A Novel Bayes Model: Hidden Naive Bayes

Liangxiao Jiang, Harry Zhang, and Zhihua Cai

Abstract—Because learning an optimal Bayesian network classifier is an NP-hard problem, learning-improved naive Bayes has attracted much attention from researchers. In this paper, we summarize the existing improved algorithms and propose a novel Bayes model: hidden naive Bayes (HNB). In HNB, a hidden parent is created for each attribute which combines the influences from all other attributes. We experimentally test HNB in terms of classification accuracy, using the 36 UCI data sets selected by Weka, and compare it to naive Bayes (NB), selective Bayesian classifiers (SBC), naive Bayes tree (NBTree), tree-augmented naive Bayes (TAN), and averaged one-dependence estimators (AODE). The experimental results show that HNB significantly outperforms NB, SBC, NBTree, TAN, and AODE. In many data mining applications, an accurate class probability estimation and ranking are also desirable. We study the class probability estimation and ranking performance, measured by conditional log likelihood (CLL) and the area under the ROC curve (AUC), respectively, of naive Bayes and its improved models, such as SBC, NBTree, TAN, and AODE, and then compare HNB to them in terms of CLL and AUC. Our experiments show that HNB also significantly outperforms all of them.

Index Terms—Naive Bayes, Bayesian network classifiers, learning algorithms, classification, class probability estimation, ranking.

1 Introduction

A Bayesian network consists of a structural model and a set of conditional probabilities. The structural model is a directed graph in which nodes represent attributes and arcs represent attribute dependencies. Attribute dependencies are quantified by conditional probabilities for each node given its parents. Bayesian networks are often used for classification problems, in which a learner attempts to construct a classifier from a given set of training examples with class labels. Assume that $A_1, A_2, \ldots, A_n$ are n attributes (corresponding to attribute nodes in a Bayesian network). An example E is represented by a vector $<a_1, a_2, \ldots, a_n>$, where a_i is the value of A_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 $c(E)$ to denote the class of E. The classifier represented by a general Bayesian network is defined in (1):

$$c(E) = \arg\max_{c \in C} P(c)P(a_1, a_2, \ldots, a_n|c). \quad (1)$$

Assume that all attributes are independent given the class (conditional independence assumption), the resulting classifier is called naive Bayes:

$$c(E) = \arg\max_{c \in C} P(c)\prod_{i=1}^{n} P(a_i|c). \quad (2)$$

Fig. 1 shows graphically the structure of naive Bayes. In naive Bayes, each attribute node has the class node as its parent, but does not have any parent from attribute nodes. Because the values of $P(c)$ and $P(a_i|c)$ can be easily estimated from training examples, naive Bayes is easy to construct. Naive Bayes is the simplest form of Bayesian networks. It is obvious that the conditional independence assumption in naive Bayes is rarely true in reality, which would harm its performance in the applications with complex attribute dependencies.

Extending its structure is a direct way to overcome the limitation of naive Bayes, since attribute dependencies can be explicitly represented by arcs. Tree-augmented naive Bayes (TAN) is an extended tree-like naive Bayes [1] in which the class node directly points to all attribute nodes and an attribute node can have only one parent from another attribute node. Fig. 2 shows an example of TAN. TAN is a specific case of general Bayesian network classifiers in which the class node also directly points to all attribute nodes, but there is no limitation on the arcs among attribute nodes (except that they do not form any directed cycle).

Learning an optimal Bayesian network classifier has been proved to be NP-hard [2]. In fact, the most time-consuming step in learning a Bayesian network is learning the structure (structure learning). In practice, imposing restrictions on the structures of Bayesian networks, such as TAN, leads to an acceptable computational complexity and a considerable improvement over naive Bayes. One main issue in learning TAN is that only one attribute parent is allowed for each attribute, ignoring the influences from other attributes. In addition, in a TAN learning algorithm, structure learning is also unavoidable. Certainly, a model that avoids structure learning, and is still able to represent attribute dependencies to some extent, is desirable. This is the main motivation of this paper.

Traditionally, the performance of a classifier is measured by its classification accuracy. In fact, some classifiers, such as naive Bayes and decision trees, also produce the estimations

- *L. Jiang and Z. Cai are with the Faculty of Computer Science, China University of Geosciences, Wuhan 430074, PR China. E-mail: ljiang@cug.edu.cn, zhcai@cug.edu.cn.*
- *H. Zhang is with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada. E-mail: hzhang@unb.ca.*

Manuscript received 12 Apr. 2008; revised 2 Dec. 2008; accepted 8 Dec. 2008; published online 10 Dec. 2008.
Recommended for acceptance by S. Singh.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2008-04-0192.
Digital Object Identifier no. 10.1109/TKDE.2008.234.

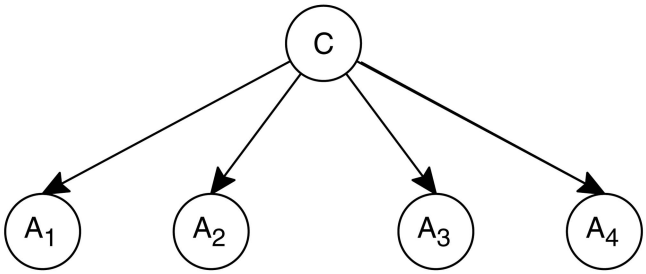


Fig. 1. An example of naive Bayes.

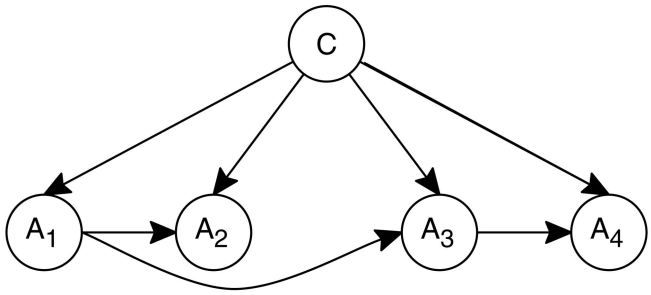


Fig. 2. An example of TAN.

of the class probability $p(c|E)$, that is, the probability of example E in class c. This information is often ignored in classification as long as the class with the highest class probability estimation is identical to the actual class. In data mining applications, however, we need accurate ranking, not just only accurate classification. For example, in direct marketing, a company usually needs to apply different promotion strategies to its customers with different likelihoods of buying its services or products. In this scenario, a classification of buyers and nonbuyers is not enough. Instead, a ranking of customers in terms of their likelihoods of buying is desired [3]. Moreover, there are some applications in which misclassification costs should be taken into account. For example, the cost of classifying a patent without cancer into the class with cancer is significantly lower than the reverse. In this kind of cost-sensitive applications, accurate class probability estimation is even required [4].

To evaluate the ranking performance of a model, AUC (the area under the ROC curve) [5], [6] is currently a well-accepted method. For binary classification, Hand and Till [7] show that AUC is equivalent to the probability that a randomly chosen example of class $-$ will have a smaller estimated probability of belonging to class $+$ than a randomly chosen example of class $+$. A simple approach to calculating the AUC of the classifier G is:

$$\hat{A} = \frac{S_0 - n_0(n_0+1)/2}{n_0 n_1}, \tag{3}$$

where n_0 and n_1 are the numbers of negative and positive examples, respectively, and $S_0 = \sum r_i$, where r_i is the rank of ith negative example in the ranked list. From (3), it is clear that AUC is essentially a measure of the quality of ranking. For example, the AUC of a ranking is 1 (the maximum value of AUC) if there is no negative example preceding a positive example. For multiple classes, Hand and Till [7] propose to calculate AUC using the following M measure:

$$\hat{A} = \frac{2}{k(k-1)} \sum_{i<j \in C} \hat{A}(i,j), \tag{4}$$

where k is the number of classes and $\hat{A}(i,j)$ is the AUC of each pair of classes i and j.

For class probability estimation, conditional log likelihood (CLL) has been recently used to evaluate the quality of class probability estimation yielded by a classifier [8].

Given a classifier G and a set of examples $D = \{E_1, E_2, \ldots, E_i \ldots, E_t\}$, where $E_i = (a_{i_1}, a_{i_2} \ldots, a_{i_n}, c_i), t$ is the number of examples, n is the number of attributes, and c_i is the class label of E_i. The conditional log likelihood $CLL(G|D)$ of a classifier G on D is defined as:

$$CLL(G|D) = \sum_{i=1}^{t} logP_G(c_i|a_{i_1}, a_{i_2} \ldots, a_{i_n}). \tag{5}$$

In this paper, we first summarize the existing improved algorithms of naive Bayes, and then propose a novel Bayes model: hidden naive Bayes (HNB). In HNB, a hidden parent is created for each attribute which combines the influences from all other attributes. Our experimental results show that HNB demonstrates remarkable performance in classification, class probability estimation, and ranking, compared to naive Bayes and its state-of-the-art improved models.

The rest of the paper is organized as follows: First, we summarize the existing improved algorithms of naive Bayes. Then, we present our novel model: hidden naive Bayes. Followed by the description of our experimental setup and results in detail, we make a conclusion and outline the main directions for future research.

2 Related Work

In order to relax the conditional independence assumption of naive Bayes effectively, an appropriate language and efficient machinery to represent and manipulate independence assertions are needed [1]. Both are provided by Bayesian networks [9]. Unfortunately, it has been proved that learning an optimal Bayesian network is NP-hard [2]. In order to avoid the intractable complexity for learning Bayesian networks, learning-improved naive Bayes has attracted much attention from researchers. Related work can be broadly divided into five main categories:

1. *Structure extension*. Extending the structure of naive Bayes to represent the dependencies among attributes.
2. *Feature selection*. Selecting an attribute subset from the whole space of attributes.
3. *Attribute weighting*. Assigning different weights to attributes in building naive Bayes.
4. *Local learning*. Employing the principle of local learning to build a local naive Bayes.
5. *Data expansion*. Expanding training data and building a naive Bayes on the expanded training data.

2.1 Structure Extension

Just as we discussed before, the main problem confronting naive Bayes is its conditional independence assumption. In real-world data mining applications, this assumption is unrealistic. Thus, extending the structure of naive Bayes by using directed arcs to explicitly represent attribute dependencies is a natural way to relax this unrealistic assumption. The resulting model is essentially a Bayesian network. Thus, learning the structure of Bayesian networks is unavoidable. However, learning the optimal structure is an NP-hard problem [2]. In practice, imposing restrictions on the structures of Bayesian networks is necessary. For example, learning TAN [1] leads to an acceptable computational complexity and a considerable improvement over naive Bayes. Friedman et al. [1] propose a TAN learning algorithm based on conditional mutual information, called TAN, in this paper. The conditional mutual information is defined as:

$$I_P(X;Y|Z) = \sum_{x,y,z} P(x,y,z) log \frac{P(x,y|z)}{P(x|z)P(y|z)}, \qquad (6)$$

where x, y, and z are the values of variables X, Y, and Z, respectively. In the TAN algorithm, $I_P(A_i; A_j|C)$ between each pair of attributes is computed, and a complete undirected weighted graph is built, in which nodes are attributes $A_1, \cdots, A_n$, and the weight of an edge connecting A_i to A_j is set to $I_P(A_i; A_j|C)$. Then, a maximum-weighted spanning tree is constructed. Finally, the undirected tree is converted to the directed one, and a node labeled as C that points to all attribute nodes is added.

The *SuperParent* algorithm [10] is another TAN learning algorithm. It is a greedy heuristic search algorithm in which an arc of achieving the highest accuracy improvement is selected in each step. According to their experimental results, the *SuperParent* algorithm significantly outperforms naive Bayes. But, one disadvantage is its relatively high time complexity.

Zhang and Ling [11] observed that the dependence among attributes tends to cluster into groups in many real-world domains with a large number of attributes. Based on their observation, they presented an improved algorithm *StumpNetwork*. Their motivation is to enhance the efficiency of the *Superparent* algorithm while maintaining a similar predictive accuracy.

One unavoidable issue in learning TAN is structure learning. Thus, a model that avoids structure learning and is still able to represent attribute dependencies to some extent, is desirable. Webb et al. presented an algorithm called averaged one-dependence estimators, simply AODE [12]. In AODE, an aggregate of one-dependence classifiers are learned and the prediction is produced by directly averaging the predictions of all the qualified one-dependence classifiers.

2.2 Feature Selection

The feature selection approach improves naive Bayes by removing redundant and/or irrelevant attributes from training data sets, and only selecting those that are the most informative in learning tasks. This approach works well with the hypothesis that it can improve naive Bayes' performance in domains that include redundant and/or irrelevant attributes without reducing naive Bayes' performance in domains that do not. In fact, any of this kind of improved algorithms is a variant of naive Bayes that only uses a subset of the given attributes in making predictions.

Obviously, the main challenge is how to efficiently select relevant attributes from a given training data set. To achieve this, several feature selection algorithms have been presented and demonstrated considerable improvement over naive Bayes.

Langley and Sage [13] presented an algorithm called selective Bayesian classifiers (simply SBC). It uses a forward greedy search method to select an attribute subset through the whole space of attributes. More specifically, it uses naive Bayes' classification accuracy to evaluate alternative subsets of attributes and adds one unselected attribute, which achieves the best improvement on the accuracy, to the subset in each iteration.

Jiang et al. [14] presented an algorithm, called evolutional naive Bayes (ENB). It selects an attribute subset through the whole space of attributes by carrying out an evolutional (genetic) search process. ENB uses naive Bayes' classification accuracy as the fitness function to evaluate alternative subsets of attributes and selects the individual (hypotheses) with the maximum classification accuracy after a fixed number of generations.

In addition, there are many general feature selection algorithms, which are not specifically for naive Bayes. For example, Kohavi and John [15] explored the relation between optimal feature subset selection and relevance, and proposed a wrapper method for feature subset selection. Ratanamahatana and Gunopulos [16] proposed another feature selection approach by building decision trees.

2.3 Attribute Weighting

Different from feature selection which completely eliminates the least relevant attributes from the attribute space, attribute weighting weights each attribute differently according to its contribution to classification. The resulting model is called weighted naive Bayes (simply WNB) [17], which uses (7) to classify the test example E:

$$c(E) = \arg\max_{c \in C} P(c) \prod_{i=1}^{n} P(a_i|c)^{w_i}, \qquad (7)$$

where w_i is the weight of attribute $A_i (i = 1, 2, \ldots, n)$.

Obviously, how to learn the weights is crucial and has attracted some attention from researchers. For example, Zhang and Sheng [18] explored various attribute weighting methods: the gain ratio method, the hill climbing method, the Markov Chain Monte Carlo method, the hill climbing method combined with the gain ratio method, and the Markov Chain Monte Carlo method combined with the gain ratio method.

Deng et al. [19] presented another attribute weighting algorithm based on the rough set theory, which weights each attribute using the attribute importance degree. They experimentally compare the attribute weighting methods from different views.

Hall [20] presented a decision-tree-based attribute weighting algorithm. The assumption is that the weight assigned to a predictive attribute should be inversely related to the degree of dependency it has on other attributes. Specifically, the proposed method estimates the degree of attribute dependency by constructing unpruned decision trees and looking at the depth at which attributes are tested in the tree. In this method, a bagging procedure is used to stabilize the estimates and the weight of an attribute is zero if it does not appear in the built decision tree.

Although attribute weighting seems a promising approach for improving naive Bayes, searching effectively and efficiently for a good weight assignment is critical. To our knowledge, the improvement achieved by the existing methods is not very impressive, which leaves significant space for future research in this direction.

2.4 Local Learning

The basic idea of the local learning approach is building a naive Bayes on a subset (called local training data) of the training data set, instead of on the whole set. Although the attribute independence assumption of naive Bayes is always violated on the whole training data, it could be expected that the dependencies within the local training data are weaker than those on the whole training data. Thus, the naive Bayes built on the local training data could perform better. In addition, it has been observed that the accuracy of naive Bayes does not scale up in large data sets [21]. This feature makes it especially fit to be a local model embedded into another model, such as a decision tree, a k-nearest neighbor (KNN).

For decision tree learning, naive Bayes tree (NBTree) presented by Kohavi [21] is a typical example. It is a hybrid algorithm combining decision tree with naive Bayes. Learning an NBTree is similar to C4.5 [22] except its score function for evaluating splitting attributes. After a tree is grown, a naive Bayes is constructed for each leaf using the data associated with that leaf. NBTree classifies a test example by sorting it to a leaf and then applying the naive Bayes in that leaf to assign a class label to it. Their experimental results proved that NBTree significantly outperforms naive Bayes and C4.5.

In local learning, KNN is the most well-recognized algorithm. The idea of combining KNN with naive Bayes is quite straightforward. Like all lazy learning methods, the training data is simply stored, and learning is deferred until the classification time. Whenever a test example is classified, a local naive Bayes is trained using the k-nearest neighbors of the test example, with which the test example is classified.

In recent years, researchers have done considerable work on combining KNN with naive Bayes. For example, Frank et al. [23] presented an algorithm, called locally weighted naive Bayes, simply LWNB. In LWNB, the k-nearest neighbors of the test example are First found and each of them is weighted in terms of its distance to the test example. Then, a local naive Bayes is built from the locally weighted training examples. Although it is a k-related algorithm, its classification performance is not particularly sensitive to the value of k as long as it is not too small.

In addition, there are other approaches to building local naive Bayes, for example, LBR (lazy Bayesian rule) [24] and SNNB (selective neighborhood naive Bayes) [25], which also demonstrate significant improvement over naive Bayes. LBR does not directly use the k-nearest neighbors of the test example as the training data for the local naive Bayes. Instead, before classifying a test example, LBR generates a rule most appropriate to the test example. The training examples that satisfy the antecedent of the rule are chosen as the training data for the local naive Bayes, and this local naive Bayes only uses those attributes that do not appear in the antecedent of the rule. In SNNB, multiple naive Bayes are learned by using different k values and a local naive Bayes is trained for each k value. The most accurate one is used to classify the test example. Compared to LWNB, SNNB is not a k-related algorithm, but it has relatively higher time complexity in searching for the best k value for each test example.

2.5 Data Expansion

One practical problem in learning a Bayesian Network classifier is the high variance due to the lack of training data. More precisely, the probability estimates are prone to being unreliable. Intuitively, when the training data are not enough, the underlying distribution would not be clearly reflected. If there are more training data even with the same pattern, the underlying distribution would be reinforced in the training data, and thus, could be more easily learned. Thus, adding more examples to the training set, called data expansion, would help the learning algorithm. To draw more training examples, however, the underlying distribution should be known, which is not the case in reality.

One way to overcome this issue is to clone the existing examples. The idea is adding some clones of training examples to the training data set, based on *similarity*. The resulting model is called instance-cloned naive Bayes (simply ICNB) [26]. ICNB produces an expanded training data set by cloning some training instances based on their similarities to the test instance, and a naive Bayes is then built from the expanded training data set and applied to classify the test instance.

Assume that an instance corresponds to a point in an n-dimensional discrete space, then the similarity between two examples x and y can simply be defined as:

$$s(x,y) = \sum_{i=1}^{n} \delta(a_i(x), a_i(y)), \tag{8}$$

where $\delta(a_i(x), a_i(y))$ is one if $a_i(x) = a_i(y)$ and zero otherwise. Obviously, the similarity $s(x,y)$ is a function which counts the number of identical attribute values of x and y.

For instance cloning, there are different strategies, such as clone all the training instances (global cloning), clone a subset of training instances (local cloning), and clone instances to improve a predefined score (such as AUC) on the resulting expanded data (cloning through search).

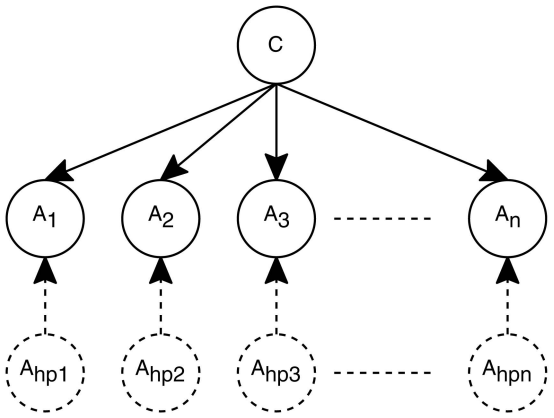


Fig. 3. The structure of HNB.

3 Hidden Naive Bayes

As discussed in previous sections, naive Bayes ignores attribute dependencies. On the other hand, although a Bayesian network can represent arbitrary attribute dependencies, it is intractable to learn it from data [2]. Thus, learning-restricted structures, such as TAN, are more practical. However, only one parent is allowed for each attribute in TAN, even though several attributes might have similar influence on it. Our motivation is to develop a novel model that can avoid the intractable computational complexity for learning an optimal Bayesian network and still take the influences from all attributes into account. The idea is to create a hidden parent for each attribute, which combines the influences from all other attributes. This model is called hidden naive Bayes (simply HNB).

Fig. 3 gives the structure of an HNB. In Fig. 3, C is the class node, and is also the parent of all attribute nodes. Each attribute A_i has a hidden parent $A_{hp_i}, i = 1, 2, \ldots, n$, represented by a dashed circle. The arc from the hidden parent A_{hp_i} to A_i is also represented by a dashed directed line, to distinguish it from the regular arcs.

The joint distribution represented by an HNB is defined as follows:

$$P(A_1, \ldots, A_n, C) = P(C) \prod_{i=1}^{n} P(A_i | A_{hp_i}, C), \tag{9}$$

where

$$P\big(A_i | A_{hp_i}, C\big) = \sum_{j=1, j \neq i}^{n} W_{ij} * P(A_i | A_j, C). \tag{10}$$

We can see that the hidden parent A_{hp_i} for A_i is essentially a mixture of the weighted influences from all other attributes.

The classifier corresponding to an HNB on an example $E = (a_1, \ldots, a_n)$ is defined as follows:

$$c(E) = \arg\max_{c \in C} P(c) \prod_{i=1}^{n} P(a_i | a_{hp_i}, c), \tag{11}$$

where

$$P\big(a_i | a_{hp_i}, c\big) = \sum_{j=1, j \neq i}^{n} W_{ij} * P(a_i | a_j, c). \tag{12}$$

In an HNB, attribute dependencies are represented by hidden parents of attributes. The way of defining hidden parents determines the capability of representing attribute dependencies. In (10), one-dependence estimators $P(A_i|A_j, C)$ are used to define hidden parents. Recall that at most one attribute parent is allowed for each attribute in TAN, and the influences from other attributes have to be ignored. In an HNB, on the other hand, the influences from all other attributes can be represented and a weight is used to represent the importance of an attribute. Thus, intuitively, HNB is a more accurate and expressive model than TAN with respect to representing attribute dependencies.

If we have an order of attributes: $A_1, A_2, \ldots, A_n$, $P(A_i|A_{hp_i}, C)$ can be thought of as an approximation of $P(A_i|A_1, \ldots, A_{i-1})$. In (10), the approximation is based on one-dependence estimators. However, in principle, arbitrary x-dependence estimators can be used to define hidden parents. If $x = n - 1$, any Bayesian network is representable by an HNB. Thus, theoretically, an HNB is equivalent to a Bayesian network in terms of expressive power. In practice, however, we would prefer a simple way to define hidden parents in order to make the learning process simple and efficient.

From (9) and (10), we can see that the approach to determining the weights $W_{ij}, i, j = 1, 2, \ldots, n$ and $i \neq j$, is crucial for learning an HNB. There are two general approaches to doing it: performing a cross-validation-based search, or directly computing the estimated values from data. We adopt the latter, and use the conditional mutual information between two attributes A_i and A_j as the weight of $P(A_i|A_j, C)$. More precisely, in our implementation, W_{ij} is defined in (13):

$$W_{ij} = \frac{I_P(A_i; A_j | C)}{\sum_{j=1, j \neq i}^{n} I_P(A_i; A_j | C)}, \tag{13}$$

where $I_P(A_i; A_j|C)$ is the conditional mutual information defined in (14).

$$I_P(A_i; A_j | C) = \sum_{a_i, a_j, c} P(a_i, a_j, c) log \frac{P(a_i, a_j | c)}{P(a_i | c) P(a_j | c)}. \tag{14}$$

Learning an HNB is quite simple, and mainly, about estimating the parameters in the HNB from the training data. The learning algorithm for HNB is depicted as follows:

Algorithm. HNB(D)
Input: a set D of training examples
Output: a hidden naive Bayes for D
  for each value c of C
    Compute $P(c)$ from D
  for each pair of attributes A_i and A_j
    for each assignment a_i, a_j, and c to A_i, A_j, and C
      Compute $P(a_i|a_j, c)$ from D
  for each pair of attributes A_i and A_j
    Compute $I_P(A_i; A_j|C)$ and W_{ij} from D

Besides, in our implementation of HNB, probabilities $P(c)$ and $P(a_i|a_j, c)$ are estimated using the M-estimation as follows:

TABLE 1
Description of Data Sets Used in the Experiments

No.	Dataset	Examples	Attributes	Classes	Missing value	Numeric value
1	anneal	898	39	6	Y	Y
2	anneal.ORIG	898	39	6	Y	Y
3	audiology	226	70	24	Y	N
4	autos	205	26	7	Y	Y
5	balance-scale	625	5	3	N	Y
6	breast-cancer	286	10	2	Y	N
7	breast-w	699	10	2	Y	N
8	colic	368	23	2	Y	Y
9	colic.ORIG	368	28	2	Y	Y
10	credit-a	690	16	2	Y	Y
11	credit-g	1000	21	2	N	Y
12	diabetes	768	9	2	N	Y
13	Glass	214	10	7	N	Y
14	heart-c	303	14	5	Y	Y
15	heart-h	294	14	5	Y	Y
16	heart-statlog	270	14	2	N	Y
17	hepatitis	155	20	2	Y	Y
18	hypothyroid	3772	30	4	Y	Y
19	ionosphere	351	35	2	N	Y
20	iris	150	5	3	N	Y
21	kr-vs-kp	3196	37	2	N	N
22	labor	57	17	2	Y	Y
23	letter	20000	17	26	N	Y
24	lymph	148	19	4	N	Y
25	mushroom	8124	23	2	Y	N
26	primary-tumor	339	18	21	Y	N
27	segment	2310	20	7	N	Y
28	sick	3772	30	2	Y	Y
29	sonar	208	61	2	N	Y
30	soybean	683	36	19	Y	N
31	splice	3190	62	3	N	N
32	vehicle	846	19	4	N	Y
33	vote	435	17	2	Y	N
34	vowel	990	14	11	N	Y
35	waveform-5000	5000	41	3	N	Y
36	zoo	101	18	7	N	Y

All of these data sets are the whole 36 UCI data sets selected by Weka. We downloaded these data sets in format of arff *from the main website of Weka.*

$$P(c) = \frac{F(c) + 1.0/k}{t + 1.0}, \tag{15}$$

$$P(a_i|a_j, c) = \frac{F(a_i, a_j, c) + 1.0/n_i}{F(a_j, c) + 1.0}, \tag{16}$$

where $F(\bullet)$ is the frequency with which a combination of terms appears in the training data, t is the number of training examples, k is the number of classes, and n_i is the number of values of attribute A_i.

From the algorithm above, we know that the training process of HNB is similar to TAN, except no structure learning. A three-dimensional table of probability estimates for each attribute value, conditioned by each other attribute value and each class, is generated. To create the hidden parent of an attribute, HNB needs to compute the conditional mutual information $I_P(A_i; A_j|C)$ for each pair of attributes. The time complexity for computing weights using (13) is $O(n^2)$. Thus, the training time complexity of HNB is $O(tn^2 + kn^2v^2)$, where t is the number of training examples, n is the number of attributes, k is the number of classes, and v is the average number of values for an attribute. At classification time, given an example, (11) is used, and it takes $O(kn^2)$.

In principle, HNB is a structure-extension-based algorithm. In HNB, we essentially extend the structure of naive Bayes by creating a layer of hidden parents. Thus, the resulting structure is more complex than naive Bayes. In that sense, it is similar to TAN and AODE.

Compared to TAN, which has a training time complexity of $O(tn^2 + kn^2v^2 + n^2logn)$ and classification time complexity of $O(kn)$, HNB does not have structure learning with the time complexity of $O(n^2logn)$ in TAN. Thus, the training time complexity of HNB is lower than that of TAN.

Compared to AODE, which has a training time complexity of $O(tn^2)$ and classification time complexity of $O(kn^2)$, HNB needs more training time and same classification time. HNB, however, has an explicit semantics. Roughly speaking, a hidden parent for an attribute can be seen as aggregating the influences from all other attributes that

TABLE 2
The Detailed Experimental Results on Accuracy and Standard Deviation

Data set	NB	SBC	NBTree	TAN	AODE	HNB
anneal	94.32±2.23	96.94±2.03	98.4±1.53	96.75±1.73	96.74±1.72	98.62±1.14
anneal.ORIG	88.16±3.06	89.68±2.92	91.27±3.03	90.48±2.16	88.79±3.17	91.6±2.63
audiology	71.4±6.37	74.06±7.07	76.66±7.47	65.3±6.81	71.66±6.42	73.15±6
autos	63.97±11.35	68.69±11.27	74.75±9.44	72.59±9.64	73.38±10.24	78.04±9.43
balance-scale	91.44±1.3	91.44±1.3	91.44±1.3	85.97±2.95	89.78±1.88	89.65±2.42
breast-cancer	72.94±7.71	72.53±7.52	71.66±7.92	69.53±6.55	72.53±7.15	70.23±6.49
breast-w	97.3±1.75	96.58±2.19	97.23±1.76	95.52±2.38	97.11±1.99	96.08±2.46
colic	78.86±6.05	83.37±5.56	82.5±6.51	80.03±5.99	80.9±6.19	81.25±6.27
colic.ORIG	74.21±7.09	74.83±6.17	74.83±7.82	67.76±6.07	75.3±6.6	75.5±6.57
credit-a	84.74±3.83	85.36±3.99	84.86±3.92	84.19±4.15	85.91±3.78	84.84±4.43
credit-g	75.93±3.87	74.76±3.85	75.54±3.92	74.84±3.86	76.42±3.86	76.86±3.64
diabetes	75.68±4.85	76±5.24	75.28±4.84	76.04±4.85	76.37±4.35	75.83±4.86
glass	57.69±10.07	56.37±10.04	58±9.42	58.64±9.06	61.13±9.79	59.33±8.83
heart-c	83.44±6.27	81.12±7.15	81.1±7.24	79.83±8.55	82.48±6.96	81.43±7.35
heart-h	83.64±5.85	80.19±7.03	82.46±6.26	81.2±5.97	84.06±5.85	80.72±6
heart-statlog	83.78±5.41	80.85±7.61	82.26±6.5	79.59±5.87	83.67±5.37	81.74±5.94
hepatitis	84.06±9.91	82.51±8.48	82.9±9.79	83±9.11	84.82±9.75	82.71±9.95
hypothyroid	92.79±0.73	93.46±0.5	93.05±0.65	93.35±0.59	93.53±0.62	93.28±0.52
ionosphere	90.86±4.33	91.25±4.14	89.18±4.82	91.4±4.5	92.08±4.24	93.02±3.98
iris	94.33±6.79	96.67±4.59	95.27±6.16	94.07±5.68	94.47±6.22	93.93±6
kr-vs-kp	87.79±1.91	94.34±1.3	97.81±2.05	92.86±1.47	91.01±1.67	92.35±1.32
labor	96.7±7.27	82.63±12.69	95.6±8.39	89±12.39	95.3±8.49	90.87±13.15
letter	65.8±2.04	67.74±2.19	74.32±2.12	77.02±1.83	77.66±2.02	82.31±1.74
lymph	85.97±8.88	80.24±9.58	82.21±8.95	84.51±9.39	86.25±9.43	82.93±8.96
mushroom	93.58±2.03	99.09±1.22	99.88±0.26	99.88±0.26	99.94±0.19	99.94±0.19
primary-tumor	47.2±6.02	44.49±6.76	45.84±6.61	44.8±6.74	47.67±6.3	47.85±6.06
segment	89.03±1.66	90.65±1.77	92.64±1.61	93.91±1.57	92.94±1.4	94.72±1.42
sick	96.78±0.91	97.51±0.72	97.86±0.69	97.69±0.69	97.51±0.73	97.78±0.73
sonar	76.35±9.94	69.78±9.74	71.4±8.8	75.39±9.47	79.04±9.42	80.89±8.68
soybean	92.2±3.23	91.99±3.16	92.3±2.7	94.93±2.44	93.28±2.84	94.67±2.25
splice	95.42±1.14	94.95±1.29	95.42±1.14	94.87±1.23	96.12±1	96.13±0.99
vehicle	61.03±3.48	60.98±3.62	68.91±4.58	73.34±3.8	71.62±3.6	73.63±3.86
vote	90.21±3.95	95.59±2.76	94.78±3.32	94.43±3.34	94.52±3.19	94.36±3.2
vowel	66.09±4.78	68.59±4.5	88.01±3.71	91.87±2.8	89.52±3.12	92.99±2.49
waveform-5000	79.8±2.97	78.69±4.25	78.77±3.71	79.1±3.48	84.87±3.07	84.31±3.02
zoo	94.37±6.79	94.04±7.34	94.55±6.54	96.63±5.84	94.66±6.38	99.9±1
Mean	82.16±4.88	82.17±5.04	84.14±4.87	83.34±4.81	84.81±4.69	85.10±4.55

are assigned higher weights with higher influences. Actually, the weights in HNB can be assigned by human experts, which allows an effective interaction between human experts and the learning program. In addition, we should notice that AODE is an ensemble learning method, in which a collection of models is built and their predictions are combined, whereas only a single model is learned in HNB.

4 Experiments and Results

We ran our experiments on all the 36 UCI data sets [27] selected by Weka [28], which represent a wide range of domains and data characteristics and are described in Table 1. We downloaded these data sets in the format of *arff* from the main Website of Weka. In our experiments, we adopted the following four preprocessing steps:

1. *Replacing missing attribute values*. We used the unsupervised filter named *ReplaceMissingValues* in Weka to replace all missing attribute values in each data set. *ReplaceMissingValues* replaces all missing values with the modes and means from the training data.
2. *Discretizing numeric attribute values*. Numeric attributes were discretized by the filter of *Discretize* in Weka using unsupervised 10-bin discretization.
3. *Removing useless attributes*. Apparently, if the number of values of an attribute is almost equal to the number of examples in a data set, it rarely contributes to classification. Thus, we used the unsupervised filter named *Remove* in Weka to remove this type of attribute. In these 36 data sets, there are only three such attributes: the attribute "Hospital Number" in the data set "colic.ORIG," the attribute "instance

TABLE 3
The Compared Results of Two-Tailed t-Test on Accuracy with the 95 Percent Confidence Level

	NB	SBC	NBTree	TAN	AODE
SBC	10/24/2				
NBTree	11/25/0	7/29/0			
TAN	13/17/6	5/27/4	5/24/7		
AODE	13/22/1	11/23/2	5/27/4	8/25/3	
HNB	17/18/1	13/21/2	10/24/2	9/27/0	8/28/0

TABLE 4
The Detailed Experimental Results on CLL and Standard Deviation

Data set	NB	SBC	NBTree	TAN	AODE	HNB
anneal	-14.22±6.16	-12.4±6.53	-18.46±17.63	-11.03±6.65	-8.75±5.15	-5.1±5.6
anneal.ORIG	-23.58±5.6	-22.9±5.66	-33.33±16.32	-24±7.45	-22.78±5.43	-17.47±6.23
audiology	-65.91±24.28	-23.33±6.96	-95.28±41.89	-82.89±22.19	-64.95±23.83	-64.77±26.43
autos	-45.57±18.12	-19.91±5.94	-34.94±16.81	-35.33±17.07	-27.26±12.29	-32.34±17.55
balance-scale	-31.75±1.51	-31.75±1.51	-31.75±1.51	-33.61±2.39	-33.26±1.6	-28.74±2.21
breast-cancer	-18.37±4.49	-16.64±2.79	-20.47±5.23	-18.81±3.62	-16.87±3.44	-18.24±3.51
breast-w	-18.28±14.16	-13.07±9.54	-17.47±13.63	-9.62±5.51	-7.19±4.87	-10.32±6.16
colic	-30.63±11.38	-16.78±4.33	-34.42±17.34	-24.49±8.74	-20.49±6.92	-21.04±7.29
colic.ORIG	-21.24±5.74	-18.58±3.58	-38.5±17.6	-35.23±9.88	-19.96±4.71	-20.58±5.42
credit-a	-28.79±8.1	-27.26±5.39	-34.52±11.89	-30.42±6.84	-26.53±7.08	-28±7.42
credit-g	-52.79±6.35	-51.99±5.62	-62.44±23.22	-59.02±7.91	-51.42±5.78	-51.3±6.34
diabetes	-40.78±7.49	-38.57±6.34	-42.7±9.11	-40.87±7.49	-38.47±6.07	-40.56±7.21
glass	-24.08±5.42	-23.75±4.76	-31.06±9.62	-27.23±5.5	-22.26±4.86	-21.81±5.52
heart-c	-13.91±6.71	-13.91±4.57	-15.7±7.49	-14.97±5.54	-12.98±5.66	-14.33±6.32
heart-h	-13.49±5.37	-14.86±4.61	-14.73±5.94	-14.07±3.97	-12.64±4.55	-13.79±4.5
heart-statlog	-12.25±4.96	-12.18±3.91	-16.31±9.29	-13.42±4.15	-11.95±4.59	-13.69±5.02
hepatitis	-8.53±5.98	-6.73±2.98	-9.18±5.78	-7.6±4.31	-6.67±4.49	-6.87±3.93
hypothyroid	-97.14±13.29	-93.89±9.11	-98.23±14.58	-95.75±13.4	-88.57±11.17	-86.94±10.29
ionosphere	-34.79±19.94	-10.31±4.79	-35.54±20.03	-19.01±13.06	-20.48±13.8	-18.01±13.05
iris	-2.56±2.35	-2.01±1.73	-2.69±2.9	-2.44±1.77	-2.62±2.16	-3±2.3
kr-vs-kp	-93.48±7.65	-97.93±4.58	-28.01±18.07	-56.91±7.74	-78.32±5.99	-69.73±5.63
labor	-0.71±0.99	-2.75±1.5	-1.03±2.27	-2.38±2.91	-0.84±1.06	-1.55±2
letter	-563.93±43.41	-522.82±38.7	-595.33±81.52	-368.29±34.34	-318.33±28.78	-260.16±26.21
lymph	-6.22±3.96	-7.52±2.98	-8.48±5.51	-7.38±4.58	-5.56±3.57	-6.7±3.74
mushroom	-34.79±13.37	-8.92±5.86	-2.52±5.92	-1.34±2.92	-0.89±2.02	-0.61±1.49
primary-tumor	-65.56±8.27	-65.07±8.18	-74.19±14.56	-69.69±8.4	-64.2±7.71	-65.96±8.7
segment	-124.32±33.74	-64.22±14.01	-111.94±45.14	-48.82±15.42	-50.97±12.88	-32.97±9.88
sick	-46.05±11.99	-34.08±9.14	-45.55±19.82	-29.05±9.2	-32.37±8.93	-25.64±7.41
sonar	-22.67±11.47	-13.71±4.28	-38.85±19.05	-21.83±9.74	-15.85±8.62	-15.89±8.55
soybean	-26.25±11.03	-18.58±5.5	-28.63±15.19	-9.35±4.29	-15.65±6.57	-9.03±3.61
splice	-46.53±12.85	-49.11±12.23	-47.11±13.57	-49.24±12.15	-42.05±15.87	-38.1±10.57
vehicle	-172.12±27.55	-84.56±11.55	-137.97±32.69	-60.31±10.2	-67.07±9.77	-53.58±7.59
vote	-27.25±13.85	-7.55±3.52	-7.35±5.41	-7.73±5.12	-7.51±4.68	-7.52±4.67
vowel	-89.8±11.38	-83.78±10.66	-45.93±16.44	-25.4±7.55	-31.16±6.8	-20.14±7.24
waveform-5000	-75.82±14.33	-50.61±7.74	-106.58±31.18	-64.39±13.54	-37.21±7.86	-42.23±9.89
zoo	-1.22±1.06	-3.22±1.66	-1.29±1.68	-1.06±1.28	-1.09±0.97	-0.43±0.39
Mean	-55.43±11.23	-44.03±6.74	-54.68±16.55	-39.53±8.523	-35.70±7.51	-32.42±7.50

name" in the data set "splice," and the attribute "animal" in the data set "zoo."

4. *Sampling large data sets*. For saving the time of running experiments, we used the unsupervised filter named *Resample*with the size of 20 percent in Weka to randomly sample each large data set having more than 5,000 examples. In these 36 data sets, there are three such data sets: "letter," "mushroom," and "waveform-5,000."

We conducted empirical experiments to compare HNB with naive Bayes (NB), SBC [13], NBTree [21], TAN [1], and averaged one-dependence estimators (AODE) [12]. Note that we did not choose the algorithms from the data expansion approach, because data expansion is actually a preprocessing method, which can be applied to any algorithm. The purpose of our experiments is to compare the essential (core) performance in extending naive Bayes. We also did not include the existing attribute-weighted naive Bayes algorithms, since they are currently not very competitive in terms of performance, to our experience.

TABLE 5
The Compared Results of Two-Tailed t-Test on CLL with the 95 Percent Confidence Level

	NB	SBC	NBTree	TAN	AODE
SBC	15/19/2				
NBTree	5/25/6	5/15/16			
TAN	13/17/6	9/15/12	11/23/2		
AODE	26/9/1	14/17/5	23/11/2	13/19/4	
HNB	21/15/0	16/17/3	19/16/1	18/17/1	11/20/5

We evaluated the selected algorithms in terms of classification (measured by accuracy), class probability estimation (measured by CLL), and ranking (measured by AUC). The accuracy of each model is based on the percentage of successful predictions on the test sets of each data set, and the CLL and AUC scores are calculated using (5) and (3), (4). We implemented HNB, SBC, and TAN within the Weka framework [28], and used the implementation of NB, NBTree, and AODE in Weka.

In all experiments, the accuracy, CLL, and AUC scores of each model on each data set were obtained via 10 runs of 10-fold cross validation. Runs with the various algorithms were carried out on the same training sets and evaluated on

TABLE 6
The Detailed Experimental Results on AUC and Standard Deviation

Data set	NB	SBC	NBTree	TAN	AODE	HNB
anneal	96.1±1.18	95.12±2.38	96.23±1.29	96.46±0.45	96.18±1.06	96.45±0.64
anneal.ORIG	94.26±4.23	94.27±4.35	95.13±2.6	94.78±3.47	94.37±4.11	94.9±3.67
audiology	71.08±0.64	70.92±0.74	71.06±0.69	71.04±0.65	71.09±0.63	71.23±0.6
autos	90.07±4.93	91.08±4.14	93.54±2.96	91.97±4.83	92.43±3.84	93.59±3.17
balance-scale	84.08±4.42	84.08±4.42	84.08±4.42	80.77±4.64	79.14±4	86.76±4.33
breast-cancer	68.24±11.93	67.43±12.68	66.01±10.94	63.13±12.08	69.03±10.37	64.96±11.18
breast-w	99.22±0.76	99.2±0.76	99.21±0.75	98.84±1.01	99.32±0.69	98.97±0.99
colic	83.22±7.46	84.54±6.75	86.04±7.65	84.78±6.53	85.54±6.68	85.68±6.57
colic.ORIG	80.57±8.03	80.47±7.82	78.81±8.76	71.43±8.57	81.67±7.67	83.06±5.59
credit-a	91.71±3.16	86.78±4.76	91.14±3.36	90.24±3.08	92.18±3	90.84±3.69
credit-g	79.02±4.22	77.72±4.92	77.49±5.34	76.56±4.97	79.35±4.15	79.31±4.42
diabetes	82.51±5	82.17±6.4	81.99±5.1	81.76±5	82.68±4.86	82.32±5
glass	80.89±5.9	81.21±6.01	82±6.08	79.43±6.06	81.98±5.91	85.02±4.66
heart-c	84.05±0.6	83.77±0.67	83.93±0.62	83.7±0.7	84.05±0.61	83.87±0.68
heart-h	83.9±0.62	83.23±0.94	83.79±0.66	83.6±0.59	83.91±0.58	83.69±0.64
heart-statlog	90.85±5.12	87.63±7.03	89.28±6.26	88.39±4.81	90.69±5.04	88.5±5.66
hepatitis	88.41±10.91	81.91±14.29	84.74±12.28	86.01±10.82	88.36±11.08	86.98±10.79
hypothyroid	87.78±6.12	85.43±5.61	87.47±6.34	86.42±6.68	86.86±6.91	88.62±5.69
ionosphere	93.4±4.79	93.06±5.33	94.04±4.42	97.95±2.52	97.19±2.47	98.11±2.04
iris	98.64±2.17	98.43±2	98.84±2.01	98.94±2.05	98.55±2.34	98.72±2.36
kr-vs-kp	95.16±1.2	96.35±0.9	99.44±0.6	98.25±0.62	97.4±0.77	98.2±0.56
labor	98.17±7.36	73.21±25.02	97.42±11.59	92.67±13.47	98.33±6.38	95.92±10.43
letter	95.58±0.56	95.99±0.55	96.57±0.63	97.97±0.35	98.3±0.29	98.82±0.21
lymph	90±1.71	87.99±3.55	89.05±2.47	89.02±3.48	90.11±1.78	89.88±2.19
mushroom	99.56±0.3	99.46±1.13	99.97±0.09	100±0.01	100±0.01	100±0.01
primary-tumor	78.88±1.76	78.28±1.89	78.12±1.8	78.4±1.75	78.87±1.82	78.98±1.71
segment	98.5±0.41	98.94±0.36	99.08±0.34	99.53±0.19	99.43±0.2	99.7±0.15
sick	95.83±2.4	94.75±3.39	94.27±3.62	98.1±1.21	96.97±1.78	98.16±1.27
sonar	84.17±9.52	75.32±11.54	77.54±9.9	82.38±9.33	87.82±8.71	89.18±7.29
soybean	99.73±0.34	98.83±1.16	99.68±0.41	99.78±0.36	99.76±0.34	99.82±0.33
splice	99.45±0.28	99.2±0.42	99.43±0.31	99.35±0.36	99.56±0.25	99.57±0.24
vehicle	80.31±3.09	81.32±3.3	85.66±3.43	90.57±2.04	89.76±2.09	90.53±2.01
vote	96.95±2.14	94.02±2.78	98.51±1.67	98.66±1.25	98.58±1.32	98.69±1.2
vowel	95.58±1.12	96.15±1.04	98.46±0.84	99.46±0.35	99.31±0.41	99.67±0.24
waveform-5000	94.99±1.66	93.19±2.1	91.19±3.15	92.62±2.15	96.58±1.31	96.53±1.35
zoo	89.48±2.37	88.68±2.66	89.48±2.37	89.48±2.37	89.48±2.37	89.48±2.37
Mean	89.45±3.57	87.78±4.55	89.41±3.77	89.23±3.58	90.41±3.22	90.69±3.16

the same test sets. In particular, the cross-validation folds are the same for all the experiments on each data set. Finally, we compared related algorithms via two-tailed t-test with a 95 percent confidence level. According to the statistical theory, we speak of two results for a data set as being "significantly different" only if the probability of significant difference is at least 95 percent [29].

Tables 2, 4, and 6, respectively, show the accuracy, CLL, and AUC scores of each model on each data set, and the average values and standard deviation on all data sets are summarized at the bottom of the table. Tables 3, 5, and 7 show the compared results of two-tailed t-test, in which each entry $w/t/l$ means that the model in the corresponding row wins in w data sets, ties in t data sets, and loses in l data sets, compared to the model in the corresponding column.

TABLE 7
The Compared Results of Two-Tailed t-Test on AUC with the 95 Percent Confidence Level

	NB	SBC	NBTree	TAN	AODE
SBC	4/22/10				
NBTree	8/26/2	11/25/0			
TAN	9/23/4	12/23/1	6/29/1		
AODE	13/22/1	16/19/1	9/25/2	10/23/3	
HNB	16/19/1	18/18/0	9/26/1	11/25/0	8/26/2

From our experiments, we can see that the performance of HNB, not only in classification but also in class probability estimation and ranking, is overall the best among the models compared in the paper. Now, we summarize some highlights briefly as follows:

1. SBC significantly outperforms NB in accuracy (10 wins and 2 losses) and CLL (15 wins and 2 losses), but performs even worse than NB in AUC (4 wins and 10 losses).
2. NBTree significantly outperforms NB in accuracy (11 wins and 0 losses), almost ties NB in CLL (5 wins and 6 losses), and slightly outperforms NB in AUC (8 wins and 2 losses).
3. TAN significantly outperforms NB in accuracy (13 wins and 6 losses) and CLL (13 wins and 6 losses), and slightly outperforms NB in AUC (9 wins and 4 losses).

4. AODE significantly outperforms NB in accuracy (13 wins and 1 loss), CLL (26 wins and 1 loss), and AUC (13 wins and 1 loss).
5. HNB significantly outperforms NB in accuracy (17 wins and 1 losses), CLL (21 wins and 0 loss), and AUC (16 wins and 1 loss).
6. HNB significantly outperforms all the other improved models of naive Bayes not only in accuracy but also in CLL and AUC.

5 Conclusions

In this paper, we summarize the existing improved algorithms for naive Bayes and propose a novel Bayes model: HNB. We conducted a systematic experimental study on the classification, class probability estimation, and ranking performance of HNB. Our experimental results show that HNB has a better overall performance compared to the other state-of-the-art models for augmenting naive Bayes. Considering its simplicity, HNB is a promising model that could be used in many real-world applications.

The HNB that we implemented is based on one-dependence estimators. It could be generalized to arbitrary dependence estimators. Thus, HNB can be seen as a general model in which structure learning plays a less important role than in Bayesian networks. In defining and learning an HNB, how to learn the weights is crucial. Currently, we use conditional mutual information to estimate the weights directly from data. We believe that the use of more sophisticated methods, such as EM, could improve the performance of the current HNB and make its advantage stronger. This is the main research direction for our future work.

In addition, the structure of HNB contains one hidden layer. It can be expected to extend to more complex structures, such as double layers [30]. This will be another topic for future research.

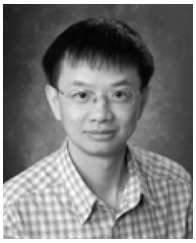
Acknowledgments

This paper is an extended version of AAAI 2005 and ICML 2005 conference papers. The authors thank anonymous reviewers for their very useful comments and suggestions.

References

[1] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Machine Learning,* vol. 29, pp. 131-163, 1997.
[2] D.M. Chickering, "Learning Bayesian Networks is NP-Complete," *Learning from Data: Artificial Intelligence and Statistics V,* D. Fisher and H. Lenz, eds., pp. 121-130, Springer-Verlag, 1996.
[3] C.X. Ling and H. Zhang, "Toward Bayesian Classifiers with Accurate Probabilities," *Proc. Sixth Pacific-Asia Conf. Knowledge Discovery and Data Mining (KDD '02),* pp. 123-134, 2002.
[4] P. Domingos, "MetaCost: A General Method for Making Classifiers Cost Sensitive," *Proc. Fifth Int'l Conf. Knowledge Discovery and Data Mining,* pp. 155-164, 1999.
[5] A.P. Bradley, "The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms," *Pattern Recognition,* vol. 30, pp. 1145-1159, 1997.
[6] C.X. Ling, J. Huang, and H. Zhang, "AUC: A Statistically Consistent and More Discriminating Measure than Accuracy," *Proc. Int'l Joint Conf. Artificial Intelligence (IJCAI '03),* pp. 329-341, 2003.
[7] D.J. Hand and R.J. Till, "A Simple Generalisation of the Area under the ROC Curve for Multiple Class Classification Problems," *Machine Learning,* vol. 45, pp. 171-186, 2001.
[8] D. Grossman and P. Domingos, "Learning Bayesian Network Classifiers by Maximizing Conditional Likelihood," *Proc. 21st Int'l Conf. Machine Learning,* pp. 361-368, 2004.
[9] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kauhmann, 1988.
[10] E. Keogh and M. Pazzani, "Learning Augmented Bayesian Classifiers: A Comparison of Distribution-Based and Classification-Based Approaches," *Proc. Int'l Workshop Artificial Intelligence and Statistics,* pp. 225-230, 1999.
[11] H. Zhang and C.X. Ling, "An Improved Learning Algorithm for Augmented Naive Bayes," *Proc. Fifth Pacific-Asia Conf. Knowledge Discovery and Data Mining (KDD '01),* pp. 581-586, 2001.
[12] G.I. Webb, J. Boughton, and Z. Wang, "Not So Naive Bayes: Aggregating One-Dependence Estimators," *Machine Learning,* vol. 58, pp. 5-24, 2005.
[13] P. Langley and S. Sage, "Induction of Selective Bayesian Classifiers," *Proc. 10th Conf. Uncertainty in Artificial Intelligence,* pp. 339-406, 1994.
[14] L. Jiang, H. Zhang, Z. Cai, and J. Su, "Evolutional Naive Bayes," *Proc. First Int'l Symp. Intelligent Computation and Its Applications (ISICA '05),* pp. 344-350, 2005.
[15] R. Kohavi and G. John, "Wrappers for Feature Subset Selection," *Artificial Intelligence J.,* vol. 97, nos. 1/2, pp. 273-324, 1997.
[16] C.A. Ratanamahatana and D. Gunopulos, "Scaling up the Naive Bayesian Classifier: Using Decision Trees for Feature Selection," *Proc. Workshop Data Cleaning and Preprocessing (DCAP '02), at IEEE Int'l Conf. Data Mining (ICDM '02),* 2002.
[17] J.T.A.S. Ferreira, D.G.T. Denison, and D.J. Hand, "Weighted Naive Bayes Modelling for Data Mining," Dept. of Math., Imperial College, 2001.
[18] H. Zhang and S. Sheng, "Learning Weighted Naive Bayes with Accurate Ranking," *Proc. Fourth IEEE Int'l Conf. Data Mining (ICDM '04),* pp. 567-570, 2004.
[19] W. Deng, G. Wang, and Y. Wang, "Weighted Naive Bayes Classification Algorithm Based on Rough Set," *Computer Science,* vol. 34, pp. 204-206, 2007.
[20] M. Hall, "A Decision Tree-Based Attribute Weighting Filter for Naive Bayes," *Knowledge-Based Systems,* vol. 20, pp. 120-126, 2007.
[21] R. Kohavi, "Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid," *Proc. Second Int'l Conf. Knowledge Discovery and Data Mining (KDD '96),* pp. 202-207, 1996.
[22] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
[23] E. Frank, M. Hall, and B. Pfahringer, "Locally Weighted Naive Bayes," *Proc. Conf. Uncertainty in Artificial Intelligence,* pp. 249-256, 2003.
[24] Z. Zheng and G.I. Webb, "Lazy Learning of Bayesian Rules," *Machine Learning,* vol. 41, no. 1, pp. 53-84, 2000.
[25] Z. Xie, W. Hsu, Z. Liu, and M. Lee, "SNNB: A Selective Neighborhood Based Naive Bayes for Lazy Learning," *Proc. Sixth Pacific-Asia Conf. Knowledge Discovery and Data Mining (KDD '02),* pp. 104-114, 2002.
[26] L. Jiang, D. Wang, H. Zhang, Z. Cai, and B. Huang, "Using Instance Cloning to Improve Naive Bayes for Ranking," *Int'l J. Pattern Recognition and Artificial Intelligence,* vol. 22, no. 6, pp. 1121-1140, 2008.
[27] C. Merz, P. Murphy, and D. Aha, "UCI Repository of Machine Learning Databases," Dept. of ICS, Univ. of California, http://www.ics.uci.edu/mlearn/MLRepository.html, 1997.
[28] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques,* second ed. Morgan Kaufmann, http://prdownloads.sourceforge.net/weka/datasets-UCI.jar, 2005.
[29] C. Nadeau and Y. Bengio, "Inference for The Generalization Error," *Advances in Neural Information Processing Systems,* vol. 12, pp. 307-313, MIT Press, 1999.
[30] J. Sun, C. Wang, and S. Chen, "A Double Layer Bayesian Classifier," *Proc. Fourth Int'l Conf. Fuzzy Systems and Knowledge Discovery (FSKD '07),* vol. 1, pp. 540-544, 2007.

Liangxiao Jiang received the MSc degree from the China University of Geosciences, Wuhan, where he is currently a lecturer in the Faculty of Computer Science. In 2004 and 2005, he was a visiting scholar in the Faculty of Computer Science at the University of New Brunswick. His research interests include data mining, machine learning, and artificial intelligence.



Harry Zhang received the PhD degree from the University of Western Ontario, Canada. Currently, he is an associate professor in the Faculty of Computer Science at the University of New Brunswick.

Zhihua Cai received the PhD degree from the China University of Geosciences, Wuhan, where he is currently a professor in the Faculty of Computer Science.