Generalized Additive Bayesian Network Classifiers

Jianguo Li[†] [‡] and **Changshui Zhang**[‡] and **Tao Wang**[†] and **Yimin Zhang**[†]
[†]Intel China Research Center, Beijing, China
[‡]Department of Automation, Tsinghua University, China
{jianguo.li, tao.wang, yimin.zhang}@intel.com, zcs@mail.tsinghua.edu.cn

Abstract

Bayesian network classifiers (BNC) have received considerable attention in machine learning field. Some special structure BNCs have been proposed and demonstrate promise performance. However, recent researches show that structure learning in BNs may lead to a non-negligible posterior problem, i.e, there might be many structures have similar posterior scores. In this paper, we propose a generalized additive Bayesian network classifiers, which transfers the structure learning problem to a generalized additive models (GAM) learning problem. We first generate a series of very simple BNs, and put them in the framework of GAM, then adopt a gradient-based algorithm to learn the combining parameters, and thus construct a more powerful classifier. On a large suite of benchmark data sets, the proposed approach outperforms many traditional BNCs, such as naive Bayes, TAN, etc, and achieves comparable or better performance in comparison to boosted Bayesian network classifiers.

1 Introduction

Bayesian networks (BN), also known as probabilistic graphical models, graphically represent the joint probability distribution of a set of random variables, which exploit the conditional independence among variables to describe them in a compact manner. Generally, a BN is associated with a directed acyclic graph (DAG), in which the nodes correspond to the variables in the domain and the edges correspond to direct probabilistic dependencies between them [Pearl, 1988].

Bayesian network classifiers (BNC) characterize the conditional distribution of the class variables given the attributes, and predict the class label with the highest conditional probability. BNCs have been successfully applied in many areas. Naive Bayesian (NB) [Langley *et al.*, 1992] is the simplest BN, which only consider the dependence between each feature x_i and the class variable y. Since it ignores the dependence between different features, NB may perform not well on data sets which violate the independence assumption. Many BNCs have been proposed to overcome NB's limitation. [Sahami, 1996] proposed a general framework to describe the limited dependence among feature variables, called k-dependence Bayesian network (kdB). [Friedman *et al.*, 1997] proposed tree augmented Naive Bayes (TAN), a structure learning algorithm which learns a maximum spanning tree (MST) from the attributes. Both TAN and kdB have tree-structure graph. K2 is an algorithm which learns general BN for classification purpose [Cooper and Herskovits, 1992].

The key differences between these BNCs are their structure learning algorithms. Structure learning is the task of finding out one graph structure that best characterizes the true density of given data. Many criteria, such as Bayesian scoring function, minimal description length (MDL) and conditional independence test [Cheng *et al.*, 2002], have been proposed for this purpose. However, it is inevitable to encounter such a situation: several candidate graph structures have very close score value, and are non-negligible in the posterior sense. This problem has been pointed out and presented theoretic analysis by [Friedman and Koller, 2003]. Since candidate BNs are all approximations of the true joint distribution, it is natural to consider aggregating them together to yield a much more accurate distribution estimation. Several works have been done in this manner. For example, [Thiesson *et al.*, 1998] proposed mixture of DAG, and [Jing *et al.*, 2005] proposed boosted Bayesian network classifiers.

In this paper, a new solution is proposed to aggregate candidate BNs. We put a series of simple BNs into the framework of generalized additive models [Hastie and Tibshirani, 1990], and adopt a gradient-based algorithm to learn the combining parameters, and thus construct a more powerful learning machine. Experiments on a large suite of benchmark data sets demonstrate the effectiveness of the proposed approach.

The rest of this paper is organized as follows. In Section 2, we briefly introduce some typical BNCs, and point out the non-negligible problem in structure learning. In Section 3, we propose the generalized additive Bayesian network classifiers. To evaluate the effectiveness of the proposed approach, extensive experiments are conducted in Section 4. Finally, concluding remarks are given in Section 5.

2 Bayesian Network Classifiers

A Bayesian network B is a directed acyclic graph that encodes the joint probability distribution over a set of random variables $\mathbf{x} = [x_1, \cdots, x_d]^T$. Denote the parent nodes of x_i by $Pa(x_i)$, the joint distribution $P_B(\mathbf{x})$ can be represented by

factors over the network structures as follows:

$$P_B(\mathbf{x}) = \prod_{i=1}^{d} P(x_i|Pa(x_i)).$$

Given data set $D = \{(\mathbf{x}, y)\}$ in which y is the class variable, BNCs characterize D by the joint distribution $P(\mathbf{x}, y)$, and convert it to conditional distribution $P(y|\mathbf{x})$ for predicting the class label.

2.1 Several typical Bayesian network classifiers

The Naive Bayesian (NB) network assumes that each attribute variable only depends on the class variable, i.e,

$$P(\mathbf{x}, y) = P(y)P(\mathbf{x}|y) = P(y)\prod_{i=1}^{d} P(x_i|y).$$

Figure 1(a) illustrates the graph structure of NB.

Since NB ignores the dependencies among different features, it may perform not well on data sets which violate the attribute independence assumption. Many BNCs have been proposed to consider the dependence among features. [Sahami, 1996] presented a more general framework for limited dependence Bayesian networks, called k-dependence Bayesian classifiers (kdB).

Definition 1: A k-dependence Bayesian classifier is a Bayesian network which allows each feature x_i to have a maximum of k feature variables as parents, i.e., the number of variables in $Pa(x_i)$ equals to k+1 ('+1' means that k does not count the class variable y).

According to the definition, NB is a 0-dependence BN. The kdB [Sahami, 1996] algorithm adopts mutual information $I(x_i; y)$ to measure the dependence between the ith feature variable x_i and the class variable y, and conditional mutual information $I(x_i, x_j|y)$ to measure the dependence between two feature variables x_i and x_j. Then kdB employs a heuristic rule to construct the network structure via these two measures.

kdB does not maximize any optimal criterion in structure learning. Hence, it yields limited performance improvement over NB. [Keogh and Pazzani, 1999] proposed super-parent Bayesian networks (SPBN), which assumes that there is an attribute acting as public parent (called super-parent) for all the other attributes. Suppose x_i is the super parent, denote the corresponding BN as $P_i(\mathbf{x}, y)$, we have

$$\begin{aligned} P_i(\mathbf{x}, y) &= P(y)P(x_i|y)P(\mathbf{x}_{\setminus i}|x_i, y) \\ &= P(y)P(x_i|y)\prod_{j=1, j\neq i}^{n} P(x_j|x_i, y). \end{aligned} \quad (1)$$

It is obvious that SPBN structure is a special case of kdB (k=1). Figure 1(b) illustrates the graph structure of SPBN. The SPBN algorithm adopts classification accuracies as the criterion to select out the best network structure.

[Friedman *et al.*, 1997] proposed tree augmented Naive Bayes (TAN), which is also a special case of kdB (k=1). TAN attempts to add edges to the Naive Bayesian network in order to improve the posterior estimation. In detail, TAN first computes the conditional mutual information $I(x_i, x_j|y)$ between any two feature variables x_i and x_j, and thus obtain a full adjacency matrix. Then TAN employs the minimum spanning tree algorithm (MST) on the adjacency matrix to obtain a tree-structure BN. Therefore, TAN is optimal in the sense of MST. Many experiments show that TAN significantly outperforms NB. Figure 1(c) illustrates one possible graph structure of TAN.

Both kdB and TAN generate tree-structure graph. [Cooper and Herskovits, 1992] proposed the K2 algorithm, which adopts the K2 score measure and exhaustive search to learn general BN structures.

2.2 The structure learning problem

Given training data D, structure learning is the task of finding a set of directed edges G that best characterizes the true density of the data. Generally, structure learning can be categorized into two levels: macro-level and micro-level. In the macro-level, several candidate graph structures are known, and we need choosing the best one out. In order to avoid overfitting, people often use model selection methods, such as Bayesian scoring function, minimum descriptive length (MDL), etc [Friedman *et al.*, 1997]. In the micro-level, structure learning cares about whether one edge in the graph should be existed or not. In this case, people usually employ the conditional independence test to determine the importance of edges [Cheng *et al.*, 2002].

However, in both cases, people may face such a situation that several candidates (graphs or edges) have very close scores. For instance, suppose MDL is used as the criterion, people may encounter a situation that two candidate BN structure G_1 and G_2 have MDL score 0.899 and 0.900, respectively. Which one should be chosen? Someone may say that it is natural to select G_1 out since it has a bit smaller MDL score, but practice may show that G_1 and G_2 have similar performance, and G_2 may perform even better in some cases. In fact, both of them are non-negligible in the posterior sense.

This problem has been pointed out and presented theoretic analysis by [Friedman and Koller, 2003]. It shows that when there are many models that can explain the data reasonably well, model selection makes a somewhat arbitrary choice between these models. Besides, the number of possible structures grows super-exponentially with the number of random variables. For these two reasons, we don't want to do structure learning directly. We hope aggregating a series of simpler and weaker BNs together to obtain a much more accurate distribution estimation of the underlying process.

We note that several researchers have proposed some schemes for this purpose, for examples, learning mixtures of DAG [Thiesson *et al.*, 1998], or ensembles of Bayesian networks by model averaging [Rosset and Segal, 2002; Webb *et al.*, 2005]. We briefly introduce them in the following.

2.3 Model averaging for Bayesian networks

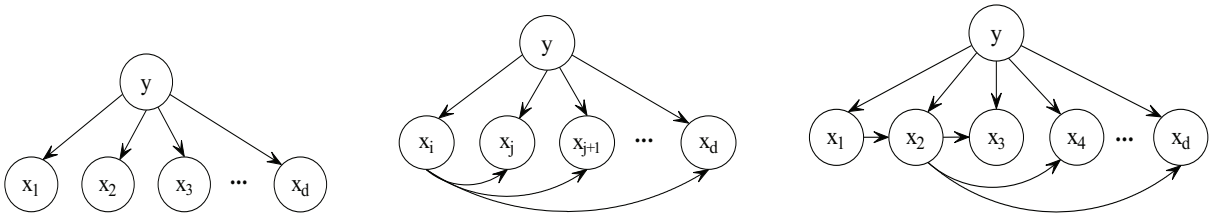
Since candidate BNs are all approximations of the true distribution, model averaging is a natural way to combine candidates together for a more accurate distribution estimation.

Mixture of DAG (MDAG)

Definition 2: If $P(\mathbf{x}|\ \theta_c, G_c)$ is a DAG model, the following equation defines a mixture of DAG model

$$P(\mathbf{x}|\theta_s) = \sum\nolimits_c \pi_c P(\mathbf{x}|\theta_c, G_c),$$

where π_c is the prior for the c-th DAG model G_c, and $\pi_c \geq 0$, $\sum_c \pi_c = 1$, θ_c is the parameter for graph G_c.



(a) Naive Bayesian (b) Super-parent kdB (k=1) (c) One possible structure of TAN

Figure 1: Typical Bayesian network structures

MDAG learns the mixture models via maximizing the posterior likelihood of given data set. In detail, MDAG combining uses the Chesseman-Stutz approximation and the Expectation-Maximization algorithm for both the mixture components structure learning and the parameter learning.

[Webb *et al.*, 2005] presented a special and simple case of MDAG for classification purpose, called the average one dependence estimation (AODE). AODE adopts a series of fix-structure simple BNs as the mixture components, and directly assumes that all mixture components in MDAG have equal mixture coefficient. Practices show that AODE outperforms Naive Bayes and TAN.

Boosted Bayesian networks

Boosting is another commonly used technique for combining simple BNs. [Rosset and Segal, 2002] employed the gradient Boosting algorithm [Friedman, 2001] to combine BNs for density estimation. [Jing *et al.*, 2005] proposed boosted Bayesian network classifiers (BBN), and adopted general AdaBoost algorithm to learn the weight coefficients.

Given a series of simple BNs: $P_i(\mathbf{x}, y),\ i = 1, \cdots, n$, BBN aims to construct the final approximation by linear additive models: $P(\mathbf{x}, y) = \sum_{i=1}^{n} \alpha_i P_i(\mathbf{x}, y)$, where $\alpha_i \geq 0$ are weight coefficients, and $\sum_i \alpha_i = 1$. More generally, the constraint on α_i can be relaxed, but only $\alpha_i \geq 0$ is kept: $F(\mathbf{x}, y) = \sum_{i=1}^{n} \alpha_i P_i(\mathbf{x}, y)$. In this case, the posterior can be defined as follows

$$P(y|\mathbf{x}) = \frac{\exp\{F(\mathbf{x}, y)\}}{\sum_{y'} \exp\{F(\mathbf{x}, y')\}}. \tag{2}$$

For general binary classification problem $y \in$ {-1,1}, this problem can be solved by the exponent loss function

$$\mathcal{L}(\alpha) = \sum\nolimits_k \exp\{-yF(\mathbf{x}_k, y)\} \tag{3}$$

via the AdaBoost algorithm [Friedman *et al.*, 2000].

3 Generalized additive Bayesian networks

In this section, we present a novel scheme that can aggregate a series of simple BNs to a more accurate density estimation of the true process. Suppose $P_i(\mathbf{x}, y),\ i = 1, \cdots, n$ are the given simple BNs, we consider putting them in the framework of generalized additive models (GAM) [Hastie and Tibshirani, 1990]. The new algorithm is called generalized additive Bayesian network classifiers (GABN).

In the GAM framework, $P_i(\mathbf{x}, y)$ are considered to be linear additive variables in the link function space:

$$F(\mathbf{x}, y) = \sum\nolimits_i^n \lambda_i f_i[P_i(\mathbf{x}, y)]. \tag{4}$$

GABN is an extensible framework since many different link functions can be considered. In this paper, we study a special link function: $f_i(\cdot) = \log(\cdot)$. Defining $z = (\mathbf{x}, y)$ and taking exponent on both sides of the above equation, we have

$$\exp[F(z)] = \exp\Big[\sum\nolimits_i^n \lambda_i f_i(z)\Big] = \prod\nolimits_i^n P_i^{\lambda_i}(z).$$

This is in fact a potential function. It can also be written as a probabilistic distribution when given a normalization factor,

$$P(z) = \frac{1}{S_\lambda(z)} \prod_i^n P_i^{\lambda_i}(z), \tag{5}$$

where $S_\lambda(z)$ is the normalization factor:

$$S_\lambda(z) = \sum_z \Big\{ \prod_i^n P_i^{\lambda_i}(z) \Big\} = \sum_z \exp\Big\{ \sum_{i=1}^n \lambda_i \log P_i(z) \Big\}. \tag{6}$$

The likelihood of $P(z)$ is called quasi-likelihood:

$$\begin{aligned} \mathcal{L}(\lambda) &= \sum\nolimits_{k=1}^N \log P(z_k) \\ &= \sum\nolimits_{k=1}^N \Big\{ \sum_{i=1}^n \lambda_i \log P_i(z_k) - \log S_\lambda(z_k) \Big\} \\ &= \sum\nolimits_{k=1}^N \Big\{ \lambda \cdot \mathbf{f}(z_k) - \log S_\lambda(z_k) \Big\}, \end{aligned} \tag{7}$$

where $\lambda = [\lambda_1, \cdots, \lambda_n]^T$, $\mathbf{f}(z_k) = [f_1(z_k), \cdots, f_n(z_k)]^T$.

3.1 The Quasi-likelihood optimization problem

Maximizing the quasi-likelihood, we can obtain the solution of the additive parameters. To make the GAM model meaningful and tractable, we add some constraints to the parameters. The final optimization problem turns to be:

$$\begin{aligned} \max \quad & \mathcal{L}(\lambda) \\ s.t. \quad & (1)\ 0 \leq \lambda_i \leq 1 \\ & (2)\ \textstyle\sum_i \lambda_i = 1 \end{aligned} \tag{8}$$

For equation constraint, the Lagrange multiplier can be adopted to transfer the problem into an unconstraint one; while for inequation constraints, classical interior point method (IPM) can be employed. In detail, the IPM utilizes barrier functions to transfer inequation constraints into a series of unconstraint optimization problems [Boyd and Vandenberghe, 2004].

Here, we adopt the most used logarithmic barrier function, and obtain the following unconstraint optimization problem:

$$\begin{aligned}\mathcal{L}(\lambda, r_k, \alpha) &= r_k \sum_{i=1}^{n} \log(\lambda_i) + r_k \sum_{i=1}^{n} \log(1-\lambda_i) \\ &+ \alpha(1 - \sum_{i=1}^{n} \lambda_i) + \mathcal{L}(\lambda) \\ &= r_k[\log(\lambda) + \log(\mathbf{1}_n - \lambda)] \cdot \mathbf{1}_n \\ &+ \alpha(1 - \lambda \cdot \mathbf{1}_n) + \mathcal{L}(\lambda), \end{aligned} \quad (9)$$

where $\mathbf{1}_n$ indicates a n-dimensional vector with all elements equal to 1, r_k is the barrier factor in the kth step of the IPM iteration, and α is the Lagrange multiplier.

Therefore, in the kth IPM iteration step, we need to maximize an unconstraint problem $\mathcal{L}(\lambda, r_k, \alpha)$. Quasi-Newton method is adopted for this purpose.

3.2 Quasi-Newton method for the unconstraint optimization problem

To solve the unconstraint problem: $\max \mathcal{L}(\lambda, r_k, \alpha)$, we must have the gradient of $\mathcal{L}$ w.r.t λ.

Theorem 1: The gradient of $\mathcal{L}(\lambda, r_k, \alpha)$ w.r.t λ is

$$\begin{aligned}\frac{\partial \mathcal{L}(\lambda, r_k, \alpha)}{\partial \lambda} &= \mathbf{g}_\lambda = \sum_{k=1}^{N} \left\{ \mathbf{f}(z_k) - E_{P(z)}[\mathbf{f}(z_k)] \right\} \\ &+ r_k[\frac{1}{\lambda} - \frac{1}{\mathbf{1}_n - \lambda}] \cdot \mathbf{1}_n - \alpha \mathbf{1}_n. \end{aligned} \quad (10)$$

Proof: In Equation (10), it is easy to obtain the gradient of the first summation term and non-summation terms. Here, we only present the gradient solution of the second summation term, i.e., $\log S_\lambda(z_k)$ in $\mathcal{L}(\lambda)$.

$$\frac{\partial \log S_\lambda(z)}{\partial \lambda} = \frac{1}{S_\lambda(z)} \frac{\partial S_\lambda(z)}{\partial \lambda}$$

$$\because S_\lambda(z) = \sum_z \exp\left\{\lambda \cdot \mathbf{f}(z)\right\}$$

$$\frac{\partial S_\lambda(z)}{\partial \lambda} = \sum_{z_k} \mathbf{f}(z_k) \exp\left\{\lambda \cdot \mathbf{f}(z_k)\right\}$$

$$\begin{aligned}\therefore \frac{\partial \log S_\lambda(z)}{\partial \lambda} &= \frac{1}{S_\lambda(z)} \sum_{z_k} \mathbf{f}(z_k) \exp\left\{\lambda \cdot \mathbf{f}(z_k)\right\} \\ &= \sum_{z_k} P(z_k)\mathbf{f}(z_k) = E_{P(z)}[\mathbf{f}(z_k)]. \end{aligned} \quad \square$$

For computational cost consideration, we did not further compute the second order derivative of $\mathcal{L}(\lambda, r_k, \alpha)$, while adopted the quasi-Newton method [Bishop, 1995] to solve the problem. In this paper, the L-BFGS procedure provided by [Liu and Nocedal, 1989] is employed for this task.

3.3 The IPM based training algorithm

The interior point method starts from a point in the feasible region, sequentially adjusts the barrier factor r_k in each iteration, and solves a series unconstraint problem $\mathcal{L}(\lambda, r_k, \alpha)$, $k = 1, 2, \cdots$. The detailed training algorithm is shown in Table 1.

Table 1: The training algorithm for GABN

Input: Given training set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$
Training Algorithm
S0: set convergence precision $\epsilon > 0$, and the maximal step M;
S1: initialize the interior point as $\lambda = [\lambda_1, \cdots, \lambda_n]^T$, $\lambda_i = 1/n$;
S2: generate a series of simple BNs: $P_i(\mathbf{x}, y)$, $i = 1, \cdots, n$;
S3: for $k = 1 : M$
  S4: select $r_k > 0$ and $r_k < r_{k-1}$,
    obtain the kth step optimization problem $\mathcal{L}(\lambda, r_k, \alpha)$;
  S5: calculate $\mathbf{g}_\lambda$ and the quasi-likelihood $\mathcal{L}(\lambda)$;
  S6: employ L-BFGS procedure to solve: $\max \mathcal{L}(\lambda, r_k, \alpha)$;
  S7: test of the barrier term
    $$a_k = r_k[\log(\lambda) + \log(\mathbf{1}_n - \lambda)] \cdot \mathbf{1}_n;$$
  S8: if $a_k < \epsilon$ jump to *S9*, else continue the loop;
S9: Output the optimal parameter λ^*,
  and obtain the final generalized models $P(z; \lambda^*)$.

3.4 A series of fix-structure Bayesian networks

There are one unresolved problem in the algorithm listed in Table 1, which is in the *S2* step, i.e, how to generate a series of simple BNs as the weak learner. There are many methods for this purpose. In our experiments, we take super parent BN as the weak learner. Readers may consider other possible strategies to generate simple BNs.

For a d-dimensional data set, when setting different attribute as the public parent node according to Equation (1), it can generate d different fix-structure super-parent BNs: $P_i(\mathbf{x}, y)$, $i = 1, \cdots, d$. Figure 1(b) depicts one example of this kind of simple BNs. To improve performance, mutual information $I(x_i, y)$ is computed for removing several BNs with lowest mutual information score. In this way, we obtain n very simple BNs, and adopt them as weak learners in GABN.

Parameters (conditional probabilistic table) learning in BNs is common, and thus details are omitted here. Note that for robust parameter estimation, Laplacian correction and m-estimate [Cestnik, 1990] are adopted.

3.5 Discussions

GABN has several advantages over the typical linear additive BN models: Boosted BN (BBN). First, GABN is much more computational efficient than BBN. Given d-dimensional and N samples training set, it is not hard to prove that the computational complexity of GABN is $O(Nd^2 + MNd)$, where M is the IPM iteration steps. On the contrary, BBN requires sequentially learning BN structures in each boosting step. This leads to a complexity of $O(KNd^2)$, where K is the boosting step, which is usually very large (in 10^2 magnitude). Therefore, GABN dominates BBN on scalable learning task. Practice also demonstrates this point.

Furthermore, GABN presents a new direction for combining weaker learners since it is a highly extensible framework. We present a solution for logarithmic link function. It is not hard to adopt other link functions under the GAM framework, and thus propose new algorithms. Many existing GAM properties, optimization methods can be seamlessly adopted to ag-

gregate simple BNs for more powerful learning machines.

4 Experiments

This section evaluates the performance of the proposed algorithm, compared it with other BNCs such as NB, TAN, K2, *k*dB, SPBN; model averaging methods such as AODE, BBN; and decision tree algorithm CART [Breiman *et al.*, 1984].

The benchmark platform was 30 data sets from the UCI machine learning repository [Newman *et al.*, 1998]. One point should be indicated here: for BNCs, when data sets have continuous features, we first adopted discretization method to transfer them into discrete features [Dougherty *et al.*, 1995]. We employed 5-fold cross-validation for the error estimation, and kept all compared algorithms having the same fold split. The final results are shown in Table 2, in which the results by TAN and K2 are obtained by the Java machine learning toolbox Weka [Witten and Frank, 2000].

To present statistical meaningful evaluation, we conducted the paired *t*-test to compare GABN with others. The last row of Table 2 shows the win/tie/lose summary in 10% significance level of the test. In addition, Figure 2 illustrates the scatter plot of the comparison results between GABN and other classifiers. We can see that GABN outperforms most other BNCs, and achieves comparable performance to BBN. Specially note, the SPBN column shows results by the best individual super-parent BN, which are significant worse than GABN. This demonstrates that it is effective and meaningful to use GAM for aggregating simple BNs.

5 Conclusions

In this paper, we propose a generalized additive Bayesian network classifiers (GABN). GABN aims to avoid the non-negligible posterior problem in Bayesian network structure learning. In detail, we transfer the structure learning problem to a generalized additive models (GAM) learning problem. We first generate a series of very simple Bayesian networks (BN), and put them in the framework of GAM, then adopt a gradient-based learning algorithm to combine those simple BNs together, and thus construct a more powerful classifiers. Experiments on a large suite of benchmark data sets demonstrate that the proposed approach outperforms many traditional BNCs such as naive Bayes, TAN, etc, and achieves comparable or better performance in comparison to boosted Bayesian network classifiers. Future work will focus on other possible extensions within the GABN framework.

References

[Bishop, 1995] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, London, 1995.

[Boyd and Vandenberghe, 2004] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[Breiman *et al.*, 1984] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification And Regression Trees*. Wadsworth International Group, 1984.

[Cestnik, 1990] B. Cestnik. Estimating probabilities: a crucial task in machine learning. In *the 9th European Conf. Artificial Intelligence (ECAI)*, pages 147–149, 1990.

[Cheng *et al.*, 2002] J. Cheng, D. Bell, and W. Liu. Learning belief networks from data: An information theory based approach. *Artificial Intelligence*, 137:43–90, 2002.

[Cooper and Herskovits, 1992] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[Dougherty *et al.*, 1995] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *the 12th Intl. Conf. Machine Learing (ICML)*, San Francisco, 1995. Morgan Kaufmann.

[Friedman and Koller, 2003] N. Friedman and D. Koller. Being Bayesian about network structure: a Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50:95–126, 2003.

[Friedman *et al.*, 1997] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2):131–163, 1997.

[Friedman *et al.*, 2000] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(337-407), 2000.

[Friedman, 2001] J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5), 2001.

[Hastie and Tibshirani, 1990] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman & Hall, 1990.

[Jing *et al.*, 2005] Y. Jing, V. Pavlović, and J. Rehg. Efficient discriminative learning of Bayesian network classifiers via boosted augmented naive Bayes. In *the 22nd Intl. Conf. Machine Learning (ICML)*, pages 369–376, 2005.

[Keogh and Pazzani, 1999] E. Keogh and M. Pazzani. Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *7th Intl. Workshop Artificial Intelligence and Statistics*, pages 225–230, 1999.

[Langley *et al.*, 1992] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *the 10th National Conf. Artificial Intelligenc (AAAI)*, pages 223–228, 1992.

[Liu and Nocedal, 1989] D. Liu and J. Nocedal. On the limited memory BFGS method for large-scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[Newman *et al.*, 1998] D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases, 1998.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Rosset and Segal, 2002] S. Rosset and E. Segal. Boosting density estimation. In *Advances in Neural Information Processing System (NIPS)*, 2002.

[Sahami, 1996] M. Sahami. Learning limited dependence Bayesian classifiers. In *the 2nd Intl. Conf. Knowledge Discovery and Data Mining (KDD)*, pages 335–338. AAAI Press, 1996.

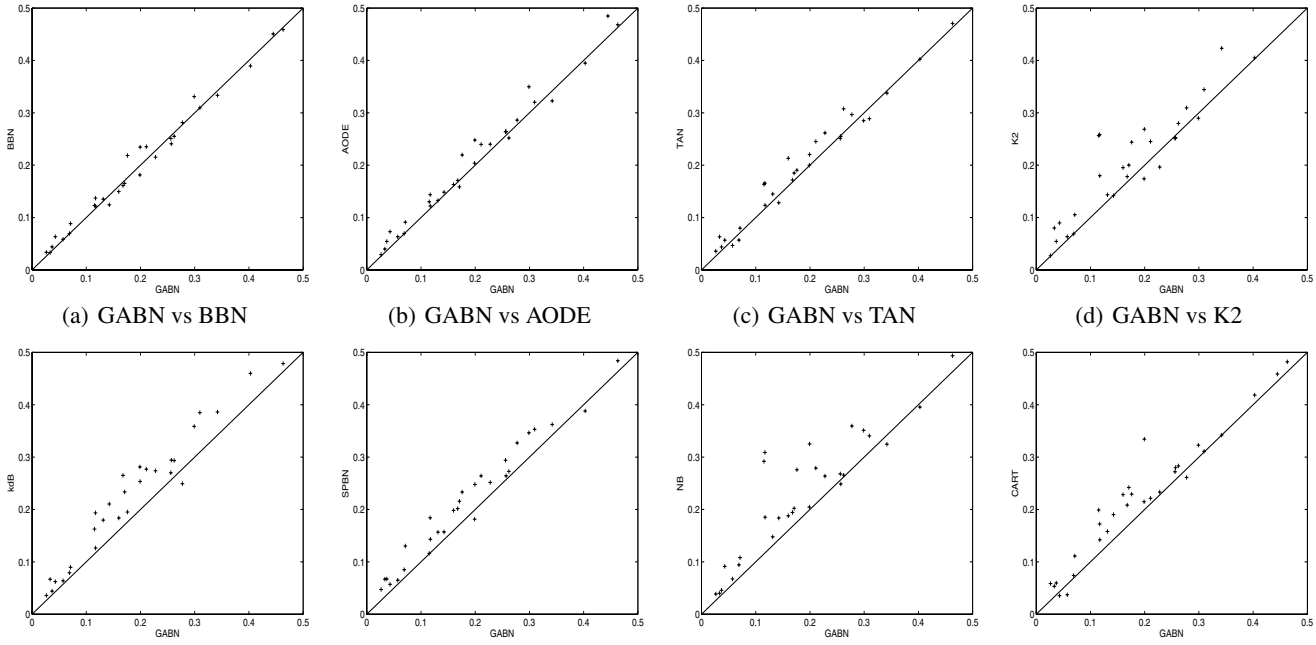
[Thiesson *et al.*, 1998] B. Thiesson, C. Meek, D. Heckerman, and et al. Learning mixtures of DAG models. In *Conf. Uncertainty in Artificial Intelligence (UAI)*, pages 504–513, 1998.

[Webb *et al.*, 2005] G. Webb, J. R. Boughton, and Zhihai Wang. Not so naive Bayes: aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24, 2005.

[Witten and Frank, 2000] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, 2000.

Table 2: Testing error on 30 UCI data sets

dataset	BBN	GABN	AODE	TAN	K2	*k*dB	SPBN	NB	CART
australia	.1352	**.1313**	.1328	.1449	.1435	.1797	.1566	.1478	.1580
autos	.2185	**.1760**	.2195	.1903	.2439	.1952	.2333	.2760	.2293
breast-cancer	.2551	.2621	**.2520**	.3077	.2797	.2935	.2727	.2656	.2832
breast-w	.0337	**.0264**	.0293	.0358	.0272	.0358	.0472	.0386	.0586
cmc	**.4589**	.4630	.4683	.4705	.5017	.4787	.4840	.4935	.4820
cylinder-band	.2345	**.1994**	.2479	.2204	.2686	.2535	.2478	.3252	.3343
diabetes	**.2408**	.2569	.2642	.2552	.2526	.2943	.2643	.2486	.2799
german	**.2510**	.2560	.2640	**.2510**	**.2510**	.2700	.2940	.2680	.2720
glass	.3311	.2990	.3497	**.2851**	.2897	.3586	.3464	.3511	.3228
glass2	.2154	.2277	.2400	.2617	**.1964**	.2740	.2516	.2637	.2331
heart-c	**.1611**	.1678	.1710	.1716	.1783	.2651	.2017	.1944	.2083
heart-stat	.1815	.1989	.2037	.2000	**.1741**	.2813	.1815	.2047	.2148
ionosphere	.0884	**.0711**	.0911	.0798	.1054	.0896	.1303	.1082	.1111
iris	**.0333**	**.0333**	.0400	.0633	.0800	.0667	.0667	.0400	.0533
letter	.1371	**.1170**	.1437	.1656	.2583	.1935	.1842	.3088	.1722
liver	.3333	.3420	**.3228**	.3376	.4232	.3862	.3623	.3246	.3420
lymph	**.1243**	.1425	.1483	.1284	.1419	.2104	.1572	.1837	.1902
page-blocks	.0585	.0572	.0634	.0468	.0636	.0639	.0650	.0674	**.0373**
post-operative	.3096	.3096	.3202	**.2889**	.3444	.3850	.3531	.3406	.3111
satimg	.1218	**.1172**	.1221	.1234	.1799	.1265	.1430	.1851	.1422
segment	.0636	.0429	.0732	.0568	.0897	.0623	.0571	.0913	**.0355**
sonar	.2353	**.2108**	.2397	.2452	.2452	.2770	.2643	.2789	.2213
soybean-big	.0696	.0690	.0696	**.0569**	.0689	.0793	.0851	.0943	.0740
tae	.4506	**.4448**	.4848	.5497	.5232	.5170	.5101	.5433	.4585
vehicle	.2814	.2773	.2860	.2967	.3095	**.2493**	.3272	.3594	.2612
vowel	.1232	**.1152**	.1303	.1636	.2566	.1626	.1162	.2919	.1990
waveform	**.1496**	.1600	.1630	.2132	.1954	.1838	.1982	.1880	.2284
wavef+noise	.1652	.1708	**.1586**	.1850	.2000	.2336	.2156	.2022	.2420
wdbc	.0441	**.0370**	.0545	.0439	.0545	.0440	.0673	.0458	.0597
yeast	.3895	.4028	.3949	.4023	.4050	.4598	**.3881**	.3955	.4185
average	.1965	.1928	.2049	.2080	.2250	.2323	.2224	.2375	.2211
win/tie/lose	11/10/9	–	17/9/4	19/6/5	21/6/3	29/0/1	27/1/2	26/1/3	23/4/3



(a) GABN vs BBN (b) GABN vs AODE (c) GABN vs TAN (d) GABN vs K2

(e) GABN vs *k*dB (f) GABN vs SPBN (g) GABN vs NB (h) GABN vs CART

Figure 2: Scatter plots for experimental results on 30 UCI data sets. Each plot shows the relative error rate of GABN and one compared algorithm. Points above the diagonal line correspond to data sets where GABN performs better than the compared algorithm.