CS 334 — Homework 4 Solutions

Jack Anstey

Problem 1: Preprocessing + Model Assessment

a) See q1.py

b) See q1.py

c) See q1.py

d) See q1.py

Problem 2: Spam Detection via Perceptron

a) See perceptron.py

b) To find the optimal epoch of the binary and count datasets created as a part of problem 1, I implemented my own version of k-fold cross-validation in crossvalidation.py. It takes the training and test data as inputs, along with the number of folds that you want to use for each iteration and the maximum epoch that you want to test.

For both datasets, I chose to have the number of folds be 5 and the maximum epoch to test to be 20. I chose 5 folds as that is an acceptable standard, and 20 to be the max as anything higher is susceptible to overtraining and something less would likely undertrain.

After running my implementation, the results are as follows:

```
Optimal epoch for the binary dataset: 4
```

```
Optimal epoch for the count dataset: 14
```

With the optimal epoch values found for both datasets, I added another method to crossvalidation.py that uses the optimal epoch and prints out the number of mistakes when predicting the training and test sub-datasets from the binary and count datasets:

```
Using the binary dataset, the results are as follows:
Number of training mistakes: 49
Number of test mistakes: 65
```

```
Using the count dataset, the results are as follows:
Number of training mistakes: 126
Number of test mistakes: 546
```

c) The same method that I used for part b not only prints the number of mistakes, but it also returns the weights from perceptron.py. Using these weights, subtracting the bias from it first, we are able to see what the highest 15 weights are and the lowest 15 weights and the words that correspond with them. The words with the highest and lowest weights for each dataset are as follows:

```
Using the binary dataset, the results are as follows:
The words with the most positive weights: ['click', 'remov', 'pleas', 'pai', 'market', 'free', 'sight', 'your', 'hour', 'these', 'help', 'death
tospamdeathtospamdeathtospam', 'death', 'name', 'dir']
The words with the most negative weights: ['wrote', 'dn', 'up', 'which', 'rob', 'seem', 'about', 'the', 'httpaddr', 'would', 'reserv', 'version
', 'too', 'http', 'prefer']
```

```
Using the count dataset, the results are as follows:
The words with the most positive weights: ['numberc', 'report', 'numberb', 'dollarnumb', 'order', 'numberdnumb', 'call', 'anumb', 'usd', 'face'
, 'pleas', 'your', 'year', 'remov', 'numbera']
The words with the most negative weights: ['numberp', 'd', 'file', 'w', 'but', 'iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
iiiiiiiiiiii', 'user', 'pjnumber', 'messag', 'if', 'date', 're', 'razor', 'cnet', 'version']
```

Problem 3: Spam Detection using Naive Bayes and Logistic Regression

a) For this problem, I found that using MultinomialNB was the best option for both the binary and count datasets as the number of mistakes were either to or less than that of BernoulliNB. With that, the number of mistakes I found were:

```
Number of mistakes when using the binary dataset and multinomial naive bayes: 70
Number of mistakes when using the count dataset and multinomial naive bayes: 546
```

When using the count dataset, the sklearn implementation matched the number of mistakes that my own implementation had when using an optimal epoch. Surprisingly, my own implementation, when using an optimal epoch, was slightly better than the sklearn implementation (65 mistakes vs 70). I believe this is due to sklearn not using the exact optimal epoch, which I think is fine as their implementation runs significantly faster and provides nearly the same result in regards to accuracy.

b) When using logistic regression, the results were even better.

```
Number of mistakes when using the binary dataset and logistic regression: 38
Number of mistakes when using the count dataset and logistic regression: 546
```

The number of mistakes again stayed the same when using the count dataset, but the number of mistakes when using the binary dataset were nearly cut in half - only 38 mistakes were made. Overall, it seems that logistic regression and the binary dataset performs the best for this classification problem.