Naive Bayes Classification for Sentiment Analysis of Movie Reviews:

Machine Learning

Elif Ilaria Yurtseven, Mazen Abusharkh

02/26/2017

Outline of the Report:

Abstract ………………………………………………………………………………………………....……… 2

1. Introduction …………………………………………………………………………………….……… 2
2. Theory & Background …………………………………………………………………………....…… 3
3. Design ………………………………………………………………………………………………… 4
   1. Figure 1 Illustrative diagram of the program ………………………………………....……… 4
   2. Figure 2 Code for frequent words …………………………………………………....……….. 5
   3. Figure 3 Table for prediction-reference output …………………………………….…………. 6
4. Results ……………………………………………………………………………………....………….. 6
   1. Table 1 Bag of words with 150 frequency …………………………………….…………… 6
   2. Table 2 Bag of words with 50 frequency ……………………………………..……………… 7
   3. Table 3 Bag of words with 10 frequency …………………………………………..………… 7
   4. Table 4 Bag of words with 5 frequency …………………………………………....………… 7
5. Conclusion ……………………………………………………………………………...……………… 8
6. Reference …………………………………………………………………………….………………… 9

***Abstract***

**The purpose of this program is to predict whether a movie review is positive or negative based on the kind of words that would appear in either type of review. The program is given a specific number of training data (1000, 1200, 1400 or 1600 reviews) that is already classified as positive or negative. Each word is taken as an input and the program checks the frequency of that word in relation to its reviews’ classification. After the training, the program will be able to generalize its experience on the different reviews of that same movie. Moreover, if the program is later developed and trained with more generalizable movie reviews it would be able to predict the sentiment of the different reviews of different movies. The movie used for training and testing the program is Pang and Lee, and the reviews are collected from IMDB movie reviews.**

1. Introduction

When looking at a movie review, one cannot necessarily determine whether the review has a positive or a negative tone without looking at the number of stars the reviewer provided. Some writing critics are trained to determine the specific tone and opinion of the writer, however, that takes a lot of education and analysis. With this program, we aim to develop a classifier that will classify a review as positive or negative by looking at the frequency of the words in both types. A human being cannot read through thousands of reviews in order to decide whether to watch a movie or not. However, as we will explain, an artificial intelligence machine could use its processing power to determine the tone in a review and the sentiment of that review. A well-developed program will have the machine learn how to determine the tone or language in a review quantitatively based on the earlier training reviews. We developed a program in R that reads through the desired number of movie reviews and by separating them as training and testing reviews, and by using Naive Bayes classifier our program was able to separate reviews into positive and negative with high accuracy levels.

1. Theory & Background

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem, which describes the probability of an event based on prior knowledge of conditions that might be related to the event, with strong (naive) independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used by many other types of classifiers.

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from a finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable.

Naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

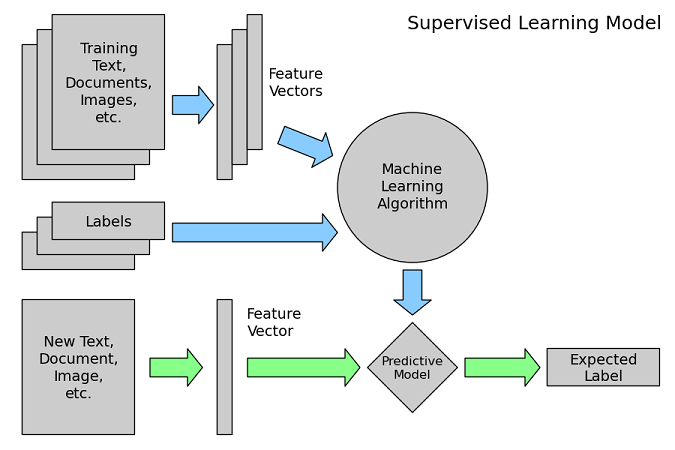
On the other hand, Bayesian classifier minimizes the probability of misclassification. A Bayesian classifier is based on the idea that the role of a (natural) class is to predict the values of features for members of that class. Examples are grouped in classes because they have common values for the features. Such classes are often called natural kinds.

The idea behind a Bayesian classifier is that, if an agent knows the class, it can predict the values of the other features. If it does not know the class, Bayes' rule can be used to predict the class given (some of) the feature values. In a Bayesian classifier, the learning agent builds a probabilistic model of the features and uses that model to predict the classification of a new example.

1. Design

R is a statistical computer science language that allows easy manipulation of values when given in the form of an excel document. This is why we made use of R and its internal libraries to write a code that can check an excel file given reviews and their class. The program determines the accuracy of the prediction of the system when it tests.,

The below diagram is a general overview of how our supervised model works. The training text is the 2000 reviews with their categories (positive and negative). The feature vectors are the bag of words that we very and apply onto the testing data. The labels are the classification of the training reviews as ‘pos’ and ‘neg’ and the result is our predicted model.

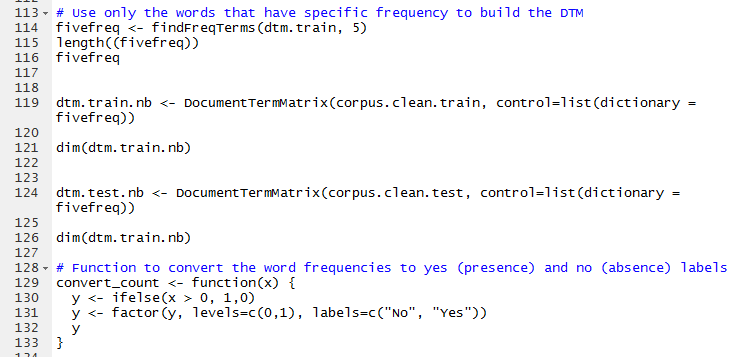


*Figure 1: Illustrative diagram of the program*

The input file was a CSV document (excel) sorted into two as reviews in one column and their classification as positive or negative in the other one. The total data is then randomized amongst itself. Once the data is ready, we split it such that the data is split into two parts, a part for training and another for testing. In order to make sure that we know which words to look for in the reviews, the system first determines which ones are the words that are most frequent in the reviews. Once the system knows which words to look for, it starts reading through all reviews and classifying the various words according to their frequencies in the negative and positive reviews separately.

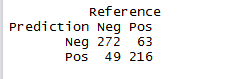
Once the entire training data is read through, the system has an idea of whether a word is more likely to be in a positive review or a negative one. Hence, once we test additional data, the system knows which class of review it is more likely to be according to the words that it knows. Our algorithms make sure that the system can determine for each word whether the existence of the word gives a higher possibility for the review to be positive or negative. The way we know which words to look for is by giving a lower bound to the number of times that word appears in the training data reviews.

The below code snippet shows how we find the most frequent words in the training data and the function use to checks whether each of those words is in the reviews in the testing data:



*Figure 2: Code for frequent words*

Once the data is finished testing, the program outputs a table of the predicted results versus the real results of each review in the testing data:



*Figure 3: Table for prediction-reference output*

From the above table, we are able to calculate the accuracy of our algorithm. This calculation is done by the division of (true positive + true negative) by (false positive + false negative).

We have tested the data multiple times by separating the ration of training and testing data as well as the bag of words. The bag of words in this program consists of the most frequent words. The frequency desired is determined by the programmer. We tested various frequencies in order to reach the maximum accuracy.

1. Results

*Table 1: Bag of words with 150 frequency*

|  |  |
| --- | --- |
| Training/ Testing Ratio | Accuracy |
| 1:1 | 0.728 |
| 3:2 | 0.73125 |
| 7:3 | 0.765 |
| 4:1 | 0.76 |

*Table 2: Bag of words with 50 frequency*

|  |  |
| --- | --- |
| Training/ Testing Ratio | Accuracy |
| 1:1 | 0.781 |
| 3:2 | 0.775 |
| 7:3 | 0.80167 |
| 4:1 | 0.8025 |

*Table 3: Bag of words with 10 frequency*

|  |  |
| --- | --- |
| Training/ Testing Ratio | Accuracy |
| 1:1 | 0.789 |
| 3:2 | 0.79375 |
| 7:3 | 0.8133 |
| 4:1 | 0.805 |

*Table 4: Bag of words with 5 frequency*

|  |  |
| --- | --- |
| Training/ Testing Ratio | Accuracy |
| 1:1 | 0.792 |
| 3:2 | 0.79875 |
| 7:3 | 0.8133 |
| 4:1 | 0.8 |

1. Conclusion

Looking at the results we can see that the most accurate results are when we have the bag of words with frequency 10. This value makes sense because as the necessary frequency is lower, there are more words that the program can use and find in the testing data. However, as the value goes too low, i.e 5, the program needs to go through most of the words that exist in the training data in the first place. And not all of these words are relevant in determining whether a review is positive or negative.

As we increase the ratio between the training and the testing data we can see that the accuracy increases between the 1:1 ratio up until 7:3. However, it gets lower for 4:1. This result is reasonable because as the amount of training data increases, the results will be more accurate. However, after a certain value, the algorithm would be too variant to have the same or higher accuracy than the 7:3 ratio for example.

In conclusion the most accurate result to predict if a review is positive or negative using Naïve Bayes classifier in R with our program is by having 7800 words out of 35634 (that have frequency in training data reviews of 10) and when the training/testing ratio was 7:3 (which is 70% training data and 30% testing data).

Considering the amount of programming and testing that we needed to do in order to optimize our accuracy for the testing data, we can think that we could have written a program which processed the same information by using Bayesian classifier rather than Naïve Bayes one. The way we could have developed such program would have been by adding weights and conditional dependency with certain words such as “and”, “not” etc. These words could change the meaning and the tone of a sentence according to words around them. For example, while “bad” could be a very negative word for a movie review, “not bad” could be a neutral word. This would affect both the class and weight of the review the word “not” is in.

1. References

Class Notes

<http://www.r-tutor.com/>

http://www.allprogrammingtutorials.com/tutorials/introduction-to-machine-learning.php