Remmy Martin Kilonzo
250750759

Assignment 1

1

See p1.m

2

See p2.m

3

a) See p3.m

b)

K	**Training Error**
1	0.0835
3	0.0790
5	0.0850
7	0.0885

It seems that as K strays from 3, the error rate increases.

c)

K = 5: err = 0.0850
Confusion Matrix:

	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	9	192	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

It seems that 7 and 1 are confused, however this is asymmetric, as 1 and 7 are never confused. I suspect this has to do with the variants of 1, particularly those with a horizontal line at the bottom. I believe that the classifier is able to correctly identify

those variants of 1, whereas it confuses those that do not have the horizontal line, with 7.
Additionally, 4 and 9 are confused, but this confusion is more symmetric. I suspect that this is because there are few, if any, explicit identifiers of either class, unlike the 1-and-7 case, with regards to the horizontal line at the bottom of 1, which is absent from any variant of 7.

4

See p4.m

5

a) See p5.m
b)

Iterations	Training Error	Test Error
100	0.2922	0.3698
1000	0.2078	0.2287
10000	0.2311	0.2190

Interestingly, training error was lowest at 1000 iterations. I believe this is just a statistical error, since this method is, in effect, trying to brute force the ideal weights, and so it is expected that the more attempts it gets, the closer it becomes. This is reflected in the constantly decreasing test error. The improvement in test error follows a logarithmic trend, which is consistent with the expectations of such a strategy (as the ideal weight is being approached, an exponentially larger number of iterations are expected.

6

a) See p6.m
b)

Iterations	Training Error	Test Error
30	0.0291	0.0608

Compared to 5b), these results are fantastic; these errors are up to an order of magnitude less than those in the brute-force model, with significantly fewer iterations. This is because an optimization strategy was used—gradient decent—which had the model learn at each iteration, instead of starting from scratch in an attempt to produce the best weights.

Remmy Martin Kilonzo
250750759

7

See p7.m

8

a) See p8.m
b)

Iterations	**Learning Rate**	**Training Error**	**Test Error**
100	0.01	0.0654	0.1790

	0	1	2	3	4	5	6	7	8	9
0	155	0	3	0	0	6	4	2	2	3
1	0	225	2	0	0	2	4	0	1	0
2	0	4	165	8	4	3	8	5	**16**	6
3	1	1	2	166	2	8	4	6	10	7
4	1	2	1	2	189	2	7	1	4	8
5	8	0	0	7	1	130	8	2	**19**	4
6	4	1	9	1	5	4	151	2	1	0
7	0	5	**12**	7	3	1	0	152	3	**22**
8	2	**7**	4	9	5	**13**	2	2	146	2
9	0	0	0	4	11	5	0	**5**	6	163

Although the training error in this model is less than those in 3c), there are significantly more confused digits in this classifier, and many are not symmetrical. Digits 7 and 2 are confused, with almost no symmetry. I suspect this is because of the horizontal line that some use to “cross” the “spine” of their 7s. Likewise, 7 and 9 are confused, and this too is hardly symmetrical, and I cite the same reason for this. 2 and 8 do share symmetrical confusion, and this is likely because they can be quite similar. Interestingly, none of the confused digits in 3c) were nearly as confused in this model.

9

a) See p9.m
b)

Iterations	**Learning Rate**	**Training Error**	**Test Error**
100	0.01	0.8542	0.8820

The training and test errors in this model are significantly worse than those in 8b). This could be an indication that the learning rate needs tweaking, or that the softmax single sample rule is not ideal for classifying this dataset. Additionally, it could simply be the result of too few iterations.

10

a) See p10a.m
b) See p10b.m
c)

Regularization	Hidden Layers	Validation Error	Test Error
0.8	[100]	0.0680	0.0915

With a validation error similar to 8b), this model has a significantly better test error than both cases. Like 8b), this model is significantly better than 9b).

d)

Regularization	Hidden Layers	Validation Error	Test Error
0.8	[150, 150]	0.0407	0.0730
0.7	[100, 50]	0.0527	0.0715

The above are examples of parameters to achieve the best validation and test errors, with the top being those for the former, and the bottom being those for the latter. Either case has significantly improved validation and test errors over any of the previous models, including those in 10c).
These were retrieved though trial and error, testing the errors for a range of hidden layers and regularization strength values to find the best parameters.

11

Completed

12

Completed

I submitted a neural network classifier on the full MINST data with a regularization strength of 0.7, two hidden layers of 100 and 50 neurons respectively.

My profile is MartinKilonzo.

13

Using the neural network classifier from 10d) with the lowest test error, I examine the confusion matrix it produced:

	0	1	2	3	4	5	6	7	8	9
0	173	0	0	0	0	0	2	0	0	0
1	0	232	0	0	0	0	2	0	0	0
2	2	1	205	0	0	0	3	2	3	3
3	0	0	5	189	0	5	1	3	3	1
4	0	0	1	0	204	0	2	1	2	**7**
5	3	0	0	2	0	167	3	1	3	0
6	3	1	1	0	5	3	164	0	1	0
7	0	3	**10**	1	3	0	0	178	0	**10**
8	2	1	4	6	4	2	1	2	167	3
9	1	0	0	4	**7**	2	0	1	1	178

This model seems to confuse 2 and 7 asymmetrically, and 7 and 9 asymmetrically. Interestingly it confuses 4 and 9 perfectly symmetrically. Understanding this, my strategy was to pair this classifier with one that confuses uniquely different classes. Unfortunately, the ideal classifier for that is the KNN classifier in 3, which is too non-performant for a dataset as large as the digit one. As such, the next best alternative was chosen: the classifier in 8.

The next step was to improve this model by picking a better learning rate. The approach for this was identical to picking the ideal neural network parameters—trial and error.

It turned out the performance of the neural network in 10 was better than this combination, and so that will be my model for my initial submission to the competition.