Pattern Recognition Letters 30 (2009) 477–485

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Class dependent feature scaling method using naive Bayes classifier for text datamining

Eunseog Youn [a], Myong K. Jeong [b,*]

[a] Department of Computer Science, Texas Tech University, Lubbock, TX 79409, USA
[b] Department of Industrial and Systems Engineering and RUTCOR (Rutgers Center for Operations Research), Rutgers, the State University of New Jersey, Piscataway, NJ 08854, USA

ARTICLE INFO

Article history:
Received 18 September 2007
Received in revised form 1 August 2008
Available online 24 December 2008

Communicated by L. Heutte

Keywords:
Classification
Feature selection
Naive Bayes classifier
Recursive feature elimination

ABSTRACT

The problem of feature selection is to find a subset of features for optimal classification. A critical part of feature selection is to rank features according to their importance for classification. The naive Bayes classifier has been extensively used in text categorization. We have developed a new feature scaling method, called class–dependent–feature–weighting (CDFW) using naive Bayes (NB) classifier. A new feature scaling method, CDFW–NB–RFE, combines CDFW and recursive feature elimination (RFE). Our experimental results showed that CDFW–NB–RFE outperformed other popular feature ranking schemes used on text datasets.

1. Introduction

Text data mining (Chakrabarti, 2000) is a research domain involving many research areas, such as natural language processing, machine learning, information retrieval (Salton, 1989), and data mining. Text categorization and feature selection are two of the many text data mining problems. The text document categorization problem has been studied by many researchers (Joachims, 1997; Joachims, 1998; Lang, 1995; Leopold and Kindermann, 2002; McCallum and Kamal Nigam, 1998; Rennie, 2001; Rifkin, 2002; Salton, 1989; Yang and Pedersen, 1997). Yang et al. showed comparative research on feature selection in text classification (Yang and Pedersen, 1997). Many machine learning methods have been used for the text categorization problem. Some of the popular methods include the naive Bayes method (Mladenic, 1998; Rennie et al., 2003), support vector machines (SVM) (Joachims, 2000), and maximum entropy classifiers (Nigam et al., 1999), to name a few. Despite naive Bayes classifier's successful applications to text document categorization problems, the feature selection using the naive Bayes classifier has been given little attention.

The naive Bayes classifier has been successful despite its crude class conditional independence assumption. Obviously, most real datasets violate this assumption. Due to the naive assumption, the naive Bayes often leads to the poor posterior probability. Webb and Pazzani (1998) and Bennett (2000) studied to get the better posteriors to accommodate the violation of the assumption. The feature independence assumption related to the naive Bayes has been studied in conjunction with the naive Bayes classification performance (Domingos and Pazzani, 1996; Rish, 2001). Many researchers, however, tried to relax this crude assumption in the hope of getting better classification accuracy. However, it seems that there is no greater advance in this direction. Friedman et al. (1997) compared the Bayesian classifier with the Bayesian network which supposedly less violates the independence assumption, and found the latter did not give significant improvement. More survey on this issue can be found in Domingos et al.'s study (Domingos and Pazzani, 1997).

Despite the naive assumption of the naive Bayes, its success has not been well explained or understood until recently. Domingos and Pazzani (1997), Domingos and Pazzani (1996), and Friedman (1997) have investigated this issue recently. Their findings are essentially the distinction between probability estimation and classification performance. Their claims are even stronger to say that detecting and relaxing this assumption is not necessarily the best way to improve performance. Keeping this in mind, we have developed the class–dependent–feature–weighting approach as a new feature ranking method using naive Bayes (CDFW–NB). We have shown its application to text document datasets and protein sequence dataset. In the following sections, we introduce naive Bayes classifier, text categorization, feature selection, our proposed feature ranking method, and its experimental results.

* Corresponding author. Tel.: +1 732 445 4858; fax: +1 732 445 5472.
E-mail address: mjeong@rci.rutgers.edu (M.K. Jeong).

0167-8655/$ - see front matter © 2008 Published by Elsevier B.V.
doi:10.1016/j.patrec.2008.11.013

2. Naive Bayes classifier

The Bayesian method is one of the most extensively used machine learning methods (Mitchel, 1997). Among them, the naive Bayes classifier is popular in text document classification. Its successful applications to text document datasets have been shown in many research articles (Joachims, 1997; McCallum and Kamal Nigam, 1998; Mitchel, 1997; Rennie, 2001). The popularity of the naive Bayes classifier in text classification problems is due to its simplicity such as a linear time complexity, and no parameters to be adjusted.

We introduce the terminology which we will use. We will denote a feature (input) vector by the symbol $\boldsymbol{X}$ and the ith component of $\boldsymbol{X}$ is written as X_i. A particular observed instance vector is denoted as $\boldsymbol{x}$ and the ith component of $\boldsymbol{x}$ is denoted as x_i. Class label variable is denoted as Y and a particular value of Y is written as y. We will use $\boldsymbol{w}$ to represent the word vector corresponding to the feature vector. The ith component of $\boldsymbol{w}$ is denoted as w_i and w_i is the word corresponding to the ith feature. Hence, X_i is a random variable denoting the number of occurrences of the word w_i. We will simplify the notation as follows:

$$P(Y=y|\boldsymbol{X}=\boldsymbol{x}) = P(Y=y|(X_1,\ldots,X_d)=(x_1,\ldots,x_d)) = P(y|x_1,\ldots,x_d) = P(y|\boldsymbol{x}).$$

$P(w_i|y)$ is the probability that a randomly drawn word from a document in class y will be the word w_i. Now let us begin with the Bayes formula, which is fundamental theorem underlying our feature ranking. Bayes theorem:

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}.$$

We take an example of an application of the Bayes theorem to a classification problem. Suppose we have a two category ($Y=+1$ or -1) classification problem. We can do the classification using the Bayes theorem. Given a test instance $\boldsymbol{x}$, we compute

$$P(+1|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|+1)P(+1)}{P(\boldsymbol{x})},$$
$$P(-1|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|-1)P(-1)}{P(\boldsymbol{x})}.$$

Then we assign $\boldsymbol{x}$ as +1 class if $P(+1|\boldsymbol{x}) > P(-1|\boldsymbol{x})$; otherwise -1. The probability $P(y|\boldsymbol{x})$ is called *a posteriori* probability (or posterior) of y and $P(\boldsymbol{x}|y)$ is called the likelihood of y with respect to $\boldsymbol{x}$. The naive Bayes classifier tries to assign a class label which maximizes *a posteriori* probability (MAP) for a given test instance. That is why this classifier is often called the MAP naive Bayes classifier. We turn to how to compute *a posteriori* probability of y, given an instance $\boldsymbol{x}$. Using the Bayes theorem,

$$P(y|\boldsymbol{x}) = P(y|(x_1,x_2\cdots x_d)) = \frac{P((x_1,x_2\cdots x_d)|y)\cdot P(y)}{P(x_1,x_2\cdots x_d)} \quad (1)$$

Since we only want to compare the posterior probabilities for different y's, and the denominator is common for different y's, we can simply ignore the denominator and compute only the numerator in (1). In the numerator, $P(y)$, called the prior probability, can be computed by simply counting the number of instances whose class labels are y, and the fraction of this number over the total number of training instances is $P(y)$. But computing $P((x_1,x_2\cdots x_d)|y)$ by the same fashion is not feasible unless the data are big enough (Mitchel, 1997). The naive Bayes approach tries to get around this problem by a simplifying assumption regarding the relationship between features (Han and Kamber, 2001; Mitchel, 1997; Tan et al., 2005; Sivia, 1996). The naive Bayes approach thus introduces the class conditional independence assumption between the features. Hence, the numerator becomes

$$P((x_1,x_2,\cdots,x_d)|y)\cdot P(y) = \prod_{i=1}^{d} P(x_i|y)\cdot P(y) \quad (2)$$

In summary, the naive Bayes approach classifies an instance $\boldsymbol{x}$ as c where $c = \text{argmax}_y \prod_{i=1}^{d} P(x_i|y)\cdot P(y)$.

We will explain how to estimate the class conditional probabilities. We apply the naive Bayes approach to text data. When it is applied particularly to the text data, the probability $P(x_i|y)$, describing the probability that the word (feature) w_i occurs x_i times, provided that $\boldsymbol{x}$ belongs to class y, is estimated (Mitchel, 1997)

$$P(x_i|y) = \left(\frac{a\cdot p + N_i^y}{a + N^y}\right)^{x_i}. \quad (3)$$

where a is the equivalent sample size parameter, p is the prior probability, N^y is the total number of words in all documents in class y, counting duplicate words multiple times, and N_i^y is the number of times the word w_i occurs in all documents in class y. Now we consider a binary classification problem using the naive Bayes. That is, Y is either +1 or -1. It was shown that the binary naive Bayes classifier is a linear classifier (Brank et al., 2002; Chakrabarti et al., 2003; Ng and Jordan, 2001; Rennie, 2001). The naive Bayes classifies $\boldsymbol{x}$ as +1 if $P(+1|\boldsymbol{x}) > P(-1|\boldsymbol{x})$, and otherwise -1. In our experiment, we have chosen the $a\cdot p = 1$ where a = total number of features (unique words) in the training samples.

3. Text datasets and preprocessing

We will show how the text document can be represented as a dataset suitable for supervised learning. For the document data to be available for a supervised learning such as the naive Bayes or SVM, the document is first converted into a vector space notation (also called "bag-of-words"). For a more compact dataset without any information loss, stop-words are removed. Such stop-words are "then," "we," "are," and so forth. Again, different forms of the same word root are processed by the stemmer program. The stemmer program converts each word to its root form. We used a stop-words list which contains 571 such words, which is available from the Internet (http://www.unine.ch/Info/clef/). For the stemmer program, we used the Porter's stemming algorithm. The Java version of the algorithm is publicly available (http://www.tartarus.org/~martin/ `PorterStemmer`). The resulting matrix converted from the documents is very high-dimensional and sparse.

For our preliminary experiment, we have collected three different text document datasets: textCL, textAB, and textCD. All three datasets were generated from PubMed (http://www.pubmed.org). By entering the keywords "acute chronic leukemia," we collected 30 abstracts and by entering "colon cancer" 30 as well. This dataset is called textCL. For the textAB, two sets of keywords are "functional genomics OR gene expression" and "structural genomics OR proteomics." For the textCD, the keywords are "HIV infection" and "cancer tumor."

After collecting the documents, each document was converted into a so-called "bag-of-words" representation. The conversion steps into a bag-of-words representation are shown as follows:

3.1. Steps for bag-of-words representation of text

1. Identify all the words in the 60 documents and make a list of words.
2. Remove ***stop-words*** such as "a," "the," "and," "then," and so forth from the list.
3. Apply the ***stemming*** algorithm to each word in the list. The ***stemming*** leaves out the root form of the words. For example, "computer," "computing," "computation," and "computes" all have the same comput root.