Structure extended multinomial naive Bayes

Liangxiao Jianga,b,∗, Shasha Wanga, Chaoqun Lic, Lungan Zhanga

aDepartment of Computer Science, China University of Geosciences, Wuhan 430074, China
bHubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China
cDepartment of Mathematics, China University of Geosciences, Wuhan 430074, China

ARTICLE INFO

Article history:
Received 8 June 2015
Revised 6 August 2015
Accepted 14 September 2015
Available online 30 September 2015

Keywords:
Text classification
Multinomial naive Bayes
Structure extension

ABSTRACT

Multinomial naive Bayes (MNB) assumes that all attributes (i.e., features) are independent of each other given the context of the class, and it ignores all dependencies among attributes. However, in many real-world applications, the attribute independence assumption required by MNB is often violated and thus harms its performance. To weaken this assumption, one of the most direct ways is to extend its structure to represent explicitly attribute dependencies by adding arcs between attributes. On the other hand, although a Bayesian network can represent arbitrary attribute dependencies, learning an optimal Bayesian network from high-dimensional text data is almost impossible. The main reason is that learning the optimal structure of a Bayesian network from high-dimensional text data is extremely time and space consuming. Thus, it would be desirable if a multinomial Bayesian network model can avoid structure learning and be able to represent attribute dependencies to some extent. In this paper, we propose a novel model called structure extended multinomial naive Bayes (SEMNB). SEMNB alleviates the attribute independence assumption by averaging all of the weighted one-dependence multinomial estimators. To learn SEMNB, we propose a simple but effective learning algorithm without structure searching. The experimental results on a large suite of benchmark text datasets show that SEMNB significantly outperforms MNB and is even markedly better than other three state-of-the-art improved algorithms including TDM, DWMNB, and $R_w,c,MNB$.

© 2015 Elsevier Inc. All rights reserved.

1. Introduction

A Bayesian network [29] consists of a structural model and a set of conditional probabilities. The structural model is a directed acyclic graph in which nodes represent attributes (i.e., features) and arcs represent attribute dependencies. Attribute dependencies are quantified by conditional probabilities for each node given its parents. Learning Bayesian networks from data has two elements: structure learning and parameter learning. Bayesian networks are often used for classification problems, in which a learner attempts to construct a classifier from a collection of labeled training instances. The resulting classifiers are called Bayesian network classifiers. Naive Bayes is the simplest one of Bayesian network classifiers. It assumes that all attributes are independent of each other given the context of the class. This is the so-called attribute independence assumption. Thanks to this assumption, naive Bayes avoids structure learning and its parameter learning can also be greatly simplified, especially when the number of attributes is very large.

∗Corresponding author. Tel.: +86 2767883716.
E-mail addresses: lijiang@cug.edu.cn (L. Jiang), chqli@cug.edu.cn (C. Li).

http://dx.doi.org/10.1016/j.ins.2015.09.037
0020-0255/© 2015 Elsevier Inc. All rights reserved.
Text classification is just such a domain with a large number of attributes (i.e., features), in which the number of attributes is the number of different words occurring in documents. To the best of our knowledge, the number of different words occurring in documents is often very large especially when the documents are the real-world text data from the Web, UseNet and Newswire articles. Therefore, a large number of naive Bayes text classifiers [19,24,27,30,31] are proposed to address the text classification problems, among which multinomial naive Bayes (MNB) [24] is widely used due to its simplicity, efficiency and efficacy. Given a test document \( d \), represented by a word vector \( \langle w_1, w_2, \ldots, w_m \rangle \), MNB uses Eq. (1) to classify the document \( d \).

\[
c(d) = \arg \max_{c \in C} \prod_{i=1}^{m} P(w_i \mid c)^{f_i},
\]

where \( c(d) \) is the class label of \( d \) predicted by MNB, \( C \) is the set of all possible class labels, \( m \) is the vocabulary size in the text collection (the number of different words in all of the documents), \( w_i \) \( (i = 1, 2, \ldots, m) \) is the \( i \)-th word that occurs in the document \( d \), \( f_i \) is the frequency count of the word \( w_i \) in the document \( d \), \( P(c) \) is the probability that the document \( d \) occurs in the class \( c \), and \( P(w_i \mid c) \) is the conditional probability that the word \( w_i \) occurs given the class \( c \), which can be estimated by Eqs. (2) and (3) with Laplace smoothing respectively:

\[
P(c) = \frac{\sum_{j=1}^{n} \delta(c_j, c) + 1}{n + s},
\]

\[
P(w_i \mid c) = \frac{\sum_{j=1}^{n} f_{ji} \delta(c_j, c) + 1}{\sum_{j=1}^{n} \sum_{j=1}^{m} f_{ji} \delta(c_j, c) + m},
\]

where \( n \) is the number of training documents, \( s \) is the number of classes, \( c_j \) is the class label of the \( j \)-th training document, \( f_{ji} \) is the frequency count of the word \( w_i \) in the \( j \)-th training document, and \( \delta(\ast) \) is a binary function, which is one if its two parameters are identical and zero otherwise.

In MNB, each attribute node has the class node as its parent, but does not have any parent from attribute nodes. Because the values of \( P(c) \) and \( P(w_i \mid c) \) can be easily estimated from training documents, MNB is very easy to learn. However, it is obvious that its attribute independence assumption is rarely true in reality. This would harm its performance in the applications with complex attribute dependencies. In order to weaken the attribute independence assumption required for MNB, numerous approaches have been proposed. The existing work can be broadly divided into five main categories: (1) frequency transforming [3,5,11,23,25,28]; (2) instance weighting [7,8,14]; (3) local learning [13,34]; (4) feature weighting [20–22,33,35]; (5) feature selection [12,39]. Note that the improved approaches to MNB are not limited to these five categories of approaches. For example, complement naive Bayes (CNB) [30] is proposed to address the skewed training data, and a topic document model approach (TDM) [17] is proposed to improve the parameter estimation of multinomial naive Bayes classifiers.

Here we would like to focus on another approach called structure extension. Extending its structure is a direct way to weaken its attribute independence assumption, since attribute dependencies can be explicitly represented by arcs. However, learning an optimal Bayesian network from high-dimensional text data is almost impossible [4]. The main reason is that learning the optimal structure of a Bayesian network from high-dimensional text data is extremely time and space consuming. Thus, a multinomial Bayesian network model that can avoid structure learning and be able to represent attribute dependencies to some extent is desirable.

Responding to this fact, we firstly make a survey of the existing improved approaches to MNB in this paper, and then propose a novel model, called structure extended multinomial naive Bayes (SEMNBN). In SEMNB, we present a novel approach to weaken the attribute independence assumption by averaging all of the weighted one-dependence multinomial estimators. SEMNB inherits the structural simplicity of MNB and can be easily learned without structure searching. To learn SEMNB, we propose an efficient algorithm based on gain ratio. The experimental results on a large suite of benchmark text datasets validate the effectiveness of SEMNB.

The rest of the paper is organized as follows. At first, we make a survey of the existing improved approaches to MNB. Then, we propose structure extended multinomial naive Bayes (SEMNBN). Followed by the description of our experimental setup and results in detail, we draw conclusions and outline main research direction for our future work.

2. Related work

2.1. Frequency transforming

The frequency transforming approach transforms word frequencies and uses transformed word frequencies in building MNB. Given a test document \( d \), MNB with frequency transforming still uses Eq. (1) to classify the document \( d \), but the conditional probability \( P(w_i \mid c) \) is estimated by Eq. (4).

\[
P(w_i \mid c) = \frac{\sum_{j=1}^{n} f'_{ji} \delta(c_j, c) + 1}{\sum_{j=1}^{n} \sum_{j=1}^{m} f'_{ji} \delta(c_j, c) + m},
\]

where \( f'_{ji} \) is the transformed frequency count of the word \( w_i \) in the \( j \)-th training document.
To learn $f_{ji}^v$ ($j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, m$), the simplest but effective method is the document length-based transforming (i.e., document length normalization) [30], which plays an important role in performance enhancement in information retrieval [16,28]. The length of the documents is usually normalized to prevent the influence the document length might have [23,32]. The document length can be normalized in various ways. Some widely used normalization methods are described in Table 1. There are many variants of them, here we just pick a few representative ones.

In Table 1, $f_{ji}$ is the frequency count of the word $w_i$ in the $j$th training document, $dl_j$ represents the document length of the $j$th training document ($dl_j = \sum_{i=1}^{m} f_{ji}$), $avl$ represents the average document length of all the training documents ($avl = \frac{1}{n} \sum_{j=1}^{n} dl_j$). The parameter $\alpha$ in RF, Okapi’s, and Pivoted normalization is often chosen as 0.2, 0.25, and 0.2 respectively [11,25,28]. The parameters $a$ and $b$ in BM25 Normalization are often set to 0.3 and 0.9 [3] respectively. The parameter $c$ in DRF Normalization takes a value from the set of {0.25, 0.5, 0.8, 1, 2, 3, 5, 8, 10} [5]. The turning factor $tf$ in RF Normalization is defined as:

$$tf = \frac{f_{ji}(1 + k)}{f_{ji} + k(1 - b) + c \frac{dal}{all}},$$

where the parameters $k$ and $b$ are free parameters, which are often chosen as 2 and 0.75 [25] respectively.

### 2.2. Instance weighting

The instance weighting approach assigns different weights to different instances (i.e., documents) in building MNB. Given a test document $d$, MNB with instance weighting still uses Eq. (1) to classify the document $d$, but the probability $P(c)$ and the conditional probability $P(w_i | c)$ are estimated by Eqs. (6) and (7) respectively.

$$P(c) = \frac{\sum_{j=1}^{n} W_j \delta(c_j, c) + 1}{\sum_{j=1}^{n} W_j + s},$$

$$P(w_i | c) = \frac{\sum_{j=1}^{n} W_j f_{ji} \delta(c_j, c) + 1}{\sum_{i=1}^{m} \sum_{j=1}^{n} W_j f_{ji} \delta(c_j, c) + m},$$

where $W_j$ is the weight of the $j$th training instance (i.e., document).

To learn $W_j$ ($j = 1, 2, \ldots, n$), boosting [7,8] maybe one of the most effective ways. The general idea of boosting is to learn an ensemble of base classifiers, where each base classifier in the ensemble pays more attention to the instances misclassified by its predecessor. For detail, boosting increases the weights of the training instances misclassified by the classifiers learned in the last iteration, and learns the classifiers from the re-weighted training instances in the next iteration. This iteration process is repeated for $T$ rounds, where $T$ is an artificially assigned parameter.

Recently, [14] worked on the approach of instance weighting and proposed an improved multinomial naive Bayes algorithm by discriminative instance weighting. They called it discriminatively weighted multinomial naive Bayes (DWMNB). In each iteration of DWMNB, different training instances are discriminatively assigned different weights according to the estimated conditional probability loss.

### 2.3. Local learning

The local learning approach builds an MNB on the neighborhood of a test instance, instead of on the whole training data. Given a test document $d$, MNB with local learning still uses Eq. (1) to classify the document $d$, but the probability $P(c)$ and the conditional probability $P(w_i | c)$ are estimated by Eqs. (8) and (9) respectively.

$$P(c) = \frac{\sum_{j=1}^{k} \delta(c_j, c) + 1}{k + s}.$$
\[ P(w_i|c) = \frac{\sum_{j=1}^{k} f_{ji} \delta(c_j, c) + 1}{\sum_{m=1}^{n} \sum_{j=1}^{k} f_{ji} \delta(c_j, c) + m} \]  

(9)

where \( k \) is the number of training instances in the neighborhood of \( d \).

The local learning approach is a kind of instance selection approach, namely the selected training instances are dropped into the neighborhood of a test instance, which helps to weaken the effects of attribute dependencies that may exist in the whole training data.

To find the neighborhood of a test instance, the \( k \)-nearest neighbor algorithm (KNN) is the most well-recognized algorithm. Thus, the idea of combining KNN with multinomial naive Bayes is quite straightforward. Like all the other lazy learning methods, the training data is simply stored, and learning is deferred until the classification time. Whenever a test instance is classified, a local multinomial naive Bayes is trained using the training data is simply stored, and learning is deferred until the classification time. Whenever a test instance is classified, a local multinomial naive Bayes is trained using the \( k \) nearest neighbors of the test instance, with which the test instance is classified. Based on this idea, [13] proposed a locally weighted version of multinomial naive Bayes called locally weighted multinomial naive Bayes (LWMNB).

Recently, [34] applied a decision tree learning algorithm to find the neighborhood of a test instance, and then deployed a multinomial naive Bayes text classifier on each leaf node of the built decision tree. The resulting model is called multinomial naive Bayes tree (MNBTree). MNBTree builds a binary tree, in which the split attributes’ values are divided into zero and nonzero. At the same time, MNBTree uses the information gain measure to build the tree for reducing the time consumption.

\[ P(c) = \sum_{j=1}^{n} \delta(c_j, c) + 1 \]

\[ \sum_{j=1}^{n} \delta(c_j, c) + n + s \]

(10)

(11)

(12)

where \( c(d) = \arg \max_{c \in C} P(c) \prod_{i=1}^{q} P(w_i|c) ^ {L_i} \) to classify the document \( d \).

2.4. Feature weighting

The feature weighting approach assigns different weights to different features (i.e., attributes) in building MNB. Given a test document \( d \), MNB with feature weighting uses Eq. (10) to classify the document \( d \).

2.5. Feature selection

The feature selection approach builds an MNB on the selected feature subset, instead of on the whole space of features. Given a test document \( d \), MNB with feature selection uses Eq. (12) to classify the document \( d \).
moderate-sized text collection, the native feature space consists of tens or hundreds of thousands of unique features (words). This is prohibitively high for many learning algorithms. For example, Bayesian network models will be computationally intractable unless an independence assumption (often not true) among features is imposed. It is highly desirable to reduce the native space without sacrificing classification accuracy.

To execute the feature subset in text classification, a large number of feature selection methods have been proposed. Yang and Pedersen [39] provided a comparative study of feature selection methods for text classification. In detail, they summarized and evaluated five feature selection methods based on document frequency (DF), information gain (IG), mutual information (MI), $\chi^2$ statistic (CHI), and term strength (TS). Their experimental results indicate that IG and CHI are most effective in aggressive feature removal without losing classification accuracy. Besides, [12] proposed another two-stage Markov blanket based feature selection algorithm.

3. Structure extension for multinomial naive Bayes

In this section, we propose another approach called structure extension. It uses directed arcs to explicitly represent the dependencies among features. Given a test document $d$, MNB with structure extension uses Eq. (15) to classify the document $d$.

$$c(d) = \arg \max_{c \in C} P(c) \prod_{i=1}^{m} P(w_i | \Pi_{w_i}, c)^{f_i},$$

where $\Pi_{w_i}$ denotes the set of feature parents of $w_i$ in the Bayesian network.

Learning an optimal $\Pi_{w_i}$ for each feature $w_i$ is similar to learning an optimal Bayesian network that has been proved to be NP-hard [4]. In addition, when the structural complexity of a Bayesian network is high, its variance is also high due to the limited training data [9], and thus its probability estimates could be still poor. Therefore, a multinomial Bayesian network model without structure learning, and is still able to represent dependencies among features to some extent, is desirable. This is the main motivation of this paper.

To retain naive Bayes’ desirable simplicity and efficiency while alleviating the problems of the attribute independence assumption, averaged one-dependence estimators (AODE) [36] and its improved version called weighted average of one-dependence estimators (WAODE) [15] have demonstrated remarkable error performance. Inspired by them, here we propose a novel model, called structure extended multinomial naive Bayes (SEMNB). SEMNB weakens the attribute independence assumption by averaging all of the weighted one-dependence multinomial estimators, and thus inherits the structural simplicity and efficiency of MNB.

In SEMNB, a one-dependence multinomial estimator is built for each present word $w_i$, in which the present word $w_i$ is directly set to be the parent of all other present words $w_t$ ($t \neq i$). Once all of the one-dependence multinomial estimators are learned, they are assigned different weights $W_i$ ($i = 1, 2, \ldots, m$). So, given a test document $d$, SEMNB uses Eq. (16) to classify it.

$$c(d) = \arg \max_{c \in C} \left( \frac{\sum_{i=1}^{m} W_i P(c) P(w_i | c)^{f_i} \prod_{i=1, t \neq i}^{m} P(w_t | w_i, c)^{f_t}}{\sum_{i=1, t \neq i}^{m} W_i} \right).$$

The only thing left now is how to efficiently learn $W_i$, $P(c)$, $P(w_i | c)$, and $P(w_t | w_i, c)$ in Eq. (16). We will explain this in details. Firstly, we use the gain ratio, GainRatio($D$, $w_i$), of the word $w_i$ splitting the given training documents set $D$ to define the weight $W_i$ as follows:

$$W_i = \text{GainRatio}(D, w_i) = \frac{\text{Gain}(D, w_i)}{\text{SplitInformation}(D, w_i)},$$

where $\text{Gain}(D, w_i)$ is the information gain of the word $w_i$ splitting $D$, and it is defined by Eq. (18).

$$\text{Gain}(D, w_i) = \text{Entropy}(D) - \sum_{f_i \in \{0, \bar{0}\}} \frac{|D_{f_i}|}{|D|} \text{Entropy}(D_{f_i}).$$

where $|D_{f_i}|$ is the number of the documents whose value of $w_i$ is $f_i$, where $f_i \in \{0, \bar{0}\}$, and $f_i = 0$ indicates the absence of $w_i$, and $f_i = \bar{0}$ indicates the presence of $w_i$. Entropy($D$) is the entropy of $D$ defined by

$$\text{Entropy}(D) = - \sum_{c \in C} P(c) \log_2 P(c),$$

where $P(c)$ is the probability of class $c$ in $D$.

SplitInformation($D$, $w_i$) is the split information of $w_i$, which is actually the entropy of $D$ with respect to the values ($f_i \in \{0, \bar{0}\}$) of $w_i$:

$$\text{SplitInformation}(D, w_i) = - \sum_{f_i \in \{0, \bar{0}\}} \frac{|D_{f_i}|}{|D|} \log_2 \frac{|D_{f_i}|}{|D|}.$$  

SplitInformation($D$, $w_i$) is sensitive to how broadly and uniformly the word $w_i$ splits the training documents set $D$. 

\[ \text{SplitInformation}(D, w_i) = - \sum_{f_i \in \{0, \bar{0}\}} \frac{|D_{f_i}|}{|D|} \log_2 \frac{|D_{f_i}|}{|D|}. \]
Secondly, the probability \( P(c) \) and the conditional probability \( P(w_i|c) \) are also estimated by Eqs. (2) and (3) respectively. It has a space complexity of \( O(sm) \), where \( s \) is the number of classes and \( m \) is the vocabulary size in the text collection (the number of different words in all of the documents).

Finally, \( P(w_i|w_j, c) \) is the conditional probability \( w_i \) appears given \( w_j \) and \( c \). To the best of our knowledge, the space complexity of directly estimating \( P(w_i|w_j, c) \) from the given training documents set \( D \) is \( O(sm^2) \). Just as discussed in Section 2.5, a major characteristic, or difficulty, of text classification problems is the high dimensionality of the feature space. Even in a moderate-sized text collection, the native feature space consists of tens or hundreds of thousands of unique features (words). Therefore, \( m \) is usually so large that it is prohibitive for saving the tables of joint feature (word) and class frequencies. From these tables, the probability estimates \( P(w_i|w_j, c) \) are derived that are required for estimating \( P(w_i|w_j, c) \). On the other hand, to our observation, the text data is often a sparse matrix, and thus the number of different (present) words in a test document \( d \) is much smaller than \( m \) (the vocabulary size in the text collection). We denote it by \( |d| \) (\( |d| = \sum_{i=1}^{m} j_{i} \)).

Based on these facts, we propose to transform some training space consumption into test time consumption. Namely, we transfer the step of estimating \( P(w_i|w_j, c) \) from the training time to the test (classification) time. At training time, we only need to calculate \( W_i \), \( P(c) \), and \( P(w_j|c) \). At test time, whenever a test document \( d \) is classified, \( P(w_i|w_j, c) \) is estimated according to the stored training documents set \( D \) and the test document \( d \). For detail, given a word \( w_i \) presented in \( d \), we only select the training documents in which \( w_i \) appears to estimate the conditional probability \( P(w_i|w_j, c) \) by Eq. (21). This only takes the space complexity of \( O(s|d|) \).

\[
P(w_i|w_j, c) = \frac{\sum_{j_{i}=1}^{n_{i}} f_{j_{i}}(c_{j_{i}}, c) + 1}{\sum_{j_{i}=1}^{n_{i}} \sum_{j_{j} \neq 0} f_{j_{i}}(c_{j_{i}}, c) + m},
\]  

(21)

Based on our idea, the algorithm framework of our proposed SEMNB can be graphically shown in Fig. 1. The whole learning algorithm for SEMNB is partitioned into an eager subsection (SEMNB-Training) and a lazy subsection (SEMNB-Test). They are depicted below in detail.

1. **Algorithm**: SEMNB-Training \((D)\)
2. **Input**: a training documents set \( D \)
3. **Output**: \( P(c) \), \( P(w_j|c) \), and \( W_i \) \((i = 1, 2, \ldots, m)\)
   1. For each class label \( c \)
      1. Compute \( P(c) \) from \( D \) using Eq. (2)
   2. For each word \( w_j \) \((i = 1, 2, \ldots, m)\) from \( D \)
      1. For each class label \( c \)
         1. Compute \( P(w_j|c) \) using Eq. (3)
      2. Compute \( W_i \) using Eq. (17)
   3. Return the computed \( P(c) \), \( P(w_j|c) \), and \( W_i \) \((i = 1, 2, \ldots, m)\)

1. **Algorithm**: SEMNB-Test \((d, D, P(c), P(w_j|c), W_i)\)
2. **Input**: a test document \( d \), the training documents set \( D \), the computed \( P(c) \), \( P(w_j|c) \), and \( W_i \) via SEMNB-Training
3. **Output**: the predicted class label of \( d \)
   1. For each present word \( w_i \) \((i = 1, 2, \ldots, |d|)\) in \( d \)
      1. Select the training documents in which \( w_i \) appears as \( D_{w_i} \)
      2. For each present word \( w_t \) \((t = 1, 2, \ldots, |d| \wedge t \neq i)\) in \( d \)
         1. For each class label \( c \)
            1. Use Eq. (21) to compute \( P(w_i|w_t, c) \) from \( D_{w_i} \)
            2. Based on \( P(c) \), \( P(w_t|c) \), \( P(w_i|w_t, c) \), and \( W_i \), use Eq. (16) to predict the class label of \( d \)
   3. Return the predicted class label of \( d \) as \( c(d) \)
Table 2
Computational complexity.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Training Time</th>
<th>Training Space</th>
<th>Test Time</th>
<th>Test Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNB</td>
<td>O(nm)</td>
<td>O(sm)</td>
<td>O(sm)</td>
<td>O(sm)</td>
</tr>
<tr>
<td>SEMNB</td>
<td>O(nm + sm)</td>
<td>O(sm)</td>
<td>O(n</td>
<td>d</td>
</tr>
<tr>
<td>AODE</td>
<td>O(nm^2)</td>
<td>O(sm^2v^2)</td>
<td>O(sm^2)</td>
<td>O(sm^2v^2)</td>
</tr>
<tr>
<td>WAODE</td>
<td>O(nm^2 + smv)</td>
<td>O(sm^2v^2 + m + sv)</td>
<td>O(sm^2)</td>
<td>O(sm^2v^2 + m)</td>
</tr>
</tbody>
</table>

n is the number of training documents; m is the vocabulary size of training documents, namely the number of attributes; s is the number of classes; |d| is the number of different (present) words in the test document d; v is the average number of values for an attribute of the traditional categorical data.

Table 3
Datasets used in the experiments.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Documents number</th>
<th>Words number</th>
<th>Classes number</th>
</tr>
</thead>
<tbody>
<tr>
<td>fbis</td>
<td>2463</td>
<td>2000</td>
<td>17</td>
</tr>
<tr>
<td>la1s</td>
<td>3204</td>
<td>13195</td>
<td>6</td>
</tr>
<tr>
<td>la2s</td>
<td>3075</td>
<td>12432</td>
<td>6</td>
</tr>
<tr>
<td>oh8</td>
<td>1003</td>
<td>3182</td>
<td>10</td>
</tr>
<tr>
<td>oh10</td>
<td>1050</td>
<td>3238</td>
<td>10</td>
</tr>
<tr>
<td>oh15</td>
<td>913</td>
<td>3100</td>
<td>10</td>
</tr>
<tr>
<td>oh5</td>
<td>918</td>
<td>3012</td>
<td>10</td>
</tr>
<tr>
<td>ohscal</td>
<td>11162</td>
<td>11465</td>
<td>10</td>
</tr>
<tr>
<td>re0</td>
<td>1657</td>
<td>3758</td>
<td>25</td>
</tr>
<tr>
<td>re1</td>
<td>1504</td>
<td>2886</td>
<td>13</td>
</tr>
<tr>
<td>tr11</td>
<td>414</td>
<td>6429</td>
<td>9</td>
</tr>
<tr>
<td>tr12</td>
<td>313</td>
<td>5804</td>
<td>8</td>
</tr>
<tr>
<td>tr21</td>
<td>336</td>
<td>7902</td>
<td>6</td>
</tr>
<tr>
<td>tr23</td>
<td>204</td>
<td>5832</td>
<td>6</td>
</tr>
<tr>
<td>tr31</td>
<td>927</td>
<td>10128</td>
<td>7</td>
</tr>
<tr>
<td>tr41</td>
<td>878</td>
<td>7454</td>
<td>10</td>
</tr>
<tr>
<td>tr45</td>
<td>690</td>
<td>8261</td>
<td>10</td>
</tr>
<tr>
<td>wap</td>
<td>1560</td>
<td>8640</td>
<td>20</td>
</tr>
</tbody>
</table>

Through the analysis and description above, we can find our transformation from training space consumption to test time consumption is pretty simple. Our learning algorithm significantly reduces the training space complexity, yet at the same time does not incur a very high test time complexity. Table 2 displays the relative computational complexity of our learning algorithm compared to the standard multinomial naive Bayes (MNB). Note that our improved idea is inspired by AODE [36] and WAODE [15], but our proposed SEMNB is totally different from them. At first, AODE and WAODE can be viewed as two improved algorithms of naive Bayes (NB) for traditional categorical data classification, while our proposed SEMNB aims at improving multinomial naive Bayes (MNB) for text classification. Secondly, AODE and WAODE can’t be applied to classify text data because all attribute values in text data are the frequency information, generally integer values, of all words in the vocabulary of training documents. At the same time, it is almost impossible to directly adapt them to text classification due to their very high space complexity and the high dimension of text data. Thirdly, our proposed SEMNB subtly transforms the training space consumption confronting AODE and WAODE to test time consumption without incurring a very high test time complexity. The detailed computational complexity comparisons can also be found in Table 2.

4. Experiments and results

In this section, we design a group of experiments to empirically validate the effectiveness of our proposed SEMNB. We compare it to the standard MNB and some other state-of-the-art improved algorithms including CNB [30], TDM [17], DWMNB [14], MNBTree [34], Rw.c.MNB [21], and FWMNB [35] in terms of classification accuracy. We use the existing implementations of MNB and CNB in WEKA platform [38] and implement our proposed SEMNB and all the other competitors, including TDM, DWMNB, MNBTree, Rw.c.MNB, and FWMNB, in WEKA platform [38].

We ran our experiments on 18 widely used text classification benchmark datasets published on the main web site of WEKA platform [38]. They represent a wide range of domains and data characteristics. The detailed description of these 18 datasets is shown in Table 3. To save the time and memory in running experiments, we don’t include the largest dataset: “new3s” in our experiments. We carefully observed this dataset and found that there exists a mass of too sparse words in it.

In our experiments, the classification accuracy of each algorithm on each dataset are obtained via 10 runs of 10-fold cross-validation. Runs with the various algorithms are carried out on the same training sets and evaluated on the same test sets. In particular, the cross-validation folds are the same for all the experiments in each dataset. Table 4 shows the detailed classification accuracy of each algorithm in each dataset. Besides, the averaged classification accuracy is summarized at the bottom of the
Table 4
Classification accuracy (%) comparisons for SEMNB versus MNB, CNB, TDM, DWMNB, MNBTree, $R_w$, MNB, and FWMNB.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MNB</th>
<th>CNB</th>
<th>TDM</th>
<th>DWMNB</th>
<th>MNBTree</th>
<th>$R_w$, MNB</th>
<th>FWMNB</th>
<th>SEMNB</th>
</tr>
</thead>
<tbody>
<tr>
<td>fbis</td>
<td>77.11</td>
<td>76.78</td>
<td>82.44</td>
<td>80.39</td>
<td>79.06</td>
<td>79.87</td>
<td>78.69</td>
<td>83.27</td>
</tr>
<tr>
<td>la1s</td>
<td>88.41</td>
<td>86.3</td>
<td>90.48</td>
<td>88.85</td>
<td>87.22</td>
<td>87.88</td>
<td>88.79</td>
<td>89.15</td>
</tr>
<tr>
<td>la2s</td>
<td>89.88</td>
<td>88.26</td>
<td>90.98</td>
<td>90.14</td>
<td>87.34</td>
<td>88.72</td>
<td>90.22</td>
<td>91.01</td>
</tr>
<tr>
<td>oh0</td>
<td>89.35</td>
<td>92.31</td>
<td>88.95</td>
<td>89.64</td>
<td>88.93</td>
<td>89.05</td>
<td>91.47</td>
<td>88.87</td>
</tr>
<tr>
<td>oh10</td>
<td>83.66</td>
<td>84.38</td>
<td>81.12</td>
<td>83.29</td>
<td>79.01</td>
<td>83.61</td>
<td>85.63</td>
<td>83.36</td>
</tr>
<tr>
<td>oh15</td>
<td>86.63</td>
<td>90.58</td>
<td>85.97</td>
<td>86.87</td>
<td>88.74</td>
<td>86.46</td>
<td>89.32</td>
<td>87.55</td>
</tr>
<tr>
<td>oh5</td>
<td>88.41</td>
<td>86.3</td>
<td>90.48</td>
<td>88.85</td>
<td>87.22</td>
<td>87.88</td>
<td>88.79</td>
<td>89.15</td>
</tr>
<tr>
<td>ohscal</td>
<td>74.74</td>
<td>76.5</td>
<td>74.6</td>
<td>74.3</td>
<td>78.00</td>
<td>74.18</td>
<td>76.31</td>
<td>76.40</td>
</tr>
<tr>
<td>re0</td>
<td>80.02</td>
<td>82.37</td>
<td>79.66</td>
<td>81.81</td>
<td>77.3</td>
<td>77.07</td>
<td>80.93</td>
<td>82.73</td>
</tr>
<tr>
<td>re1</td>
<td>83.31</td>
<td>84.99</td>
<td>81.98</td>
<td>83.13</td>
<td>84.26</td>
<td>82.72</td>
<td>85.38</td>
<td>82.22</td>
</tr>
<tr>
<td>tr11</td>
<td>85.21</td>
<td>82.64</td>
<td>86.8</td>
<td>85.81</td>
<td>85.79</td>
<td>85.44</td>
<td>86.83</td>
<td>87.62</td>
</tr>
<tr>
<td>tr12</td>
<td>80.99</td>
<td>86.32</td>
<td>83.28</td>
<td>82.46</td>
<td>85.3</td>
<td>84.76</td>
<td>82.62</td>
<td>86.64</td>
</tr>
<tr>
<td>tr21</td>
<td>61.9</td>
<td>85.94</td>
<td>88.79</td>
<td>78.45</td>
<td>86.15</td>
<td>69.63</td>
<td>65.12</td>
<td>90.36</td>
</tr>
<tr>
<td>tr23</td>
<td>71.15</td>
<td>70.59</td>
<td>85.51</td>
<td>84.02</td>
<td>93.04</td>
<td>73.82</td>
<td>73.4</td>
<td>89.05</td>
</tr>
<tr>
<td>tr31</td>
<td>94.6</td>
<td>94.67</td>
<td>97.06</td>
<td>96.28</td>
<td>96.48</td>
<td>94.2</td>
<td>95.54</td>
<td>95.94</td>
</tr>
<tr>
<td>tr41</td>
<td>94.65</td>
<td>94.23</td>
<td>95.13</td>
<td>95.21</td>
<td>94.38</td>
<td>93.05</td>
<td>95.61</td>
<td>94.97</td>
</tr>
<tr>
<td>tr45</td>
<td>83.6</td>
<td>87.2</td>
<td>91.99</td>
<td>87.36</td>
<td>90.36</td>
<td>88.88</td>
<td>86.59</td>
<td>91.54</td>
</tr>
<tr>
<td>wap</td>
<td>81.22</td>
<td>77.53</td>
<td>81.04</td>
<td>81.83</td>
<td>75.42</td>
<td>76.31</td>
<td>82.53</td>
<td>80.53</td>
</tr>
<tr>
<td>Average</td>
<td>82.62</td>
<td>84.63</td>
<td>85.87</td>
<td>85.03</td>
<td>85.56</td>
<td>83.12</td>
<td>84.29</td>
<td>86.82</td>
</tr>
</tbody>
</table>

Table 5
Ranks computed by the Wilcoxon test.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MNB</th>
<th>CNB</th>
<th>TDM</th>
<th>DWMNB</th>
<th>MNBTree</th>
<th>$R_w$, MNB</th>
<th>FWMNB</th>
<th>SEMNB</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNB</td>
<td>-</td>
<td>53.0</td>
<td>47.0</td>
<td>16.0</td>
<td>58.0</td>
<td>91.0</td>
<td>0.0</td>
<td>19.0</td>
</tr>
<tr>
<td>CNB</td>
<td>118.0</td>
<td>-</td>
<td>63.0</td>
<td>84.0</td>
<td>82.5</td>
<td>114.0</td>
<td>61.0</td>
<td>46.0</td>
</tr>
<tr>
<td>TDM</td>
<td>124.0</td>
<td>108.0</td>
<td>-</td>
<td>105.0</td>
<td>99.0</td>
<td>141.0</td>
<td>87.0</td>
<td>33.0</td>
</tr>
<tr>
<td>DWMNB</td>
<td>155.0</td>
<td>87.0</td>
<td>66.0</td>
<td>-</td>
<td>96.28</td>
<td>96.48</td>
<td>94.2</td>
<td>95.54</td>
</tr>
<tr>
<td>MNBTree</td>
<td>113.0</td>
<td>88.5</td>
<td>72.0</td>
<td>95.0</td>
<td>-</td>
<td>127.0</td>
<td>77.0</td>
<td>47.0</td>
</tr>
<tr>
<td>$R_w$, MNB</td>
<td>80.0</td>
<td>57.0</td>
<td>30.0</td>
<td>28.0</td>
<td>44.0</td>
<td>-</td>
<td>42.0</td>
<td>7.5</td>
</tr>
<tr>
<td>FWMNB</td>
<td>171.0</td>
<td>110.0</td>
<td>84.0</td>
<td>103.0</td>
<td>94.0</td>
<td>129.0</td>
<td>-</td>
<td>64.0</td>
</tr>
<tr>
<td>SEMNB</td>
<td>152.0</td>
<td>125.0</td>
<td>138.0</td>
<td>141.0</td>
<td>124.0</td>
<td>163.0</td>
<td>107.0</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6
Summary of the Wilcoxon test. $\ast$ = the method in the row improves the method of the column. $\circ$ = the method in the column improves the method of the row. Lower diagonal level of significance $\alpha = 0.05$, Upper diagonal of level significance $\alpha = 0.1$.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MNB</th>
<th>CNB</th>
<th>TDM</th>
<th>DWMNB</th>
<th>MNBTree</th>
<th>$R_w$, MNB</th>
<th>FWMNB</th>
<th>SEMNB</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNB</td>
<td>-</td>
<td>$\ast$</td>
<td>$\circ$</td>
<td>$\circ$</td>
<td>$\circ$</td>
<td>$\circ$</td>
<td>$\circ$</td>
<td>$\circ$</td>
</tr>
<tr>
<td>CNB</td>
<td>$\circ$</td>
<td>-</td>
<td>$\circ$</td>
<td>$\circ$</td>
<td>$\circ$</td>
<td>$\circ$</td>
<td>$\circ$</td>
<td>$\circ$</td>
</tr>
<tr>
<td>TDM</td>
<td>$\circ$</td>
<td>$\ast$</td>
<td>-</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
</tr>
<tr>
<td>DWMNB</td>
<td>$\ast$</td>
<td>$\circ$</td>
<td>$\ast$</td>
<td>-</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
</tr>
<tr>
<td>MNBTree</td>
<td>$\ast$</td>
<td>$\circ$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>-</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
</tr>
<tr>
<td>$R_w$, MNB</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>-</td>
<td>$\ast$</td>
<td>$\ast$</td>
</tr>
<tr>
<td>FWMNB</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>-</td>
<td>$\ast$</td>
</tr>
<tr>
<td>SEMNB</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>$\ast$</td>
<td>-</td>
</tr>
</tbody>
</table>

The average (arithmetic mean) across all datasets provides a gross indication of relative performance in addition to other statistics.

We take advantage of KEEL Data-Mining Software Tool [1,2] to finish the Wilcoxon signed-ranks test [6,37] for comparing each pair of algorithms. The Wilcoxon signed-ranks test [37] is a non-parametric alternative to the paired t-test [26]. It ranks the differences in performances of two classifiers for each data set, ignoring the signs, and compares the ranks for the positive and the negative differences [6]. Table 5 shows the detailed ranks computed by the Wilcoxon test. In Table 5, each number below the diagonal is the sum of ranks for the data sets on which the algorithm in the row outperformed the algorithm in the corresponding column (the sum of ranks for the positive differences, denoted by $R^+$), and each number above the diagonal is the sum of ranks for the data sets 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 $\alpha = 0.05$ and $N = 18$ data sets, we speak of two classifiers as being “significantly different” if the smaller of $R^+$ and $R^-$ is equal or less than 40 and thus we reject the null-hypothesis. Table 6 summarizes the detailed compared results of the Wilcoxon test.
test. In Table 6, • indicates that the algorithm in the row improves the algorithm in the corresponding column, and ◦ indicates that the algorithm in the column improves the algorithm in the corresponding row.

From our experimental results, we can see that SEMNB significantly outperforms MNB and is even markedly better than other three state-of-the-art improved algorithms including TDM, DWMNB, and Rw, c MNB. Now, we summarize the highlights as follows:

1. SEMNB outperforms the original MNB with $R^+ = 152$ and $R^- = 19$, and the smaller of which is much less than 40. We can draw a conclusion that the difference between SEMNB and MNB is significant. Further, the averaged classification accuracy (86.82%) of SEMNB is also much higher than that of MNB (82.62%).

2. SEMNB is markedly better than other three state-of-the-art improved algorithms: TDM ($R^+ = 138$ and $R^- = 33$), DWMNB ($R^+ = 141$ and $R^- = 30$), and Rw, c MNB ($R^+ = 163.5$ and $R^- = 7$). Additionally, the averaged classification accuracy of TDM (85.87%), DWMNB (85.03%), and Rw, c MNB (83.12%) are also much lower than that of SEMNB.

3. According to the ranks and summary of the Wilcoxon test, SEMNB is generally the best one among all of the algorithms used to compare.

Finally, we observe the efficiency of our proposed SEMNB and compare it to its competitors in terms of elapsed training and test time in seconds. Our experiments were performed on a desktop PC with 64-bit Microsoft Windows 7 with Intel(R) Core(TM) i7-4770 Quad core CPU @ 3.40 GHz and 12 GB RAM. The detailed compared results are shown in Tables 7 and 8 respectively.
Note that a small number is better than a large number in terms of elapsed training and test time. From these compared results, we can see that our proposed SEMNB does not incur a very high training and test time complexity, and is even faster than the previous published FWMNB. This conclusion also indicates that our transformation from training space consumption to test time consumption is successful and valuable.

5. Conclusions and future work

In this paper, we summarize all categories of the existing improved approaches to multinomial naive Bayes (MNB) and then propose a novel model called structure extended multinomial naive Bayes (SEMN). SEMNB alleviates the attribute independence assumption by averaging all of the weighted one-dependence multinomial estimators. To learn SEMNB, we propose a simple but effective learning algorithm without structure searching. Our experimental results show that SEMNB has a better overall performance compared to the other state-of-the-art models for improving MNB. Considering the effectiveness and computational simplicity of SEMNB, it could be a promising model for real-world text classification applications.

The existing improved approaches to MNB, including frequency transforming, instance weighting, local learning, feature weighting, and feature selection, are essentially meta learning methods, and thus they could be used for improving SEMNB. We believe that the use of these meta learning methods, such as frequency transforming and instance weighting, could enhance the classification performance of the current SEMNB. This is the main research direction for our future work.

Acknowledgments

The work was partially supported by the National Natural Science Foundation of China (61203287), the Program for New Century Excellent Talents in University (NCET-12-0953), the Chenguang Program of Science and Technology of Wuhan (2015070404010202), and the Fundamental Research Funds for the Central Universities (CUG130504, CUG130414).

References


