Report: Text Classification and Naive Bayes

Kobee Raveendran

September 5, 2020

1 Multinomial Naive Bayes

Below is a snippet of my code for computing the prior probabilities of each class and the likelihood estimates of each feature (a unique word). NOTE: This code may not be completely safe to copy-paste, as some whitespace characters may be ignored.

```
# count the number of occurrences of each class in training set,
# convert into probability
prior = np.array([np.count_nonzero(y == [i]) / len(y) for i in classes])

# keep track of total number of words belonging to class c
bag_sizes = [0] * n_classes

# count category-wise occurrences of words, track num. of words per category
for doc in range(len(x)):
    c = y[doc][0]
    for word in range(len(x[doc])):
        likelihood[word, c] += x[doc][word]
        bag_sizes[c] += 1

# convert word frequencies into probabilities (with or without smoothing)
for word in range(len(likelihood)):
    for c in range(len(likelihood[0])):
        if self.smooth:
            likelihood[word, c] += self.smooth_param
            likelihood[word, c] /= (bag_sizes[c] + n_words)
        else:
            likelihood[word, c] /= bag_sizes[c]
```

2 Train/Test Split

Using the original train/test split of 80%/20%, my training set accuracy was 95.5% and my test set accuracy was 78%. After switching to a 50%/50% split, my training set accuracy slightly improved, jumping to 96%, while my test set accuracy dipped to 75.9%. Below is the modified line of code from `run_classifier.py`:

```
dataset = SentimentCorpus(train_per = 0.5, test_per = 0.5)
```