Learning Weighted Naive Bayes with Accurate Ranking

Harry Zhang
Faculty of Computer Science
University of New Brunswick
Fredericton, NB, Canada E3B 5A3
hzhang@unb.ca

Shengli Sheng
Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 5B7
ssheng@uwo.ca

Abstract

Naive Bayes is one of most effective classification algorithms. In many applications, however, a ranking of examples are more desirable than just classification. How to extend naive Bayes to improve its ranking performance is an interesting and useful question in practice. Weighted naive Bayes is an extension of naive Bayes, in which attributes have different weights. This paper investigates how to learn a weighted naive Bayes with accurate ranking from data, or more precisely, how to learn the weights of a weighted naive Bayes to produce accurate ranking. We explore various methods: the gain ratio method, the hill climbing method, and 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. Our experiments show that a weighted naive Bayes trained to produce accurate ranking outperforms naive Bayes.

1 Introduction

Naive Bayesian Classifier, or simply naive Bayes, is one of the most effective and efficient classification algorithms. In classification, a classifier, which assigns a class label to an example, is built from a set of training examples with class labels. Assume that A_1, A_2,$\cdots$, A_n are n attributes. An example 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 that corresponds to the class, and c represent the value that C takes. In naive Bayes, all attributes are assumed independent given the value of the class variable (conditional independence assumption):

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

An example E is classified to the class with the maximum posterior probability. More precise, the classification on E given by naive Bayes, denoted by $V_{nb}(E)$, is defined as below:

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

Since the conditional independence assumption is rarely true in reality, it is natural to extend naive Bayes to relax the conditional independence assumption. There are two major ways to do it: (1) The structure of naive Bayes is extended to represent explicitly the dependences among attributes, and the resulting model is called augmented naive Bayes (ANB) [3]. (2)Attributes are weighted differently, and the resulting model is called weighted naive Bayes (WNB). A WNB is formally defined as below.

$$V_{wnb}(E) = \arg\max_c p(c) \prod_{i=1}^{n} p(a_i|c)^{w_i}, \quad (2)$$

where $V_{wnb}(E)$ denotes the classification give by the WNB, and w_i is the weight of attribute A_i.

In recent years, AUC has been noticed by machine learning and data mining community, and some researchers believe [4] that AUC is a more discriminant evaluation method than error rate for learning algorithms that also produce class probability estimates. Since naive Bayes performances well in terms of accuracy [1], a natural question is: What is the performance of naive Bayes, in terms of ranking, or AUC? Can we improve its AUC by using some sort of extended naive Bayes, such as WNBs?

The remainder of this paper is organized as follows: In Section 2, we present several algorithms for learning a WNB with high AUC. We implemented these learning algorithms and the related experimental results are presented in Section 3.

Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)
0-7695-2142-8/04 $ 20.00 IEEE

2 Learning Weighted Naive Bayes with High AUC

As we discussed in Section 1, a WNB is an extension of naive Bayes in which attributes are assigned different weights. How to learn the weight vector in a WNB is the key in learning a WNB. In this section, we investigate various methods for learning the weight vector in a WNB.

2.1 Gain Ratio

Gain ratio was originally used to choose an attribute that classifies best among a set of attributes in the decision tree algorithm [7]. We argue that an attribute of higher gain ratio deserves higher weight in a WNB. Thus, we propose a gain ratio based method that calculate the weight of an attribute from a dataset as follows:

$$w_i = \frac{GainRatio(Tr, A_i) \times n}{\sum_{i=1}^{n} GainRatio(Tr, A_i)}, \quad (3)$$

where n is the number of attributes, and Tr is the dataset. The gain ratio of attribute A_i is defined as [7].

The WNB based on gain ratio is denoted by WNB-G.

2.2 Hill Climbing

This method has been widely used in many areas. In a WNB, the weight w_i of attribute A_i is found by a search process consisting of a sequence of steps. In each step, the weight is revised to achieve higher AUC, according to the rule below:

$$w_i(n) \leftarrow w_i(n-1) + \Delta w(n), \quad (4)$$

where $w_i(n)$ and $w_i(n-1)$ are the weights at step n and step $n-1$ respectively, and $\Delta w(n)$ is the weight update performed at step n. In our implementation, $\Delta w(n)$ is defined as follows:

$$\Delta w(n) = \eta O(auc)(1 - O(auc))^2, \quad (5)$$

where η is the learning rate, auc is the current value of AUC, and $O(auc)$ is defined as:

$$O(auc) = \frac{1}{1 + e^{-auc}}.$$

The initial weight of an attribute is assigned to 1. We adjust the weight of each attribute separately. For each attribute A_i, the weight w_i is repeatedly revised until the increase between the current AUC and the previous AUC is less than a small value ϵ.

The WNB based on hill climbing is denoted by WNB-HC.

2.3 Markov Chain Monte Carlo

The weights vector of a WNB can be also learned through a random walk. First, we initialize the weight vector as $< 1, 1, \cdots, 1 >$. Then we revise the weights by randomly choosing adding or subtracting a constant value Δw or unchanging it in each step. More precisely, we randomly choose one from the three equations below:

$$w_i(n) \leftarrow w_i(n-1) + \Delta w,$$
$$w_i(n) \leftarrow w_i(n-1) - \Delta w,$$
$$w_i(n) \leftarrow w_i(n-1).$$

The constant value Δw can be adjusted manually until it achieves the best performance in AUC in most datasets. In experiments, we find an appropriate Δw value manually for each dataset.

Like the hill climbing method, we stop adjusting the weights when the increase between the current AUC and the previous AUC is less than a very small value ϵ. However, we adjust all the weights in a WNB simultaneously, unlike the hill climbing method, in which we adjust each weight individually.

We investigate both Monte Carlo and Markov Chain Monte Carlo in a random walk. In the Monte Carlo method, we can try to achieve a higher AUC by adjusting the direction randomly in each step. However, in the Markov Chain Monte Carlo method, we only randomly choose the direction at the beginning. We then adjust the weights in the same direction, as long as the current direction causes the AUC to increase. We readjust the direction only when it does not achieve a higher AUC. Our experiments show that the Markov Chain Monte Carlo method outperforms the Monte Carlo method.

A WNB with Markov Chain Monte Carlo is denoted by WNB-MCMC.

2.4 Combined Methods

In the hill climbing method and the Markov Chain Monte Carlo method, the initial weight vector is $< 1, 1, \cdots, 1 >$. Since the weight vector can be calculated directly by the gain ratio method, we can use it as the initial value, and apply hill climbing or Markov Chain Monte Carlo to search for a better weight vector. Thus, we have two corresponding combined methods: hill climbing with gain ratio, denoted by WNB-G-HC, and Markov Chain Monte Carlo with gain ratio, denoted by WNB-G-MCMC.

3 Experiment

The process of learning a WNB consists of two steps: (1)learning $p(a_i|c)$; (2)learning the weight vector. The first

Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)
0-7695-2142-8/04 $ 20.00 IEEE

Table 1. Description of the datasets used in the experiments.

Dataset	# Attributes	# Classes	# examples
Abalone	8	7	4177
Australia	11	2	690
Breast	9	10	683
Cars	6	2	446
Dermatology	33	6	366
Ecolidis	6	2	332
Hepatitis	3	2	320
Importdis	23	2	205
Iris	4	3	150
Lungcancer	56	2	32
Pima	6	2	392
Segment	18	9	2310
Vehicle	18	3	846
Vote	16	2	232

step is straightforward just as in naive Bayes. In this section, we conduct experiments to investigate the methods for learning the weight vector of a WNB described in Section 2, and each of them corresponds to a variant of WNBs. We also compare them with naive Bayes and the decision tree learning algorithm C4.4 [6], the revised version of the decision tree learning algorithm C4.5 [7] for accurate probability estimation.

All the experiments are based on eight datasets from the UCI repository [5]. Table 1 shows the properties of the datasets. In these datasets, all continuous attributes are discretized using the entropy-based method [2].

Since the Laplace correction used in C4.4 significantly increases the AUC value, we use it in naive Bayes and the five variants of WNBs. Instead of simply estimating $p(a_i|c)$ by the percentage of the number of examples with $A_i = a_i$ among the number of examples in class c, the Laplace correction used in our experiments is:

$$p(a_i|c) = \frac{N_{a_i} + 1}{N_c + k},$$

where N_c is the number of examples in class c, N_{a_i} is the number of examples in class c and with $A_i = a_i$, and k is the number of classes.

For each dataset, we ran all of the five variants of WNBs, naive Bayes (NB in short), and C4.4 with the 5-fold cross-validation 6 times. Our experiments follow the procedure below:

1. For each dataset, discretize the continuous values of the attributes by the entropy-based method [2].

Table 2. Experimental results on the variants of weighted naive Bayes in AUC. In this table, G, HC, MCMC, G-HC and G-MCMC stand for WNB-G, WNB-HC, W-MCMC, WNB-G-HC and WNB-G-MCMC, respectively.

Dataset	G	HC	MCMC	G-HC	G-MCMC
Abalone	98.1±0.04	98.3±0.04	98.0 ±0.26	98.3±0.05	98.1±0.24
Australia	77.4±0.32	76.9±0.28	76.6±0.42	77.9±0.24	76.7±0.17
Breast	81.9±1.48	80.6±1.70	80.8±1.64	80.7±1.89	81.4±1.69
Cars	91.0±0.48	91.1±0.53	91.1±0.48	91.1±0.59	91.0±0.52
Dermatology	100.0±0.05	99.9±0.04	99.9±0.05	99.9±0.0	100.0±0.05
Ecolidis	99.5±0.12	99.4±0.08	99.4±0.08	99.4±0.08	99.4±0.09
Hepatitis	62.7±0.77	62.7±0.77	62.4±0.92	62.4±0.92	62.3±0.84
Importdis	99.9±0.08	99.9±0.08	100.0±0.05	100.0±0.04	100.0±0.05
Iris	90.9±1.22	90.9±1.22	90.9±1.22	90.9±1.22	90.9±1.22
Lungcancer	71.9±8.41	84.9±3.09	84.9±3.09	84.7±7.34	76.1±4.27
Pima	77.1±0.29	77.1±0.39	76.8±0.44	77.4±0.21	76.9±0.56
Segment	89.4±1.10	91.0±1.03	90.6±1.21	91.1±0.89	90.4±1.19
Vehicle	95.2±0.40	95.5±0.50	96.1±0.30	95.8±0.40	95.9±0.30
Vote	87.9±0.84	87.2±0.44	86.9±0.53	87.7±0.91	87.7±0.46

Table 3. Experimental results on naive Bayes and C4.4 in AUC .

Dataset	C4.4	NB
Abalone	75.3±3.43	97.9±0.0
Australia	72.3±0.33	75.7±0.30
Breast	59.0±3.17	80.4±1.66
Cars	86.1±0.99	91.1±0.48
Dermatology	99.6±0.13	99.9±0.0
Ecolidis	97.2±0.48	99.2±0.12
Hepatitis	61.6±1.17	62.7±0.7
Importdis	100.0±0.0	99.5±0.18
Iris	86.2±2.80	90.9±1.22
Lungcancer	78.4±2.28	70.9±4.5
Pima	74.2±0.69	76.8±0.32
Segment	80.7±3.11	88.9±1.19
Vehicle	92.4±0.48	92.0±0.51
Vote	82.8±2.04	86.6±0.34

2. Run each variant of WNBs, naive Bayes, and C4.4 with 5-fold cross-validation, and obtain the accuracy and AUC on the test sets.

3. Repeat step 2 above 6 times and obtain an average accuracy and AUC on the test sets for each variant of WNBs, naive Bayes, and C4.4.

After the experiments, we analyzed the results with ANOVA contrasts by using 90% as the confidence level. We compared the five variants of WNBs with naive Bayes and C4.4 separately. Further more, we also compared the five variants of WNBs each other. The experimental results are shown in Table 2 and 3 (we use two tables due to the space limitation.) The comparisons among all the methods are summarized in Table 4.

Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)
0-7695-2142-8/04 $ 20.00 IEEE

Table 4. Summary of the experimental results in AUC. An entry w-t-l means that the algorithm at the corresponding row wins in w datasets, ties in t datasets, and loses in l datasets, compared to the algorithm at the corresponding column.

Variant	C4.4	NB	G	HC	MCMC	G-HC
NB	11-1-2					
G	12-0-2	7-7-0				
HC	13-1-0	7-7-0	3-9-2			
MCMC	12-2-0	6-8-0	3-8-3	1-12-1		
G-HC	12-2-0	9-5-0	5-8-1	1-13-0	4-10-0	
G-MCMC	11-3-0	8-6-0	1-12-1	0-12-2	1-12-1	1-9-4

From Table 4, we can observe four interesting facts. Firstly, all variants of WNBs outperform naive Bayes in terms of AUC. More precisely, all of WNB-G, WNB-HC and WNB-MCMC outperform naive Bayes in 7 datasets and does not lose in any dataset; WNB-G-HC outperforms naive Bayes in 9 datasets and loses in 0 dataset; WNB-G-MCMC outperforms naive Bayes in 8 datasets and lose in 0 dataset.

The second observation is that the combined methods WNB-G-HC and G-MCMC outperform slightly the original methods WNB-G, WNB-HC and WNB-MCMC. We can see that WNB-G-HC performs better than WNB-G (5 wins, 8 ties, 1 loss). It is also slightly better than WNB-HC (1 wins, 13 ties, 0 loss). In addition, compared with naive Bayes, it outperforms naive Bayes (9 wins, 5 ties, 0 loss) in more datasets than WNB-HC (7 wins, 7 ties, 0 loss). If just looking at the values of AUC, WNB-G-HC has higher values of AUC than WNB-HC in 7 datasets. WNB-G-MCMC also performs slightly better than WNB-MCMC. Compared with naive Bayes, it also outperforms naive Bayes (8 wins, 6 ties, 0 loss) in more datasets than WNB-MCMC (6 wins, 8 ties, 0 loss).

Another interesting fact is that both the five variants of WNBs and naive Bayes outperform C4.4 in terms of AUC. All WNBs outperform C4.4 on Most of 14 datasets. Besides WNB-G loses to C4.4 in two datasets, other variants of WNBs don't lose to C4.4. Naive Bayes outperforms C4.4 on 11 out of 14 datasets, ties with C4.4 in one dataset Vehicle, and loses to C4.4 in two datasets. This observation indicates that naive Bayes and its extensions may have better performance than the decision tree algorithm in accurate ranking.

Lastly, WNB-G-HC is the best WNB among the five variants of WNBs, according to the experimental results, although the differences among them are not that big. This observation suggests us that WNB-G-HC is the first choice in practice.

4 Conclusions and Discussions

In this paper, we studied various methods for learning the weight vector in a WNB for accurate ranking. Our experiments show us a few interesting facts illustrated below.

1. WNBs outperform naive Bayes in terms of accurate ranking. Hence it should be used if the goal is to achieve good ranking.

2. Among the WNBs studied in this paper, the hill climbing method with gain ratio is the best one.

3. WNBs and naive Bayes have better performance than the decision tree learning algorithm C4.4 in ranking.

To our knowledge, this paper has two major contributions:

1. Although there are some recent works addressing the issue of learning a WNB in term of accuracy, none of them directly addresses learning a WNB with accurate ranking, measured by AUC. This paper studied systematically various methods for learning a WNB with high AUC.

2. We proposed using gain ratio to calculate the weight of an attribute, which can be combined with other search based methods to achieve better performance. Our experiments show that gain ratio based method (WNB-G-HC) achieves the best performance among the methods studied in this paper.

References

[1] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. *Machine Learning*, 29:103–130, 1997.

[2] U. Fayyad and K. Irani. *Multi-interval discretization of continuous-valued attributes for classification learning*. Proceedings of Thirteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann, 1993.

[3] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[4] C. X. Ling, J. Huang, and H. Zhang. *AUC: a statistically consistent and more discriminating measure than accuracy*. Proceedings of the International Joint Conference on Artificial Intelligence IJCAI03. Morgan Kaufmann, 2003.

[5] C. Merz, P. Murphy, and D. Aha. UCI repository of machine learning databases. 1997.

[6] F. J. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, 2003.

[7] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)
0-7695-2142-8/04 $ 20.00 IEEE