TJHSST Computer Systems Lab Senior Research Project
Naive Bayes Classification
2008-2009

Christina Wallin

October 30, 2008

Abstract

One part of computational linguistics is the classification and comparison of texts into classes. This comparison could be author classification, spam filtering, or in the case of this project, the classification of texts from different genres of news stories. The goal of the project is to be able to classify a text as belonging to one of two classes based on the actual words in the text. This method of classification is called a naive Bayes classifier. In this age of huge amounts of data available online, a classifier which could discriminate between two types of news could be increasingly useful.

Keywords: naive Bayes classifier, computational linguistics, multivariate Bernoulli event model, Laplace smoothing

1 Introduction - Elaboration on the problem statement, purpose, and project scope

1.1 Scope of Study

The project will first be to make a program which generates a probability vector for each class, which measures the probability that a vector (in this

case a word) is in a member of that class. In order to do this, a list of all of the words in each text and their relative frequencies needs to be made, and then a global frequency table needs to be created for each class of texts. The next part of the naive Bayes classifier makes the assumption that all of the vectors are independent, and so if you know the probability that the vectors are in a specific class of text, given another text you can figure out the class given a list of variables (words). In this way the class of an unknown text can be determined. To start out with, I am writing the first section, and hopefully for second quarter I can complete the second part. In latter quarters I hope to be able to optimize the algorithm by making a multivariate Bernoulli event model (explained later) and discriminating words based on stems instead of the actual form by using a stemmer (probably the Porter stemmer)–e.g. walking and walked would be counted as the same word–and perhaps, if time permits, coding another classification method in order to compare its effectiveness to that of the naive Bayes.

1.2   Expected results

My prospective results would be the number/percentage of texts classified correctly. The 20 Newsgroup database is divided into training and test sets, so I will be able to use that as a constant for training and testing differnet versions of the program. The results would be comparing the effectiveness of the method for two different sets of genres of news (with the data coming from the 20 Newsgroups) and later comparing the effectiveness of the naive Bayes classifier with one using the multivariate Bernoulli model or another classification system. This study would add more data as to whether or not the Naive Bayes classification system, sometimes even called the "Idiot's Bayes," is effective compared to other methods.

1.3   Type of research

My research is use-inspired basic research. It is done to achieve fundamental understanding concerning the effectiveness of the naive Bayes as compared to other methods. However, it is also done to try to provide a way for news stories from different genres to be classified. In this age with huge amounts of data popping up on the internet every day, an effective way to sift through the texts with classification will be essential.

2 Background and review of current literature and research

There have been many studies which have used the naive Bayesian classification method, for the method itself has been around for a very long time. The first paper that made the Bayesian assumption that all of the variables were independent was written in 1966: "Construction of a model for computer-assisted diagnosis: an application to the problem of non-toxic goitre" by Boyle et al. A reference to the previous paper was found in the paper "Idiot's Bayes–Not So Stupid After All?" by David Hand and Keming Yu, which using theoretical and real data situations shows that the Naive Bayes is not a horrible method because of its false assumption that all of the variables (in my case, occurances of words) are independent. This paper is essentially a review of past uses of naive Bayes and the conclusions of those researchers and a theoretical treatise as to why the naive Bayes is effective. Most of the studies talked about in this paper are medical diagnosis studies, and so text classification is a slightly different way of approaching naive Bayes classifiers. /par Another element of this project is the fact that since it deals with the actual text of the words, it is also computational linguistics. So, it is actually the intersection of linguistics and machine learning. For a different linguistic classification problem, dialectology, or the study of how language varies over geographical regions, the study "Linguistic Variation and Composition" by John Nerbonne details how computational methods can be used to more effectively study language variation. One method which was used to increase the effectiveness of the computational linguistic techniques used was the Porter stemmer, which when given a string gives back its stem. The stemmer extracts the essential information out of a string and allows words which are just forms of the same word (e.g. stops, stopping) to be considered as such. This stemming eliminates morphological data, but it helps with understanding the lexical data, and for text classification the morphological part is inessential. Thus, I could see whether or not using a Porter stemmer would help with classification. I was not able to find any other uses of this stemmer, so perhaps this experimentation would be original research.

3   Procedures and Methodology

For this first quarter version, I have focused on researching how to make a naive Bayes classifier and creating the first section of it. The first quarter version reads in a text and parses the string to remove all punctuation and capitalization. Then, it creates a dictionary of words occurring in that text and their frequencies. This step is in anticipation of the next part of the program, which is to create a probability vector (PFX) for each class. This first model is not multivariate, which means that it does not take into account multiple occurances of a word in a document. So, in this case, the probability vector is a dictionary of all of the words occurring in the test cases with the probability for each variable (word) the number of times the word occurs in a text of the class divided by the total number of texts in the class. Then, 1 needs to be added to both the top and the bottom, so that there is never a probability of zero that a term occurs. That process is called Laplace smoothing, and it accounts for words which are not found in the training cases but may be found in the test cases. Thus, for each individual text, the probability vector is binary, with a 1 if the word is in the text and a 0 if it is not.

Then with this probability vector for each class, by Bayes' theorum which states that each vector is independent, you can calculate the probability that a text is of a specific class by generating the probability vector for that specific text and comparing it to the PFX for each class. For this, each variable (i.e. the occurance or non-occurance of each word) has to be calculated in order to form the probability that it is in a particular class. Then, the individual probabilities would usually be multiplied in order to determine what the probability would be. However, the multiplication can sometimes have problems, and so it is better to take the sum of the logarithms of the probabilities that the variable is an element of the class.

Once the second part of the project is completed and the program can convert from the PFX to the probability that a given text is in a class, the basic naive Bayes is complete. It can then be tested with different genres of news (see testing below). An extension is the multivariate Bernoulli event model, which essentially is the same thing except it takes into account the frequencies of the words within the texts themselves.

The programming language used for these manipulations is Python. An aid for the linguistic sorts of operations is the nltk package, the Natural Language Toolkit. This package provides capabilities like the Porter stemmer

programmed into it, and also has methods for displaying the data with the PyLab package. The PyLab package is a method of making graphs and charts, very much like matlab. Visuals can be made of the probabilities that selected words occur in each class, as compared to a text to be classified. With the multivariate model, a graph of selected frequencies of words in the sample text would also be beneficial. These charts would help the viewer understand the processes the computer is going through as it compares the text and the probability vectors. However, the actual probability that a text is in a specific class is also significant, because it represents the degree of confidence the program has. The average number could be displayed as a pie chart along with the percentage of texts correctly classified.

As far as testing goes, the 20 Newsgroup database from which the news stories come is separated into training and testing sets. The testing sets come in folders with the correct class in the name, and so with Python methods to check the folder from which the text came whether or not the class was correctly determined can be showed. However, the fact that the program produces the correct class does not mean that the algorithm is working correctly. So, the tests cannot solely be based on that.

One way to test the program is to test the components. You can see whether or not the probability vector is determined properly for a very small data set, and if it works on a small scale it should also work on a large scale. The naive Bayes itself can be tested with specific small data sets. One example for the multivariate model from http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html contains only 4 training texts and one testing text for a total of 14 words, but it can classify certain words correctly.

The requirements of naive Bayes program alone are a class-like structure for each text, a program which calculates the probability vector for each class and is able to update it as the training cases progress, and a program which, when given a text, can classify it. The second two programs would probably have to be separate, since they need to run separately and both in the final version would probably take a good deal of time to run, but the PFX would need to be saved somehow, perhaps via a text file. If it has these things, and it is able to classify the 20 Newsgroups data relatively accurately, the program is rather successful. Then, it could be modified to be multivariate and/or to use the Porter stemmer, and the relative effectiveness could be compared.

For the most part I describe the algorithms above, but one thing which

needs to be clarified is the assumption central to the naive Bayes classifier–Bayes' Theory. This theory states that all of the variables are independent in their determination of the class. In this case, the variables are individual words. The assumption is obviously false, because certain words usually occur together. For example, in a news article in the genre of news about space, if the word 'rover' occurs, the word 'Mars' is more likely to occur. However, even though the assumption is false the naive Bayes classifier is a rather effective tool for classification.

4   Expected Results

I expect to get a percentage of test cases in which the Naive Bayes classifier is successful after being trained with the training cases. Using Pylab, I can plot the percentage of test cases successful and failed for classification as a pie chart. I can also give the reader a view of what is happening with the algorithm by plotting the probability of selected words from each class, which is made with the training data. On this plot I can also include whether or not that word is present in a sample test case. The plot of the most probable words would also gain insight as to what words are key to each class.

My prospective results could help other researchers know the relationship of the naive Bayes classifier to the multivariate Bernoulli event model and perhaps another classification system. They would be able to build upon this information by potentially doing another classification method (perhaps using neural networks) and using the same testing and training data and comparing the results.