Russell Stirling
250 757 946

Assignment 1: Hand Written Digit Classification

Question 1)

Code for question one can be found in file **p1.m**

Question 2)

Code for question one can be found in file **p2.m**

Question 3)

Part A:

Code for question one can be found in file **p3.m**

Part B:

K Value	1	3	5	7
Test Error	0.083500000000000	0.075000000000000	0.084500000000000	0.088500000000000

Test error seems to be best when using our three closest neighbors for comparison. It gets slightly less accurate with either just 1 neighbor, or 5 neighbors. It is convenient to use function p2 to check the error and confusion. Matlab also allows me to keep all results stored in variables so I can easily compare and contrast them.

Part C:

Confusion matrix for k=5:

Digit	0	1	2	3	4	5	6	7	8	9
0	173	0	0	0	0	0	1	1	0	0
1	0	234	0	0	0	0	0	0	0	0
2	4	8	193	0	1	0	3	8	2	0
3	0	0	0	188	0	6	2	4	3	4
4	0	2	0	0	199	0	3	1	0	12
5	2	1	1	5	2	158	3	1	2	4
6	3	1	0	0	2	1	171	0	0	0
7	1	12	3	0	2	1	0	178	0	8
8	3	1	1	9	3	6	2	3	160	4
9	0	0	0	2	5	4	0	4	2	177

4 (class 5) seems to be classified as 9 (class 10) very frequently. However this does not happen as much in reverse. Similarly, 7 (class 8) seems to be classified as 1 (class 2) fairly frequently too however 1 is never (in this trial) classified as 7.

Data gathered using commands:

```
clear
load('A1.mat')
Ck1 = p3(X_train, Y_train, X_test, 1);
Ck3 = p3(X_train, Y_train, X_test, 3);
Ck5 = p3(X_train, Y_train, X_test, 5);
Ck7 = p3(X_train, Y_train, X_test, 7);
[errK1,~] = p2(Ck1, Y_test);
```

Russell Stirling
250 757 946

```
[errK3,~] = p2(Ck3, Y_test);
[errK5,confK5] = p2(Ck5, Y_test);
[errK7,~] = p2(Ck7, Y_test);
```

Question 4)

Code for question one can be found in file **p4.m**

Question 5)

Part A:

Code for question one can be found in file **p5.m**

Part B:

Error table for each iteration value:

iterNum Value	100	1,000	10,000
Best Training Error	0.270157068062827	0.217801047120419	0.161256544502618
Test Error	0.315789473684211	0.238095238095238	0.195488721804511

In each case training error is somewhat better than testing error. This is due to the W being developed using the training data, meaning the weights are likely to fit the data it's error percentage was tested on better than new data. The error also decreases with more iterations. This is because more iterations gives more chances to find random W that fits well.

Data gathered using commands:

```
clear
load('A1.mat')
[Xtrain, Ytrain] = p1(X_train, Y_train, 4, 9)
[Xtest, Ytest] = p1(X_test, Y_test, 4, 9)

w_i100 = p5(Xtrain,Ytrain, 100)
w_i1000 = p5(Xtrain,Ytrain, 1000)
w_i10000 = p5(Xtrain,Ytrain, 10000)

c_i100_train = p4(w_i100, Xtrain);
c_i1000_train = p4(w_i1000, Xtrain);
c_i10000_train = p4(w_i10000, Xtrain);

c_i100_test = p4(w_i100, Xtest);
c_i1000_test = p4(w_i1000, Xtest);
c_i10000_test = p4(w_i10000, Xtest);

[errTrain100,~] = p2(c_i100_train, Ytrain);
[errTrain1000,~] = p2(c_i1000_train, Ytrain);
[errTrain10000,~] = p2(c_i10000_train, Ytrain);
[errTest100,~] = p2(c_i100_test, Ytest);
[errTest1000,~] = p2(c_i1000_test, Ytest);
[errTest10000,~] = p2(c_i10000_test, Ytest);
```

Russell Stirling
250 757 946

Question 6)

Part A:

Code for question one can be found in file **p6.m** and **mySigmoid.m**

Part B:

Training Error = 0.035602094240838

Testing Error = 0.062656641604010

Even when compared to the 10,000 iteration data collected in 5b, the training and testing error is much better in this implementation with just 30 iterations. This shows that the logistic regression batch rule implementation is much more efficient at finding accurate weights then a random guessing algorithm.

Data gathered using commands:

```
clear
load('A1.mat')
[Xtrain, Ytrain] = p1(X_train, Y_train, 4, 9)
[Xtest, Ytest] = p1(X_test, Y_test, 4, 9)
winit = ones(257,1)
wp6 = p6(Xtrain, Ytrain, 30, winit, 0.1);

c6train = p4(wp6, Xtrain);
c6test = p4(wp6, Xtest);

[errTrain,~] = p2(c6train, Ytrain);
[errTest,~] = p2(c6test, Ytest);
```

Question 7)

Code for question one can be found in file **p7.m**

Question 8)

Part A:

Code for question one can be found in file **p8.m**

Part B:

Training Error = 0.059800000000000

Testing Error = 0.158500000000000

Confusion matrix for test data:

Digit	0	1	2	3	4	5	6	7	8	9
0	158	0	3	0	0	3	4	2	0	5
1	0	228	2	0	1	0	2	0	1	0
2	1	4	189	6	3	3	4	2	4	3
3	1	0	5	172	1	7	2	5	8	6
4	2	2	1	1	191	0	5	0	4	11
5	8	0	2	4	4	146	3	1	8	3
6	5	1	4	0	5	5	156	1	1	0
7	0	6	8	5	3	2	0	134	1	46
8	1	4	10	8	7	8	3	4	137	10
9	1	0	0	2	8	4	0	5	2	172

The digits being confused the most are 7 (class 8) being classified as 9 (class 10) by far the most often. This is different from the highest confusion in part 3. For 4 (class 5) being classified as 9 (class 10) and 7 (class 8) being classified as 1 (class 2) the actual number of errors seems to be fairly similar to that of question 3; though 7 as class 1 has gone down slightly.

In order to gather data for this part the following commands were run:

```
clear
load('A1.mat')
randW = rand(10,257);
wp8 = p8(X_train, Y_train, 100, randW, 0.01);
cp8_train = p8(wp8, X_train);
cp8_test = p8(wp8, X_test);
[errTrain,~] = p2(cp8_ train, Y_train);
[errTest,confTest] = p2(cp8_ test, Y_test);
```

Russell Stirling
250 757 946

Question 9)

Part A:

Code for question one can be found in file **p9.m** and **mySoftmax.m**

Part B:

Training Error = 0.046400000000000

Testing Error = 0.121500000000000

Both training error and testing error have improved slightly over the error results in question 8. This implies that the gradient descent softmax single sample rule implementation is more effective than the Perceptron single sample rule.

In order to gather data for this part the following commands were run:

```
clear
load('A1.mat')
randW = rand(10,257);
wp9 = p9(X_train, Y_train, 100, randW, 0.01);
cp9_train = p7(wp9, X_train);
cp9_test = p7(wp9, X_test);
[errTrain,~] = p2(cp9_ train, Y_train);
[errTest,~] = p2(cp9_ test, Y_test);
```

Question 10)

Part A:

Code for question one can be found in file **p10a.m**

Part B:

Code for question one can be found in file **p10b.m**

Part C:

Validation Error: 0.071333333333333

Test Error: 0.089000000000000

Compared to both 8 and 9 error values, the validation error run on training data is slightly higher in the neural network than 8 or 9. However the test error, run on new data, is better than in both 8 and 9. This means that the algorithm's in 8 and 9 over-fitted to the training data when compared to the neural network and so the neural network created a better implementation for classifying new data.

In order to gather data the following commands were used:

```
load('A1.mat')
[net, valErr] = p10a(X_train, Y_train, [100;], 0.8)
[testErr, CONF] = p10b(X_test, Y_test, net)
```

Russell Stirling
250 757 946

Part D:

My best result was achieved with:

H = [150, 17]

regularizerWeight = 0.7

Validation Error: 0.056000000000000

Test Error: 0.069500000000000

Adding a second hidden layer and adjusting the weights allowed me to improve on my test error. Making the number of units per layer too high tended to result in an overfitting network that had good validation error but poor test error. Similarly, I found I got the best results when the second hidden layer was 10-15% the size of the first hidden layer. More than this and over fitting also occurred.

On the other hand, lowering the regularization weight too much caused over fitting as well, while raising it too much caused under fitting and poor results. I found a happy medium with about 0.7.

Question 11 [Extra Credit])

Extra credit data collection completed and submitted on January 22nd.

Question 12 [Extra Credit])

Did not participate in the Kaggle hosted competition.