Augmenting Naive Bayes for Ranking

Harry Zhang HZHANG@UNB.CA
Faculty of Computer Science, University of New Brunswick, NB, Canada, E3B 5A3

Liangxiao Jiang LJIANG@CUG.EDU.CN
Faculty of Computer Science, China University of Geosciences, Wuhan, China, 430074

Jiang Su K4KM1@UNB.CA
Faculty of Computer Science, University of New Brunswick, NB, Canada, E3B 5A3

Abstract

Naive Bayes is an effective and efficient learning algorithm in classification. In many applications, however, an accurate ranking of instances based on the class probability is more desirable. Unfortunately, naive Bayes has been found to produce poor probability estimates. Numerous techniques have been proposed to extend naive Bayes for better classification accuracy, of which selective Bayesian classifiers (SBC) (Langley & Sage, 1994), tree-augmented naive Bayes (TAN) (Friedman et al., 1997), NBTree (Kohavi, 1996), boosted naive Bayes (Elkan, 1997), and AODE (Webb et al., 2005) achieve remarkable improvement over naive Bayes in terms of classification accuracy. An interesting question is: Do these techniques also produce accurate ranking? In this paper, we first conduct a systematic experimental study on their efficacy for ranking. Then, we propose a new approach to augmenting naive Bayes for generating accurate ranking, called *hidden naive Bayes* (HNB). In an HNB, a hidden parent is created for each attribute to represent the influences from all other attributes, and thus a more accurate ranking is expected. HNB inherits the structural simplicity of naive Bayes and can be easily learned without structure learning. Our experiments show that HNB outperforms naive Bayes, SBC, boosted naive Bayes, NBTree, and TAN significantly, and performs slightly better than AODE in ranking.

1. Introduction

Traditionally, the performance of a classifier is measured by its classification accuracy (or error rate). Some classifiers, such as naive Bayes and decision trees, also produce the estimates of the class probability $p(c|E)$ that is the probability of instance E in the class c. This information is often ignored in classification, as long as the class with the highest class probability estimate is identical to the actual class. In many applications, however, classification and error rate are not enough. For example, a CS department usually needs a ranking of its students in terms of their performance in order to award various scholarships. Thus, a ranking is more desirable.

If a ranking is desired and only a training data set with class labels is given, the area under the ROC (Receiver Operating Characteristics) curve (Swets, 1988; Provost & Fawcett, 1997), or simply AUC, can be used to evaluate the quality of rankings generated by a classifier. AUC is a good "summary" for comparing two classifiers across the entire range of class distributions and error costs. Bradley (Bradley, 1997) shows that AUC is a proper metric for the quality of classifiers averaged across all possible probability thresholds. It has been shown that, for binary classification, AUC is equivalent to the probability that a randomly chosen instance of class $-$ will have a smaller estimated probability of belonging to class $+$ than a randomly chosen instance of class $+$ (Hand & Till, 2001). Thus, AUC measures the quality of a ranking. The AUC of a ranking is 1 (the maximum AUC value) if no positive instance precedes any negative instance.

Naive Bayes is one of the most effective and efficient classification algorithms. In classification learning problems, a learner attempts to construct a classifier from a given set of training instances with class la-

Appearing in *Proceedings of the 22^{nd} International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

bels. Assume that A_1, A_2,$\cdots$, A_n are n attributes. An instance E is represented by a vector $(a_1, a_2, , \cdots, a_n)$, where a_i is the value of A_i. Let C represent the class variable and c represent the value that C takes. A naive Bayesian classifier, or simply naive Bayes, is defined as follows.

$$g(E) = \arg\max_c p(c) \prod_{i=1}^{n} p(a_i|c).$$

In naive Bayes, all attributes are assumed independent given the class; that is,

$$p(a_1, a_2, \cdots, a_n|c) = \prod_{i=1}^{n} p(a_i|c).$$

Figure 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 $P(a_i|c)$ can be easily estimated from training instances, naive Bayes is easy to construct. It is also, however, surprisingly effective (Kononenko, 1990; Langley et al., 1992; Domingos & Pazzani, 1997). It is obvious that the conditional independence assumption is rarely true in reality. Indeed, naive Bayes has been found to work poorly for regression problems (Frank et al., 2000), and produces poor probability estimates (Bennett, 2000).

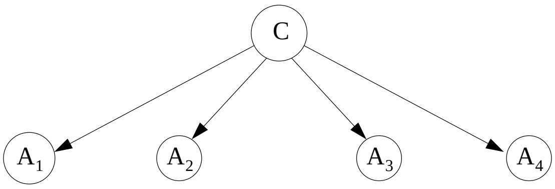

Figure 1. An example of naive Bayes

One way to alleviate the conditional independence assumption is to extend the structure of naive Bayes to represent explicitly attribute dependencies by adding arcs between attributes. Tree augmented naive Bayes (TAN) is an extended tree-like naive Bayes (Friedman et al., 1997), in which the class node directly points to all attribute nodes and an attribute node can have only one parent from another attribute node. Figure 2 shows an example of TAN.

TAN is a specific case of general augmented naive Bayesian networks, or simply ANB, in which the class node also directly points to all attribute nodes, but there is no limitation on the links among attribute nodes (except that they do not form any directed cycle). ANB can represent arbitrary attribute dependencies. Intuitively, ANB yields more accurate probability estimates and thus generates more accurate rankings. In reality, however, when the structural complexity of an ANB is high, its variance is also high due to the limited training data (Friedman et al., 1997), and thus its probability estimates could be still poor. In addition, learning an optimal ANB is similar to learning an optimal Bayesian network that has been proved to be NP-hard (Chickering, 1996). In practice, learning TAN is more feasible, due to its acceptable computational complexity and considerable improvement over naive Bayes. One main issue in learning TAN is that only one attribute parent is allowed for each attribute, and the influences from other attributes have to be ignored. Thus the rankings yielded by TAN could be inaccurate. Certainly, a model that can represents the influences on an attribute from all other attributes and still retains the structural simplicity, is desirable.

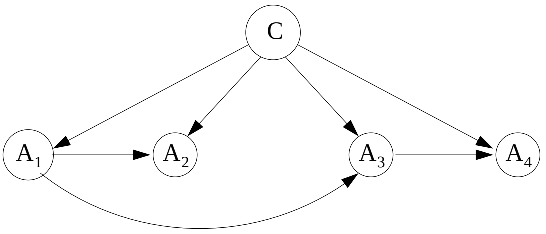

Figure 2. An example of TAN

The rest of the paper is organized as follows. In Section 2, we introduce the related work. In Section 3, we describe an empirical study on the ranking performance of the widely used techniques for augmenting naive Bayes. In Section 4, we present a novel model *hidden naive Bayes* for augmenting naive Bayes to produce accurate ranking. In Section 5, we make a conclusion and outline the main directions for future research.

2. Related Work

The ranking addressed in this paper is based on the class probabilities of instances. Ranking is different from both classification and probability estimation. For example, assume that E_+ and E_- are a positive and a negative instance respectively, and that the actual class probabilities are $p(+|E_+) = 0.9$ and $p(+|E_-) = 0.1$. An algorithm that gives class probability estimates: $\hat{p}(+|E_+) = 0.55$ and $\hat{p}(+|E_-) = 0.54$, gives a correct order of E_+ and E_- in the ranking. Notice that the probability estimates are poor and the classification for E_- is incorrect (assume that the threshold for classification is 0.5). However, If a learning algorithm produces accurate class probability estimates, it certainly produces an accurate ranking. Thus, aiming at learning a model to yield accurate probability estimates will usually lead to a model yielding accurate probability-based ranking.

Recently, a considerable amount of research work has been done on learning decision trees with accurate ranking. Traditional decision tree algorithms, such as C4.5 (Quinlan, 1993), have been observed to produce poor estimates of probabilities (Pazzani et al., 1994; Provost et al., 1998). Provost and Domingos (Provost & Domingos, 2003) propose the two techniques to improve the AUC of C4.5: smooth probability estimates by Laplace correction and turning off pruning and collapsing.

Naive Bayes is easy to construct and has surprisingly good performance in classification, even though the conditional independence assumption is rarely true in real-world applications. On the other hand, naive Bayes has been found to produce poor probability estimates (Bennett, 2000). Some work has been proposed to produce well-calibrated probabilities (Platt, 1999; Zadrozny & Elkan, 2001; Zadrozny & Elkan, 2002; R.Caruana & Niculescu-Mizil, 2005).

Many techniques have been proposed for extending naive Bayes. One approach to improving naive Bayes is selecting an attribute subset in which attributes are conditionally independent. Of the proposed techniques, *selective Bayesian classifier* (SBC) by (Langley & Sage, 1994) demonstrates remarkable improvement over naive Bayes. In SBC, a forward greedy search method is used to select a subset of attributes. A more effective and straightforward approach to improving naive Bayes is to extend its structure to represent dependencies among attributes, such as TAN and ANB. Friedman et al. (1997) propose a TAN learning algorithm based on conditional mutual information, which is an extension of the Chow-Liu algorithm (Chow & Liu, 1968). The conditional mutual information is defined as

$$I_P(X;Y|Z) = \sum_{x,y,z} P(x,y,z) log \frac{P(x,y|z)}{P(x|z)P(y|z)}, \quad (1)$$

where x, y, and z are the values of variables X, Y, and Z respectively. In TAN, $I_P(A_i;A_j|C)$ 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 by C that points to all attribute nodes is added.

When we aim at classification (discriminative learning), maximizing conditional likelihood, instead of maximizing likelihood, is desired. However, the computational cost is very high. In recent years, some research work has been done to deal with this problem (Greiner & Zhou, 2002; Grossman & Domingos, 2004).

Kohavi (1996) presents a model NBTree to combine a decision tree with naive Bayes. In an NBTree, a local naive Bayes is deployed on each leaf of a traditional decision tree, and an instance is classified using the local naive Bayes on the leaf into which it falls. The experiments show that NBTree outperforms naive Bayes significantly in accuracy.

Elkan (1997) proposes to apply boosting technique to naive Bayes. The resulting model is called boosted naive Bayes. The experiments show that boosted naive Bayes performs better than naive Bayes in accuracy.

The most recent work on improving naive Bayes is AODE (averaged one-dependence estimators) (Webb et al., 2005). In AODE, an ensemble of one-dependence classifiers are learned and the prediction is produced by aggregating the predictions of all qualified classifiers. The notion of x-dependence is introduced by Sahami (Sahami, 1996). An x-dependence estimator means that the probability of an attribute is conditioned by the class variable and at most x other attributes, which corresponds to an ANB with at most x attribute parents. In AODE, a one-dependence classifier is built for each attribute, in which the attribute is set to be the parent of all other attributes. Their experimental results show that AODE performs surprisingly well compared to other classification algorithms.

3. Empirical Study

Most techniques for augmenting naive Bayes aim at improving its classification accuracy. Do these techniques also result in accurate ranking? Which of them do? We conduct an empirical study to answer these two questions.

In our experiments, the AUC of a classifier on a data set with two classes is computed using the following formula.

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

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 the i_{th} positive example in the ranking.

Table 1. Description of the data sets used in the experiments.

DATA SET	SIZE	ATTR.	CLASSES	MISSING
ANNEAL	898	39	6	Y
ANNEAL.O	898	39	6	Y
AUDIOLOGY	226	70	24	Y
AUTOS	205	26	7	Y
BALANCE-S.	625	5	3	N
BREAST-C.	286	10	2	Y
BREAST-W	699	10	2	Y
COLIC	368	23	2	Y
COLIC.O	368	28	2	Y
CREDIT-A	690	16	2	Y
CREDIT-G	1000	21	2	N
DIABETES	768	9	2	N
GLASS	214	10	7	N
HEART-C	303	14	5	Y
HEART-H	294	14	5	Y
HEART-S	270	14	2	N
HEPATITIS	155	20	2	Y
HYPOTH.	3772	30	4	Y
IONOSPH.	351	35	2	N
IRIS	150	5	3	N
KR-VS-KP	3196	37	2	N
LABOR	57	17	2	Y
LETTER	20000	17	26	N
LYMPH	148	19	4	N
MUSHROOM	8124	23	2	Y
P.-TUMOR	339	18	21	Y
SEGMENT	2310	20	7	N
SICK	3772	30	2	Y
SONAR	208	61	2	N
SOYBEAN	683	36	19	Y
SPLICE	3190	62	3	N
VEHICLE	846	19	4	N
VOTE	435	17	2	Y
VOWEL	990	14	11	N
WAVEFORM	5000	41	3	N
ZOO	101	18	7	N

Multi-class AUC is calculated by M-measure in (Hand & Till, 2001).

Our experiments are conducted on the 36 datasets form Weka (Witten & Frank, 2000), which are listed in Table 1. All these data sets are from the UCI repository (Blake & Merz, 2000). We downloaded these data sets in the format of *arff* from main web of Weka (Witten & Frank, 2000). We adopted the following preprocessing stages on each data set.

1. The missing values of attributes in each data set were replaced by using the filter of *ReplaceMissingValues* in Weka.

2. Numeric attributes were discretized by the filter of *Discretize* in Weka using unsupervised ten-bin dicretization. Thus, all attributes were treated as nominal.

3. We notice that, if the number of values of an attribute is almost equal to the number of instances in a data set, this attribute does not contribute any information to classification. For example, the student ID numbers is not useful for classification purpose. So we used the filter of *Remove* in Weka to delete this type of attributes. In these 36 data sets, there only exists three this type of attributes, namely "Hospital Number" in data set horse-colic.ORIG, "Instance Name" in data set Splice and "Animal" in data set zoo.

We conducted experiments to compare naive Bayes (Langley et al., 1992), SBC (Langley & Sage, 1994), NBTree (Kohavi, 1996), TAN (Friedman et al., 1997), boosted naive Bayes (Elkan, 1997), and AODE (Webb et al., 2005) in terms of AUC. We implemented SBC, boosted naive Bayes, and TAN within the Weka framework (Witten & Frank, 2000), and used the implementation of naive Bayes, NBTree, and AODE in Weka. In all experiments, the AUC of an algorithm on a data set was obtained via 10 runs of ten-fold cross validation. Runs with the various algorithms were carried out on the same training sets and evaluated on the same test sets. Finally, we conducted two-tailed t-test with a 95% confidence level to compare each pair of algorithms.

Table 2 shows the AUC scores of the algorithms on each data set, and the average AUC and standard deviation on all data sets are summarized at the bottom of the table. Table 3 shows the results of two-tailed t-test, in which each entry $w/t/l$ means that the algorithm in the corresponding row wins in w data sets, ties in t data sets, and loses in l data sets, compared to the algorithm in the corresponding column. From our experiments, we have the following observations:

1. AODE achieves a better performance compared to naive Bayes, TAN, NBTree, and boosted naive Bayes.

2. NBTree achieves considerable improvement over naive Bayes in AUC (8 wins and 2 losses). Notice that a local naive Bayes is deployed on each leaf to calibrate probability estimates in NBTree. In addition, it is also believed that the instances in a leaf has less chance of having strong attribute dependencies.

3. TAN does not achieve significant improvement over naive Bayes in AUC (9 wins and 7 losses).

This indicates that representing the influence from only one attribute is not sufficient to produce accurate rankings. Thus, more sophisticated approach for augmenting naive Bayes should be considered.

4. The performance of SBC is even worse than naive Bayes in AUC (5 wins and 9 losses), although it achieves significant improvement in classification (Langley & Sage, 1994).

5. Boosted naive Bayes also performs worse than naive Bayes in AUC (3 wins and 12 losses), although its performance in accuracy is better than naive Bayes (9 wins and 5 losses in the 36 data sets).

4. Hidden Naive Bayes

As we discussed in Section 1, ANB can represent arbitrary attribute dependencies. The joint distribution represented by an ANB is depicted as follows.

$$P(A_1, A_2, \cdots, A_n, C) = P(C) \prod_{i=1}^{n} P(A_i | pa_i, C), \quad (3)$$

where pa_i denotes the parents of A_i from attribute nodes.

Naive Bayes ignores attribute dependencies. More concretely, pa_i is empty in Equation 3. In a TAN, pa_i consists of at most one attribute. Since for each attribute the influence from only one other attribute can be represented, an obvious problem is that several attributes may have the similar influence and all the influences except one have to be ignored in TAN. Thus, its ranking performance could be still poor, which has been verified by the experimental results in the previous section. From our experimental results, TAN does not perform significantly better than naive Bayes in AUC. This indicates that the rankings yielded by TAN can be improved further. Intuitively, if for each attribute the influences from all other attributes are taken into account, a more accurate model can be expected. ANB is such a model that can represent arbitrary attribute dependencies. However, there are two main problems in learning ANB. First, learning the optimal structure of an ANB is intractable. Learning the structure of an ANB essentially means to determine pa_i for each attribute A_i in Equation 3. Second, the probability estimates of an ANB could be still poor even pa_i for each A_i is known due to the high variance in estimating $P(A_i|pa_i, C)$ from limited training data. Our key idea to tackle the second problem is to approximate $P(A_i|pa_i, C)$ using the weighted one-dependence estimators as follows.

$$P(A_i|pa_i, C) \approx \sum_{A_j \in pa_i} W_{ij} * P(A_i|A_j, C), \quad (4)$$

where $\sum W_{ij} = 1$.

Moreover, we want to avoid the structure learning that is an extremely time-consuming. Our idea is to use the one-dependence estimators from all other attributes. That is, let $pa_i = \{A_1, \cdots, A_{i-1}, A_{i+1}, \cdots, A_n\}$ in Equation 4. The resulting model is called *hidden naive Bayes* (HNB). HNB represents an approximation of the joint distribution defined as follows.

$$\hat{P}(A_1, \cdots, A_n, C) = P(C) \prod_{i=1}^{n} P(A_i|A_{hp_i}, C), \quad (5)$$

where

$$P(A_i|A_{hp_i}, C) = \sum_{j=1, j \neq i}^{n} W_{ij} * P(A_i|A_j, C), \quad (6)$$

and $\sum_{j=1, j \neq i}^{n} W_{ij} = 1$.

In our implementation, we use conditional mutual information to set W_{ij} as follows.

$$W_{ij} = \frac{I_P(A_i; A_j|C)}{\sum_{j=1, j \neq i}^{n} I_P(A_i; A_j|C)}. \quad (7)$$

$I_P(A_i; A_j|C)$ is the conditional mutual information between A_i and A_j, defined in Equation 1.

HNB can be viewed in such a way that a hidden (virtual) parent A_{hp_i} is created for each attribute A_i. A_{hp_i} is a mixture of the weighted influences from all other attributes, and $P(A_i|A_{hp_i}, C)$ is an approximation of $P(A_i|A_1, \cdots, A_{i-1}, A_{i+1}, \cdots, A_n, C)$. In an HNB, attribute dependencies are actually represented through hidden parents of attributes. It is easy to show that TAN is a special case of HNB. Thus, HNB is more expressive than TAN. Compared to TAN, HNB can represent the influences on each attribute from all other attributes and assign higher weights to more important attributes. Thus, HNB should be a more accurate model than TAN with respect to representing attribute dependencies, and the rankings yielded by HNB should be more accurate.

The training process of HNB is very similar to TAN. 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 Equation 7 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 instances, n is the number of attributes, k is the number of classes, and v is the average number of values for an attribute. In practice, t is usually greater than kv^2. Thus HNB is in $O(tn^2)$. Compared to TAN that has a training time complexity of $O(tn^2+kn^2v^2+n^2logn)$, HNB does not have structure learning, and thus is more efficient. Compared to AODE that has the training time complexity of $O(tn^2)$, HNB has the same order in time complexity. However, we should notice that AODE is an ensemble learning method in which a collection of models are built and their predictions are combined, whereas only a single model is learned in HNB.

We implemented HNB in Weka Framework and compared it to naive Bayes, SBC, NBTree, TAN, boosted naive Bayes, and AODE. In the implementation of HNB, we used the Laplace estimation to avoid the zero-frequency problem. More precisely, we estimated the probabilities $P(c)$, $P(a_i|c)$, and $P(a_i|a_j, c)$ using Laplace estimation as follows.

$$\hat{P}(c) = \frac{n_c + 1}{t + k},$$

$$\hat{P}(a_i|c) = \frac{n_{ic} + 1}{n_c + v_i},$$

$$\hat{P}(a_i|a_j, c) = \frac{n_{ijc} + 1}{n_{jc} + v_i},$$

where t is the total number of training instances, k is the number of classes, v_i is the number of values of attribute A_i, n_c is the number of instances in class c, n_{ic} is the number of instances in class c and with $A_i = a_i$, n_{jc} is the number of instances in class c and with $A_j = a_j$, and n_{ijc} is the number of instances in class c and with $A_i = a_i$ and $A_j = a_j$.

The detailed results displayed in Table 2 and Table 3 show that the performance of HNB in ranking is overall the best among the algorithms compared in the paper. Now, we summarize the highlights briefly as follows:

1. HNB achieves significant improvement over naive Bayes in AUC (15 wins and 0 loss).

2. HNB performs significantly better than SBC (17 wins and 0 loss), TAN (13 wins and 1 loss), boosted naive Bayes (18 wins and 2 losses), and NBTree (8 wins and 1 loss).

3. HNB performs slightly better than AODE (5 wins and 2 losses). Considering that HNB is a single model in contrast to an ensemble of models in AODE, HNB is overall more effective.

4. HNB achieves the highest average AUC among all the algorithms.

5. Conclusions

In this paper, we conducted a systematic experimental study on the ranking performance of the widely used techniques for augmenting naive Bayes. We found that some techniques, such as NBTree, TAN and AODE, achieve improvement over naive Bayes in ranking, just as they do in classification. But some others, such as SBC and boosted naive Bayes, do not. We proposed a novel model *hidden Naive Bayes* (HNB) for accurate ranking. In HNB, a hidden parent is created for each attribute and added to naive Bayes. Our experimental results show that HNB has a better overall ranking performance compared to the state-of-the-art algorithms, measured by AUC. Considering the simplicity of HNB, HNB is a promising model that could be used in many real world applications in which an accurate ranking is desired.

Although HNB achieves significant improvement over naive Bayes in AUC, its performance in probability estimation is still unknown. In an HNB, learning high-quality weights is crucial. We believe that some more sophisticated techniques, such as gradient descent and SVM, can be applied to achieve a better performance.

References

Bennett, P. N. (2000). *Assessing the calibration of naive bayes' posterior estimates* (Technical Report CMU-CS-00-155). Carnegie Mellon University, School of Computer Science, Pittsburgh,PA.

Blake, C., & Merz, C. J. (2000). UCI repository of machine learning databases. In *Dept of ics, university of california, irvine*. http://www.ics.uci.edu/ m̃learn/MLRepository.html.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*, 1145–1159.

Chickering, D. M. (1996). Learning bayesian networks is NP-complete. In D. Fisher and H. Lenz (Eds.), *Learning from data: Artificial intelligence and statistics v*, 121–130. Springer-Verlag.

Chow, C. K., & Liu, C. N. (1968). Approximating discrete probability distributions with dependence

Table 2. Experimental results on AUC.

Datasets	NB	BoostNB	SBC	TAN	NBTree	AODE	HNB
anneal	96.1±1.18	94.7±3.81	95.12±2.38	96.56±0.21	96.23±1.29	96.18±1.06	96.19±1.31
anneal.O	94.26±4.23	91.8±4.78	94.27±4.35	95.1±2.89	95.13±2.6	94.37±4.11	94.56±4.19
audiology	71.08±0.64	71.05±0.71	70.92±0.74	70.95±0.6	71.06±0.69	71.09±0.63	71.09±0.61
autos	90.07±4.93	90.68±4.69	91.08±4.14	92.45±4.34	93.54±2.96	92.43±3.84	92.79±3.89
balance-s	84.08±4.42	96.07±2.66	84.08±4.42	77.31±5.44	84.08±4.42	79.14±4	86.55±4.07
breast-c	68.24±11.9	62.75±12.2	67.43±12.7	61.54±10.6	66.01±10.9	69.03±10.4	66.35±11.7
breast-w	99.22±0.76	98.3±1.65	99.2±0.76	98.71±1.07	99.21±0.75	99.32±0.69	99.21±0.79
colic	83.22±7.46	82±6.45	84.54±6.75	84.59±6.56	86.04±7.65	85.54±6.68	85.83±6.71
colic.O	80.57±8.03	78.77±8.09	80.46±7.83	73.02±9.77	78.81±8.76	81.67±7.67	81.65±7.2
credit-a	91.71±3.16	88.25±3.88	86.78±4.76	89.57±3.68	91.14±3.36	92.18±3	91.78±3.34
credit-g	79.02±4.22	73.96±5.33	77.72±4.92	76.45±5.08	77.49±5.34	79.35±4.15	79.94±4.47
diabetes	82.51±5	78.14±6.33	82.17±6.4	80.54±4.96	81.99±5.1	82.68±4.86	82.75±4.88
glass	80.89±5.9	77.95±4.88	81.17±5.95	83.03±6.42	82±6.08	81.98±5.91	79.23±6.23
heart-c	84.05±0.6	83.44±0.73	83.77±0.67	83.56±0.76	83.93±0.62	84.05±0.61	83.97±0.63
heart-h	83.9±0.62	83.42±0.8	83.23±0.94	83.45±0.69	83.79±0.66	83.91±0.58	83.82±0.57
heart-s	90.85±5.12	83.59±7.15	87.63±7.03	87.48±5.24	89.28±6.26	90.69±5.04	90.09±5.03
hepatitis	88.41±10.9	80.72±14.7	81.96±14.28	84.67±11.0	84.74±12.3	88.36±11.1	86.35±11.6
hypoth.	87.78±6.12	86.21±6.45	85.43±5.61	87.25±6.87	87.47±6.34	86.86±6.91	86.41±6.69
ionosph.	93.4±4.79	93.4±5.22	93.06±5.33	98.09±2.17	94.04±4.42	97.19±2.47	97.31±2.57
iris	98.64±2.17	96.59±3.9	98.43±2	98.52±2.46	98.84±2.01	98.55±2.34	98.65±2.22
kr-vs-kp	95.16±1.2	98.76±0.44	96.35±0.9	98.22±0.56	99.44±0.6	97.4±0.77	98.2±0.56
labor	98.17±7.36	92.29±12.5	73.21±25.0	88.75±17.0	97.42±11.6	98.33±6.38	97±8.9
letter	96.88±0.21	88.88±1.1	97.04±0.2	98.91±0.11	98.49±0.17	99.4±0.06	99.34±0.07
lymph	90±1.71	89.75±2.04	87.99±3.55	89.3±3.05	89.05±2.47	90.11±1.78	90.02±2.4
mushroom	99.79±0.07	100±0	99.98±0.02	100±0	100±0	100±0	100±0
p.-tumor	78.88±1.76	76.3±2.68	78.28±1.89	78.3±1.81	78.12±1.8	78.87±1.82	78.75±1.75
segment	98.5±0.41	97.7±0.49	98.94±0.36	99.53±0.22	99.08±0.34	99.43±0.2	99.56±0.18
sick	95.83±2.4	95.81±2.57	94.75±3.39	98.31±1.01	94.27±3.62	96.97±1.78	97.46±1.7
sonar	84.17±9.52	81.44±10.3	75.32±11.5	79.84±10.5	77.54±9.9	87.82±8.71	89.37±7.39
soybean	99.73±0.34	94.48±1.83	98.81±1.16	99.71±0.41	99.68±0.41	99.76±0.34	99.8±0.33
splice	99.45±0.28	98.52±0.71	99.2±0.42	99.37±0.35	99.43±0.31	99.56±0.25	99.54±0.25
vehicle	80.31±3.09	78.55±3.26	81.32±3.3	89.42±1.95	85.66±3.43	89.76±2.09	89.9±1.97
vote	96.95±2.14	96.39±2.95	94.02±2.78	98.27±1.48	98.51±1.67	98.58±1.32	98.7±1.18
vowel	95.58±1.12	94.9±1.72	96.15±1.04	99.59±0.28	98.46±0.84	99.31±0.41	99.49±0.33
waveform	95.29±0.68	89.42±1.32	95.14±0.67	93.81±0.82	93.69±0.96	96.44±0.56	96.39±0.55
zoo	89.48±2.37	89.33±2.45	88.68±2.66	89.48±2.37	89.48±2.37	89.48±2.37	89.48±2.37
Mean	89.50±3.52	87.62±4.19	87.88±4.47	88.99±3.69	89.53±3.69	90.44±3.19	90.49±3.29

trees. *IEEE Trans. on Information Theory*, *14*, 462–467.

Domingos, P., & Pazzani, M. (1997). Beyond independence: Conditions for the optimality of the simple bayesian classifier. *Machine Learning*, *29*, 103–130.

Elkan, C. (1997). *Boosting and Naive Bayesian learning* (Technical Report CS97-557). University of California, San Diego.

Frank, E., Trigg, L., Holmes, G., & Witten, I. H. (2000). Naive bayes for regression. *Machine Learning*, *41*, 5–15.

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

Greiner, R., & Zhou, W. (2002). Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Proceedings of the 17th National Conference on Artificial Intelligence* (pp. 167–173). AAAI Press.

Grossman, D., & Domingos, P. (2004). Learning bayesian network classifiers by maximizing conditional likelihood. *Proceedings of the 21th international conference on Machine learning.* Banff, Alberta, Canada: ACM Press.

Hand, D. J., & Till, R. J. (2001). A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, *45*, 171–186.

Kohavi, R. (1996). Scaling up the accuracy of naive-

Table 3. Summary of experimental results: AUC comparisons.

	NB	BoostNB	SBC	TAN	NBTree	AODE
BoostNB	3/21/12					
SBC	5/22/9	11/19/6				
TAN	9/20/7	13/20/3	12/21/3			
NBTree	8/26/2	14/21/1	12/23/1	4/26/6		
AODE	13/22/1	18/16/2	17/18/1	13/20/3	9/25/2	
HNB	15/21/0	18/16/2	17/19/0	13/22/1	8/27/1	5/29/2

bayes classifiers: A decision-tree hybrid. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 202–207). AAAI Press.

Kononenko, I. (1990). Comparison of inductive and naive bayesian learning approaches to automatic knowledge acquisition. In B. Wielinga (Ed.), *Current trends in knowledge acquisition.* IOS Press.

Langley, P., Iba, W., & Thomas, K. (1992). An analysis of bayesian classifiers. *Proceedings of the Tenth National Conference of Artificial Intelligence* (pp. 223–228). AAAI Press.

Langley, P., & Sage, S. (1994). Induction of selective bayesian classifiers. *Proceedings of Uncertainty in Artificial Intelligence* (pp. 400–406). Morgan Kaufmann.

Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., & Brunk, C. (1994). Reducing misclassification costs. *Proceedings of the 11th International conference on Machine Learning* (pp. 217–225). Morgan Kaufmann.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. *Advances in Large Margin Classifiers* (pp. 61–74). MIT Press.

Provost, F., & Fawcett, T. (1997). Analysis and visualization of classifier performance: comparison under imprecise class and cost distribution. *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (pp. 43–48). AAAI Press.

Provost, F., Fawcett, T., & Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 445–453). Morgan Kaufmann.

Provost, F. J., & Domingos, P. (2003). Tree induction for probability-based ranking. *Machine Learning, 52*, 199–215.

Quinlan, J. (1993). *C4.5: Programs for machine learning.* San Mateo: Morgan Kaufmann.

R.Caruana, & Niculescu-Mizil, A. (2005). Predicting good probabilities with supervised learning. *Proceedings of the American Meteorology Conference.* San Diego.

Sahami, M. (1996). Learning limited dependence bayesian classifiers. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 335–338). AAAI Press.

Swets, J. (1988). Measuring the accuracy of diagnostic systems. *Science, 240*, 1285–1293.

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

Witten, I. H., & Frank, E. (2000). *Data mining-practical machine learning tools and techniques with java implementation.* Morgan Kaufmann.

Zadrozny, B., & Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining* (pp. 204–213). San Francisco, California: ACM Press.

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining* (pp. 694–699). Edmonton, Alberta, Canada: ACM Press.