MSAN 621 - Homework 2

Andre Guimaraes Duarte

November 10, 2016

For this assignment, we implemented four different machine learning classification algorithms:

- Logistic Regression
- Linear Discriminant Analysis
- Quadratic Discriminant Analysis
- k Nearest Neighbors

Here, the data set comprises of a certain number of tweets, as well as their description based on sentiment analysis: one of {negative, neutral, positive} according to the perceived sentiment of the text in the tweet.

In order to implement said machine learning algorithms, we first need to create a set of features for each tweet. The objective is to predict the sentiment based on these features. Performance of the classification is assessed by the misclassification rate (lower is better).

Feature 1

The first feature is the rates of 20 English "function words": {"I", "the", "and", "to", "a", "of", "that", "in", "it", "my", "is", "you", "was", "for", "have", "with", "he", "me", "on", "but"}.

There are 20 words that we are interested in. So this feature accounts for 20 columns in our feature space: each column represents the rate of said word in the tweet.

Feature 2

The second feature is the rates of 3 punctuation symbols: {".", ",", "!"}

There are 3 punctuation symbols that we are interested in. So this feature accounts for 3 columns in our feature space: each column represents the rate of said punctuation symbol in the tweet.

Basline model

Taking these two features, we can create our baseline model. We achieved the following baseline misclassification rates (1 minus cross-validation scores):

- Logistic Regression: 0.36 ± 0.01
- Linear Discriminant Analysis: 0.36 ± 0.01
- Quadratic Discriminant Analysis: 0.50 ± 0.14
- k Nearest Neighbors: 0.41 ± 0.03

Additional features

In order to try to improve the accuracy of the algorithms, we can try to add additional features. We have to be careful with overfitting though.

The extra features added here were:

- the rate of occurrence of the 1000 most common words/tokens used in the training set

- the total length of the tweet in number of words/tokens
- the rate of occurrence of a short list of common positive/negative words (such as *awesome, great, fantastic, horrible, terrible...*)

Improvement (or worsening) over the baseline

By adding these extra features, we obtained (on the same train and test sets), the following scores:

- Logistic Regression: 0.31 ± 0.02
- Linear Discriminant Analysis: 0.27 ± 0.02
- Quadratic Discriminant Analysis: 0.54 ± 0.16
- k Nearest Neighbors: 0.41 ± 0.02

We can see that the accuracy improved significantly for Logistic Regression and LDA. For QDA and kNN, the accuracy either improved little or even got worse. This can be explained by overfitting. Since boundaries can be non-linear for these last two algorithms, there is a higher chance of overfitting, and it seems like this is what happened for this specific test and train sets. We can see that linear bounds are less prone to this behavior, as seen by the significant improvement in accuracy for the first two algorithms.

Best classfier

From these results, it seems that LDA and Logistic Regression perform better than the other two classification algorithms. As explained previously, this is due to the fact that non-linear classification methods are more susceptible to overfitting, which may be what happened here. Indeed, we can see that the variability in the CV score for QDA is especially high, meaning that changing the train and validation sets changes the result significantly.

In general, LDA was the best classifier here, and it saw an improvement of around 25% over the baseline by adding extra features.

Further explorations

To improve the performance of the model, we could imagine using a TFIDF algorithm to find the relative importance of words in a tweet, and classify those as *positive, neutral, negative.* Each tweet would be equivalent, in this case, to a document. This would possibly be faster and more reliable than the approach used in this exercise.

Running the code To run the code, use the following syntax:

```
python misclassification.py "<train-data.csv>" "<test-data.csv>" <algorithm> <number of
neighbors for knn (optional)>
```

where `<algorithm>` is one of `{logit, lda, qda, knn}`.