# Question 3 – KNN Classifier

b) For increasing neighbours of K, [1, 3, 5, 7],

errors = [0.0835, 0.0790, 0.0850, 0.0885]

As a trend I notice that there is not much decrease in error after K = 3

c) With K = 5

CONF =

173 0 4 0 0 2 3 1 3 0

0 234 9 0 2 1 1 12 1 0

0 0 192 0 0 1 0 3 1 0

0 0 0 188 0 5 0 0 9 2

0 0 1 0 199 2 2 2 3 5

0 0 0 6 0 158 1 1 6 4

1 0 3 2 3 3 171 0 2 0

1 0 8 4 1 1 0 178 3 4

0 0 2 3 0 2 0 0 160 2

0 0 0 4 12 4 0 8 4 177

(2, 8), and (10, 5) occurs 12 times.

This is the digit 7 being mistaken for 1, and the digit 4 being mistaken for 9, which makes sense considering how similar they appear. The matrix is not symmetric in those elements, as (8, 2) occurs 0 times, and (5, 10) only occurring 5 times.

# Question 5 – Randomized weight optimization

b) With 100, 1000, and 10000 iterations

train\_errors =

0.2748 0.2165 0.1087

test\_errors =

0.3601 0.2360 0.1314

Works as expected. By keeping track of the lowest error rate given a randomly generated weight vector, one would expect that increasing the iterations would decrease the error rate.

# Question 6 – Logistic Regression

b)

test\_err = 0.0560

train\_err = 0.0301

Both are significantly better than randomized optimization, given only 30 iterations, as compared to 10,000 iterations. Which is to be expected. Logistic regression takes into consideration the loss per the batch of training data and adjusts the weight vector accordingly.

# Question 8 – Perceptron Loss

b)

train\_err = 0.0622

test\_err = 0.1620

CONF =

164 0 1 1 2 4 4 0 1 0

0 230 9 3 3 0 1 8 10 2

3 2 182 5 2 1 7 6 9 0

0 0 7 163 1 7 0 5 11 1

0 0 1 0 189 3 4 6 5 6

2 0 4 19 0 144 5 1 10 4

2 2 6 1 4 3 154 0 3 0

2 0 2 6 0 2 2 143 3 5

0 0 5 4 3 11 1 1 126 1

2 0 2 5 13 4 0 35 14 175

(6, 4) occurs 19 times, (10, 8) occurs 35 times.  
This represents digit 5 mistaken for digit 3, and 9 being mistaken for 7. This is not consistent with 3c, which makes sense considering 5NN has a lower error rate than Perceptron loss in this case (0.08 versus 0.16, almost half as much of an error rate).

# Question 9 – Softmax Loss

b)

train\_err = 0.0472

test\_err = 0.1275

Training and Test error are about 0.02 units less than those of Perceptron. Which makes sense since Softmax normalizes the features.

## Question 12 – Competition

I submit the Softmax gradient descent multiclass classifier on the full MINST dataset (and training on both train and test data), with alpha of 0.01 and 100 iterations.

My user name is jgaebel.