Data Science Decal: Project 2 - Sentiment Analysis

Machine Learning at Berkeley

October 10, 2017

1 Introduction

Sentiment Analysis is a popular Natural Language Processing (NLP) task which allows us to extract the overall opinion in a text. In this project, you will be performing Sentiment Analysis on some IMDB movie reviews, to classify the overall review as positive or negative. When dealing with text data, a prevalent issue is how to encode the words as a numeric feature that can be used to compute the output of a classification algorithm. Especially because words don't naturally lend themselves to a numeric ordering, there have been many approaches on how to featurize a text. In this project, you will use the bag of words model, which uses the count of a word in a text as a feature. You will begin by using logistic regression to perform this task, followed by a decision tree approach.

2 Getting Started

Download the dataset and the iPython notebook from the Github page

3 Reading in Data

You are given the function, `readFile(filename)`, that will read the contents of a text file and output the contents of the file as a list containing single words. Using this function, read all the files into a Pandas Dataframe, in which each row represents a text file and the columns contain the counts of each word in that specific text file. Note that there should be a column for every possible word that occurs throughout all text files, so all the columns together form the unique vocabulary for the dataset. If a word does not appear in a particular file, let its count be 0. In addition, add a column in this Dataframe with the document label, containing either the value 'positive' or 'negative'. You may also want to add a column with the file name, so you can later check which reviews were incorrectly classified.

3.1 Training/Validation Split

Because you don't have access to the labels of the test set, randomly shuffle the dataset and split the data into a training set and validation set, so you can test your trained model on the validation set. In general, even if you have access to the labels of the test set, it is a good idea to use a validation set to prevent overfitting to the test set. (Hint: Use `train_test_split` from `sklearn.model_selection`)

4 Logistic Regression

Train a basic logistic regression model to classify the sentiment of the reviews. Make sure you do not use the filename as a feature if you previously included it in the Dataframe. Compare the accuracy of this basic model on the training set and the validation set. Are you overfitting? Try changing the parameters of the logistic regression, such as adding a regularization term, to reduce the overfitting.

4.1 Backward Stepwise Selection to Reduce Features

Overfitting mainly occurs if the model is too expressive for the given task. As a result, it is able to not only fit the pattern in the training data, but also the randomness of the dataset, which thereby causes a high accuracy on the training data and a low accuracy on the test data. One way to reduce the expressiveness of the model is by reducing the number of features. Currently, we used the count of every word present as a feature, but perhaps some words are more indicative of the sentiment of the review than others. Those other words, which don't contribute much to determining the sentiment, may be overfitting to the noise in the training set. One way to identify these features is to look at the features whose weights are close to 0 (remember to normalize the weights if using this method). This process is called Backward Stepwise Selection, which aims to remove the features whose removal would reduce the test error. Use Backward Stepwise Selection or any other method of removing unnecessary features so the accuracy on the validation set increases. Compare the new and old accuracies. Write a few sentences about the method you used and why it helped reduce overfitting. You can find some more methods of feature reduction on page 73 of this resource (https://people.eecs.berkeley.edu/ jrs/189/lec/13.pdf). Feel free to try out any other ideas you may have to reduce overfitting as an extra challenge and provide a description of your method and intuition.

5 Decision Trees

Train a standard single decision tree for this problem. Compare the accuracy of this basic model on the training set and the validation set. Are you overfitting? Change some of the parameters of the decision tree to reduce the overfitting and write a few sentences on why the change you made reduced the overfitting. Show how you came to the final value of that parameter. For example, if you changed the maximum depth of the tree, show a plot of how the training accuracy and validation accuracy changed as the maximum depth of the tree changed. You may notice that despite these changes, the validation accuracy is still much lower than the training accuracy. Write a few sentences on why the single decision tree is so prone to overfitting, especially for this dataset.

6 Random Forest Decision Trees

Train a random forest decision tree and compare the accuracy on the training set and validation set. Try changing some of the parameters of the random forest classifier and write a few sentences on why you chose to change those parameters and why they improved the validation accuracy. In addition, write a few sentences on why a random forest classifier does a better job at preventing overfitting than a single decision tree classifier.

7 Submission

Please submit a writeup including an analysis of how linear regression, single decision trees, and random forest decision trees performed on this dataset. Include answers to the questions in the above sections, and feel free to describe any other observations you made during this project. In addition, please submit a pdf of the iPython notebook.