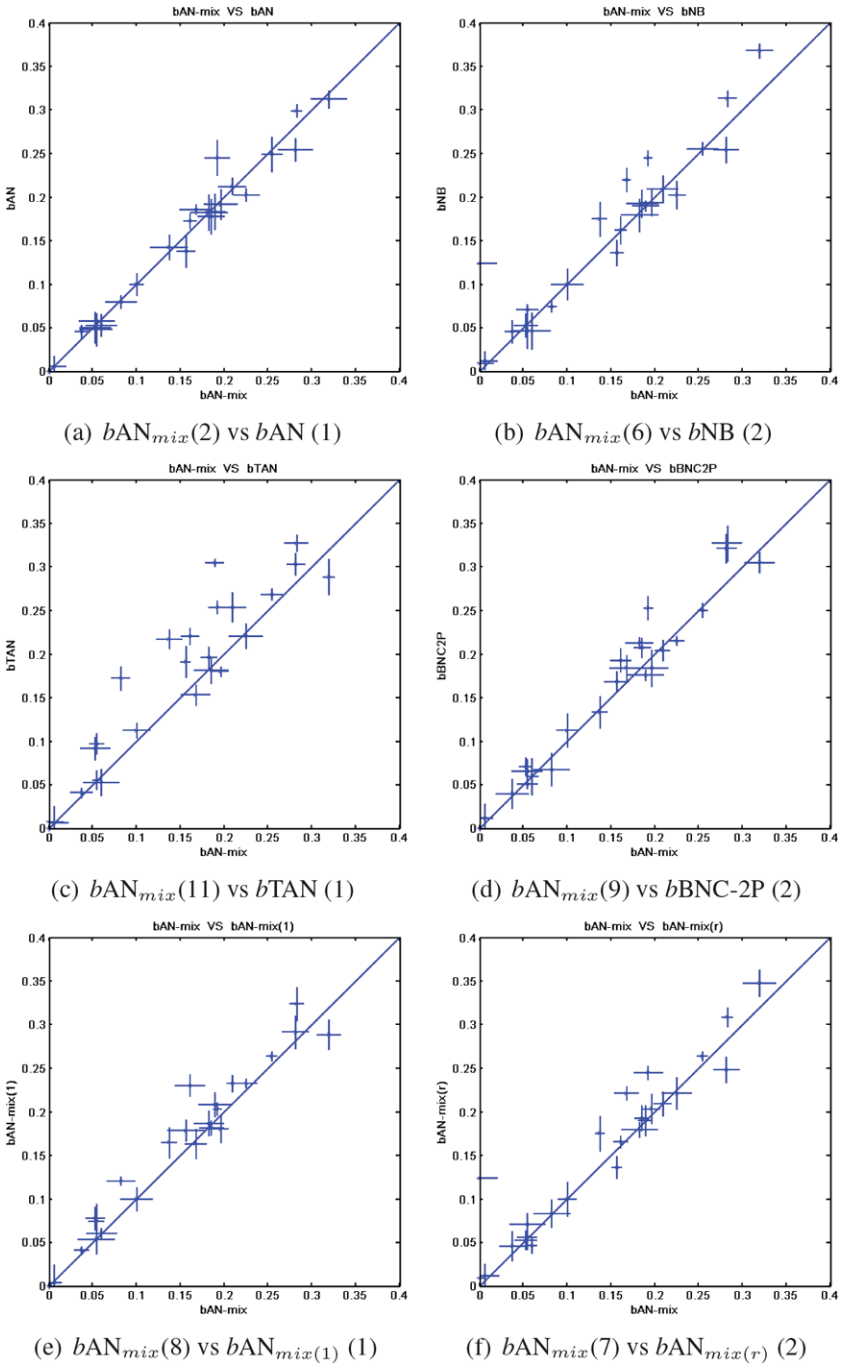
Fig. 6 Scatter plots of classification errors on UCI datasets. We also measure the significance of the error difference for each datasets between two competing algorithms. The number of statistically significant "winners" are accumulated and placed in parentheses next to the name of the algorithm. *b*AN outperforms NB, TAN, ELR-NB, ELR-TAN, and *b*NB, and is comparable with BNC-2P

Table 5 Testing errors and confidence scores on the UCI datasets. Since we did not observe a significant difference in classification accuracy between $b\text{AN}_{MH}$ and $b\text{AN}_{M2}$, we used $b\text{AN}_{M2}$ in this set of experiments. "⇐" indicates the cases where $b\text{AN}_{M2}$ outperforms the competing model in a statistically-significant manner under the given test

Data	Variance	*b*AN	$b\text{AN}_{mix}$	*b*TAN	*b*BNC-2P	$b\text{AN}_{mix(1)}$	$b\text{AN}_{mix(r)}$	$b\text{AN}_{mcmc(k2)}$
australian	(0.006)	0.1725	0.1608	0.2203	0.1928	0.2301	0.1654	0.1687
breast	(0.002)	0.0483	0.0542	0.0556	0.0512	0.0541	0.0564	0.0556
chess	(0.000)	0.0460	0.0375	0.0413	0.0403	0.0413	0.0464	0.0375
cleve	(0.008)	0.1825	0.1825	0.1960	0.2129	0.1867	0.1792	0.1825
corral	(0.000)	0.0000	0.0000	0.0080	0.0000	0.0000	0.1240	0.0000
crx	(0.004)	0.1378	0.1561	0.1913	0.1684	0.1785	0.1362	0.1701
diabetes	(0.123)	0.2487	0.2539	0.2682	0.2500	0.2634	0.2634	0.2634
flare	(0.112)	0.1914	0.1960	0.1801	0.1838	0.1801	0.2034	0.1801
german	(0.001)	0.2540	0.2810	0.3030	0.3210	0.2913	0.2480	0.2934
glass	(0.113)	0.2989	0.2822	0.3268	0.3268	0.3234	0.3084	0.3287
glass2	(0.121)	0.2023	0.2250	0.2208	0.2148	0.2323	0.2212	0.2307
heart	(0.045)	0.1778	0.1851	0.1815	0.2074	0.1815	0.1926	0.1815
hepatitis	(0.000)	0.1000	0.1000	0.1125	0.1125	0.1000	0.1000	0.1125
iris	(0.023)	0.0533	0.0600	0.0533	0.0600	0.0610	0.0467	0.0600
letter	(0.000)	0.1858	0.1674	0.1532	0.1842	0.1634	0.2209	0.1545
lymphography	(0.000)	0.1830	0.1894	0.2043	0.1759	0.2081	0.1897	0.1897
mofn-3-7-10	(0.000)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0096	0.0000
pima	(0.124)	0.2449	0.1922	0.2536	0.2526	0.2032	0.2449	0.1987
satimage	(0.000)	0.1425	0.1370	0.1570	0.1335	0.1650	0.1754	0.1370
segment	(0.014)	0.0584	0.0545	0.0970	0.0662	0.0750	0.0714	0.0567
shuttle-small	(0.000)	0.0062	0.0060	0.0072	0.0124	0.0045	0.0124	0.0045
soybean-large	(0.001)	0.0800	0.0823	0.0923	0.0676	0.1209	0.0834	0.0823

Table 5 (*Continued*)

Data	Variance	*b*AN	*b*AN$_{mix}$	*b*TAN	*b*BNC-2P	*b*AN$_{mix(1)}$	*b*AN$_{mix(r)}$	*b*AN$_{mcmc(k2)}$
vehicle	(0.043)	0.3120	0.3191	0.2883	0.3049	0.2883	0.3475	0.3021
vote	(0.000)	0.0506	0.0529	0.0920	0.0713	0.0780	0.0529	0.0529
waveform	(0.009)	0.2121	0.2089	0.2532	0.2043	0.2323	0.2087	0.2187
Average		0.1436	0.1433	0.1583	0.1526	0.1545	0.1563	0.1465
				*b*TAN	*b*BNC-2P	*b*AN$_{mix(1)}$	*b*AN$_{mix(r)}$	*b*AN$_{mcmc(k2)}$
Wilcoxon SR test		*b*AN		$\Leftarrow -3.713$	$\Leftarrow -2.718$	$\Leftarrow -3.484$	$\Leftarrow -3.606$	-0.61
Friedman/Bonferroni		*b*AN		$\Leftarrow -3.28$	$\Leftarrow -1.66$	$\Leftarrow -2.41$	$\Leftarrow -2.34$	-0.94



(a) $b\text{AN}_{mix}(2)$ vs bAN (1)

(b) $b\text{AN}_{mix}(6)$ vs bNB (2)

(c) $b\text{AN}_{mix}(11)$ vs bTAN (1)

(d) $b\text{AN}_{mix}(9)$ vs bBNC-2P (2)

(e) $b\text{AN}_{mix}(8)$ vs $b\text{AN}_{mix(1)}$ (1)

(f) $b\text{AN}_{mix}(7)$ vs $b\text{AN}_{mix(r)}$ (2)

Fig. 7 Scatter plots of classification difference on UCI datasets. We also measure the significance of error difference for each datasets between two competing algorithms. The number of statistically significant "winners" are accumulated and placed in parenthesis next to the name of the algorithm. $b\text{AN}_{mix}$ outperforms bNB, bTAN, bBNC-2P, bNB, $b\text{AN}_{mix(1)}$, $b\text{AN}_{mix(r)}$ and is comparable with bAN

Table 6 Classification error on the testing data. We divided the 25 datasets into three categories (each containing 7 to 9 sets) with respect to its classification variance (noise). For each category, the first row contains the testing error, and the second row is the error difference between each competing algorithm and *b*AN (positive differences correspond to lower error for *b*AN). *b*AN significantly outperforms *b*TAN and *b*BNC-2P in datasets with variance by having a sparser structure. Also, *b*AN is relatively resistant to noise, demonstrated by having comparable performance with *b*NB in datasets with mid or high variances

Datasets	*b*AN	*b*AN$_{mix}$	*b*NB	*b*TAN	*b*BNC-2P	*b*AN$_{mix(1)}$	*b*AN$_{mix(r)}$	*b*AN$_{mcmc(k2)}$
low	0.0767	0.0793	0.1032	0.0861	0.0811	0.0844	0.1034	0.0765
($\sigma^2 < 0.001$)		(+.0026)	(+.0265)	(+.0094)	(+.0044)	(+.0077)	(+.0267)	(−0.002)
mid	0.1586	0.1535	0.1582	0.1756	0.1688	0.1727	0.1578	0.1614
($0.001 \leq \sigma^2 \leq 0.1$)		(−.0051)	(−.0004)	(+.0170)	(+.0102)	(+.0141)	(−.0008)	(+.0028)
high	0.2299	0.2372	0.2417	0.2499	0.2456	0.2405	0.2483	0.2403
($\sigma^2 > 0.1$)		(+.0073)	(+.0118)	(+.0200)	(+.0157)	(+.0106)	(+.0184)	(+.0104)

the two basic strategies used in our formulation of boosted Bayesian network classifiers.

First, structures are beneficial in improving the performance of boosted Bayesian network classifiers. For example, *b*AN significantly outperformed *b*NB, especially in data-sets where features are highly correlated with each other (such as Letter, etc). Also, *b*AN$_{mix}$ significantly outperformed *b*AN$_{mix(r)}$. We observed in the experiment that the edges randomly selected by *b*AN$_{mix(r)}$ often did not improve the classification accuracy on the training data. As a result, *b*AN$_{mix(r)}$ usually terminated in the early stage of structure selection, with similar classification performance and model structures as *b*NB.

Second, since AdaBoost can overfit on noisy datasets, parsimonious structure selection provides the best compromise between modeling the structure of the data and controlling the classifier capacity. For example, both *b*AN and *b*NB significantly outperform *b*BNC-2P and *b*TAN. This is also observed in the comparison between *b*AN$_{mix}$ and *b*AN$_{mix(1)}$. In noisy datasets like "Australian" and "crx", *b*AN$_{mix(1)}$, having a larger number of edges, performs significantly worse than *b*AN$_{mix}$. Table 6 divides the datasets into three categories based on the classification variance in the data. It is evident that *b*AN and *b*AN$_{mix}$ significantly outperform *b*TAN and *b*BNC2P in the noisy datasets in Table 6.

Also, the combination of structure optimization with boosting *b*AN$_{mcmc(k2)}$ does not outperforms *b*AN$_{mix}$. On the contrary, *b*AN$_{mix}$ slightly outperforms *b*AN$_{mcmc(k2)}$ on datasets with larger variance, an indication that *b*AN$_{mix}$ is more resistant to overfitting.

7 Comparison of computational cost

7.1 Computational complexity of *b*NB and *b*AN

Given a naive Bayes classifier with N attributes and training data with M samples, the ML training complexity is $O(NM)$, which is optimal when every attribute is observed and used for classification. Parameter boosting for a naive Bayes model is $O(NMT)$ where T is the number of iterations of boosting. In our experiments, boosting seems to give good performance with a constant number (10–30) of iterations irrespective of the number of attributes (ranging from 5 to 50). This is consistent with the finding of Elkan (1997).

*b*AN has a higher training complexity than *b*NB. Step 1 in the *b*AN algorithm has the computational complexity $O(N^2M)$, where N is the number of attributes and M is the

amount of training data. Since we only add a maximum of $N-1$ edges to the network, steps 2–4 have a worst case complexity $O(NMTS_{max})$, where S_{max} is the number of structures analyzed. Therefore bAN has $O(N^2M + NMTS_{max})$ complexity. In our experiments, bAN evaluates a very small number of additional structures as base structure.[9] Therefore, bAN is very efficient in practice.

$b\text{AN}_{mix}$ on the other hand, has similar computational complexity as bAN. However, since $b\text{AN}_{mix}$ incrementally add new structures into the existing ensemble, $b\text{AN}_{mix}$ does not evaluate each new structure in all T steps as in bAN. Therefore, the computational complexity for $b\text{AN}_{mix}$ is $O(N^2M + NMTkS_{max})$, where $k < 1$. We will show in the next section that $b\text{AN}_{mix}$ indeed requires less classifier evaluations than bAN.

7.2 Empirical analysis of computational cost

The variations in possible optimization schemes make a theoretical comparison of the training cost for different Bayesian network classifiers challenging. Therefore we employ empirical training time in our analysis. All of the algorithms are implemented in C++ to share common code bases. The experiments were conducted on a cluster of 3 GHz machines.

BNC-2P has previously been shown to be computationally expensive in training (Pernkopf and Bilmes 2005). However, in cases where data the are fully observed, there are two ways to reduce the training cost. First, we can pre-compute the ML parameter values for all possible edges such that parameter learning given a structure becomes simple indexing. Second, since Heckerman's greedy structure search algorithm is an incremental process, we can avoid full inference in classifier evaluation by only partially updating the posterior. We denote the optimized implementation as BNC-2P-FAST. Figure 8 shows the training time comparison between BNC-2P-FAST and the original BNC algorithm on a small collection of UCI datasets. BNC-2P-FAST is roughly 2 orders of magnitude faster than BNC. However, the speedup is not possible when used in conjunction with the EM algorithm to handle missing data.

By taking advantage of the ensemble learning, boosted Bayesian network classifiers avoided the expensive process of structure and parameter optimization. Figure 9 shows the convergence rates for bAN, ELR and BNC-2P-FAST algorithm on the "Chess" data-set. Compared with other UCI datasets, Chess data-set contains a large number of features, making parameter and structure optimization particularly expensive. Here, bAN is roughly 15 times faster than ELR and BNC-2P-FAST. This speedup is attributed to the lower training cost per iteration of boosting comparing with structure or parameter search. bAN only requires a single classifier evaluation for each iteration of boosting. On the other hand, both BNC-2P and ELR algorithm require numerous classifier evaluations (plus the costly computation of gradient for ELR).

A more detailed comparison is shown in Table 7. We divided the 25 datasets into two parts based on their relative sample size and number of attributes.[10] We also itemized the individual training time for "large" datasets as shown in Figs. 10(a) and 10(b). As shown in Table 7, in larger datasets, bNB is roughly a factor of 20 faster than ELR and a factor of 50 faster than BNC-2P-FAST algorithm.

Also shown in Figs. 10(a) and 10(b), bAN is 2 to 5 times faster than ELR, and 2 to 20 times as fast as than BNC-2P-FAST. Also, BNC-2P-FAST is only applicable when all

[9] As shown in Table 4, bAN evaluates an average of less than 2 additional structures.

[10] We categorize datasets containing 2000 or more data points or 30 or more features as "large" datasets, the rest as "small" datasets.

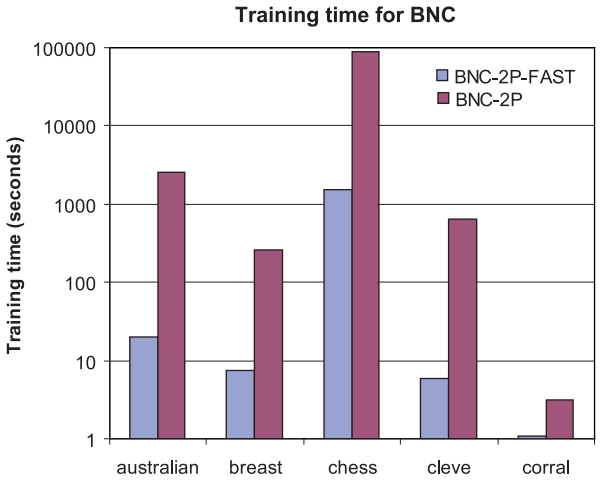


Fig. 8 Training time for an optimized version of BNC algorithm based on caching the parameters for structure evaluation. As a result, the optimized version of BNC-2P (BNC-2P-FAST) enjoys 2 orders or magnitude improvement in training efficiency

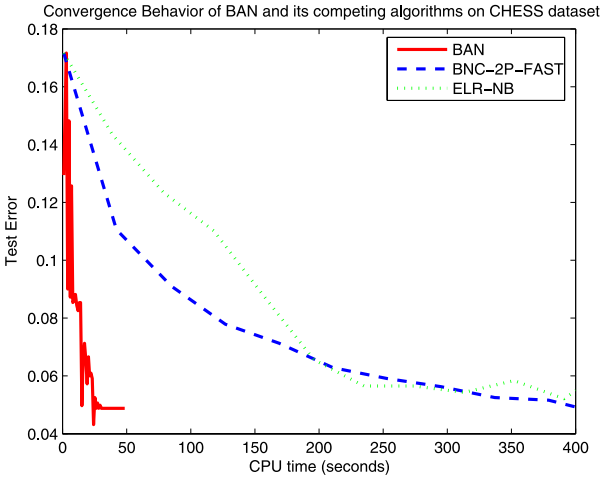


Fig. 9 Classification error on the training data converges faster with *b*AN than ELR and BNC

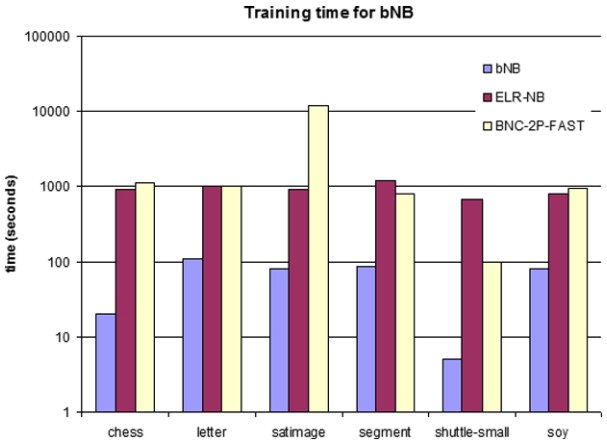
Table 7 The training time comparison between *b*AN and competing algorithms, in seconds

	*b*NB	*b*AN	ELR-NB	BNC-2P-FAST
Small-datasets	2.5	14.0	29.7	94.3
Large-datasets	59.1	266.4	925.5	2701.8

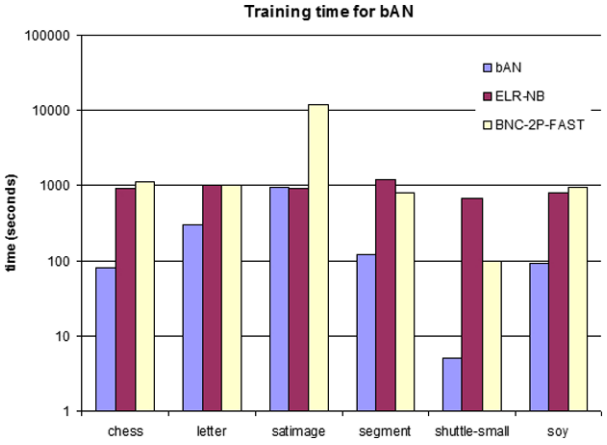
We define large-dataset as those with more than 2000 data points or more than 30 features

data are fully observed. In case of missing data, we can only use BNC-2P, which has a computational cost roughly 2 orders of magnitude larger than *b*AN. On the other hand, *b*AN can simply replace ML parameter learning with EM.

Figure 11 demonstrates that $b\text{AN}_{mix}$ can significantly reduce the number of classifier evaluation in the construction of boosted Bayesian network classifiers. Instead of re-constructing a new ensemble at each iteration of structure search, $b\text{AN}_{mix}$ keeps the



(a) *b*NB is roughly a factor of 20 faster than ELR and a factor of 50 faster than BNC-2P-FAST algorithm.



(b) *b*AN is 2 to 5 times faster than ELR, and 2 to 20 times faster than BNC-2P-FAST

Fig. 10 A break down of training time on large datasets

Bayesian network classifiers generated so far and the new structure only has to model the underlying distribution of data incorrectly classified by the previous classifiers. This strategy avoids reclassifying data points that are far away from boundaries with the new structure. Since $b\text{AN}_{mix}$ and *b*AN are comparable in classification accuracy, we believe reduced computational complexity is the main benefit of $b\text{AN}_{mix}$.

8 Discussion and conclusion

This paper proposes a family of boosting models to improve Bayesian Network classifier framework to improve the classification accuracy of Bayesian network classifiers. Our experiments demonstrate that boosted parameter optimization in conjunction with greedy structure optimization can improve the classification performance. Unlike previous experiments results that combining ELR with structure learning (Greiner and Zhou 2002), we

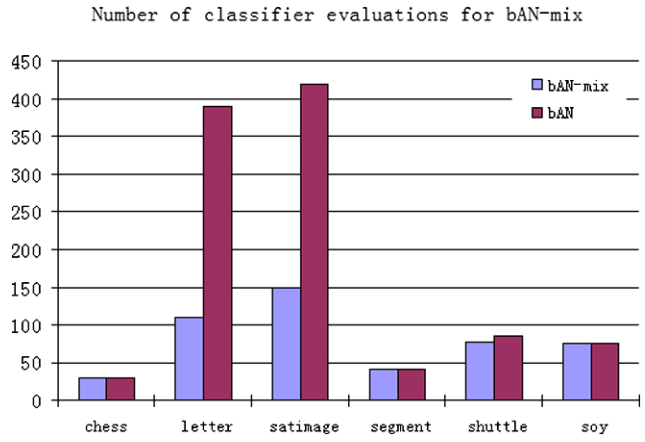


Fig. 11 In cases where *b*AN augments naive Bayes with a significant number of edges ("letter", "satimage"), $b\text{AN}_{mix}$ can significantly reduce the number of classifier evaluations by incrementing adding more complex structures into the ensemble, without re-starting boosting given each new structure

find significant benefit in combining parameter boosting with structure learning. However, full structure search at each stage of boosting can be both computationally expensive and counterproductive—better performance and less overfitting can be achieved when additional edges are gradually and parsimoniously added to the Bayesian network structure in the course of boosting.

We attribute the success of our approach to the following two reasons. First, *b*AN takes advantage of AdaBoost's resistance to over-fitting (Schapire and Singer 1999) and the variance reduction and bias reduction property of ensemble classifiers (Webb and Zheng 2004). Also, as a result of the parameter boosting, the base Bayesian network classifier constructed by *b*AN is simpler than BNC-2P and TAN. In our experiments, *b*AN adds 2 edges to naive Bayes on average while BNC-2P typically adds more than 10 edges.

We believe that the primary advantage of our approach is its simplicity and computational efficiency that scales well in datasets with large class cardinality and feature space dimension, coupled with its good performance in practice. Its use of weighted maximum likelihood parameter learning uniquely determines the parameters of the Bayesian network (in contrast to minimizing a non-convex loss function in ELR), providing an efficient mechanism for discriminative training.

One of the main advantages of using generative models for classification is the ease of encoding domain knowledge into the classifier design. *b*AN relies on the construction of TAN to provide an initial set of structure candidates. However, a user can readily translate his or her domain knowledge into a collection of structure candidates, from which the *b*AN algorithm forms the boosted Bayesian network classifier. In particular, in classification situations where domain information is potentially incomplete or noisy, instead of designing (or learning) a single model, the users may encode the most obvious (and likely to be correct) dependence relationships, while leaving the rest to the discriminative training via AdaBoost.

Our future work will focus on further studying the effects of combining boosting with Bayesian network structure selection. On important issue, in particular, is to understand the relationship between complex Bayesian network structures and the approximate decision boundaries learned by *b*AN from the data generated by those distributions.

Acknowledgements The authors would like to thank Matt Mullin for several fruitful discussions and for suggesting the chain-structured model in Fig. 3. We would like to thank S. Charles Brubaker for his help in constructing the factor graph in Fig. 12. We would also like to thank Pedro Domingos for his helpful suggestions on improving the speed and performance of the BNC-2P algorithm.

This material is based upon work which was supported in part by the National Science Foundation under NSF Grant IIS-0205507 and IIS-0413105.

Appendix: Graphical representation of boosted Bayesian network classifier

In Fig. 12, we present a graphical representation of boosted Bayesian network classifiers using a factor graph. In addition to the class variable X_c and feature variables X_a, we introduce two sets of hidden variables, W and Z. $W_k \in [0, 1]$ corresponds to the estimated posterior of the Bayesian network classifier at the kth iteration of boosting. $Z_k \in \{-1, 1\}$ is the estimated class label after applying MAP estimation to W_k.[11] For binary classification, we use t as the threshold. The factorized functions are written as the following:

$$
\begin{aligned}
f1_k &= \delta\left(w_k, \frac{P_{\theta_k}(x_c, x_a)}{\sum_{x_c \in X_c} P_{\theta_k}(x_c, x_a)}\right), \\
f2_k &= \delta(z_k, \chi(w_k > t)), \quad \chi(w_k > t) = \left\{ \begin{array}{ll} 1 & \text{if } w_k > t \\ -1 & \text{if } w_k \leq t \end{array} \right\}, \\
f3_k &= \exp[\beta_k z_k x_c].
\end{aligned}
\tag{14}
$$

The joint distribution of above factor graph can be written as the follows:

$$
\begin{aligned}
P^*(x_c, x_a) &= \sum_{z \in Z} \int_{w \in [0,1]} P^*(x_c, x_a, z, w) \\
&= \sum_{z \in Z} \int_{w \in [0,1]} \prod_k \exp[\beta_k z_k x_c]\, \delta(z_k, \chi(w_k > t))\, \delta(w_k, P_{\theta_k}(x_c | x_a))
\end{aligned}
$$

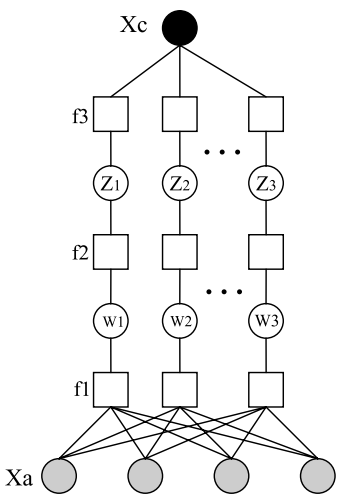


Fig. 12 Representing boosted naive Bayes (binary class) as factor graphs

[11] We want point out that variables W and Z are deterministic given the instantiation of the features.

$$= \prod_k \exp\left[\beta_k \psi_k x_c\right], \quad \text{where } \psi_k = \chi\left(P_{\theta_k}(x_c|x_a) > t\right)$$

where δ refers to Kronecker delta function.

The ratio of the conditional distribution can then be expressed as:

$$\frac{P^*(x_c = 1|x_a)}{P^*(x_c = -1|x_a)} = \frac{P^*(x_c = 1, x_a)}{P^*(x_c = -1, x_a)} = \prod_k \frac{\exp[\beta_k \psi_k]}{\exp[-\beta_k \psi_k]} = \prod_k \exp[2\beta_k \psi_k]$$

$$= \prod_k \exp\left[\psi_k \log \frac{1 - err_k}{err_k}\right] = \prod_k \left(\frac{1 - err_k}{err_k}\right)^{\psi_k} \tag{15}$$

Equation 15 represents the posterior ratio of Discrete AdaBoost. We want to also point out that given the values for ψ, Eq. 15 has similar functional form as naive Bayes under certain constraints.[12]

References

Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003). Hidden Markov support vector machines. In *Proceedings of the 20th international conference on machine learning (ICML)* (pp. 3–10), Washington, DC.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: bagging, boosting, and variants. *Machine Learning*, *36*(1–2), 105–139.

Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html

Chelba, C., & Acero, A. (2004). *Conditional maximum likelihood estimation of naive Bayes probability models using rational function growth transform* (Technical Report MSR-TR-2004-33). Microsoft.

Chickering, D. M., & Heckerman, D. (1997). Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, *29*(2–3), 181–212.

Choudhury, T., Rehg, J. M., Pavlović, V., & Pentland, A. (2002). Boosting and structure learning in dynamic Bayesian networks for audio-visual speaker detection. In *Proceedings of the 16th international conference on pattern recognition* (vol. 3, pp. 789–794).

Chow, C. K., & Liu, C. N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, *14*(3), 462–467.

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

Cutting, D., Kupiec, J., Pedersen, J., & Sibun, P. (1992). A practical part-of-speech tagger. In *Proceedings of the 3rd conference on applied natural language processing* (pp. 133–140). Morristown: Association for Computational Linguistics.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Devroye, L., Györfi, L., & Lugosi, G. (1996). *A probabilistic theory of pattern recognition* (p. 260).

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, *40*(2), 139–157.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th international conference on machine learning (ICML)* (pp. 194–202).

Drucker, H., & Cortes, C. (1996). Boosting decision trees. In *Proceedings of the 8th advances in neural information processing systems (NIPS)* (pp. 470–485).

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley-Interscience.

[12] When the positive error equals the negative error, and when x_c is uniformly distributed, $\left(\frac{1-err_k}{err_k}\right)^{\psi_k} = \left(\frac{\widehat{P}_{D_k}(\psi_k = x_c)}{\widehat{P}_{D_k}(\psi_k \neq x_c)}\right)^{\psi_k} = \frac{\widehat{P}_{D_k}(\psi_k|x_c=1)}{\widehat{P}_{D_k}(\psi_k|x_c=-1)}$, which is analogous to the Maximum Likelihood parameters in symmetric naive Bayes. Instead of x_a, ψ are the features for naive Bayes in this case.

Elkan, C. (1997). *Boosting and naive Bayesian learning* (Technical Report CS97-557). Department of Computer Science and Engineering, University of California, San Diego.

Friedman, J. H. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, *1*(1), 55–77.

Friedman, N., & Koller, D. (2000). Being Bayesian about network structure. In *Proceedings of the 16th conference on uncertainty in artificial intelligence (UAI)* (pp. 201–210), June 2000.

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

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. In *Proceedings of the 16th international joint conference on artificial intelligence (IJCAI)* (p. 1300–1309).

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, *38*(2), 337–374.

Greiner, R., & Zhou, W. (2002). Structural extension to logistic regression: discriminative parameter learning of belief net classifiers. In *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. In *Proceedings of the 21st international conference on machine learning (ICML)* (pp. 361–368). Banff: ACM Press.

Heckerman, D. (1995). *A tutorial on learning with Bayesian networks* (Technical Report MSR-TR-95-06). Microsoft Research.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, *20*(3), 197–243.

Jing, Y., Pavlović, V., & Rehg, J. M. (2005). Efficient discriminative learning of Bayesian network classifiers via boosted augmented naive Bayes. In *Proceedings of the 22nd international conference on machine learning (ICML)* (pp. 369–376).

Jojic, V., Jojic, N., Meek, C., Geiger, D., Siepel, A., Haussler, D., & Heckerman, D. (2004). Efficient approximations for learning phylogenetic HMM models from data. *Bioinformatics*, *20*(1), 161–168.

Keogh, E., & Pazzani, M. (1999). Learning augmented Bayesian classifiers: a comparison of distribution-based and classification-based approaches. In *Proceedings of the 7th international workshop on artificial intelligence and statistics* (pp. 225–230).

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*(1–2), 273–324.

Kohavi, R., & Wolpert, D. H. (1996). Bias plus variance decomposition for zero-one loss functions. In L. Saitta (Ed.), *Machine learning: proceedings of the thirteenth international conference* (pp. 275–283). San Mateo: Morgan Kaufmann.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th international conference on machine learning (ICML)* (pp. 282–289).

Lam, W., & Bacchus, F. (1992). Learning Bayesian belief networks. an approach based on the mdl principle. *Computational Intelligence*, *10*, 269–293.

Langley, P., Iba, W., & Thompson, K. (1992). An analysis of Bayesian classifiers. In *Proceedings of the tenth national conference on artificial intelligence* (pp. 223–228). San Jose: AAAI Press.

McCallum, A., Freitag, D., & Pereira, F. (2000). Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the 17th international conference on machine learning (ICML)* (pp. 591–598), San Francisco, CA.

Nadas, A. (1983). A decision theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood. *IEEE Transactions on Acoustics, Speech, & Signal Processing*, *31*(4), 814.

Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. In *Proceedings of the 14th conference on advances in neural information processing systems (NIPS)* (pp. 841–848).

Niculescu-Mizil, A., & Caruana, R. (2005). Obtaining calibrated probabilities from boosting. In *Proceedings of the 21st conference on uncertainty in artificial intelligence*. Corvallis: AUAI Press.

Pavlović, V., Garg, A., Rehg, J. M., & Huang, T. (2000). Multimodal speaker detection using error feedback dynamic Bayesian networks. In *2000 IEEE computer society conference on computer vision and pattern recognition (CVPR)* (Vol. 2, pp. 34–41).

Pavlović, V., Garg, A., & Kasif, S. (2002). A Bayesian framework for combining gene predictions. *Bioinformatics*, *18*(1), 19–27.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Mateo: Morgan Kaufmann.

Pernkopf, F., & Bilmes, J. (2005). Discriminative versus generative parameter and structure learning of Bayesian network classifiers. In *Proceedings of the 22nd international conference on machine learning (ICML)* (pp. 657–664).

Rabiner, L., & Juang, B. H. (1993). *Fundamentals of speech recognition*. Englewood Cliffs: Prentice Hall.

Rehg, J. M., Pavlović, V., Huang, T. S., & Freeman, W. T. (2003). Special section on graphical models in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(7).

Ridgeway, G., Madigan, D., Richardson, T., & O'Kane, J. (1998). Interpretable boosted naive Bayes classification. In *Proceedings of the 4th international conference on knowledge discovery and data mining*.

Schapire, R. E., & Singer, Y. (1999). Improved boosting using confidence-rated predictions. *Machine Learning*, *37*(3), 297–336.

Schapire, R. E., & Singer, Y. (2000). BoosTexter: a boosting-based system for text categorization. *Machine Learning*, *39*(2/3), 135–168.

Schneiderman, H. (2004). Learning a restricted Bayesian network for object detection. In *Proceedings of the computer society conference on computer vision and pattern recognition (CVPR)* (pp. 639–646), June 2004.

Segal, E., Yelensky, R., & Koller, D. (2003). Genome-wide discovery of transcriptional modules from dna sequence and gene expression. *Bioinformatics*, *19*(1), 273–282.

Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin Markov networks. In *Proceedings of the 16th conference on advances in neural information processing systems (NIPS)* (p. 102). Cambridge: MIT Press.

Torralba, A., Murphy, K. P., & Freeman, W. T. (2004). Sharing features: efficient boosting procedures for multiclass object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (Vol. 2, pp. 762–769), Washington, DC, June 2004.

Webb, G., & Zheng, Z. (2004). Multistrategy ensemble learning: reducing error by combining ensemble learning techniques. *IEEE Transactions on Knowledge and Data Engineering*, *16*(8), 980–991.