Pattern Recognition 111 (2021) 107674

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Attribute and instance weighted naive Bayes

Huan Zhang [a], Liangxiao Jiang [a,b,*], Liangjun Yu [c]

[a] School of Computer Science, China University of Geosciences, Wuhan 430074, China
[b] Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China
[c] School of Computer Science, Hubei University of Education, Wuhan 430205, China

ARTICLE INFO

Article history:
Received 21 December 2019
Revised 24 June 2020
Accepted 22 September 2020

Keywords:
Naive Bayes
Attribute weighting
Instance weighting
Eager learning
Lazy learning

ABSTRACT

Naive Bayes (NB) continues to be one of the top 10 data mining algorithms, but its conditional independence assumption rarely holds true in real-world applications. Therefore, many different categories of improved approaches, including attribute weighting and instance weighting, have been proposed to alleviate this assumption. However, few of these approaches simultaneously pay attention to attribute weighting and instance weighting. In this study, we propose a new improved model called attribute and instance weighted naive Bayes (AIWNB), which combines attribute weighting with instance weighting into one uniform framework. In AIWNB, the attribute weights are incorporated into the naive Bayesian classification formula, and then the prior and conditional probabilities are estimated using instance weighted training data. To learn instance weights, we single out an eager approach and a lazy approach, and thus two different versions are created, which we denote as AIWNB^{E} and AIWNB^{L}, respectively. Extensive experimental results show that both AIWNB^{E} and AIWNB^{L} significantly outperform NB and all the other existing state-of-the-art competitors.

1. Introduction

Classification is a basic task in data mining and pattern recognition [12]. Due to its easiness to construct but surprising effectiveness, naive Bayes (NB) continues to be one of the top 10 data mining algorithms [33].

Given a test instance $\mathbf{x}$, represented by an attribute value vector $\langle a_1, a_2, \cdots, a_m\rangle$, NB utilizes Eq. 1 to predict its class label.

$$c(\mathbf{x}) = \arg\max_{c\in C} P(c)\prod_{j=1}^{m} P\left(a_j|c\right), \tag{1}$$

where C is the collection of all possible class labels c, m is the number of attributes, a_j is the value for the jth attribute A_j of $\mathbf{x}$, $P(c)$ is the prior probability of the class c, and $P(a_j|c)$ is the conditional probability of $A_j = a_j$ given the class c, which can be estimated by the following m-estimation:

$$P(c) = \frac{\sum_{i=1}^{n}\delta(c_i, c) + \frac{1}{q}}{n+1}, \tag{2}$$

$$P(a_j|c) = \frac{\sum_{i=1}^{n}\delta(a_{ij}, a_j)\delta(c_i, c) + \frac{1}{n_j}}{\sum_{i=1}^{n}\delta(c_i, c) + 1}, \tag{3}$$

where n is the number of training instances, q is the number of classes, c_i is the class label of the ith training instance, a_{ij} is the jth attribute value of the ith training instance, n_j is the number of values for the jth attribute A_j, and $\delta(\bullet)$ is a binary function, which is 1 if its two parameters are identical and 0 otherwise.

NB predicts the class label with the highest posterior probability, and its performance is competitive with other state-of-the-art classifiers such as C4.5 [9]. Nonetheless, NB makes the conditional independence assumption that all attributes are fully independent given the class. Since the assumption required by NB hardly holds true in real-world applications, a mass of enhancements have been proposed to relax this assumption. We summarize them into six categories as shown in Fig. 1.

Attribute weighting is practical to alleviate NB's primary weakness [17,21,37]. In the attribute set, some attributes are more important than others, so they should have more influences to the final model than less important attributes. Thus, attribute weighting assigns a discriminative weight to each attribute. Then the attribute weights are incorporated into the naive Bayesian classification formula to build an attribute weighted NB.

* Corresponding author.
E-mail address: ljiang@cug.edu.cn (L. Jiang).

https://doi.org/10.1016/j.patcog.2020.107674
0031-3203/

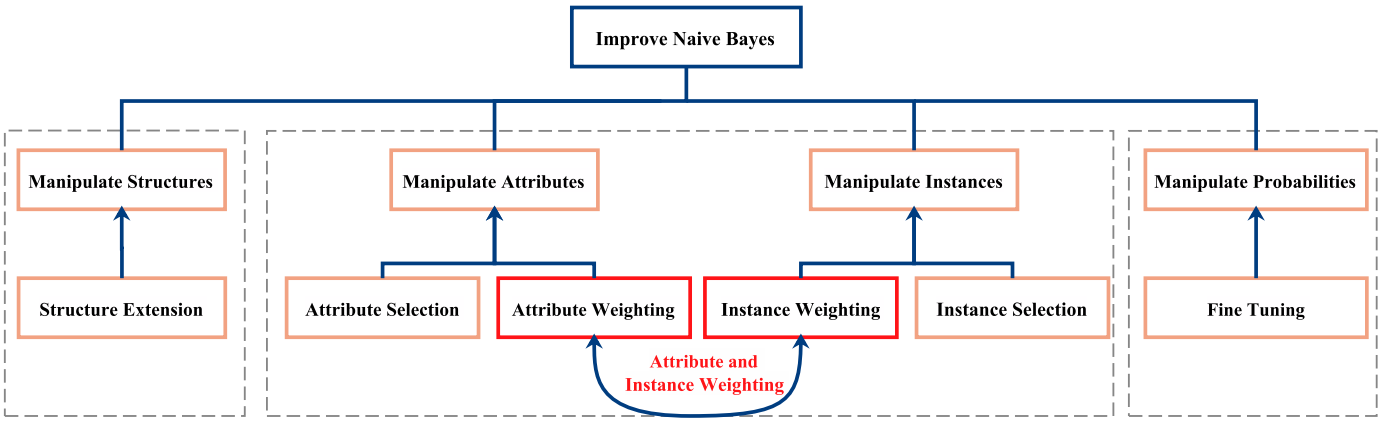

Fig. 1. Different paradigms of improved naive Bayes.

Instance weighting is another effective method to relax NB's primary weakness [16,18]. Some training instances are more reliable than others, and they should have more influences to the final model than less reliable instances. Thus, instance weighting assigns different weights to different instances. Then the weight of each instance is incorporated into the formula of the prior and conditional probabilities to get a more accurate probability estimation.

To the best of our knowledge, however, essentially all the existing weighting approaches are either attribute weighting or instance weighting. Few of them simultaneously pay attention to attribute weighting and instance weighting. We argue that different attributes should have different importance, meanwhile different instances should have different reliability. Thus, in this study, we propose a new improved model called attribute and instance weighted naive Bayes (AIWNB), which combines attribute weighting with instance weighting into one uniform framework. To learn attribute weights, a correlation-based attribute weighting approach is used. To learn instance weights, we single out an eager approach and a lazy approach, and thus two different versions are created, which we denote as AIWNB$^{\mathrm{E}}$ and AIWNB$^{\mathrm{L}}$, respectively. In AIWNB$^{\mathrm{E}}$, we employ an eager learning approach called attribute value frequency-based instance weighting to obtain instance weights. In AIWNB$^{\mathrm{L}}$, we employ a lazy learning approach called similarity-based instance weighting to obtain instance weights. Finally, the learned attribute weights are incorporated into the naive Bayesian classification formula, meanwhile the prior and conditional probabilities are estimated using instance weighted training data. We expect AIWNB could inherit the advantages of both attribute weighting and instance weighting simultaneously. To validate the effectiveness of AIWNB, we conduct two groups of empirical studies on a collection of 36 benchmark datasets from the University of California at Irvine (UCI) repository. The experimental results demonstrate that both AIWNB$^{\mathrm{E}}$ and AIWNB$^{\mathrm{L}}$ perform much better than their state-of-the-art competitors.

In summary, the main contributions and innovations of this paper can be highlighted as follows:

1) We summarize the improved NB paradigms into six categories and conduct a comprehensive survey on them. For each category, we provide detailed analysis and review on some representative approaches.
2) We propose a new improved model called attribute and instance weighted naive Bayes (AIWNB) and give the general framework of AIWNB, which pays attention to attribute weighting and instance weighting simultaneously.
3) We single out a correlation-based attribute weighting approach to learn attribute weights. Meanwhile, we single out a frequency-based eager learning approach and a similarity-based lazy learning approach to learn instance weights. Thus two different versions are created, which we denote as AIWNB$^{\mathrm{E}}$ and AIWNB$^{\mathrm{L}}$, respectively.
4) We conduct two groups of extensive experiments to validate the effectiveness of AIWNB$^{\mathrm{E}}$ and AIWNB$^{\mathrm{L}}$, respectively. Furthermore, we make a thorough discussion to give some instructive suggestions on where and when do our AIWNB algorithms perform better than their competitors.

The remainder of the paper is organized as follows. Section 2 provides a comprehensive survey of the existing improved approaches for NB. Section 3 proposes our AIWNB approach. Next, Section 4 presents the experimental setup and results. Finally, Section 5 concludes the study and outlines the main directions for future work.

2. Related work

To alleviate the primary weakness in NB, appropriate structures are needed to mitigate the conditional independence assumption. Unfortunately, learning an optimal Bayesian network has been verified a typical NP-hard problem [3]. Thus, to learn Bayesian networks easily, numerous enhancements to NB have been proposed as shown in Fig. 1. We divide them into six categories and make a comprehensive survey of them in this section.

2.1. Structure extension

Due to the limitation of the conditional independence assumption in NB, extending the structure of NB is a direct method to expect better performance of the final model. In this method, some directed arcs can be added to explicitly represent the dependencies among attributes. Different from NB, this method uses Eq. 4 to classify the test instance $\mathbf{x}$. For simplicity, we assume that each attribute node can have at most another one attribute parent node.

$$c(\mathbf{x}) = \arg\max_{c\in C} P(c)\prod_{j=1}^{m} P\left(a_j|a_{jp},c\right), \tag{4}$$

where a_{jp} is the attribute value of A_{jp}, which is the attribute parent of A_j. The prior probability of $P(c)$ is estimated by Eq. 2 as well, but the conditional probability of $P(a_j|a_{jp}, c)$ is estimated by the following Eq. 5.

$$P(a_j|a_{jp},c) = \frac{\sum_{i=1}^{n}\delta(a_{ij},a_j)\delta(a_{ijp},a_{jp})\delta(c_i,c)+\frac{1}{n_j}}{\sum_{i=1}^{n}\delta(a_{ijp},a_{jp})\delta(c_i,c)+1}, \tag{5}$$

where a_{ijp} is the attribute value of A_{jp} of the ith training instance.

To find each attribute's parent node, Friedman et al. [9] singled out tree-augmented naive Bayes (TAN). They argue that the attribute dependencies can be modeled as a tree-like structure. The same as NB, the class node directly points to all attributes nodes. However, unlike NB, each attribute has one parent node from another attribute. In TAN, the parent node of each attribute is assigned according to the conditional mutual information between each pair of attributes.

In order to avoid structure learning, Webb et al. [30] proposed an improved model called averaged one-dependence estimators (AODE). In AODE, each attribute is in turn regarded as the parent node of all other attributes. The AODE model weakens the conditional independence assumption by averaging all of the qualified one-dependence classifiers and thus it improves the classification accuracy but without incurring the high computational complexity.

In addition to AODE, Jiang et al. [19] proposed another novel improved NB model called hidden naive Bayes (HNB). In HNB, a hidden parent is created for each attribute which combines the influences from all other attributes. The experimental results demonstrate that HNB is an expressive model because it can represent the influences of all attributes.

In a word, extending the structure of NB can, to some extent, mitigate the conditional independence assumption [32], but it is a rather difficult problem to obtain a suitable structure of extended NB. Also, the structure extension method is computationally intensive.

2.2. Attribute selection

The attribute selection method is the process of identifying and removing some irrelevant or redundant attributes [10]. This method also utilizes Eqs. 2 and 3 to calculate the prior and conditional probabilities. However, different from the standard NB, this method only adopts an attribute subset of the most informative attributes when making a prediction by Eq. 6.

$$c(\mathbf{x}) = \arg\max_{c\in C} P(c)\prod_{j=1}^{s} P\left(a_j|c\right), \tag{6}$$

where s is the number of selected attributes.

To find an optimal attribute subset from the original attribute set, Langley & Sage [24] proposed an improved model called selective Bayesian classifiers (SBC). It selects an attribute subset from the whole space of attributes via a forward greedy search method. More specifically, it employs the classification accuracy of NB to evaluate alternative attribute subsets, and considers adding each unselected attribute which can improve the classification accuracy of NB at most in each iteration.

However, SBC always chooses a so-called best attribute into the selected attribute subset in each iteration. Jiang et al. [15] argued that the search strategy used in SBC is so greedy that it often falls into a local optimization. So they presented an enhanced attribute selection model by carrying a random search through the whole space of attributes called randomly selected naive Bayes (RSNB). RSNB has some randomness and can avoid local optimization in SBC to some extent.

Recently, Chen et al. [2] presented an efficient attribute selection model called selective naive Bayes (SNB). In SNB, attribute selection models are built in a way that each one is a trivial extension of another. Specifically, each attribute would be added into the candidate model in the descending order according to their mutual information with the class. Then, the most predictive SNB model can be selected by the measures of incremental leave-one-out cross validation. Experimental results show that SNB has a higher classification accuracy, yet maintains the simplicity and efficiency.

2.3. Attribute weighting

Distinguished from attribute selection which completely removes the most irrelevant or redundant attributes from the attribute space, attribute weighting [25,26,35] is more flexible which assigns a continuous weight to each attribute between 0 and 1. In this method, the prior and conditional probabilities are also calculated by Eqs. 2 and 3, but the weight of each attribute is incorporated into the naive Bayesian classification formula as shown in Eq. 7.

$$c(\mathbf{x}) = \arg\max_{c\in C} P(c)\prod_{j=1}^{m} P\left(a_j|c\right)^{w_j^{att}}, \tag{7}$$

where w_j^{att} represents the weight of the jth attribute.

To define the weight of each attribute, Hall [11] proposed a decision tree-based attribute weighted naive Bayes (DTAWNB) model, which calculates the dependencies between attributes through an unpruned decision tree from the training instances with randomly sample. In DTAWNB, the weight for an attribute is inversely proportional to the minimum depth of the decision tree. Then a bagging procedure is employed to stabilize the estimated weights across the ensemble.

Besides, Zaidi et al. [36] proposed an attribute weighted NB model named weighting attributes to alleviate NB' independence assumption (WANBIA) to optimize attribute weights using gradient descent searches, by either maximizing the conditional log-likelihood or minimizing the mean squared error. This wrapper can generally achieve a better classification accuracy, since the weights are determined based on the performance feedback from the classifier itself.

Recently, Jiang et al. [20] proposed another improved model called correlation-based attribute weighted naive Bayes (CAWNB). In CAWNB, the weight of each attribute is firstly defined as the difference between the mutual relevance (correlation between attribute and class) and the average mutual redundancy (redundancy among attributes). Secondly, a sigmoid transformation must be performed to ensure that the weight is within a realistic range. The classification accuracy of CAWNB is higher than NB, meanwhile it maintains the simplicity of the final model.

2.4. Instance weighting

Distinguished from attribute weighting, instance weighting assigns different weights to different instances. Instance weighted NB also utilizes Eq. 1 to predict its class label, but the prior and conditional probabilities are estimated using Eqs. 8 and 9, respectively.

$$P(c) = \frac{\sum_{i=1}^{n} w_i^{ins}\delta(c_i, c) + \frac{1}{q}}{\sum_{i=1}^{n} w_i^{ins} + 1}, \tag{8}$$

$$P(a_j|c) = \frac{\sum_{i=1}^{n} w_i^{ins}\delta(a_{ij}, a_j)\delta(c_i, c) + \frac{1}{n_j}}{\sum_{i=1}^{n} w_i^{ins}\delta(c_i, c) + 1}, \tag{9}$$

where w_i^{ins} is the weight of the ith training instance.

Instance weighting can be divided into two subcategories: eager learning and lazy learning, depending on when the main computational cost occurs. Eager learning spends the main computational costs at training phase, whilst lazy learning spends the main computational costs at classification phase.

For eager learning, Jiang et al. [18] presented an instance weighting model called discriminatively weighted naive Bayes (DWNB). In each iteration, training instances are discriminatively increased different weights associated with the estimated conditional probability loss, and then a new instance weighted NB classifier is learned from the re-weighted training instances in the next iteration. The number of iterations is usually set to 15 to guarantee its effectiveness.

Although DWNB is an eager learning approach, it also needs some iterations to get the final model. To obtain an instance weighting model more efficiently, Xu et al. [34] proposed a simple, efficient, and effective instance weighting model called attribute value frequency-based instance weighted naive Bayes (AVFWNB). In AVFWNB, the weight of each training instance is defined as the scalar product of its attribute value frequency vector and the attribute value number vector. AVFWNB significantly outperforms the standard NB in classification accuracy, meanwhile maintains the simplicity of the final model.

For lazy learning, Jiang & Guo [16] presented a lazy naive Bayes (LNB) model, in which each training instance is weighted (cloned) according to the similarity between the test instance and each training instance. Specifically, each training instance is cloned a different number of times according to its similarity to the test instance. Finally, a NB classifier is built using the expanded instances as the training data. The cloning techniques improved both the ranking performance and the classification performance of NB.

2.5. Instance selection

The main idea of instance selection is based on building a NB model on a subset of the training instances instead of on the whole training instances. Though the conditional independence assumption hardly holds true in the whole space of the training instances, a local NB model is fit to a subset of data that is in the neighbourhood of test instance. Therefore, the local NB model which built on a small neighbor dataset might perform better. Instance selection can be regarded as a special case of instance weighting. This method also utilizes Eq. 1 to predict its class label, but the prior and conditional probabilities are estimated using Eqs. 10 and 11, respectively.

$$P(c) = \frac{\sum_{i=1}^{l} \delta(c_i, c) + \frac{1}{q}}{\sum_{i=1}^{l} l + 1}, \tag{10}$$

$$P(a_j|c) = \frac{\sum_{i=1}^{l} \delta(a_{ij}, a_j)\delta(c_i, c) + \frac{1}{n_j}}{\sum_{i=1}^{l} \delta(c_i, c) + 1}, \tag{11}$$

where l is the number of selected training instances.

Instance selection can also be divided into eager learning and lazy learning. For eager learning, Kohavi [22] proposed a hybrid model of decision tree and NB classifier called NBTree. Learning a NBTree model is similar to C4.5 [28] except its score function for evaluating splitting attributes. After a tree is grown, a local NB model is built based on each leaf of the grown tree. A new test instance can be classified by the local NB based on the corresponding leaf of decision tree. This approach retains the interpretability of the model and extensive experimental results have verified that NBTree significantly outperforms NB and C4.5.

Inspired by the good performance of NBTree, Wang et al. [29] proposed an improved model called multinomial naive Bayes tree (MNBTree) by employing a MNB classifier on each leaf node of the built decision tree. Unlike NBTree, MNBTree builds a binary tree, in which the split attributesvalues are just divided into zero and nonzero. Moreover, MNBTree utilizes the information gain measure instead of the classification accuracy measure to build the binary tree for reducing the time complexity.

For lazy learning, the k-nearest neightbor (KNN) algorithm is one of the most well-recognized for local learning. The idea of combining KNN with NB is quite straightforward. Frank et al. [8] proposed a hybrid model called locally weighted naive Bayes (LWNB), which is another instance selection approach for learning a local NB model. LWNB constructs a unique NB according to the k-nearest neighbors of the test instance. The distance between the test instance and its neighbors can be calculated using various difference metrics. LWNB enhances NB in classification accuracy, and it is not highly sensitive to the value of k unless k is excessively small.

2.6. Fine tuning

For NB, an accurate estimation of needed probabilities terms is essential to get the high classification accuracy. Thus, this category method is to manipulate probabilities to find a good estimation of needed probabilities terms. In this method, the standard NB model is built in the first phase, thus Eq. 1 is also utilized to predict its class label, and the prior and conditional probabilities are also estimated using Eqs. 2 and 3, respectively. But different from NB, in the fine tuning stage, each training instance is classified, if a training instance is mistakenly classified, the probability terms $P(a_j|c_{actu})$ and $P(a_j|c_{pred})$ will be adjusted by Eqs. 12 and 13.

$$P_{t+1}\left(a_j|c_{actu}\right) = P_t\left(a_j|c_{actu}\right) + \delta_{t+1}\left(a_j, c_{actu}\right), \tag{12}$$

$$P_{t+1}\left(a_j|c_{pred}\right) = P_t\left(a_j|c_{pred}\right) - \delta_{t+1}\left(a_j, c_{pred}\right), \tag{13}$$

where t is the iterative number, c_{actu} and c_{pred} are the actual class and the predicted class of the misclassified training instance, respectively. $\delta_{t+1}\left(a_j, c_{actu}\right)$ and $\delta_{t+1}\left(a_j, c_{pred}\right)$ are the update step size of each iteration.

To determine better estimations for the probability terms, Hindi [13] at first proposed the fine tuned naive Bayes (FTNB) model. In FTNB, a standard NB model is at first built using the training instances. If a training instance is mistakenly classified by this model, that means the instance's predicted class has a higher computed probability than its actual class. Then in the fine tuning stage, given the instance's attribute values, the value of $P(a_j|c_{actu})$ will be increased, and the value of $P(a_j|c_{pred})$ will be decreased. The update step size $\delta_{t+1}\left(a_j, c_{actu}\right)$ will be proportional to the difference between $P(a_j|c_{actu})$ and the probability of the attribute value with the maximum probability given c_{actu}. Similarly, the update step size $\delta_{t+1}\left(a_j, c_{pred}\right)$ will be proportional to the difference between the probability of the attribute value with the minimum probability given c_{pred}. The fine tuning process is repeated as long as the classification accuracy of training instances continues to increase. FTNB improves the classification accuracy compared with NB, especially in the problem of lacking of training instances.

To further improve FTNB, Diab & Hindi [5] transforms the probability estimation problem into an optimization problem. They use three different metaheuristic methods to solve this problem, namely, differential evolution (DE), genetic algorithms (GA) and simulated annealing (SA). Besides, they also proposed a novel DE approach that used multi-parent mutation and crossover operations (MPDE). Experimental results demonstrate that the approaches are widely useful in different domains, especially when the training instances are insufficient to reflect the underlying distribution of the instances. They also indicate that the proposed MPDE approach is more convenient for fine tuning NB than other approaches.

Recently, Hindi et al. [14] proposed an ensemble of FTNB classifiers using the bagging approach for text classification. They modified the update equations in FTNB to make the approach unstable enough to produce diverse classifiers. Then an ensemble of FTNB classifiers is built to expect a remarkable improvement in classification accuracy. The empirical experiments presented significant improvement in 16 benchmark text classification datasets.

3. Attribute and instance weighted naive Bayes (AIWNB)

3.1. General framework of AIWNB

Just as we discussed above, both attribute weighting and instance weighting can alleviate the conditional independence assumption in NB. Attribute weighting incorporates the attribute weights into the naive Bayesian classification formula to discriminatively assign a different weight to each attribute, but regards each instance as the same importance. Instance weighting incorporates the instance weights to estimate the prior and conditional probabilities more accurately, but regards each attribute as the same importance.

Nonetheless, essentially all the existing weighting approaches are either attribute weighting or instance weighting. These two kinds of weighting approaches are used separately rather than utilizing both the predictive capabilities of attributes and instances. But we argue that different attributes should have different importance, meanwhile different instances should have different reliability. Motivated by the effectiveness of attribute weighting and instance weighting, we expect that if we employ an attribute weighting approach to discriminatively assign different attributes with different weights, and then employ an instance weighting approach to estimate the prior and conditional probabilities more accurately, we can inherit both the advantage of attribute weighting and the advantage of instance weighting simultaneously, and thus obtain better performance than using either single attribute weighting or single instance weighting. We call this improved model as attribute and instance weighted naive Bayes (AIWNB).

In AIWNB, the weight of each attribute w_j^{att} $(j = 1, 2, \cdots, m)$ is incorporated into the naive Bayesian classification formula, and the weight of each instance w_i^{ins} $(i = 1, 2, \cdots, n)$ is used to estimate the prior and conditional probabilities. Now, given a test instance $\mathbf{x}$, AIWNB utilizes Eq. 7 to classify it, and yet at the same time utilizes Eqs. 8 and 9, respectively, to estimate the prior probability $P(c)$ and the conditional probability $P(a_j|c)$.

The general framework of AIWNB is described in Algorithm 1. According to Algorithm 1, we can see that how to learn the attribute weight vector $\mathbf{w^{att}}$ and the instance weight vector $\mathbf{w^{ins}}$ are two crucial problems. To learn $\mathbf{w^{att}}$, we single out a correlation-based attribute weighting approach in Subsection 3.2. To learn $\mathbf{w^{ins}}$, we single out an eager approach and a lazy approach in Subsection 3.3.

Algorithm 1 AIWNB(T, $\mathbf{x}$).

Input: T-a training dataset, $\mathbf{x}$-a test instance
Output: the class label $c(\mathbf{x})$ of $\mathbf{x}$

1: Employ an attribute weighting approach to learn the weightof each attribute w_j^{att} $(j = 1, 2, \cdots, m)$
2: Employ an instance weighting approach to learn the weightof each instance w_i^{ins} $(i = 1, 2, \cdots, n)$
3: Estimate the prior probability $P(c)$ by Eq. 8
4: Estimate the conditional probability $P(a_j|c)$ by Eq. 9
5: Predict the class label $c(\mathbf{x})$ of $\mathbf{x}$ by Eq. 7
6: **return** $c(\mathbf{x})$

3.2. The weight of each attribute

Attribute weighting assigns a different weight to each attribute. Recently, Jiang et al. [20] proposed a correlation-based attribute weighting approach. They argued that strongly predictive attributes should be closely related to the class label, yet uncorrelated with other attributes. In this approach, the mutual information is used to measure the normalized value reflecting the attribute-class relevance and the average attribute-attribute redundancy. The weight of each attribute is finally calculated by a sigmoid transformation of the difference between the attribute-class relevance and the average attribute-attribute redundancy. Motivated by this work, in this study, we adopt this approach to calculate the weight of each attribute. The detailed processes are described as follows.

In the first step, the mutual information is used to measure the correlation between each pair of random discrete variables. The attribute-class relevance and the attribute-attribute inter-correlation are respectively defined as:

$$I(A_j; C) = \sum_{a_j} \sum_{c} P(a_j, c) \log \frac{P(a_j, c)}{P(a_j)P(c)}, \tag{14}$$

$$I(A_j; A_k) = \sum_{a_j} \sum_{a_k} P(a_j, a_k) \log \frac{P(a_j, a_k)}{P(a_j)P(a_k)}, \tag{15}$$

where C is the collection of all possible class labels c, a_j and a_k are the value for the attribute A_j and A_k, respectively. $I(A_j; C)$ is the attribute-class relevance between the attribute A_j and the class C. $I(A_j; A_k)$ is the attribute-attribute inter-correlation between two different attributes A_j and A_k.

In the second step, to guarantee the results realistic, $I(A_j; C)$ is normalized into $NI(A_j; C)$ and $I(A_j; A_k)$ is normalized into $NI(A_j; A_k)$ by Eqs. 16 and 17, respectively.

$$NI(A_j; C) = \frac{I(A_j; C)}{\frac{1}{m}\sum_{j=1}^{m} I(A_j; C)}, \tag{16}$$

$$NI(A_j; A_k) = \frac{I(A_j; A_k)}{\frac{1}{m(m-1)}\sum_{j=1}^{m}\sum_{k=1\wedge k\neq j}^{m} I(A_j; A_k)}, \tag{17}$$

where $NI(A_j; C)$ and $NI(A_j; A_k)$ are the normalized values representing the mutual relevance and the mutual redundancy, respectively.

In the third step, the attribute weight is defined as the difference between the mutual relevance and the average mutual redundancy. The detailed formula is defined by Eq. 18.

$$\Delta w_j^{att} = \underbrace{NI(A_j; C)}_{\text{relevance}} - \underbrace{\frac{1}{m-1}\sum_{k=1\wedge k\neq j}^{m} NI(A_j; A_k)}_{\text{average redundancy}}, \tag{18}$$

where Δw_j^{att} is the weight of the jth attribute A_j.

The weight Δw_j^{att} calculated by Eq. 18 may be negative, but an attribute weight should be in the range of [0, 1]. Therefore, a standard logistic sigmoid function is used to transform Δw_j^{att} into the range of [0, 1] by Eq. 19.

$$w_j^{att} = \frac{1}{1 + e^{-\Delta w_j^{att}}}. \tag{19}$$

Now, each attribute is assigned a discriminative weight. In the following subsection, we will introduce how to learn a discriminative weight for each instance.

3.3. The weight of each instance

To learn instance weights, we single out an eager approach and a lazy approach, thus two different versions are proposed, which are denoted as AIWNB^{E} and AIWNB^{L}, respectively. Next, we will introduce the detailed processes of our proposed AIWNB^{E} and AIWNB^{L} approaches, respectively.

3.3.1. AIWNB^{E} approach

To maintain the computational simplicity that characterizes NB, we adopt a simple, efficient and effective approach called attribute value frequency-based instance weighting filter to learn the weight of each instance. This approach is motivated by such observations: 1) The frequency of each attribute value contains some important information that can be utilized to define training instances' weights. 2) The weight of each training instance has positive correlation with its attribute value frequency vector and the attribute value number vector of the whole training dataset.

At first, we focus on the frequency of each attribute value, and then employ Eq. 20 to calculate the frequency of each attribute value:

$$f_{ij} = \frac{\sum_{r=1}^{n} \delta\left(a_{rj}, a_{ij}\right)}{n}, \tag{20}$$

where f_{ij} is the frequency of a_{ij} (the jth attribute value of the ith training instance), n is the number of training instances, a_{rj} is the jth attribute value of the rth training instance.

Then, let n_j be the number of values of the jth attribute, and the attribute value number vector can be represented by $\langle n_1, n_2, \cdots, n_m\rangle$. Then, the weight of the ith training instance is defined as the inner product (scalar product) of its attribute value frequency vector and its attribute value number vector. The detailed definition is:

$$\begin{aligned} w_i^{ins} &= < f_{i1}, f_{i2}, \cdots, f_{im} > \bullet < n_1, n_2, \cdots, n_m > \\ &= \sum_{j=1}^{m} (f_{ij} * n_j). \end{aligned} \tag{21}$$

Now, the entire learning algorithm for our AIWNB^{E} can be partitioned into training (AIWNB^{E}-Training) and classification (AIWNB^{E}-Classification) algorithms. They are briefly depicted in Algorithms 2 and 3. AIWNB^{E} is an eager learning approach, thus spends the main computational costs during training phase. The attribute weight vector $\mathbf{w^{att}}$ and the instance weight vector $\mathbf{w^{ins}}$ are both obtained during training phase. In classification phase, we utilize Eqs. 7–9 to predict the class label of the test instance $\mathbf{x}$.

According to Algorithm 2, the execution time for getting $\mathbf{w^{att}}$ is proportional to $mqv + m^2v^2 + m + m^2 + m$, where v is the average number of values for an attribute. The execution time for getting $\mathbf{w^{ins}}$ is proportional to $2nm$. Therefore, the whole time complexity of getting $\mathbf{w^{att}}$ and $\mathbf{w^{ins}}$ is $O(mqv + m^2v^2 + m + m^2 + m + 2nm)$. If we only take the highest order term, the whole time complexity is only $O(m^2v^2)$. Besides, the auxiliary space for getting $\mathbf{w^{att}}$ is proportional to $m^2v^2 + mv + m^2 + m$. The auxiliary space for getting $\mathbf{w^{ins}}$ is proportional to $mv + m$. Therefore, the whole space complexity of getting $\mathbf{w^{att}}$ and $\mathbf{w^{ins}}$ is $O(m^2v^2 + 2mv + m^2 + 2m)$. If we only take the highest order term, the whole space complexity is only $O(m^2v^2)$. In a word, this approach is quite simple, efficient, and effective.

3.3.2. AIWNB^{L} approach

Generally speaking, eager learning can receive a lower time complexity than lazy learning, but lazy learning can receive bigger boost in terms of classification accuracy. Now, a lazy learning approach is also proposed to learn the weight of each instance as follows.

Algorithm 2 AIWNB^{E}-Training(T).

Input: T-a training dataset
Output: $\mathbf{w^{att}}$-the attribute weight vector,$\mathbf{w^{ins}}$-the instance weight vector

for each attribute $A_j (j = 1, 2, \cdots, m)$ **do**
  Calculate $I(A_j; C)$ by Eq. 14
end for
for each pair of attributes A_j and $A_k (k \neq j)$ **do**
  Calculate $I(A_j; A_k)$ by Eq. 15
end for
for each attribute $A_j (j = 1, 2, \cdots, m)$ **do**
  Calculate $NI(A_j; C)$ by Eq. 16
end for
for each pair of attributes A_j and $A_k (k \neq j)$ **do**
  Calculate $NI(A_j; A_k)$ by Eq. 17
end for
for each attribute $A_j (j = 1, 2, \cdots, m)$ **do**
  Calculate Δw_j^{att} by Eq. 18
  Calculate w_j^{att} by Eq. 19
end for
for each training instance $\mathbf{y_i}$ $(i = 1, 2, \cdots, n)$ from T **do**
  for each attribute value a_{ij} $(j = 1, 2, \cdots, m)$ from $\mathbf{y_i}$ **do**
    Calculate f_{ij} using Eq. 20
  end for
  Calculate w_i^{ins} by Eq. 21
end for
return $\mathbf{w^{att}}$ and $\mathbf{w^{ins}}$

Algorithm 3 AIWNB^{E}-Classification($\mathbf{w^{att}}$, $\mathbf{w^{ins}}$, $\mathbf{x}$).

Input: $\mathbf{w^{att}}$-the attribute weight vector, $\mathbf{w^{ins}}$-the instance weight vector, $\mathbf{x}$-a test instance
Output: $c(\mathbf{x})$-the predicted class label of $\mathbf{x}$

Estimate the prior probability $P(c)$ by Eq. 8
Estimate the conditional probability $P(a_j|c)$ by Eq. 9
Predict the class label $c(\mathbf{x})$ of $\mathbf{x}$ by Eq. 7
return $c(\mathbf{x})$

At first, let us define a very important concept of **similarity** between two instances with nominal attributes. Let $\mathbf{x}$ be a test instance and $\mathbf{y_i}$ be the ith training instance, the similarity, denoted by $s(\mathbf{x}, \mathbf{y_i})$, between them is defined as:

$$s(\mathbf{x}, \mathbf{y_i}) = \sum_{j=1}^{m} \delta\left(a_j(\mathbf{x}), a_j(\mathbf{y_i})\right), \tag{22}$$

where

$$\delta\left(a_j(\mathbf{x}), a_j(\mathbf{y_i})\right) = \begin{cases} 1 & a_j(\mathbf{x}) = a_j(\mathbf{y_i}) \\ 0 & a_j(\mathbf{x}) \neq a_j(\mathbf{y_i}) \end{cases} \tag{23}$$

It can be seen that $s(\mathbf{x}, \mathbf{y_i})$ is a function that simply counts the number of identical attributes of $\mathbf{x}$ and $\mathbf{y_i}$. Roughly speaking, this function measures the extent of similarity between these two instances. Then, the weight of the ith training instance is defined as:

$$w_i^{ins} = 1 + s(\mathbf{x}, \mathbf{y_i}). \tag{24}$$

Now, the entire learning algorithm for our AIWNB^{L} can also be partitioned into training (AIWNB^{L}-Training) and classification (AIWNB^{L}-Classification) algorithms. They are briefly depicted in Algorithm 4 and Algorithm 5. Unlike AIWNB^{E}, AIWNB^{L} is a lazy learning approach, thus spends the main computational costs during classification phase. For each test instance, AIWNB^{L} will build a special model. Thus, the instance weight vector $\mathbf{w^{ins}}$ is obtained

Algorithm 4 AIWNB^{L}-Training(T).

Input: T-a training dataset
Output: $\mathbf{w}^{\mathbf{att}}$-the attribute weight vector
for each attribute $A_j (j = 1, 2, \cdots, m)$ **do**
  Calculate $I(A_j; C)$ by Eq. 14
end for
for each pair of attributes A_j and $A_k (k \neq j)$ **do**
  Calculate $I(A_j; A_k)$ by Eq. 15
end for
for each attribute $A_j (j = 1, 2, \cdots, m)$ **do**
  Calculate $NI(A_j; C)$ by Eq. 16
end for
for each pair of attributes A_j and $A_k (k \neq j)$ **do**
  Calculate $NI(A_j; A_k)$ by Eq. 17
end for
for each attribute $A_j (j = 1, 2, \cdots, m)$ **do**
  Calculate Δw_j^{att} by Eq. 18
  Calculate w_j^{att} by Eq. 19
end for
return $\mathbf{w}^{\mathbf{att}}$

Algorithm 5 AIWNB^{L}-Classification($\mathbf{w}^{\mathbf{att}}$, $\mathbf{x}$).

Input: $\mathbf{w}^{\mathbf{att}}$-the attribute weight vector,$\mathbf{x}$-a test instance
Output: $c(\mathbf{x})$-the predicted class label of $\mathbf{x}$
for each training instance $\mathbf{y_i} (i = 1, 2, \cdots, n)$ from T **do**
  Calculate $s(\mathbf{x}, \mathbf{y_i})$ by Eq. 22
  Calculate w_i^{ins} by Eq. 24
end for
Estimate the prior probability $P(c)$ by Eq. 8
Estimate the conditional probability $P(a_j|c)$ by Eq. 9
Predict the class label $c(\mathbf{x})$ of $\mathbf{x}$ by Eq. 7
return $c(\mathbf{x})$

during classification phase, and then we use Eqs. 7–9 to predict the class label of the test instance $\mathbf{x}$.

According to Algorithm 4, the execution time for getting $\mathbf{w}^{\mathbf{att}}$ is also proportional to $mqv + m^2v^2 + m + m^2 + m$, where v is the average number of values for an attribute. According to Algorithm 5, the execution time for getting $\mathbf{w}^{\mathbf{ins}}$ is proportional to nm. It is obvious that, compared to AIWNB^{E}, AIWNB^{L} spends more time in classification phase, but could also obtain better classification performance. Besides, the auxiliary space for getting $\mathbf{w}^{\mathbf{att}}$ is proportional to $m^2v^2 + mv + m^2 + m$. For getting $\mathbf{w}^{\mathbf{ins}}$, we only need a constant auxiliary space, thus the space complexity is $O(1)$. Therefore, the whole space complexity of getting $\mathbf{w}^{\mathbf{att}}$ and $\mathbf{w}^{\mathbf{ins}}$ is $O(m^2v^2 + mv + m^2 + m)$. If we only take the highest order term, the whole space complexity is only $O(m^2v^2)$, which is the same as AIWNB^{E}.

4. Experiments and results

4.1. Experimental setting and benchmark data

The purpose of this section is to investigate the classification performance of our proposed AIWNB^{E} and AIWNB^{L}. Therefore, we conduct two groups of experiments to compare our proposed AIWNB^{E} and AIWNB^{L} with the standard NB and some other state-of-the-art competitors. In the first group, we compare AIWNB^{E} with CAWNB, AVFWNB, and NB. In the second group, we compare AIWNB^{L} with CAWNB, LNB, and NB. Now, we introduce these competitors and their abbreviations as follows.

- CAWNB: NB with the correlation-based attribute weighting [20].
- AVFWNB: NB with the eager attribute value frequency-based instance weighting [34].
- LNB: NB with the lazy similarity-based instance weighting [16].
- NB: the standard naive Bayes [23].

Table 1
Descriptions of 36 UCI datasets used in the experiments.

Dataset name	Instance number	Attribute number	Nominal attributes (%)	Class number	Missing values
anneal	898	38	84.21	5	N
anneal.ORIG	898	38	84.21	6	Y
audiology	226	69	100.00	24	Y
autos	205	25	40.00	7	Y
balance-scale	625	4	0.00	3	N
breast-cancer	286	9	100.00	2	Y
breast-w	699	9	0.00	2	Y
colic	368	22	68.18	2	Y
colic.ORIG	368	27	74.07	2	Y
credit-a	690	15	60.00	2	Y
credit-g	1000	20	65.00	2	N
diabetes	768	8	0.00	2	N
glass	214	9	0.00	7	N
heart-c	303	13	53.85	5	Y
heart-h	294	13	53.85	5	Y
heart-statlog	270	13	0.00	2	N
hepatitis	155	19	68.42	2	Y
hypothyroid	3772	29	20.69	4	Y
ionosphere	351	34	0.00	2	N
iris	150	4	0.00	3	N
kr-vs-kp	3196	36	100.00	2	N
labor	57	16	50.00	2	Y
letter	20000	16	0.00	26	N
lymphography	148	18	83.33	4	N
mushroom	8124	22	100.00	2	Y
primary-tumor	339	17	100.00	21	Y
segment	2310	19	0.00	7	N
sick	3772	29	75.86	2	Y
sonar	208	60	0.00	2	N
soybean	683	35	100.00	19	Y
splice	3190	61	100.00	3	N
vehicle	846	18	0.00	4	N
vote	435	16	100.00	2	Y
vowel	990	13	23.08	11	N
waveform-5000	5000	40	0.00	3	N
zoo	101	17	94.12	7	N

Our experiments were conducted on a collection of 36 benchmark classification datasets from the University of California at Irvine (UCI) repository [6], which represent a wide range of domains and data characteristics listed in Table 1. Please note that, the nominal attributes (%) column shows the percentage of nominal attributes in a dataset (number of nominal attributes ÷ number of all attributes × 100%). We downloaded these datasets in the format of *arff* from the main website[1] of the Waikato Environment for Knowledge Analysis (WEKA) platform [31]. In our experiments, missing attribute values were replaced with the modes of nominal attribute values and means of numerical attribute values from the available data. As all algorithms could only deal with discrete value data, numerical attribute values were discretized by the Fayyad & Irani's MDL method [7]. Besides, we manually deleted three useless attributes in advance: "Hospital Number" in "colic.ORIG", "instance name" in "splice" and "animal" in "zoo". Apparently, if an attribute varies so much that the number of its attribute values is just (most) equal to the number of instances in a dataset, it is a useless attribute.

Table 2
Classification accuracy comparisons for AIWNB^E versus CAWNB, AVFWNB, and NB.

Dataset	AIWNB^E	CAWNB	AVFWNB	NB
anneal	**98.94 ± 1.05**	98.50 ± 1.29	98.62 ± 1.15	96.36 ± 1.97 •
anneal.ORIG	**95.06 ± 2.23**	94.60 ± 2.48	93.32 ± 2.65	92.71 ± 2.70 •
audiology	**83.93 ± 7.00**	74.22 ± 6.36 •	78.58 ± 8.44 •	75.74 ± 6.58 •
autos	**78.04 ± 9.02**	77.95 ± 8.95	77.27 ± 9.43	77.02 ± 9.69
balance-scale	73.75 ± 4.22	**73.76 ± 4.15**	71.10 ± 4.30	71.08 ± 4.29
breast-cancer	71.90 ± 7.55	**72.46 ± 7.25**	71.41 ± 7.98	72.32 ± 7.91
breast-w	97.17 ± 1.68	97.14 ± 1.81	**97.48 ± 1.68**	97.25 ± 1.79
colic	**83.45 ± 5.45**	83.34 ± 5.62	81.47 ± 5.86	81.20 ± 5.80
colic.ORIG	**73.87 ± 6.40**	73.70 ± 6.46	72.91 ± 6.34	73.43 ± 6.27
credit-a	**87.03 ± 3.83**	86.99 ± 3.81	86.23 ± 3.85	86.17 ± 3.94
credit-g	**75.81 ± 3.60**	75.70 ± 3.53	75.38 ± 3.90	75.40 ± 4.01
diabetes	77.87 ± 4.86	**78.01 ± 4.89**	77.89 ± 4.66	77.88 ± 4.65
glass	74.02 ± 8.41	73.37 ± 8.38	**76.25 ± 8.07**	74.20 ± 8.11
heart-c	82.71 ± 6.61	82.94 ± 6.57	83.04 ± 6.68	**83.73 ± 6.46**
heart-h	84.29 ± 5.85	83.82 ± 6.16	**84.90 ± 5.68**	84.43 ± 5.88
heart-statlog	83.22 ± 6.61	83.44 ± 6.69	**83.78 ± 6.29**	83.74 ± 6.25
hepatitis	85.75 ± 8.97	**85.95 ± 9.25**	85.38 ± 9.00	85.05 ± 9.45
hypothyroid	**99.07 ± 0.48**	98.56 ± 0.56 •	98.98 ± 0.48	98.74 ± 0.57 •
ionosphere	**92.40 ± 4.13**	91.82 ± 4.34	91.94 ± 4.09	91.37 ± 4.55
iris	**94.40 ± 5.50**	**94.40 ± 5.50**	**94.40 ± 5.50**	94.33 ± 5.56
kr-vs-kp	**93.73 ± 1.28**	93.58 ± 1.32	88.18 ± 1.86 •	87.81 ± 1.90 •
labor	**94.33 ± 9.30**	92.10 ± 10.94	**94.33 ± 10.13**	93.83 ± 10.41
letter	**75.56 ± 0.89**	75.22 ± 0.83 •	75.07 ± 0.84 •	74.67 ± 0.86 •
lymphography	84.68 ± 7.99	84.81 ± 8.13	85.49 ± 7.83	**85.70 ± 7.95**
mushroom	**99.53 ± 0.23**	99.19 ± 0.32 •	99.12 ± 0.31 •	98.03 ± 0.49 •
primary-tumor	**47.76 ± 5.25**	47.20 ± 5.27	45.85 ± 6.53	47.11 ± 5.65
segment	**94.16 ± 1.38**	93.47 ± 1.46 •	93.69 ± 1.41 •	92.91 ± 1.56 •
sick	97.33 ± 0.85	**97.36 ± 0.84**	97.02 ± 0.86 •	97.07 ± 0.84
sonar	82.23 ± 8.65	82.56 ± 8.25	84.49 ± 7.79	**84.96 ± 7.57**
soybean	**94.74 ± 2.19**	93.66 ± 2.73 •	94.52 ± 2.36	93.53 ± 2.79 •
splice	**96.21 ± 0.99**	96.19 ± 0.99	95.61 ± 1.11 •	95.58 ± 1.12 •
vehicle	**63.59 ± 3.92**	62.91 ± 3.88	63.36 ± 3.87	62.64 ± 3.84
vote	**92.18 ± 3.76**	92.11 ± 3.74	90.25 ± 3.95 •	90.30 ± 3.89 •
vowel	**69.98 ± 4.11**	68.84 ± 4.30 •	67.46 ± 4.62 •	66.00 ± 4.58 •
waveform-5000	82.98 ± 1.37	**83.11 ± 1.38**	80.65 ± 1.46 •	80.76 ± 1.49 •
zoo	**96.05 ± 5.60**	95.96 ± 5.61	**96.05 ± 5.60**	95.75 ± 5.68
Average	**84.94**	84.41	84.21	83.86
W/T/L	-	7/29/0	10/26/0	13/23/0

4.2. Experimental results and analysis

Tables 2 and 5 present the detailed results of the comparisons in terms of the classification accuracy. All of the classification accuracy estimates were obtained by averaging the results from 10 separate runs in a stratified ten-fold cross-validation. Runs with the various algorithms are carried out on the same training sets and evaluated on the same test sets. The highest classification accuracy on each dataset is highlighted in bold. The symbol • in these two tables denotes that our proposed AIWNB^E and AIWNB^L significantly outperform their competitors with a corrected paired two-tailed t-test at the $p = .05$ significance level [27]. Besides, the averages and the *Win*/*Tie*/*Lose* (W/T/L) values over 36 datasets are summarized at the bottom of the tables. Each average (arithmetic mean) across all datasets provides a gross indication of relative performance in addition to other statistics. Each entry's W/T/L in the tables implies that, compared to their competitors, our proposed AIWNB^E and AIWNB^L win on W datasets, tie on T datasets, and lose on L datasets.

Yet at the same time, based on the classification accuracy results presented in Tables 2 and 5, we applied the Knowledge Extraction based on Evolutionary Learning (KEEL) data-mining software tool [1] to complete the Wilcoxon signed-ranks test [4] for thoroughly comparing each pair of algorithms. The Wilcoxon signed-ranks test is a non-parametric statistical test, which ranks the differences in performances of two algorithms for each dataset, ignoring the signs, and compares the ranks for positive and negative differences.

Table 3
Ranks of the Wilcoxon test with regard to AIWNB^E.

Algorithm	AIWNB^E	CAWNB	AVFWNB	NB
AIWNB^E	-	493.5	488.0	555.5
CAWNB	136.5	-	371.0	477.5
AVFWNB	142.0	259.0	-	498.5
NB	110.5	188.5	167.5	-

Tables 3 and 6 present the detailed ranks computed by the Wilcoxon test. In Tables 3 and 6, each number above the diagonal is the sum of ranks for the datasets on which the algorithm in the row is better than the algorithm in the corresponding column (the sum of ranks for the positive differences, denoted by R^+), and each number below the diagonal is the sum of ranks for the datasets on which the algorithm in the column is worse than the algorithm in the corresponding row (the sum of ranks for the negative differences, denoted by R^-). According to the table of exact critical values for the Wilcoxon test, for a confidence level of $\alpha = 0.05$ ($\alpha = 0.1$) and $N = 36$ datasets, we speak of two classifiers as being "significantly different" if the smaller of R^+ and R^- is equal or less than 208 (227) and thus we reject the null-hypothesis. Tables 4 and 7 summarize the detailed comparison results. In Tables 4 and 7, ◦ indicates that the algorithm in the column improves the algorithm in the corresponding row, and • indicates that the algorithm in the row improves the algorithm in the corresponding column.

[1] https://waikato.github.io/weka-wiki/datasets/.

Table 4
Summary of the Wilcoxon test with regard to AIWNB^{E}.

Algorithm	AIWNB^{E}	CAWNB	AVFWNB	NB
AIWNB^{E}	-	•	•	•
CAWNB	○	-		•
AVFWNB	○		-	•
NB	○	○	○	-

Lower diagonal level of significance $\alpha = 0.05$; upper diagonal level of significance $\alpha = 0.1$.

From all of these comparison results, the conclusion is evident that our proposed AIWNB^{E} and AIWNB^{L} significantly outperform the standard NB and all the other competitors including CAWNB, AVFWNB, and LNB. For simplicity, we summarize the highlights as follows:

1) AIWNB^{E} has the highest classification accuracy on 23 datasets, which is considerably better than those of CAWNB (7 datasets), AVFWNB (7 datasets), and the standard NB (3 datasets).
2) AIWNB^{L} has the highest classification accuracy on 23 datasets, which is considerably better than those of CAWNB (5 datasets), LNB (8 datasets), and the standard NB (4 datasets).
3) The averaged classification accuracy of AIWNB^{E} on 36 datasets is 84.94%, which is considerably higher than those of CAWNB (84.41%), AVFWNB (84.21%), and the standard NB (83.86%). This suggests that our proposed AIWNB^{E} approach is effective.

Table 6
Ranks of the Wilcoxon test with regard to AIWNB^{L}.

Algorithm	AIWNB^{L}	CAWNB	LNB	NB
AIWNB^{L}	-	591.5	462.0	578.0
CAWNB	74.5	-	258.0	477.5
LNB	168.0	372.0	-	520.5
NB	88.0	188.5	109.5	-

Table 7
Summary of the Wilcoxon test with regard to AIWNB^{L}.

Algorithm	AIWNB^{L}	CAWNB	LNB	NB
AIWNB^{L}	-	•	•	•
CAWNB	○	-		•
LNB	○		-	•
NB	○	○	○	-

4) The averaged classification accuracy of AIWNB^{L} on 36 datasets is 85.52%, which is remarkably higher than those of CAWNB (84.41%), LNB (84.81%), and the standard NB (83.86%). This suggests that our proposed AIWNB^{L} approach is effective.
5) Based on the two-tailed t-tests, AIWNB^{E} is the best overall. AIWNB^{E} is better than CAWNB (7 wins and zero loss), AVFWNB (10 wins and zero loss), and NB (13 wins and zero loss).

Table 5
Classification accuracy comparisons for AIWNB^{L} versus CAWNB, LNB, and NB.

Dataset	AIWNB^{L}	CAWNB		LNB		NB	
anneal	**98.90 ± 1.10**	98.50 ± 1.29		98.62 ± 1.15		96.36 ± 1.97	•
anneal.ORIG	**95.06 ± 2.23**	94.60 ± 2.48		93.37 ± 2.65		92.71 ± 2.70	•
audiology	**84.81 ± 6.83**	74.22 ± 6.36	•	79.02 ± 8.36	•	75.74 ± 6.58	•
autos	**79.80 ± 8.63**	77.95 ± 8.95		78.14 ± 9.69		77.02 ± 9.69	
balance-scale	73.52 ± 4.39	**73.76 ± 4.15**		70.30 ± 4.30	•	71.08 ± 4.29	
breast-cancer	71.52 ± 7.23	**72.46 ± 7.25**		71.83 ± 7.59		72.32 ± 7.91	
breast-w	97.15 ± 1.77	97.14 ± 1.81		**97.28 ± 1.72**		97.25 ± 1.79	
colic	**83.40 ± 5.44**	83.34 ± 5.62		81.71 ± 5.86		81.20 ± 5.80	
colic.ORIG	**74.38 ± 6.70**	73.70 ± 6.46		73.24 ± 6.07		73.43 ± 6.27	
credit-a	86.93 ± 3.85	**86.99 ± 3.81**		86.42 ± 3.81		86.17 ± 3.94	
credit-g	**75.86 ± 3.67**	75.70 ± 3.53		75.65 ± 4.06		75.40 ± 4.01	
diabetes	**78.32 ± 4.67**	78.01 ± 4.89		77.98 ± 4.48		77.88 ± 4.65	
glass	74.90 ± 8.25	73.37 ± 8.38		**75.68 ± 8.12**		74.20 ± 8.11	
heart-c	82.81 ± 6.61	82.94 ± 6.57		83.04 ± 6.59		**83.73 ± 6.46**	
heart-h	84.26 ± 5.89	83.82 ± 6.16		**85.00 ± 5.68**		84.43 ± 5.88	
heart-statlog	83.19 ± 6.71	83.44 ± 6.69		**83.74 ± 6.34**		**83.74 ± 6.25**	
hepatitis	**86.00 ± 9.07**	85.95 ± 9.25		85.44 ± 9.09		85.05 ± 9.45	
hypothyroid	**99.05 ± 0.50**	98.56 ± 0.56	•	98.98 ± 0.47		98.74 ± 0.57	•
ionosphere	**92.68 ± 3.76**	91.82 ± 4.34		92.31 ± 4.00		91.37 ± 4.55	
iris	**94.40 ± 5.50**	**94.40 ± 5.50**		**94.40 ± 5.50**		94.33 ± 5.56	
kr-vs-kp	**94.06 ± 1.27**	93.58 ± 1.32	•	88.54 ± 1.93	•	87.81 ± 1.90	•
labor	93.80 ± 10.17	92.10 ± 10.94		**94.17 ± 10.45**		93.83 ± 10.41	
letter	79.60 ± 0.85	75.22 ± 0.83	•	**79.74 ± 0.85**		74.67 ± 0.86	•
lymphography	85.08 ± 7.72	84.81 ± 8.13		85.56 ± 7.92		**85.70 ± 7.95**	
mushroom	**99.71 ± 0.20**	99.19 ± 0.32	•	99.68 ± 0.18		98.03 ± 0.49	•
primary-tumor	**47.76 ± 5.21**	47.20 ± 5.27		46.44 ± 6.39		47.11 ± 5.65	
segment	**95.32 ± 1.32**	93.47 ± 1.46	•	94.80 ± 1.37	•	92.91 ± 1.56	•
sick	**97.36 ± 0.83**	**97.36 ± 0.84**		97.09 ± 0.85		97.07 ± 0.84	•
sonar	82.28 ± 8.57	82.56 ± 8.25		84.35 ± 7.83		**84.96 ± 7.57**	
soybean	**94.82 ± 2.24**	93.66 ± 2.73	•	94.60 ± 2.33		93.53 ± 2.79	•
splice	**96.55 ± 1.01**	96.19 ± 0.99	•	95.97 ± 1.09	•	95.58 ± 1.12	•
vehicle	**67.57 ± 3.27**	62.91 ± 3.88	•	67.55 ± 3.45		62.64 ± 3.84	•
vote	**93.68 ± 3.52**	92.11 ± 3.74	•	90.74 ± 3.88	•	90.30 ± 3.89	•
vowel	**74.48 ± 3.93**	68.84 ± 4.30	•	74.28 ± 4.30		66.00 ± 4.58	•
waveform-5000	**83.51 ± 1.38**	83.11 ± 1.38	•	81.43 ± 1.39	•	80.76 ± 1.49	•
zoo	96.05 ± 5.60	95.96 ± 5.61		**96.25 ± 5.57**		95.75 ± 5.68	
Average	**85.52**	84.41		84.81		83.86	
W/T/L	-	12/24/0		7/29/0		15/21/0	

Table 8
The percentages of the datasets on which our AIWNB algorithms and all of their competitors obtain the highest classification accuracy under each data circumstance.

Data Characteristics		AIWNB^E	Competitors	AIWNB^L	Competitors
Instance number	< 500	45.45	**54.55**	42.86	**57.14**
	≥ 500	**72.22**	27.78	**73.68**	26.32
Attribute number	< 20	41.67	**58.33**	34.78	**65.22**
	≥ 20	**81.25**	18.75	**88.89**	11.11
Nominal (%) attributes	< 50	47.06	**52.94**	**50.00**	**50.00**
	≥ 50	**65.22**	34.78	**63.64**	36.36

6) Based on the two-tailed t-tests, AIWNB^L is the best overall. AIWNB^L is better than CAWNB (12 wins and zero loss), LNB (7 wins and zero loss), and NB (15 wins and zero loss).
7) According to the Wilcoxon signed-ranks test results, we can draw the conclusion that our proposed AIWNB^E is markedly better than CAWNB (R^+=493.5 and R^-=136.5), AVFWNB (R^+=488.0 and R^-=142.0), and NB (R^+=555.5 and R^-=110.5).
8) According to the Wilcoxon signed-ranks test results, we can draw the conclusion that our proposed AIWNB^L is notably better than CAWNB (R^+=591.5 and R^-=74.5), LNB (R^+=462.0 and R^-=168.0), and NB (R^+=578.0 and R^-=88.0).
9) Seen from all these comparison results, our proposed AIWNB^E and AIWNB^L are generally the best among all of the algorithms used to compare. This could also fully verify the universal advantages of AIWNB for a wide range of domains.

4.3. Discussion

Moreover, to further analyze the relationship between the performances of our AIWNB algorithms and the characteristics of datasets, we observed their performances from the perspective of instance number, attribute number, and nominal attributes (%). In terms of instance number, we divide the datasets into two categories: less than 500 instances and greater than or equal to 500 instances. Similarly, we also divide the datasets into two categories according to whether they have more than 20 attributes. The third classification criteria is that whether nominal attributes (%) is greater than 50. Then we calculate the percentages of the datasets on which our AIWNB algorithms and all of their competitors obtain the highest classification accuracy under each data circumstance, respectively. Table 8 shows the detailed results. From all of these results, we can see where and when do our AIWNB algorithms perform better than their competitors. Now, we summarize the highlights as follows:

1) On the datasets whose instance number is greater than or equal to 500, the percentages with the highest classification accuracy of AIWNB^E (72.22%) and AIWNB^L (73.68%) are much higher than those on the datasets whose instance number is less than 500 (45.45% and 42.86%).
2) On the datasets whose attribute number is greater than or equal to 20, the percentages with the highest classification accuracy of AIWNB^E (81.25%) and AIWNB^L (88.89%) are also much higher than those on the datasets whose attribute number is less than 20 (41.67% and 34.78%).
3) On the datasets whose nominal attributes (%) is greater than or equal to 50, the percentages with the highest classification accuracy of AIWNB^E (65.22%) and AIWNB^L (63.64%) are also higher than those on the datasets whose nominal attributes (%) is less than 50 (47.06% and 50.00%).

In a word, our AIWNB algorithms usually perform better on the datasets whose instance number is greater than or equal to 500, attribute number is greater than or equal to 20, and nominal attributes (%) is greater than or equal to 50, such as the dataset "splice". By contrast, our AIWNB algorithms do not perform so well on the datasets whose instance number is less than 500, attribute number is less than 20, and nominal attributes (%) is less than 50, such as the dataset "glass". In summary, our AIWNB algorithms are overall more suitable for solving high-dimensional large data classification problems with nominal attributes. Conversely, for low-dimensional small data classification problems with numeric attributes, the improvements of our AIWNB algorithms are not always so obvious.

5. Conclusions and future work

Numerous approaches, including attribute weighting and instance weighting, have been proposed to improve NB. However, almost all the existing weighting approaches are essentially either attribute weighting or instance weighting, and few of them focus on attribute weighting and instance weighting simultaneously. We argue that different attributes should have different importance, meanwhile different instances should have different reliability, and thus, in this paper, we propose a new improved model called attribute and instance weighted naive Bayes (AIWNB), which combines attribute weighting with instance weighting into one uniform framework. To learn attribute weights, a correlation-based attribute weighting approach is exploited. To learn instance weights, we single out an eager approach and a lazy approach, and thus two different versions are created, which we denote as AIWNB^E and AIWNB^L, respectively. The extensive comparison experiments on a large number of classification problems validate the effectiveness of the proposed AIWNB^E and AIWNB^L.

How to learn attribute weights and instance weights are two crucial problems in AIWNB. Currently, we estimate attribute weights and instance weights directly from training data, which is efficient but quite simple. We believe that the use of more sophisticated optimizing search approaches to learn weights for both attributes and instances simultaneously could further improve the performance of the current versions and make their advantages stronger. This is the main direction for our future work. In addition, applying the attribute and instance weighting method to improve some other classification models is another interesting topic for our future work.

Declaration of Competing Interest

Author confirm that there is no conflict-of-interest in the submission, and the manuscript has been approved by all authors for publication.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China (U1711267) and Fundamental Research Funds for the Central Universities (CUG2018JM18 and CUGGC03).

References

[1] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Multiple-Valued Logic Soft Comput. 17 (2–3) (2011) 255–287.
[2] S. Chen, G.I. Webb, L. Liu, X. Ma, A novel selective naïve bayes algorithm, Knowl. Based Syst. 192 (2020) 105361.
[3] D.M. Chickering, Learning bayesian networks is np-complete, in: Learning from Data - Fifth International Workshop on Artificial Intelligence and Statistics, 1995, pp. 121–130.
[4] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (1) (2006) 1–30.

[5] D.M. Diab, K.E. Hindi, Using differential evolution for fine tuning naïve bayesian classifiers and its application for text classification, Appl. Soft Comput. 54 (2017) 183–199.
[6] D. Dua, C. Graff, UCI machine learning repository, University of California, School of Information and Computer Science, Irvine, CA, 2017.
[7] U.M. Fayyad, K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: Proceedings of the 13th International Joint Conference on Artificial Intelligence, 1993, pp. 1022–1029.
[8] E. Frank, M.A. Hall, B. Pfahringer, Locally weighted naive bayes, in: Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence, 2003, pp. 249–256.
[9] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Mach. Learn. 29 (2–3) (1997) 131–163.
[10] M.A. Hall, Correlation-based feature selection for discrete and numeric class machine learning, in: Proceedings of the Seventeenth International Conference on Machine Learning, 2000, pp. 359–366.
[11] M.A. Hall, A decision tree-based attribute weighting filter for naive bayes, Knowl. Based Syst. 20 (2) (2007) 120–126.
[12] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, 3rd ed., Morgan Kaufmann, 2011.
[13] K.E. Hindi, Fine tuning the naïve bayesian learning algorithm, AI Commun. 27 (2) (2014) 133–141.
[14] K.E. Hindi, H. AlSalman, S. Qasem, S.A. Al-Ahmadi, Building an ensemble of fine-tuned naive bayesian classifiers for text classification, Entropy 20 (11) (2018) 857.
[15] L. Jiang, Z. Cai, H. Zhang, D. Wang, Not so greedy: randomly selected naive bayes, Expert Syst. Appl. 39 (12) (2012) 11022–11028.
[16] L. Jiang, Y. Guo, Learning lazy naive bayesian classifiers for ranking, in: 17th IEEE International Conference on Tools with Artificial Intelligence, 2005, pp. 412–416.
[17] L. Jiang, C. Li, S. Wang, L. Zhang, Deep feature weighting for naive bayes and its application to text classification, Eng. Appl. Artif. Intell. 52 (2016) 26–39.
[18] L. Jiang, D. Wang, Z. Cai, Discriminatively weighted naive bayes and its application in text classification, Int. J. Artif. Intell. Tools 21 (1) (2012) 1250007.
[19] L. Jiang, H. Zhang, Z. Cai, A novel bayes model: hidden naive bayes, IEEE Trans. Knowl. Data Eng. 21 (10) (2009) 1361–1371.
[20] L. Jiang, L. Zhang, C. Li, J. Wu, A correlation-based feature weighting filter for naive bayes, IEEE Trans. Knowl. Data Eng. 31 (2) (2019) 201–213.
[21] L. Jiang, L. Zhang, L. Yu, D. Wang, Class-specific attribute weighted naive bayes, Pattern Recognit. 88 (2019) 321–330.
[22] R. Kohavi, Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996, pp. 202–207.
[23] P. Langley, W. Iba, K. Thompson, An analysis of bayesian classifiers, in: Proceedings of the 10th National Conference on Artificial Intelligence, 1992, pp. 223–228.
[24] P. Langley, S. Sage, Induction of selective bayesian classifiers, in: Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence, 1994, pp. 399–406.
[25] C. Lee, An information-theoretic filter approach for value weighted classification learning in naive bayes, Data Knowl. Eng. 113 (2018) 116–128.
[26] C. Lee, F. Gutierrez, D. Dou, Calculating feature weights in naive bayes with kullback-leibler measure, in: 11th IEEE International Conference on Data Mining, 2011, pp. 1146–1151.
[27] C. Nadeau, Y. Bengio, Inference for the generalization error, Mach. Learn. 52 (3) (2003) 239–281.
[28] J.R. Quinlan, C4.5: programs for Machine Learning, Elsevier, 2014.
[29] S. Wang, L. Jiang, C. Li, Adapting naive bayes tree for text classification, Knowl. Inf. Syst. 44 (1) (2015) 77–89.
[30] G.I. Webb, J.R. Boughton, Z. Wang, Not so naive bayes: aggregating one-dependence estimators, Mach. Learn. 58 (1) (2005) 5–24.
[31] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, Elsevier, 2011.
[32] J. Wu, S. Pan, X. Zhu, P. Zhang, C. Zhang, SODE: Self-adaptive one-dependence estimators for classification, Pattern Recognit. 51 (2016) 358–377.
[33] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A.F.M. Ng, B. Liu, P.S. Yu, Z. Zhou, M. Steinbach, D.J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (1) (2008) 1–37.
[34] W. Xu, L. Jiang, L. Yu, An attribute value frequency-based instance weighting filter for naive bayes, J. Exp. Theoretical Artif. Intell. 31 (2) (2019) 225–236.
[35] L. Yu, L. Jiang, D. Wang, L. Zhang, Toward naive bayes with attribute value weighting, Neural Comput. Appl. 31 (10) (2019) 5699–5713.
[36] N.A. Zaidi, J. Cerquides, M.J. Carman, G.I. Webb, Alleviating naive bayes attribute independence assumption by attribute weighting, J. Mach. Learn. Res. 14 (1) (2013) 1947–1988.
[37] H. Zhang, L. Jiang, L. Yu, Class-specific attribute value weighting for naive bayes, Inf. Sci. 508 (2020) 260–274.

Huan Zhang is currently a Ph.D. student at the School of Computer Science, China University of Geosciences (Wuhan). His research interests mainly include machine learning and data mining (MLDM). In MLDM domains, he has already published an article in Information Sciences and an article in International Journal of Machine Learning and Cybernetics.

Liangxiao Jiang is currently a professor at the School of Computer Science, China University of Geosciences (Wuhan). His research interests mainly include machine learning and data mining (MLDM). In MLDM domains, he has already published more than 70 papers in peer-reviewed international journals and conferences, such as TKDE, TSMC, PR, INS, KAIS, KBS, EAAI, ESWA, NCA, JIIS, PRL, JETAI, IJPRAI, FCS, IJCAI, AAAI, ICML, ICDM, DASFAA.

Liangjun Yu is currently an associate professor at the School of Computer Science of Hubei University of Education. Her research interests mainly include machine learning and data mining (MLDM). In MLDM domains, she has already published more than 10 papers in peer-refereed international journals and conferences.