Homework 1

Evan Sidrow

CSCI 5622

1 K-nearest Neighbors

1.1 Programming questions

See knn.py (attached to this submission).

1.2 Analysis

1.2.1 *What is the role of number of training instances to training accuracy?*

It is clear that, when more training data is included in the model formulation, the accuracy improves. This is because the model "learns" more possible instances, and is able to find closer matches to the test data in KNN. Figure 1 shows the relationship between training set size and accuracy for $k \in \{1, 3, 5, 7, 9\}$. Note that for all k, the accuracy improves as the amount of training data increases, with the accuracy getting closer and closer to one.

1.2.2 *Which numbers get confused with each other most easily?*

When running the program with a limit of 10000 training instances and k = 3, the five most mistaken numbers were:

1. 4 confused for a 9 (3.76% of all 4's)
2. 5 confused for a 3 (2.84% of all 5's)
3. 9 confused for a 7 (2.39% of all 9's)
4. 5 confused for a 6 (2.19% of all 5's)
5. 8 confused for a 3 (2.18% of all 8's)

Interestingly, only 4's were confused with 9's, not the other way around. This is could be because many of the training 9's were messy and looked like 4's, so many 4's in the test set were close to these messy 9's. However, the 4's in the training set could have all been very clean, nice-looking 4's, so no 9's in the test set came close to these 4's.

Figure 2 shows the full confusion matrix.

1.2.3 *What is the role of k to training accuracy?*

Figure 3 shows the accuracy versus k for several different training sets. It appears as though for smaller limits, a larger k tends to correspond to a lower accuracy (i.e. $k = 1$ is ideal). This is likely due to the fact that the 1^{st} nearest neighbor of a number is much closer to that number than the 2^{nd}, 3^{rd}, etc. nearest neighbors. Therefore, as we include neighbors that are farther away, we get different numbers more easily.

Note, however, that this effect is diminished when we have a larger training set, since the 2^{nd}, 3^{rd}, etc. nearest neighbors to a test instance are closer when we have more training data. In fact, $k = 3$ is the ideal k when the limit is 5000 or 10000, indicating that the training data is closer together.

1.2.4 *In general, does a small value for k cause "overfitting" or "underfitting"?*

In general, a smaller value for k causes "overfitting". This is because for a small k the model will fit the training data extremely closely and "overfit" the training data. However, a larger k will smooth the model's fit to the training data, and in the extreme case of $k = n$, where n is the size of the training set, the model will classify every test point the same number. This is clearly "underfitting" our training data.

In terms of the Bias-Variance trade-off, a small k means that the model will *vary* greatly depending upon whatever numbers are close to it in the training set, but the *bias* of the model will be low, since we are hugging our training data very closely. However, a large k ($k = n$) means that the model won't *vary* much at all- every point will be classified as the most common number in the training set. However, this is clearly very *biased* for each point, since the most common number is not necessarily the correct number at a given point.

2 Cross Validation

2.1 Programming questions

See cross_validation.py (attached to this submission).

2.2 Analysis

2.2.1 *What is the best k chosen from the 5-fold cross validation with "limit 500"?*

After running cross_validation.py, I get the results given in figure 4. The best k appears to be $k = 3$.

2.2.2 *What is the best k chosen from the 5-fold cross validation with "limit 5000"?*

After running cross_validation.py, I get the results given in figure 5. The best k appears to be $k = 1$.

2.2.3 *Is the best k consistent with the best performance k in problem 1?*

The best k for cross validation is not consistent with the best k from problem 1. Here a the best k for limit = 500 is $k = 3$ and the best k for limit = 5000 is $k = 1$. However, in problem 1, the best k for limit = 500 is $k = 1$ and the best k for limit = 5000 is $k = 3$. This could be because the training data and the test data come from different distributions, and could also be because of random sampling error.

3 Bias-variance tradeoff

We have:

$$Err(x_0) = E((y_0 - h_S(x_0))^2)$$

and since $E(X^2) = V(X) + E(X)^2$:

$$Err(x_0) = V(y_0 - h_S(x_0)) + E(y_0 - h_S(x_0))^2$$

$$Err(x_0) = V(f(x_0) + \epsilon - h_S(x_0)) + E(f(x_0) + \epsilon - h_S(x_0))^2$$

$$Err(x_0) = \sigma_\epsilon^2 + V(\frac{1}{k}\sum_{l=1}^{k}(f(x_{(l)}) + \epsilon)) + [f(x_0) - \frac{1}{k}\sum_{l=1}^{k} f(x_{(l)})]^2$$

$$Err(x_0) = \sigma_\epsilon^2 + \frac{1}{k^2}\sum_{l=1}^{k}\sigma_\epsilon^2 + [f(x_0) - \frac{1}{k}\sum_{l=1}^{k} f(x_{(l)})]^2$$

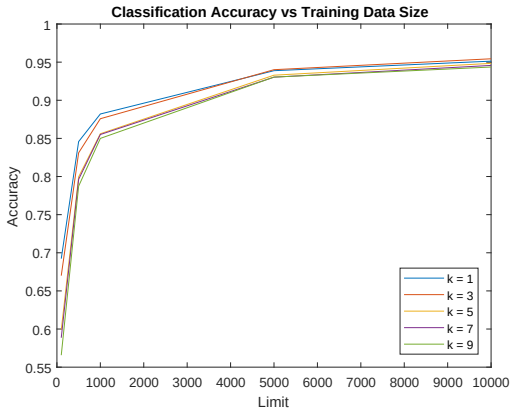

Figure 1: Accuracy versus training data size for various k values

$$Err(x_0) = \sigma_\epsilon^2 + [f(x_0) - \frac{1}{k}\sum_{l=1}^{k} f(x_{(l)})]^2 + \frac{\sigma_\epsilon^2}{k}$$

which is what we were trying to prove.

4 Syllabus Quiz

This has been done!

5 Figures

```
      0     1     2     3     4     5     6     7     8     9
-------------------------------------------------------------
0:  982     0     3     0     0     0     2     2     1     1
1:    0  1059     4     0     0     0     0     1     0     0
2:    2    11   936     9     1     1     3    20     4     3
3:    2     1     7   984     0     9     1     6    13     7
4:    0    16     0     0   922     0     0     8     0    37
5:    4     3     2    26     2   845    20     3     6     4
6:    3     1     0     0     0     4   959     0     0     0
7:    0    15     3     0     5     0     0  1059     0     8
8:    5    20     4    22     6    20     9    10   899    14
9:    5     3     0    10    16     4     0    23     1   899
Accuracy: 0.954400
```

Figure 2: Confusion Matrix where $k = 3$, $L = 10000$

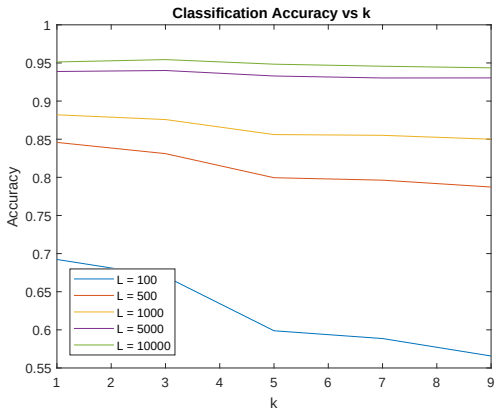

Figure 3: Accuracy versus k for several different training data sizes

```
$ py -3 cross_validation.py --limit 500
1-nearest neighber accuracy: 0.836000
3-nearest neighber accuracy: 0.858000
5-nearest neighber accuracy: 0.822000
7-nearest neighber accuracy: 0.824000
9-nearest neighber accuracy: 0.794000
Accuracy for chosen best k= 3: 0.831100
```

Figure 4: output of cross_validation.py for limit = 500

```
$ py -3 cross_validation.py --limit 5000
1-nearest neighber accuracy: 0.941800
3-nearest neighber accuracy: 0.937600
5-nearest neighber accuracy: 0.929400
7-nearest neighber accuracy: 0.926800
9-nearest neighber accuracy: 0.925000
Accuracy for chosen best k= 1: 0.938800
```

Figure 5: Output of cross_validation.py for limit = 5000