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; some tab/spacing may be required after pasting.

```
# 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

Table 1 shows the results obtained on the training set, and Table 2 shows my results on the test set, both evaluated under multiple conditions. A general trend to be noted is that using Laplacian smoothing (or perhaps any non-zero smoothing parameter) leads to better performance at test-time. An additional trend is that the size of the training set greatly affects the model's ability to generalize (or vulnerability to overfitting), as a comparison of test set accuracies between the 80%/20% and 50%/50% train/test splits show.

		Smoothing	
		Laplacian	None
Split	80/20	95.50%	98.13%
	50/50	96.00%	99.60%

Table 1: Model accuracy on the training set on various configurations

		Smoothing	
		Laplacian	None
Split	80/20	78.00%	68.25%
	50/50	75.90%	57.10%

Table 2: Model accuracy on the test set on various configurations