Introduction to Deep Learning - Assignment 1

Brandão, Daniel
s3276694

Abelha, João
s3276708

1

1.1

The accuracy of the classifier is mediocre since most of the digits have a small distance to at least another one. The digit '1' is the one that has consistently high distance to all others. The digit '9' has many smalls distances with many of the other digits, making it hard to separate. So the digit 9 is expected to have a worse accuracy and the digit 1 is probably going to be classified correctly most of the time. The hardest pairs to separate are [9,8] (distance = 40.97), [3,5] (distance = 37.44), [9,4] (distance = 36.13) and [9,7] (distance = 29.45). (Appendix-Table 1)

1.2

Using dimensionality reduction algorithms it is possible visualize the clusters corresponding to each digit, however, the LLE algorithm has many overlaps between digits, their boundaries are not well defined making it hard to separate them. We can see how 9 is often overlapping other clusters mainly the 8 and 7 cluster, corresponding to the small distance between the numbers. The digit 1 is very isolated (except in the LLE algorithm visualization). The closest pairs in the distance matrix are also close when dimensionality reduced. For example, the clusters 3 and 5 are, in all 3 visualizations,very close or overlapping and the 0 and 1 clusters are very far from each other and do not overlap. The t-SNE algorithm takes more time to execute(since it is nonlinear) but produces better and clearer results, separating quite precisely all digit clusters.

set.

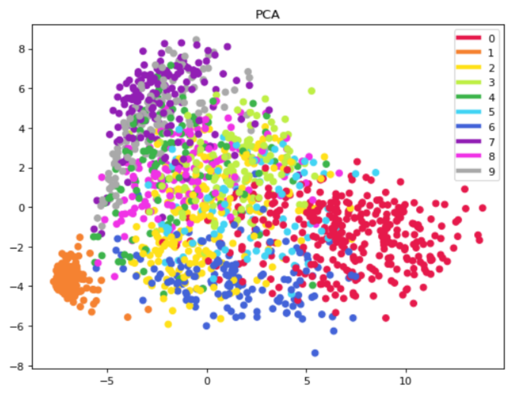

Figure 1: Visualization using PCA

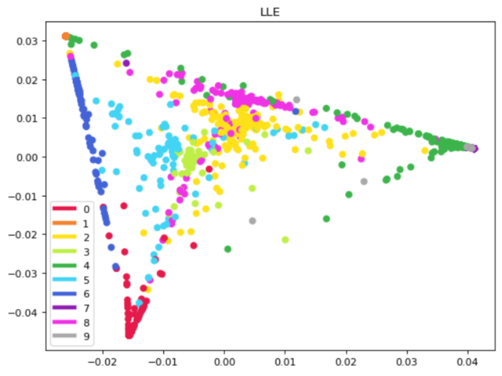

Figure 2: Visualization using LLE

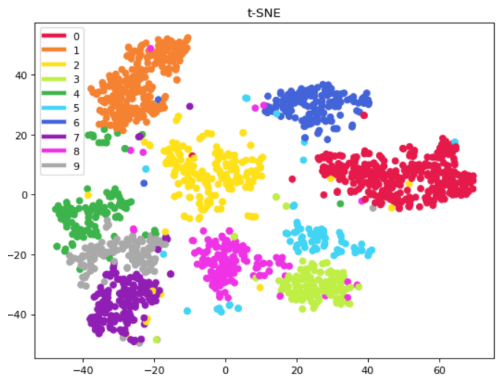

Figure 3: Visualization using t-SNE

1.3

The distance-based classifier accurately predicted 86.35% of the training set and 80.4% of the test set.

1.4

The k used was k=1 since it had the smallest error value.

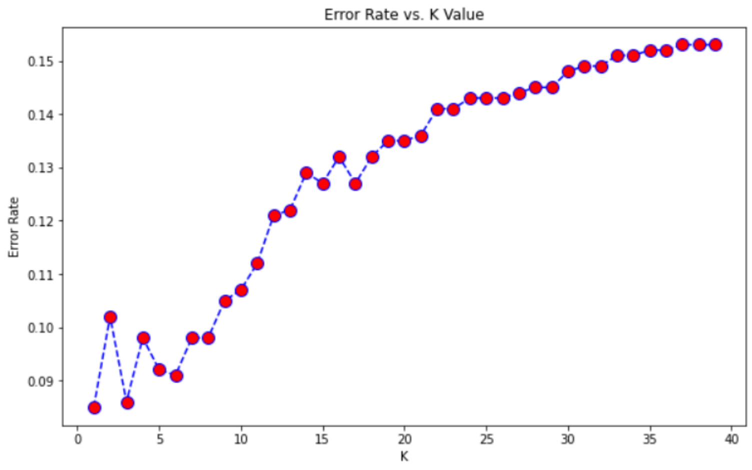

Figure 4: Error rate for each k value

The digits that were most difficult to classify correctly in the test set are 5 (25.455% miss-classifications), 2 (16.832% miss-classifications) and 8(15.217% miss-classifications). (Appendix-Table 2)

Despite the high miss-classification values for some digits, the classifier still has a high accuracy in the training set (0.915), since it classifies with very high accuracy the most common digits. The accuracy for the train set was 1.0. Since the k=1 the nearest neighbour to each element of the training set is itself, so it is always accurate, and also making this value not representative of an actual accuracy. This classifier performed significantly better than the distance-based one (0.915 compared to 0.804).

Training set Performance:

Accuracy: 1.0

Test set Performance:

Accuracy: 0.915

(Full measurements in the Appendix)

2

In main loop of the algorithm we implemented, first calculate the output of the all the nodes for an input, and compare them to the output of the node that is supposed to be the most activated. If other nodes output a higher value than the supposed node, their weights are decreased with the input values. The supposed node weights are increased with the input values. The algorithm presented in the slides had a stop condition for the loop that requires all the training set to be classified correctly, the algorithm could not satisfy that condition, so we ended up only doing a predefined number of epochs In each epoch this process is done to all training images. In the training set, the accuracy obtained was 0.708

3 Question 3

The function described in the document were slightly modified and have different parameters. The xor_net function also receives as parameter a function that should be the activation function. The grmdse function receives also a function as a parameter, the derivative of the activation function to be used in the back-propagation. During the training, data regarding the MSE, mean square error, and the classified examples were collected and is presented in the following charts:

Different activation function were also tried and compared. The sigmoid function outputs a value between 1 and 0. However, the output is never 0 or 1, which means that there is always going to be an error associated because we can only get close to those values but never reach them. Tanh shares a similar trend. For the data showed, the sigmoid seemed better, however, when struggling to find more rapidly the local minimum the tanh may be better since it has larger derivatives.

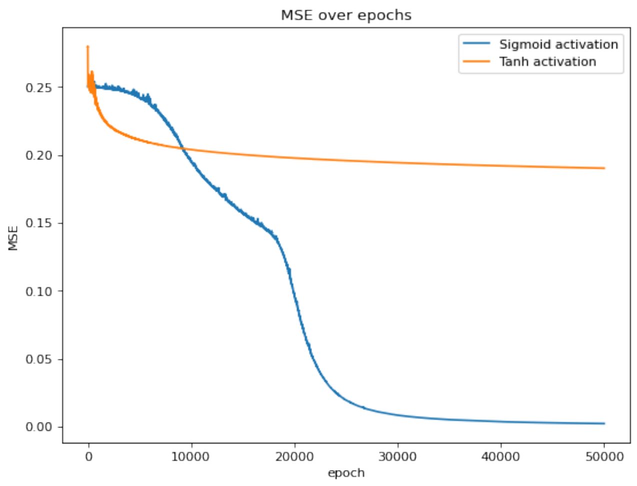

Figure 5: MSE over epochs, random weight initialization (-1,1), different activation functions, learning rate = 0.1

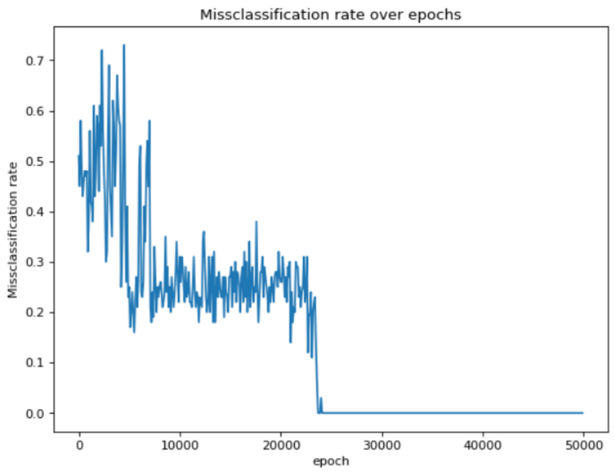

Figure 6: Miss classification rate over epochs, random weight initialization (-1,1), learning rate = 0.1

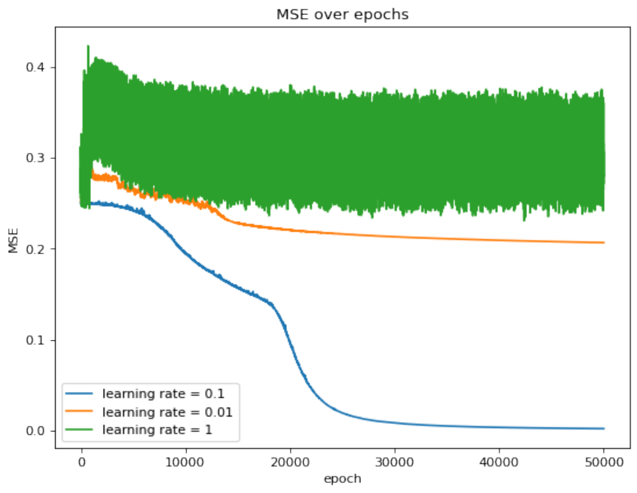

Figure 7: MSE over epochs, random weight initialization (-1,1), different learning rates

The network was trained with different learning rates. With a high training rate (training rate = 1) the gradient descent did not decrease during training and fluctuated a lot, taking steps to large to find the local minima. With a low of (training rate = 0.01) the training was slower and the error stopped decreasing while still high. A good middle ground that was found was 0.1, where the training was rather quick and also able to find the minimum. We concluded that the xor problem is quite sensible to the learning rate.

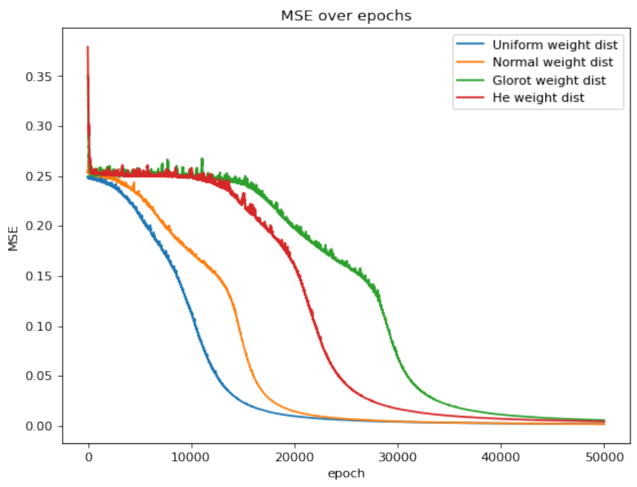

Figure 8: MSE over epochs, different weight initialization methods, learning rate = 0.1

We implemented different strategies to initialize weights to see how it affected training and to experiment with the lazy approach. In the training of the network, the uniform weight distribution performed a better than the others but the difference was not substantial. The glorot initialization would perform better if the activation function was ReLU.

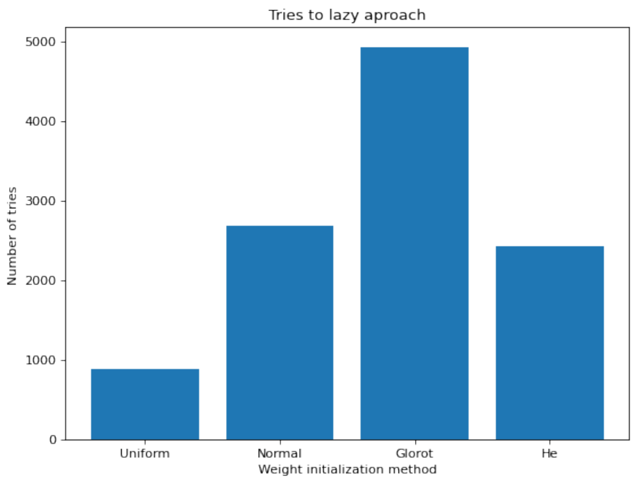

Figure 9: Average number of tries for each weight initialization method for the lazy approach

For the lazy approach, the uniform weight distribution also performed better since it outputs more varied values. These values were taken from a network that used the sigmoid activation function.

A Appendix

	0	1	2	3	4	5	6	7	8	9
0	0.00	208.79	87.13	83.61	115.99	56.54	66.49	140.77	98.17	131.99
1	208.79	0.00	102.52	137.67	103.51	123.63	112.67	115.42	101.74	98.65
2	87.13	102.52	0.00	66.88	62.93	62.52	53.76	78.72	50.09	78.99
3	83.61	137.67	66.88	0.00	82.58	37.44	86.53	79.61	49.29	69.80
4	115.99	103.51	62.93	82.58	0.00	64.02	77.13	57.50	54.48	36.13
5	56.54	123.63	62.52	37.44	64.02	0.00	44.87	84.86	48.54	68.20
6	66.49	112.67	53.76	86.53	77.13	44.87	0.00	118.55	73.74	108.99
7	140.77	115.42	78.72	79.61	57.50	84.86	118.55	0.00	71.70	29.45
8	98.17	101.74	50.09	49.29	54.48	48.54	73.74	71.70	0.00	40.97
9	131.99	98.65	78.99	69.80	36.13	68.20	108.99	29.45	40.97	0.00

Table 1: Distance matrix.

	0	1	2	3	4	5	6	7	8	9
0	219	0	2	0	0	0	1	1	0	1
1	0	118	0	0	0	0	2	1	0	0
2	5	1	84	2	0	0	1	3	4	1
3	3	0	2	71	0	2	0	0	0	1
4	0	2	3	0	77	0	0	2	0	2
5	3	0	0	5	0	41	0	3	2	1
6	4	0	0	0	2	0	84	0	0	0
7	0	1	0	1	3	0	0	59	0	0
8	1	1	0	6	0	2	1	1	78	2
9	1	0	0	0	0	0	0	2	1	84

Table 2: Test set confusion matrix.

K-neighbours classifier performance

Training set Performance:

Accuracy: 1.0

macro recall: 1.0

micro recall: 1.0

weighted recall: 1.0

macro f1-score: 1.0

micro f1-score: 1.0

weighted f1-score: 1.0

macro precision: 1.0

micro precision: 1.0

weighted precision: 1.0

Test set Performance:

Accuracy: 0.915

macro recall: 0.8981685786029999

micro recall: 0.915

weighted recall: 0.915

macro f1-score: 0.90177911633946

micro f1-score: 0.915

weighted f1-score: 0.9142472803684764

macro precision: 0.90897777443517789

micro precision: 0.915

weighted precision: 0.9161915409575275