STAT 5630: FINAL PROJECT

SENTIMENT CLASSIFICATION ON MOVIE REVIEWS

Xin Jin & Frank Woodling
University of Virginia
5/5/16

I. Introduction

The problem we are addressing is a sentiment analysis problem with the objective of classifying movie reviews. The data was taken from IMDB[3] (International Movie Database) website. The goal is to be able to predict reviews depending on the text and words used in the reviews. We will explore different machine learning methods in order to try to find the best method of review prediction.

II. Dataset

The original dataset was collected by Pang and Lee in 2004, two of the earliest proponents of Sentiment Classification. We used a modified dataset built for this specific Kaggle competition that was further modified for our own use. The data is composed of 25,000 movie reviews for the training set, and another 11,000 for the testing set. IMDB reviews are rated from 1-10, with 10 being the best. The reviews do not have a length limit, and many reviews are very long.

The data we obtained is classified as either a negative or positive review. Ratings that are less than or equal to 4 are deemed negative reviews, and for those whose rating is greater than or equal to 7 are thought of as positive reviews, which is the personal sentiment/attitude toward the movie. For the purposes of this project we will split the given training data into our training and testing sets. The way the contest is set up we would get no information from submitting our results. All other aspects of our work will be the same as the contest. We will end with a vector of 1s and 0s to compare with the correct predictions in order to find out how many we were able to predict correctly.

III. What is Sentiment Analysis?

Sentiment analysis is a subset of natural language processing. ”Natural Language Processing (NLP) is the computerized approach to analyzing text that is based on both a set of theories and a set of technologies. Natural Language Processing is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications.” [5]

The field of sentiment analysis is relatively new, but it becoming an increasingly active field. The paper, *Thumbs up? Sentiment Classification using Machine Learning Techniques*[6] written by Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan was published in 2002, and marked the birth of research in this field. Note that Pang and Lee data mined the original dataset for this IMDB movie review data. In recent years, many new methods have been developed such as Latent Dirichlet Allocation in 2003 and Vector space models and 2010.

The methods used for sentiment analysis will allow us to take our movie reviews, and decide which words are likely to be a positive or negative review. For example, words such as ”good” or ”bad” will give us much more information than words like ”a” or ”the”. In practice it is more complicated as there can be tens of thousands of words, so we will need to pick the most predictive features.

IV. Data Preparation

The most difficult part of this project was getting the raw data to a usable form. The 25,000 reviews were each in a different text file. The reviews contained many different characters, symbols, emoticons, punctuation, and of course spacing. Here are some examples of the raw data:

1) Another good Stooge short!Christine McIntyre is so lovely and evil and the same time in this one!She is such a great actress!The Stooges are very good and especially Shemp and Larry!This to is a good one to watch around Autumn time!
2) If only to avoid making this type of film in the future. This film is interesting as an experiment but tells no cogent story.One might feel virtuous for sitting thru it because it touches on so many IMPORTANT issues but it does so without any discernable motive. The viewer comes away with no new perspectives (unless one comes up with one while one's mind wanders, as it will invariably do during this pointless film). One might better spend one's time staring out a window at a tree growing.

The two examples show both a positive and negative review. As a human it is easy to tell what they thought of the movie, but the goal of natural language processing and sentiment analysis is to find a way for the machine to make sense of those words. Looking at the examples we can see typos, proper nouns, and differing spacing/formatting. We have to find a way to normalize this data.

The first step was to read the data in. We created a cell array that included each file name as a different element. Next we went through each text file, and added each word as a different element chosen by gaps in spacing. Each text file or review is a column of the cell array. The data then needed to be normalized, so we used a function to remove everything that was not a letter. The data was then all converted to lower case. This is because we expect ”good”, ”Good”, and ”GOOD” to be considered the same.

The data is then tabulated separately for the negative and positive training data. This allows us to greatly reduce the size of the dataset even though it is still quite large. The tabulated array includes each unique word, the frequency of the word, and the probability of that word occuring. After combining the positive and negative tabulated arrays we can compare them.

Our main strategy for choosing features was the ratio of frequencies between negative and postive reviews of the same word. A word such as ”a” or ”the” may occur many more times than other words. This does not necessarily mean that those words are stronger predictors though. Those words should occur in positive and negative reviews at roughly the same frequency. We need words that are stronger indicators. Our idea was the larger the difference between the postive freqency and the negative frequency of a word the better feature it would be. Setting the threshold for a larger reduces the total number of features, however, they should be stronger features. Another benefit of a smaller feature set is that it runs much faster. We experimented with different sized features sets that used varying frequency ratio thresholds to see what has the greatest effect.

V. Method 1: Support Vector Machines

The first method we used is a support vector machine method. An SVM uses a hyperplane to classify data into groups. In our case we wanted to classify data into either a positive or negative group, though you can have more groups with SVM. The first usage of SVM used a nonseparable case defined by:

$$\max_{\lambda,\mu} \sum_{i=1}^{n} \lambda_i - 0.5\lambda^T Y^T X X^T Y \lambda \text{ such that:}$$
$$\sum_{i=1}^{n} y_i \lambda_i = 0$$
$$0 \leq \lambda \leq C.$$

Our first SVM test was a test without intercept. SVM without interception is one which has been normalized. It is easier to manipulate and compute. Our correct prediction percent turned out to be 0.5040 so this was not a good result at all. If we flipped a coin we could get around 0.5 prediction percent. We next tried another SVM method, this time with a linear kernel using a Matlab built-in function. We achieved a 0.8128 probability of a correct prediction, or just over 81%. This is a good result, and much better than the previous SVM test.

VI. Method 2: Decision Trees

The other method we used involved decision trees. "A tree-based method involves stratifying or segmenting the predictor space into a number of simple regions. The set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as decision tree methods." [1] We will start with a type of decision tree called bagging. Bagging utilizes a bootstrap method along with a decision tree. Bootstrapping will reduce the variance of a method. The equation for a bagged decision tree is:

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$$

We have B training data sets. "We then train our method on the *b*th bootstrapped training set in order to get $\hat{f}^{*b}(x)$, and finally average all the predictions." [1]

In order to perform the bagging decision tree test we first created an array of features. The features were selected depending on the ratio between the ratio between the postive and negative training reviews. We tried the tests with different ratios. A threshold with a higher minimum and lower maximum would select less features, as fewer of them would meet the requirements. It may seem that having less, but more powerful predictors may give us a greater chance of predicting correctly, however, we had more success with more features even if they were not considered as strong.

The next step is to create a giant array of all training sentences and all words. We want to know how many times a word is present in each sentence. We also need to count up the words in the same way with the testing sentences. The test can then be ran, resulting in vector of 1s and 0s depending if it predicted a positive or negative review. We can then compare our predictions to the actual predictions in order to find out what percentage we were able to correctly identify.

One last method we tried with decision trees is to run tests on different sample sizes. We tried using only 10,000 training sentences (5000 positive, 5000 negative), and also with 20,000 training sentences. The number of features stayed the same, but we should presumably get better information on frequencies from a greater sample size.

The table below lists our results using the decision tree bagging method. The greatest prediction rate we achieved was with a fuller feature set. We were able to attain a percent of 84.08% with the 7990 feature test. Interestingly the lesser sample test did slightly better, but they are very comparable. The prediction percent seems to almost go down linearly with the number of features used. The sample size does not make a huge difference like the number of features does. It is impossible to say exactly why this is the case, but one could speculate that 10,000 sentences gave enough information. Many of the movie reviews use the same vocabulary, and there are only so many ways used to describe and review a movie.

10,000 Sample Test	Features	Ratio Thresholds	Correct Prediction %
1	7990	≥ 1.5 or $\leq 1/2$	0.8408
2	2212	≥ 3 or $\leq 1/3$	0.7936
3	1543	≥ 3.5 or $\leq 1/3.5$	0.7824
4	1120	≥ 4 or $\leq 1/4$	0.7696
20,000 Sample Test	**Features**	**Ratio Thresholds**	**Correct Prediction %**
1	7990	≥ 1.5 or $\leq 1/2$	0.8360
2	2212	≥ 3 or $\leq 1/3$	0.8006
3	1543	≥ 3.5 or $\leq 1/3.5$	0.7856
4	1120	≥ 4 or $\leq 1/4$	0.7700

TABLE I

BAGGED DECISION TREE ATTEMPTS

VII. Results

The best result we achieved was with decision trees using a large feature set. We ended up with a 0.8408 probability of correctly deciding whether a movie review is negative or positive. As mentioned before the size of the feature set was the main factor in the prediction power. It would be interesting to see how different feature selection methods would change the prediction percent if we had more time.

One thing to keep in mind is that a perfect 100% prediction is never going to happen. Due to the nature of movie reviews anything over 90% would be very difficult. For comparison, in the Kaggle competition we modeled this project after not one person on the leaderboard achieved a percent over 90. Each reviewer has a different idea of how to rate a movie. Terms such as ”not good” or ”not bad” are deceptive as most of our weighting will come from ”good” or ”bad” ignoring the not part. It may be possible to scan for certain phrases like that, however, it could take a very long time to scan the data in this way.

Our best SVM test was close to many of our decision tree attempts. The best SVM is 0.8128 probability of a correct prediction with the SVM test using a linear kernel. The SVM test using no intercept did not yield a good result so we moved on to different methods.

Decision trees seem to be built for this type of test. Given more time and a more powerful computer we could definitely improve on our prediction with the bagging method. We would simply adjust our features, or find a different way to rank them other than with ratios.

VIII. Conclusion

The greatest difficulty with the project was purely the size of the dataset. It takes a lot of calculations to perform any of these tests. This makes the tests take extremely long, and completely eliminates some machine learning methods from being used. If we had more time we could attempt to find a better way to run those tests.

We could also come up with a different system to rank the strength of each word. There have been similar tests ran before where the testers used only a handful of words in order to try to predict a review, Twitter sentiment, etc. For example some use a dictionary or listing of words. You can make these the strongest words you can think of, and easily test each word. It would be interesting to try to find the best possible words to predict a review. The statisticians that wrote the paper *Learning Word Vectors for Sentiment Analysis.*[4] also used a synonym system in order to predict IMDB ratings. There are many possibilities, and it is hard to say exactly how good a method will be before trying it.

REFERENCES

[1] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, *An Introduction to Statistical Learning, 1st edition*. New York: Springer, 2013. Print.

[2] UMass Lowell. *"Sentiment Classification on Large Movie Review."* Kaggle in Class. N.p., 24 Feb. 2016. Web. 10 Apr. 2016.

[3] IMDb. IMDb.com, n.d. Web. 15 Apr. 2016.

[4] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). *Learning Word Vectors for Sentiment Analysis.* The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[5] Liddy, Elizabeth D. *"Natural Language Processing."* Syracuse University Surface. 2001. Web. 1 May 2016.

[6] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs Up? Sentiment Classification Using Machine Learning Techniques." IJSR International Journal of Science and Research (IJSR) 5.4 (2016): 819-21. Web.