Machine Learning **Homework 1**
CSCI 5622 Fall 2017 **Due Time Sep 15, 2017**
Name: Luke Meszar **CU identitykey: lume0018**

1 K-nearest Neighbor (40pts)

Solution.

1. What is the role of the number of training instances to accuracy?

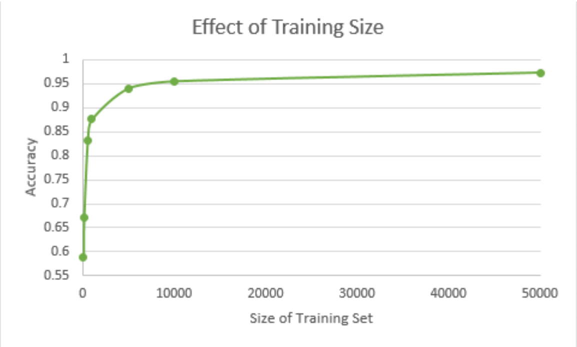

Figure 1: Training data size vs. accuracy with $k = 3$

We see from Figure 1 that the accuracy increases as the size of the training set increases. We notice there is a quick increase in accuracy when we change the size of the training set for small training set sizes. For instance, between size 50 *and size* 100*, the accuracy increases from* 58% *to* 67%*. When the training size is already large, making it bigger doesn't have much effect on the accuracy. As an example, between size* $10,000$ *and* $50,000$*, there is only about a* 2% *increase in accuracy. Looking at the graph, it seems that around a training set size of* $10,000$ *is where the diminishing returns begin.*

2. What numbers get confused with each other most easily?

I tested with $k = 3$ *and a limit of* 500 *and* 5000*. For a limit of* 500*, the most misclassified number was a* 9 *being confused for a* 4 *with a* 155 *misclassifications. The* 2*nd most misclassified number was a 9 being confused with a* 7 *with* 95 *misclassifications. There was a tie for the third most misclassified number, each with* 63 *misclassifications. These were a* 7 *being confused with a* 2 *and and a* 3 *being confused with an* 8*.*

When the training set size was increased to 5000*, the most misclassified number was still a* 9 *being confused with* 4*. However, the number of times this confusion happened when down to* 55*.*

3. What is the role of k to training accuracy?

Three tests were run. One where the size of the training data was 500*, one where the size was* 5000 *and one where the size was* 10000*. The value of k varied over odd values between* 1 *and* 13*. The results were as follows:*

k	Accuracy
1	0.8458
3	0.8311
5	0.8033
7	0.7986
9	0.7900
11	0.7825
13	0.7690

Table 1: Accuracy vs k for training set size of 500

k	Accuracy
1	0.9388
3	0.9401
5	0.9341
7	0.9307
9	0.9308
11	0.9271
13	0.9246

Table 2: Accuracy vs k for training set size of 5000

k	Accuracy
1	0.9513
3	0.9544
5	0.9492
7	0.9461
9	0.9441
11	0.9417
13	0.9411

Table 3: Accuracy vs k for training set size of 10000

The most accurate k changes depending on the size of the training set. When the size of the training set is 500, then the most accurate is $k = 1$. When the size of the training set is either 5000 or 10000, then the most accurate is $k = 3$.

4. In general, does a small value for k cause "overfitting" or "underfitting"?

In general, a small value of k will cause underfitting. When k is small, the algorithm only sees a few neighbors and cannot make a accurate prediction and won't perform *well on the training data. As the size of k increases, then the algorithm learns the training data too well and it will be unable to predict well on new data.*

2 Cross Validation (30pts)

Solution.

1. What is the best k chosen from 5-fold cross validation with "`--limit 500`"? *At* `--limit 500` *the best k is* 3 *with accuracy* 0.8311.

2. What is the best k chosen from 5-fold cross validation with "`--limit 5000`"?

   At `--limit 5000` *the best k is* 1 *with accuracy* 0.9388.

3. Is the best k consistent with the best performance k in problem 1?

   The optimal k is different for size 500 *and* 5000 *as found in problem* 1. *This is not surprising since the cross validation breaks up the training data differently so the testing conditions are not the same as in problem* 1.

3 Bias-variance tradeoff (20pts)

Proof. From the lecture, we know that the general form for bias-variance tradeoff is as follows:

$$\mathrm{Err}(x_0) = \sigma_\varepsilon^2 + [\mathrm{E}(h(x_0)) - f(x_0)]^2 + \mathrm{Var}(h(x_0)).$$

We begin by computing Bias^2:

$$\begin{aligned}
[\mathrm{E}(h(x_0)) - f(x_0)]^2 &= \left[\mathrm{E}\left(\frac{1}{k}\sum_{l=1}^{k}(f(x_l)+\varepsilon_l)\right) - f(x_0)\right]^2 \\
&= \left[\frac{1}{k}\mathrm{E}\left(\sum_{l=1}^{k} f(x_l)\right) + \mathrm{E}\left(\sum_{l=1}^{k}\varepsilon_l\right) - f(x_0)\right]^2 \\
&= \left[\frac{1}{k}\left(\sum_{l=1}^{k} f(x_l)\right) - f(x_0)\right]^2 \quad \text{assuming fixed nearest neighbors and } \mathrm{E}(\varepsilon) = 0.
\end{aligned}$$

Then, we compute the Variance:

$$\begin{aligned}
\mathrm{Var}(h(x_0)) &= \mathrm{Var}\left(\frac{1}{k}\sum_{l=1}^{k}(f(x_l)+\varepsilon_l)\right) \\
&= \frac{1}{k^2}\mathrm{Var}\left(\sum_{l=1}^{k}(f(x_l)+\varepsilon_l)\right) \\
&= \frac{1}{k^2}\left(\sum_{l=1}^{k}\left(\mathrm{Var}(f(x_l)) + \mathrm{Var}(\varepsilon_l)\right)\right) \\
&= \frac{1}{k^2}\left(0 + k\sigma_\varepsilon^2\right) \text{ assuming fixed nearest neighbors and } \mathrm{Var}(\varepsilon) = \sigma_\varepsilon^2 \\
&= \frac{\sigma_\varepsilon^2}{k}.
\end{aligned}$$

Combining these two steps, we get that

$$\mathrm{Err}(x_0) = \sigma_\varepsilon^2 + \left[\frac{1}{k}\left(\sum_{l=1}^{k} f(x_l)\right) - f(x_0)\right]^2 + \frac{\sigma_\varepsilon^2}{k}.$$

■