# Randomized Bayesian Network Classifiers

Qing Wang[1,2] and Ping Li[2]

[1] School of Computer Science, Fudan University, Shanghai, China
wangqing@fudan.edu.cn
[2] School of Management Science and Engineering, Anhui University of Technology, Ma'anshan, China
pingli@ahut.edu.cn

**Abstract.** In this paper, we propose Randomized Bayesian Network Classifiers (RBNC). It borrows the idea of ensemble learning by constructing a collection of semi-naive Bayesian network classifiers and then combines their predictions as the final output. Specifically, the structure learning of each component Bayesian network classifier is performed by just randomly choosing the parent of each attribute in addition to class attribute, and parameter learning is performed by using maximum likelihood method. RBNC retains many of naive Bayes' desirable property, such as scaling linearly with respect to both the number of instances and attributes, needing a single pass through the training data and robust to noise, etc. On the 60 widely used benchmark UCI datasets, RBNC outperforms state-of-the-art Bayesian classifiers.

## 1 Introduction

A Bayesian network [1] encodes the joint probability distribution of a set of variables as a directed acyclic graph (DAG) and a set of conditional probability tables (CPTs). Its modularity and intuitive graphical representation make it an attractive model for real world problems, and their use for classification has received considerable attentions [2,3]. Assume that $X_1, X_2, ..., X_a$ are $a$ attributes (corresponding to attribute nodes in a Bayesian network). An instance $I$ is represented by a vector $(x_1, x_2, ..., x_a)$, where $x_i$ is the value of $X_i$. Let $C$ represent the class variable (corresponding to the class node in a Bayesian network). We use $c$ to represent the value that $C$ takes and $c(I)$ to denote the class label of $I$. A Bayesian network classifier predicts the class label of instance $I$ using Equation 1.

$$c(I) = \arg\max_{c \in C} P(c)P(x_1, x_2, ..., x_a|c) \tag{1}$$

Assume that all attributes are independent given the class, that is,

$$c(I) = \arg\max_{c \in C} P(c) \prod_{i=1}^{a} P(x_i|c) \tag{2}$$

This assumption is called conditional independence assumption and the resulting classifier is called a naive Bayesian classifier, or simply naive Bayes.

Naive Bayes is the simplest form of Bayesian network classifier and has been widely applied to many real world applications [4,5,6,7,8]. Despite the fact that the conditional

Z.-H. Zhou, F. Roli, and J. Kittler (Eds.): MCS 2013, LNCS 7872, pp. 319–330, 2013.

independence assumptions are often inaccurate, the naive Bayes classifier has several properties that make it surprisingly useful in practice. In particular, both the time and space complexity grow linearly with respect to both the number of instances and attributes, the learning can be done with a single pass through the training data and the performance is robust to noise, etc.

It is obvious that the conditional independence assumption in naive Bayes is rarely true. To relax this assumption, many techniques have been proposed. Extending its structure is a direct way to overcome the limitation of naive Bayes, since attribute dependencies can be explicitly represented by adding arcs. Learning Bayesian networks has become an active research in the past decade [3,9,10]. The goal of learning a Bayesian network is to determine both the structure of the network (structure learning) and the set of CPTs (parameter learning). Since the number of possible structures is extremely huge, structure learning often has high computational complexity. Thus, heuristic and approximate learning algorithms are the realistic solution. A variety of learning algorithms have been proposed, such as TAN [2], BNC[9], HNB [11], $\hat{f}$CLL[3], A$n$DE[6], etc. Most of these algorithms achieve improved accuracy over naive Bayes. However, this is achieved at the cost of **increasing the order of computational complexity** which severely limits its applicability in practice, especially for large-scale and high-dimensional data.

In fact, a model that could relax conditional independence assumption and also retain many of naive Bayes' desirable computational and theoretical properties, is more desirable. In this paper, we present a new model Randomized Bayesian Network Classifiers (RBNC). It borrows the idea from ensemble learning paradigms by constructing a collection of semi-naive Bayesian network classifiers and then combining their predictions as the final output. Specifically, the structure learning of each component Bayesian network classifier is performed by just randomly choosing the parent of each attribute in addition to class attribute, and parameter learning is performed by using maximum likelihood method (i.e. frequency counting). Our experimental results show that RBNC demonstrates remarkable accuracy compared to other state-of-the-art algorithms.

The rest of the paper is organized as follows. We first introduce the related work. Then we present our new model RBNC, followed by the description of our experimental setup and results in detail. Finally, the paper is concluded in section 5.

## 2 Related Work

Numerous techniques have been proposed to improve or extend naive Bayes, mainly in two approaches: selecting or forming new attribute subsets in which attributes are conditionally independent, and extending the structure of naive Bayes to represent attribute dependencies.

The idea of selecting a subset of attributes or forming new attributes is to convert the data to a new form that satisfies the conditional independence assumption. Of the proposed techniques, selective naive Bayes (SBC) by [12] demonstrates a remarkable improvement over naive Bayes. SBC uses forward selection to find a good subset of attributes, and then uses this subset to construct a naive Bayes.

Learning Bayesian networks has become an active research in the past decade. The goal of learning consists of determining both the structure of the network and the set of CPTs. Since the number of possible structures is extremely huge, structure learning often has high computational complexity. Moreover, learning unrestricted Bayesian network seems to not necessarily lead to a classifier with good performance. Thus, heuristic and approximate learning algorithms are the realistic solution. For example, [2] proposed Tree Augmented Naive Bayes (TAN), a structure learning algorithm that learns a maximum spanning tree from the attributes, but retains naive Bayes model as a part of its structure to bias towards the estimation of conditional distribution. BNC-2P [9], on the other hand, is a heuristic discriminative structure learning method with conditional log likelihood as scoring function. Although the structures in TAN and BNC-2P are selected discriminatively, the parameters are trained via maximum likelihood training for computational efficiency.

Factorized conditional log-likelihood ($\hat{f}$CLL) [3] is the most recently proposed score function for learning Bayesian network classifiers. It is an approximation of the conditional log-likelihood criterion, and is devised in order to guarantee decomposability over the network structure as well as efficient estimation of the optimal parameters. This discriminative criteria achieves the same time and space complexity as the log-likelihood scoring function. The experimental results show that $\hat{f}$CLL trained TAN achieves improved accuracy over other discriminatively trained Bayesian network classifiers.

Hidden Naive Bayes (HNB) [13,11] using a predefined network structure to take the influences from all attributes into account. In HNB, each attribute $X_i$ has a hidden parent $X_{hp_i}$ which combines the influences from all other attributes. The classifier corresponding to an HNB on an instance $I = (x_1, ..., x_a)$ is defined as follows:

$$c(I) = \arg\max_{c \in C} P(c) \prod_{i=1}^{a} P(x_i | X_{hp_i}, c) \tag{3}$$

where

$$P(x_i | X_{hp_i}, c) = \sum_{j=1, j \neq i}^{a} w_{ij} P(x_i | x_j, c) \tag{4}$$

The weight $w_{ij}$ is defined by the conditional mutual information between two attributes $X_i$ and $X_j$. The hidden parent $X_{hp_i}$ for $X_i$ is essentially a mixture of the weighted influences from all other attributes. Since there is no structure learning, learning an HNB is mainly about estimating the parameters from the training data. To create the hidden parent of an attribute, HNB needs to compute the conditional mutual information for each pair of attributes.

The most recent work on improving naive Bayes is A*n*DE (averaged *n*-dependence estimators) [6] which is an generalization of the well-known AODE (averaged one-dependence estimators) [5] algorithm. In A*n*DE, an ensemble of *n*-dependence classifiers are learned and the prediction is produced by aggregating the predictions of all qualified classifiers. An *x*-dependence estimator means that the probability of an attribute is conditioned by the class variable and at most *x* other attributes. In A*n*DE, a *n*-dependence classifier is built for every combination of $n$ attributes, in which the given

*n* attributes are set to be the parent of all other attributes. A*n*DE predicts the class label of instance *I* using Equation 5.

$$c(I) = \arg\max_{c \in C} \sum_{s \in S^n} P(c, \mathbf{s}) \prod_{j=1, j \notin s}^{a} P(x_j | c, \mathbf{s}) \tag{5}$$

where $S^n$ indicates the set of all size-*n* subsets of $\{x_1, ..., x_a\}$. The experimental results show that the bias-variance trade-off for A2DE results in strong predictive accuracy over a wide range of data sets. Another reason for the authors presenting primarily results for A2DE is because the computational complexity (both space and time) of A*n*DE($n \geq 3$) is very high and defeats their Weka implementation on most data sets [6]. The ensemble size is $a$ (the number of attributes) for both AODE and A2DE.

Table 1 shows the training time and space complexity of some algorithms discussed.

**Table 1.** Computational complexity of algorithms

| | Training Complexity | | Testing Complexity | |
|---|---|---|---|---|
| Algorithm | Time | Space | Time | Space |
| NB | $O(ta)$ | $O(kav)$ | $O(ka)$ | $O(kav)$ |
| TAN | $O(ta^2 + k(av)^2 + a^2 \log a)$ | $O(k(av)^2)$ | $O(ka)$ | $O(kav^2)$ |
| HNB | $O(ta^2 + k(av)^2)$ | $O(k(av)^2)$ | $O(ka^2)$ | $O(k(av)^2)$ |
| AODE | $O(ta^2)$ | $O(k(av)^2)$ | $O(ka^2)$ | $O(k(av)^2)$ |
| A*n*DE | $O(t\binom{a}{n+1})$ | $O(k\binom{a}{n+1}v^{n+1})$ | $O(kn\binom{a}{n})$ | $O(k\binom{a}{n+1}v^{n+1})$ |
| RBNC-*n* | $O(Ntan)$ | $O(Nkav^{n+1})$ | $O(Nkan)$ | $O(Nkav^{n+1})$ |

*k* is the number of classes.
*a* is the number of attributes.
*v* is the average number of values for an attribute.
*t* is the number of training examples.
*n* is the number of parent nodes except class.
*N* is the number of component models of RBNC.

## 3 The RBNC Algorithm

In this section, we introduce the RBNC family of algorithms and analyze its computational complexity.

### 3.1 Algorithm Definition

Instead of searching for a single Bayesian network classifier model by optimizing some (discriminative or generative) score on data, RBNC randomly constructs multiple Bayesian network classifier models and then simply average their probability predictions as the final output.

We focus on *augmented naive Bayes classifiers*, that is, Bayesian network classifiers where the class attribute has no parents and all attributes have at least the class attribute as parent. In addition, we introduce a parameter $n$ to control the maximum number of

**Algorithm 1.** RBNC algorithm

**Input:** Training data $D$, where $< X_1, ..., X_a >$ and $C$ represent $a$ input attributes and class attribute, respectively. Maximum number of parents (except class) per node $n$ and number of component models $N$.
**Output:** A set of Bayesian network classifier models $E$.

Initialize $E = \{\}$.
$/*$ structure learning $*/$
**for** $i = 1$ **to** $N$ **do**
  Generate a random permutation $< A_1, .., A_a >$ of the given $a$ input attributes.
  Initialize an empty Bayesian network model $M_i$ with $a + 1$ node.
  For $M_i$, set class attribute $C$ as parent for all other attributes.
  **for** $j = 2$ **to** $a$ **do**
    **if** $j \leq n$ **then**
      For $M_i$, set all attributes in $\{A_1, ..., A_{j-1}\}$ as parent of attribute $A_j$.
    **else**
      For $M_i$, randomly select $n$ attributes in $\{A_1, ..., A_{j-1}\}$ as parent of attribute $A_j$.
    **end if**
  **end for**
  $E$=$E \bigcup M_i$.
**end for**
$/*$ parameter learning $*/$
Compute the CPTs for all $M_i \in E$ on data $D$ using maximum likelihood.
**return** $E$

parents per node in the network. The structure of each component Bayesian network classifier in RBNC is constructed by just randomly choosing $n$ other attributes as the parents for each attribute in addition to class attribute. To ensure the generated structure is DAG, first, all the attributes are ordered, then each attribute can only select those ahead of it as parents. The parameters in each component network are set to their maximum likelihood values, i.e. observed frequency counting over the data. The detailed learning process of RBNC is depicted in Algorithm 1.

RBNC predicts the class label of instance *I* using:

$$c(I) = \arg\max_{c \in C} \sum_{q=1}^{N} P_q(c|I) \tag{6}$$

where $P_q(c|I)$ is the posterior probability estimation of the $q$-th component model in RBNC, and is defined as:

$$P_q(c|I) = P(c) \prod_{i=1}^{a} P(x_i|\pi_i, c) \tag{7}$$

where $\pi_i$ is the set of parents values of attribute $X_i$.

It should be noted that RBNC-0 is just naive Bayes and in RBNC-*n* ($n \geq 1$), each component models define a weaker conditional independence assumption than naive Bayes, as it is necessarily true if the naive Bayes' assumption is true and may also

be true when the naive Bayes' assumption is not. As this is a weaker assumption than Equation 2, the bias of the model should be lower than that of naive Bayes. However, it is derived from higher-dimensional probability estimates and hence its variance should be higher.

Similar to A*n*DE, RBNC utilizes parameter $n$ that transforms the approach between a low-variance high-bias learner (naive Bayes) and a high-variance low-bias learner with Bayes optimal asymptotic error. So, RBNC actually defines a family of algorithms. Successive members of the family will be best suited to differing quantities of data, starting with low variance for small data set, with successively lower bias but higher variance suiting to increasing data quantities.

### 3.2 Computational Complexity of RBNC

Each component Bayesian network model in RBNC forms an $(n+2)$-dimensional probability table containing the observed frequency for the given combination of $n+1$ attribute values and the class labels. The space complexity of the table is $O(kav^{n+1})$ and the time complexity of compiling it is $O(tan)$, as we need to update each entry for the combination of the $n+1$ attribute-values for every instance. The time complexity for classifying a single instance is O$(kan)$ as we need to consider each attribute for the combination of *n* parent attributes within each class.

Assume the number of component models in RBNC is $N$, then for RBNC, the space complexity is $O(Nkav^{n+1})$, time complexity of compiling it is $O(Ntan)$ and classifying a single instance is O$(Nkan)$.

## 4 Experiments and Results

### 4.1 Experiment Setup

We conduct our experiments under the framework of Weka [14] on a PC with Intel Core 2 Duo P8600 2.4G CPU and 4G RAM. In our experiments, we use the 60 well-recognized datasets from the UCI repositories[15], which include all the datasets recommended by Weka and the benchmark datasets used by related works [2,9,10,3]. A brief description of the data sets is in Table 2. Numeric variables are discretized using supervised discretization method implemented in Weka. Missing values are also processed using the mechanism in Weka, which replaces all missing values with the modes and means from the training data. In addition, all the preprocessing is done with the default parameters in Weka implementation.

We compared RBNC-*n* (*n*=1,2,3, and ensemble sizes $N$ are all set to 20) with the following algorithms:

1. The naive Bayes classifier (**NB**).
2. The discriminatively trained tree-augmented naive Bayes (**TAN-$\hat{f}$CLL**) algorithm using factorized conditional log-likelihood [3].
3. The Hidden naive Bayes classifier (**HNB**) [11].

**Table 2.** Description of the data sets used for experiments

| Datasets | Size | Attribute | Classes | Datasets | Size | Attribute | Classes |
|---|---|---|---|---|---|---|---|
| adult | 48842 | 15 | 2 | ionosphere | 351 | 35 | 2 |
| albalone | 4177 | 9 | 28 | iris | 150 | 5 | 3 |
| anneal | 898 | 39 | 6 | kr-vs-kp | 3196 | 37 | 2 |
| anneal.ORIG | 898 | 39 | 6 | labor | 57 | 17 | 2 |
| audiology | 226 | 70 | 24 | letter | 20000 | 17 | 26 |
| australian | 690 | 15 | 2 | lymph | 148 | 19 | 4 |
| autos | 205 | 26 | 7 | mofn | 1324 | 11 | 2 |
| badges | 294 | 11 | 2 | mushroom | 8124 | 23 | 2 |
| balance-scale | 625 | 5 | 3 | nursery | 12960 | 9 | 5 |
| breast-cancer | 286 | 10 | 2 | optical | 5620 | 65 | 10 |
| breast-w | 699 | 10 | 2 | ozone | 2536 | 73 | 2 |
| car | 1728 | 7 | 4 | page-blocks | 5473 | 11 | 5 |
| chess | 28056 | 7 | 18 | pendigital | 10992 | 17 | 10 |
| cleve | 296 | 14 | 2 | pima | 768 | 9 | 2 |
| cmc | 1473 | 10 | 3 | primary-tumor | 339 | 18 | 21 |
| colic | 368 | 23 | 2 | segment | 2310 | 20 | 7 |
| colic.ORIG | 368 | 28 | 2 | shuttle | 5800 | 10 | 7 |
| corral | 128 | 7 | 2 | sick | 3772 | 30 | 2 |
| credit-a | 690 | 16 | 2 | sonar | 208 | 61 | 2 |
| credit-g | 1000 | 21 | 2 | soybean | 683 | 36 | 19 |
| dermatology | 366 | 35 | 6 | spambase | 4601 | 58 | 2 |
| diabetes | 768 | 9 | 2 | splice | 3190 | 62 | 3 |
| ecoli | 336 | 8 | 8 | tic-tac-toe | 958 | 10 | 2 |
| flare | 1066 | 11 | 2 | vehicle | 846 | 19 | 4 |
| glass | 214 | 10 | 7 | vote | 435 | 17 | 2 |
| heart-c | 303 | 14 | 5 | vowel | 990 | 14 | 11 |
| heart-h | 294 | 14 | 5 | waveform-5000 | 5000 | 41 | 3 |
| heart-statlog | 270 | 14 | 2 | wine | 178 | 14 | 3 |
| hepatitis | 155 | 20 | 2 | yeast | 1484 | 10 | 10 |
| hypothyroid | 3772 | 30 | 4 | zoo | 101 | 18 | 7 |

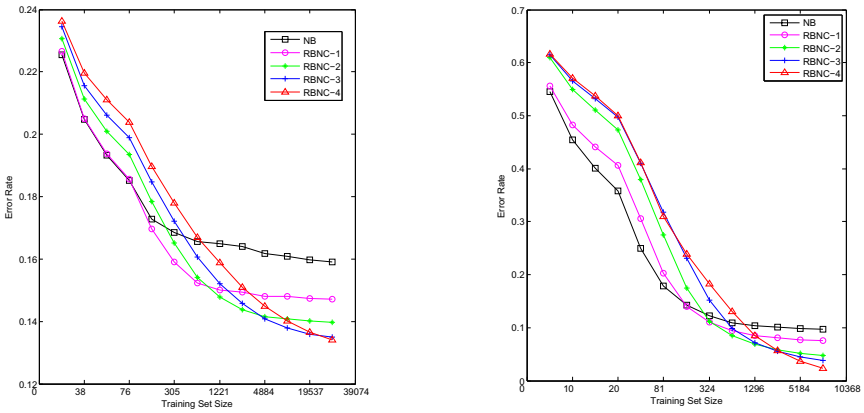
4. The Averaged one-dependence estimators (**AODE**) and Averaged two-dependence estimators (**A2DE**). We do not present the results of A$n$DE ($n \geq 3$) since even the computational requirements of A3DE defeat the Weka implementation except in cases of low dimensional data, and this is the same issue encountered by [6].
5. The Random Forests classifier with both the default setting of 10 trees (**RF-10**) and with 100 trees (**RF-100**).

The naive Bayes, HNB, AODE, A2DE and RF are already implemented in Weka, and the source code of TAN-$\hat{f}$CLL algorithm is available at the author's homepage `http://kdbio.inesc-id.pt/~asmc/software/fCLL.html`. So, we only implemented RBNC within the Weka framework and uploaded the **source codes** of RBNC at[1]. We used the **laplace estimation** to avoid the zero-frequency problem for all compared methods. In our experiment, the performance of an algorithm on each data set has been calculated via 10 runs of 10-fold stratified cross validation.

### 4.2 The Effect of Varying *n* within RBNC

To investigate how increasing $n$ within the RBNC framework affects performance as the quantity of data increases, we form learning curves for NB, RBNC-1, RBNC-2, RBNC-3 and RBNC-4 on the Adult and Nursery dataset, respectively.

[1] `http://homepage.fudan.edu.cn/wangqing/files/2011/10/rbnc1.zip`




(a) Adult (b) Nursery

**Fig. 1.** Learning curves as function of training set size

First, 10% instances are selected at random as a test set and training sets were sampled from the remaining instances. The training sets size consist of $\frac{1}{2^{11}}$, $\frac{1}{2^{10}}$,..., 1 fraction of the remaining instances, respectively. This process is repeated 100 times and each algorithm is evaluated on the resulting training-test set pairs. The learning curves of error rate for NB, RBNC-1, RBNC-2, RBNC-3 and RBNC-4 are presented in Figure 1.

Figure 1 clearly show the predicted trade-off for increasing $n$. At the smallest data size, where low variance is more important than low bias, error rate is minimized by $n$ = 0 (NB) and increases as $n$ increases. At the largest data size, where low bias is most important, this dimensionality is reversed.

### 4.3 Experimental Results

Table 3 shows the comparison results of two-tailed $t$-test with a 95% confidence level between each pair of algorithms on data, in which each entry *w/t/l* means that the

**Table 3.** Summary of experimental results under pairwise two-tailed t-test with 95% confidence level. Each cell contains the number of wins, ties and losses between the algorithm in that row and the algorithm in that column.

| *w/t/l* | RBNC-3 | RBNC-2 | RBNC-1 |
|---|---|---|---|
| RBNC-2 | 1/44/15 | – | – |
| RBNC-1 | 4/33/23 | 1/38/21 | – |
| NB | 4/24/32 | 2/25/33 | 1/31/28 |
| TAN-$\hat{f}$CLL | 6/28/26 | 8/30/22 | 14/31/15 |
| HNB | 3/40/17 | 8/39/13 | 13/42/5 |
| AODE | 5/33/22 | 6/39/15 | 9/45/6 |
| A2DE | 6/37/17 | 9/40/11 | 17/41/2 |
| RF-10 | 6/29/25 | 8/27/25 | 19/20/21 |
| RF-100 | 11/33/16 | 19/23/18 | 25/26/9 |

algorithm at the corresponding row wins in $w$ data sets, ties in $t$ data sets, and loses in $l$ data sets, compared to the algorithm at the corresponding column. Table 4 and 5 show the detailed accuracies of the algorithms on each data set. The mean accuracy and standard deviation, together with the overall rank on all data sets are summarized at the bottom of the table.

From the experimental results, we can see that RBNC-$n$ algorithms can achieve substantial improvement over naive Bayes (32 wins and 4 losses, 33 wins and 2 losses, 28 wins and 1 losses, respectively). This results show that many data sets in our experiments contain strong dependencies, and conditional independence assumption failed to capture these dependencies. In addition, RBNC-1 and RBNC-2 are comparable to AODE (6 wins and 9 losses) and A2DE (11 wins and 9 losses), respectively. For Random Forests algorithms, RBNC-$n$ ($n$=1, 2, 3) all outperform RF-10 and RBNC-3 significantly outperforms RF-100 (16 wins and 11 losses). Overall, the performance of RBNC-3 is the best among all the algorithms compared. Considering that RBNC-$n$ scales linearly with respect to both the number of instances and attributes of the training data, RBNC-$n$ are overall more efficient.

To study the robustness of our algorithm, we test it on these 60 UCI data sets under artificial noise in the class labels. Following the method in [16], the noisy version of each training data set is generated by choosing 10% instances and changing their class labels to other incorrect labels randomly. Due to space limited, we do not list the detailed results of the accuracy and standard deviation on each data set here. The experimental results show that the RBNC algorithms are all robust to noise and also achieve substantial improvement over naive Bayes. And RBNC-3 still to be the best among the algorithms compared.

To further understand the working mechanism of RBNC-$n$ and the difference compared with Random Forests, we use the bias-variance decomposition to analysis them. The results again demonstrate that with $n$ increasing, RBNC-$n$ evolves from low variance coupled with high bias through to high variance coupled with low bias. The bias terms for RBNC-$n$ ($n$=1, 2, 3) and RF (10 and 100) are 0.0963, 0.0760, 0.0696, 0.0663 and 0.0669, respectively. So RBNC-3 could achieve the same level of bias compared with Random Forests. This is of interest because it demonstrates that it is possible to create low-bias high-variance generative learners without discriminative learning.

## 5 Conclusion

In this paper, we propose the RBNC family of algorithms which utilize a single parameter $n$ to control over a bias-variance trade-off, such that higher values of $n$ are appropriate for greater numbers of training cases. RBNC retains many of naive Bayes' desirable property, such as the time and space complexity are linear with respect to both the number of training instances and attributes, the learning can be done by a single pass through the training data and the performance is robust to noise. Our experimental results show that RBNC has a better overall performance compared to the state-of-the-art Bayesian network classifier algorithms. Considering the simplicity and efficiency, RBNC is a promising model that could be used in many applications.

**Table 4.** Accuracy and standard deviation of the ten algorithms on the 60 UCI data sets

| Data sets | RBNC-3 | RBNC-2 | RBNC-1 | NB | TAN-$\widehat{f}$CLL | HNB | AODE | A2DE | RF-10 | RF-100 |
|---|---|---|---|---|---|---|---|---|---|---|
| adult | 86.48±0.39 | 86.00±0.39 | 85.27±0.39 | 84.07±0.42 | 86.21±0.37 | 84.76±0.36 | 85.20±0.41 | 85.95±0.39 | 85.27±0.42 | 85.66±0.37 |
| albalone | 25.15±1.83 | 25.53±1.80 | 26.16±1.82 | 26.18±1.97 | 25.23±1.89 | 24.79±1.92 | 26.49±1.77 | 26.44±1.84 | 23.6±1.66 | 23.42±1.72 |
| anneal | 99.10±0.90 | 98.95±1.04 | 98.63±1.18 | 96.13±2.16 | 98.98±1.10 | 98.29±1.26 | 98.05±1.37 | 98.31±1.24 | 99.2±0.94 | 99.48±0.75 |
| anneal.ORIG | 93.97±2.47 | 94.03±2.45 | 93.39±2.58 | 92.66±2.72 | 94.47±2.32 | 94.82±2.23 | 93.35±2.53 | 93.31±2.55 | 94.72±2.13 | 94.9±1.99 |
| audiology | 76.61±6.22 | 76.61±6.44 | 76.04±6.53 | 71.40±6.37 | 69.48±6.62 | 73.15±6.00 | 71.66±6.42 | 71.88±6.47 | 75.95±8.23 | 79.97±6.85 |
| australian | 86.42±3.57 | 86.74±3.34 | 86.23±3.44 | 85.38±3.49 | 85.58±3.81 | 85.04±4.04 | 86.09±3.54 | 86.19±3.63 | 84.12±3.97 | 84.9±4.12 |
| autos | 88.48±6.95 | 85.56±7.32 | 81.65±8.65 | 72.3±10.31 | 80.55±8.68 | 85.51±7.62 | 81.14±8.50 | 82.56±7.90 | 86.28±7.26 | 87.11±6.89 |
| badges | 100.0±0.0 | 100.0±0.0 | 100.0±0.0 | 100.0±0.0 | 100.0±0.0 | 100.0±0.0 | 100.0±0.0 | 100.0±0.0 | 99.97±0.33 | 100.0±0.0 |
| balance-scale | 68.7±3.72 | 68.75±3.74 | 69.47±3.81 | 71.08±4.29 | 70.91±4.06 | 69.02±3.76 | 69.34±3.82 | 68.75±3.70 | 69.92±3.98 | 68.99±3.76 |
| breast-cancer | 74.59±6.97 | 73.58±6.36 | 71.93±6.96 | 72.94±7.71 | 67.29±7.35 | 70.23±6.49 | 72.73±7.01 | 73.97±6.19 | 69.17±6.8 | 70.09±7.36 |
| breast-w | 96.45±2.21 | 96.98±1.87 | 97.24±1.69 | 97.25±1.79 | 97.21±1.82 | 96.27±2.12 | 96.97±1.87 | 96.55±2.12 | 95.19±2.64 | 95.67±2.57 |
| car | 95.27±1.41 | 93.92±1.77 | 91.74±2.06 | 85.46±2.56 | 94.44±1.79 | 93.01±2.02 | 91.41±2.06 | 93.56±1.70 | 93.18±1.86 | 94.7±1.66 |
| chess | 75.57±0.81 | 67.31±0.85 | 55.37±0.85 | 36.11±0.81 | 55.85±0.76 | 58.89±0.86 | 52.72±0.91 | 65.88±0.84 | 65.03±1.11 | 69.57±0.93 |
| cleve | 82.12±6.52 | 82.19±6.48 | 83.17±6.58 | 83.47±6.80 | 81.21±6.53 | 82.13±6.56 | 83.34±6.35 | 82.80±6.51 | 80.13±7.52 | 81.33±7.08 |
| cmc | 52.70±3.84 | 53.29±4.01 | 53.37±3.58 | 52.17±3.80 | 54.48±3.55 | 54.66±3.42 | 51.80±3.54 | 52.38±3.75 | 50.12±4.29 | 50.84±4.16 |
| colic | 82.39±6.38 | 82.39±5.87 | 82.91±5.51 | 81.39±5.74 | 79.59±6.25 | 81.60±6.04 | 82.37±5.73 | 82.28±5.90 | 82.2±6.52 | 82.99±5.72 |
| colic.ORIG | 76.28±6.39 | 77.15±5.99 | 76.42±6.07 | 74.46±6.81 | 71.31±5.23 | 68.49±6.39 | 75.66±6.65 | 75.79±6.09 | 71.93±5.92 | 73.73±5.35 |
| corral | 99.23±2.99 | 99.69±1.89 | 95.74±5.76 | 86.60±8.87 | 99.85±1.08 | 99.85±1.08 | 88.72±9.01 | 96.05±5.96 | 98.68±3.69 | 99.77±1.71 |
| credit-a | 86.96±3.76 | 86.88±3.59 | 86.00±3.87 | 86.25±4.01 | 85.36±3.83 | 85.38±4.01 | 86.55±3.82 | 86.83±3.59 | 83.77±4.38 | 84.46±3.94 |
| credit-g | 75.56±3.81 | 76.21±3.87 | 75.63±3.88 | 75.43±3.84 | 73.68±4.31 | 76.04±3.64 | 76.45±3.89 | 76.31±3.61 | 72.92±3.67 | 74.31±3.58 |
| dermatology | 98.11±2.16 | 98.44±1.91 | 98.20±2.03 | 97.93±2.12 | 96.69±2.81 | 98.61±1.85 | 97.82±2.29 | 97.85±2.27 | 95.9±3.06 | 97.07±2.62 |
| diabetes | 77.98±4.38 | 78.20±4.32 | 78.37±4.48 | 77.85±4.67 | 78.54±4.15 | 76.73±4.05 | 78.07±4.56 | 78.58±4.34 | 76.11±4.33 | 76.61±4.69 |
| ecoli | 83.91±4.93 | 83.67±4.72 | 84.27±5.14 | 85.57±4.95 | 84.86±4.95 | 82.54±4.99 | 85.16±5.05 | 84.02±4.87 | 82.99±5.59 | 82.93±5.38 |
| flare | 82.35±2.99 | 82.31±3.01 | 81.43±3.24 | 80.19±3.41 | 82.40±2.68 | 82.70±2.13 | 81.83±3.12 | 82.33±2.80 | 80.89±2.71 | 80.94±2.52 |
| glass | 79.09±8.90 | 78.86±8.90 | 76.44±8.51 | 74.39±7.95 | 77.64±8.76 | 78.27±8.97 | 76.49±7.71 | 76.59±8.80 | 76.25±8.79 | 77.32±8.21 |
| heart-c | 81.78±6.10 | 82.02±6.49 | 83.07±6.22 | 83.60±6.42 | 81.28±6.54 | 82.38±6.79 | 83.26±6.19 | 82.97±6.37 | 80.96±6.4 | 81.13±5.56 |
| heart-h | 84.67±5.90 | 84.39±6.17 | 84.53±6.01 | 84.46±5.92 | 84.84±5.94 | 85.31±5.92 | 84.43±5.92 | 84.60±6.09 | 82.25±6.66 | 81.4±6.66 |
| heart-statlog | 82.19±6.33 | 82.56±6.54 | 83.74±6.29 | 83.74±6.25 | 82.70±6.41 | 82.37±6.54 | 83.33±6.61 | 82.52±6.60 | 83.52±6.93 | 83.59±7.01 |
| hepatitis | 85.51±8.93 | 85.45±9.40 | 84.86±9.56 | 84.22±9.41 | 84.79±9.31 | 87.90±8.09 | 85.43±8.96 | 85.58±8.55 | 83.92±9.05 | 85.02±8.76 |
| hypothyroid | 99.04±0.50 | 99.00±0.51 | 98.82±0.54 | 98.48±0.59 | 99.28±0.39 | 99.13±0.41 | 98.74±0.55 | 98.79±0.49 | 99.11±0.46 | 99.29±0.4 |
| ionosphere | 93.68±3.88 | 93.19±4.01 | 92.91±3.92 | 90.77±4.76 | 92.85±4.23 | 92.62±4.12 | 92.97±4.32 | 92.99±4.16 | 92.45±3.83 | 93.5±3.92 |

**Table 5.** (**Continued**) Accuracy and standard deviation of the ten algorithms on the 60 UCI data sets. Bottom rows of the table present the mean values and overall ranks, respectively.

| Data sets | RBNC-3 | RBNC-2 | RBNC-1 | NB | TAN-$\widehat{f}$CLL | HNB | AODE | A2DE | RF-10 | RF-100 |
|---|---|---|---|---|---|---|---|---|---|---|
| iris | 93.07±6.06 | 93.60±5.60 | 93.33±5.61 | 94.47±5.61 | 93.60±5.91 | 93.33±5.92 | 93.20±5.76 | 93.33±5.76 | 94.07±5.76 | 94.4±5.74 |
| kr-vs-kp | 93.10±1.65 | 91.35±1.65 | 89.73±1.74 | 87.79±1.91 | 94.21±1.37 | 92.35±1.32 | 91.03±1.66 | 93.08±1.41 | 98.87±0.61 | 99.27±0.44 |
| labor | 95.47±8.74 | 94.93±9.00 | 95.27±9.61 | 93.13±10.56 | 95.57±9.36 | 95.60±9.47 | 95.43±8.80 | 94.67±10.04 | 92.47±11.5 | 92.8±10.98 |
| letter | 93.65±0.53 | 92.35±0.56 | 86.59±0.72 | 74.00±0.88 | 85.94±0.67 | 89.77±0.63 | 88.76±0.70 | 90.82±0.64 | 90.21±0.75 | 93.5±0.54 |
| lymph | 88.40±8.18 | 87.79±8.15 | 86.52±8.39 | 84.97±8.30 | 83.60±9.73 | 86.78±8.47 | 87.53±8.13 | 87.61±8.67 | 79.65±9.2 | 82.89±8.49 |
| mofn | 95.67±1.92 | 93.88±2.16 | 89.31±2.06 | 85.28±1.95 | 92.01±1.93 | 95.19±2.18 | 88.60±2.10 | 92.54±2.10 | 99.69±0.6 | 99.99±0.08 |
| mushroom | 100.0±0.00 | 99.96±0.07 | 99.77±0.16 | 95.52±0.78 | 99.77±0.15 | 99.96±0.06 | 99.95±0.07 | 99.99±0.03 | 100.0±0.0 | 100.0±0.0 |
| nursery | 96.19±0.44 | 95.25±0.47 | 92.51±0.59 | 90.30±0.72 | 93.41±0.55 | 94.25±0.50 | 92.73±0.62 | 94.73±0.52 | 98.36±0.38 | 99.17±0.32 |
| optical | 96.44±0.71 | 95.85±0.81 | 94.81±0.86 | 92.42±1.07 | 95.92±0.81 | 96.35±0.79 | 96.99±0.71 | 97.12±0.68 | 91.67±1.14 | 96.18±0.77 |
| ozone | 89.56±1.93 | 86.53±2.29 | 83.10±2.56 | 80.04±2.71 | 94.16±1.37 | 88.96±1.92 | 88.59±2.05 | 93.91±1.49 | 97.05±0.6 | 97.08±0.61 |
| page-blocks | 97.41±0.64 | 97.18±0.69 | 95.73±0.79 | 93.53±0.98 | 96.30±0.79 | 96.91±0.69 | 97.27±0.67 | 97.35±0.65 | 97.14±0.67 | 97.29±0.64 |
| pendigital | 98.45±0.38 | 98.43±0.37 | 97.14±0.49 | 87.89±0.99 | 96.71±0.63 | 97.83±0.43 | 97.94±0.45 | 98.23±0.40 | 96.36±0.63 | 97.64±0.46 |
| pima | 78.22±4.30 | 78.16±4.35 | 78.50±4.11 | 77.97±4.27 | 78.84±4.07 | 77.03±4.09 | 78.31±4.16 | 79.06±4.05 | 76.59±4.54 | 77.28±4.44 |
| primary-tumor | 49.00±5.49 | 47.88±5.53 | 48.67±5.99 | 47.20±6.02 | 45.76±6.35 | 47.85±6.06 | 47.87±6.37 | 48.20±6.11 | 39.68±5.96 | 41.3±6.05 |
| segment | 96.66±1.14 | 96.60±1.14 | 95.84±1.32 | 91.71±1.68 | 93.15±1.98 | 96.86±1.08 | 95.77±1.24 | 95.90±1.20 | 96.25±1.11 | 96.96±1.06 |
| shuttle | 99.82±0.13 | 99.86±0.12 | 99.88±0.11 | 99.35±0.30 | 99.83±0.14 | 99.84±0.14 | 99.84±0.13 | 99.82±0.13 | 99.74±0.17 | 99.76±0.16 |
| sick | 97.53±0.80 | 97.30±0.79 | 97.27±0.80 | 97.10±0.84 | 97.41±0.75 | 97.55±0.76 | 97.39±0.79 | 97.47±0.76 | 97.64±0.75 | 97.77±0.73 |
| sonar | 86.12±7.30 | 86.31±7.23 | 85.16±7.43 | 85.16±7.52 | 85.16±7.62 | 84.63±7.64 | 86.60±6.91 | 86.84±6.84 | 79.72±9.02 | 82.83±7.8 |
| soybean | 94.41±2.38 | 94.25±2.40 | 93.66±2.64 | 92.20±3.23 | 94.33±2.45 | 94.67±2.25 | 93.31±2.85 | 93.43±2.69 | 92.8±2.45 | 93.81±2.72 |
| spambase | 92.72±1.09 | 91.79±1.17 | 91.10±1.17 | 90.24±1.23 | 92.73±1.11 | 92.42±1.13 | 93.36±1.12 | 94.19±1.03 | 94.01±1.15 | 94.88±1.02 |
| splice | 87.40±1.77 | 94.07±1.28 | 95.93±1.12 | 95.42±1.14 | 94.83±1.05 | 96.13±0.99 | 96.12±1.00 | 96.41±0.85 | 90.09±1.78 | 95.88±1.14 |
| tic-tac-toe | 92.79±2.67 | 86.37±3.55 | 73.63±4.02 | 69.64±4.40 | 74.76±3.71 | 77.46±3.53 | 73.86±3.93 | 90.79±3.17 | 92.11±2.59 | 97.06±1.76 |
| vehicle | 73.76±3.53 | 73.35±3.73 | 71.89±3.15 | 62.52±3.81 | 70.98±3.98 | 73.31±3.38 | 72.31±3.62 | 73.06±3.71 | 72.99±4.4 | 73.19±3.5 |
| vote | 95.77±2.92 | 94.78±3.20 | 92.94±3.54 | 90.21±3.95 | 90.76±3.73 | 94.36±3.20 | 94.52±3.19 | 94.75±3.29 | 95.95±2.83 | 96.18±2.85 |
| vowel | 89.34±2.98 | 87.75±3.35 | 83.21±3.81 | 65.23±4.53 | 89.16±3.39 | 89.11±2.98 | 80.88±3.81 | 83.99±3.41 | 88.98±3.01 | 90.13±2.73 |
| waveform-5000 | 85.21±1.42 | 84.26±1.36 | 82.47±1.35 | 80.72±1.50 | 81.26±1.45 | 86.26±1.45 | 86.03±1.56 | 86.37±1.48 | 80.58±1.73 | 84.99±1.47 |
| wine | 98.88±2.66 | 98.32±2.94 | 98.88±2.26 | 98.82±2.30 | 98.08±3.13 | 98.14±2.89 | 98.37±2.92 | 98.32±2.94 | 97.97±3.24 | 98.65±2.55 |
| yeast | 60.13±3.34 | 60.26±3.60 | 59.79±4.01 | 59.16±3.80 | 59.88±3.68 | 59.64±3.82 | 59.72±3.86 | 59.63±4.01 | 59.1±3.56 | 59.59±3.41 |
| zoo | 97.03±5.19 | 96.05±5.60 | 94.85±6.39 | 93.21±7.35 | 93.20±6.83 | 97.11±4.97 | 94.66±6.38 | 94.66±6.38 | 91.75±7.78 | 93.55±6.83 |
| Mean | 86.37±3.44 | 85.91±3.47 | 84.67±3.63 | 82.12±3.92 | 84.65±3.60 | 85.34±3.44 | 84.74±3.67 | 85.81±3.56 | 84.95±3.71 | 86.08±3.43 |
| Overall Rank | 3.72 | 4.48 | 5.73 | 7.59 | 5.91 | 5.23 | 5.75 | 4.63 | 7.01 | 4.95 |

**Acknowledgements.** We would like to thank the anonymous reviewers for their helpful comments. This work was supported by the Natural Science Foundation of Anhui Provincial Education Department under Grant No.KJ2013A053.

## References

1. Pearl, J.: Probabilistic reasoning in intelligent systems: Networks of plausible inference. Morgan Kaufmann, San Francisco (1988)
2. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning 29, 131–163 (1997)
3. Carvalho, A.M., Roos, T., Oliveira, A., Myllymaki, P.: Discriminative learning of bayesian networks via factorized conditional log-likelihood. Journal of Machine Learning Research 12, 2181–2210 (2011)
4. Domingos, P., Pazzani, M.J.: On the optimality of the simple bayesian classifier under zero-one loss. Machine Learning 29(2), 103–130 (1997)
5. Webb, G.I., Boughton, J.R., Wang, Z.: Not so naive bayes: Aggregating one-dependence estimators. Machine Learning 58(1), 5–24 (2005)
6. Webb, G.I., Boughton, J.R., Zheng, F., Ting, K., Salem, H.: Learning by extrapolation from marginal to full-multivariate probability distributions: Decreasingly naive bayesian classification. Machine Learning 86(2), 233–272 (2012)
7. Salem, H., Suraweera, P., Webb, G.I., Boughton, J.R.: Techniques for efficient learning without search. In: Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part I. LNCS, vol. 7301, pp. 50–61. Springer, Heidelberg (2012)
8. Wu, X.D., Kumar, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.H., Steinbach, M., Hand, D.J., Steinberg, D.: Top 10 algorithms in data mining. Knowledge and Information Systems 14(1) (2008)
9. Grossman, D., Domingos, P.: Learning bayesian network classifiers by maximizing conditional likelihood. In: Proceedings of the 21st International Conference on Machine Learning, pp. 46–53 (2004)
10. Jing, Y.S., Pavlovi, V., Rehg, J.M.: Efficient discriminative learning of bayesian network classifier via boosted augmented naive bayes. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 369–376 (2005)
11. Jiang, L., Zhang, H., Cai, Z.: A novel bayes model: hidden naive bayes. IEEE Transations on Knowledge and Data Engineering 21(10), 1361–1371 (2009)
12. Langley, P., Sage, S.: Induction of selective bayesian classifiers. In: Proceedings of the Uncertainty in Artificial Intelligence, pp. 399–406 (1994)
13. Zhang, H., Jiang, L.X., Su, J.: Hidden naive bayes. In: The Twentieth National Conference on Artificial Intelligence (AAAI 2005), pp. 919–924 (2005)
14. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco (2000)
15. Blake, C., Merz, C.J.: UCI repository of machine learning databases. Department of ICS, University of California, Irvine, `http://www.ics.uci.edu/~mlearn/MLRepository.html`
16. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)