ECE 174 Mini Project 1

Alexander Dagman, A15484465

November 2020

Introduction

The code for this project is included in the submission folder in "main.py". This file generates several output files which are the saved weights of the trained models and the computed confusion matrices.

Problem 1: Least Squares Classifier

After running both the one vs. all and the one vs. one classifiers, the following confusion matrices were created. The overall training error of the one-versus-one classifier was 13.93% and the overall training error of the one-versus-all classifier was 7.13%.

	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Predicted 6	Predicted 7	Predicted 8	Predicted 9	Total
Actual 0	944	0	1	2	2	8	13	2	7	1	**980**
Actual 1	0	1107	2	2	3	1	5	1	14	0	**1135**
Actual 2	18	54	815	26	16	0	38	22	39	4	**1032**
Actual 3	4	18	22	884	5	16	10	22	20	9	**1010**
Actual 4	0	22	6	0	883	3	9	1	12	46	**982**
Actual 5	24	19	3	74	24	656	24	13	38	17	**892**
Actual 6	17	9	10	0	22	17	876	0	7	0	**958**
Actual 7	5	43	14	6	25	1	1	883	1	49	**1028**
Actual 8	14	48	11	31	26	40	17	13	756	18	**974**
Actual 9	16	10	3	17	80	0	1	75	4	803	**1009**
Total	**1042**	**1330**	**887**	**1042**	**1086**	**742**	**994**	**1032**	**898**	**947**	**10000**
% Right	0.96326531	0.9753304	0.78972868	0.87524752	0.89918534	0.73542601	0.91440501	0.85894942	0.7761807	0.79583746	

Figure 1: Confusion Matrix One Vs. All

	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Predicted 6	Predicted 7	Predicted 8	Predicted 9	Total
Actual 0	962	0	3	1	1	4	5	3	1	0	**980**
Actual 1	0	1122	3	3	1	1	2	1	2	0	**1135**
Actual 2	8	22	930	14	13	5	11	7	22	0	**1032**
Actual 3	8	1	16	925	3	21	3	10	18	5	**1010**
Actual 4	2	2	7	1	934	2	6	3	2	23	**982**
Actual 5	12	6	2	31	9	799	14	2	13	4	**892**
Actual 6	8	5	11	0	8	20	904	0	2	0	**958**
Actual 7	1	17	17	3	10	1	0	955	2	22	**1028**
Actual 8	8	16	10	21	11	34	12	10	839	13	**974**
Actual 9	6	5	1	10	31	11	0	23	5	917	**1009**
Total	**1015**	**1196**	**1000**	**1009**	**1021**	**898**	**957**	**1014**	**906**	**984**	**10000**
% Right	0.98163265	0.98854626	0.90116279	0.91584158	0.95112016	0.89573991	0.94363257	0.92898833	0.8613963	0.90882061	

Figure 2: Confusion Matrix One Vs. One

The one vs. one classifier performs better than the one vs. all classifier. It has almost half the overall training error of the one vs. all classifier. Additionally, it performs better than the one vs. one classifier on every single digit. This makes sense to me because the one vs. one classifier has much more precise information to work with since it has knowledge of every single binary comparison instead of only having 10 one-vs-all comparisons.

The most difficult digits to classify in both cases were the 5 and the 8. For the one vs. all case, the 5 was followed by the 8 as the most incorrectly classified digit while in the one vs. one case, it was the 8 followed by the 5. My initial thought was that the 4 or the 9 would be most difficult because can look very similar to each other while the 5 and 8 are less similar to the other digits. After looking at the confusion matrix though, 5 and 8 also make sense as being difficult to classify. The 8 is commonly mistaken for a 1, 3, or 5. What surprises me is the amount of times that a 5 was mistaken for a 3. They don't seem very similar to me, but it seems like the classifiers thought otherwise.

Problem 2: K-means Clustering

The plots on the following pages are for K=20, P-30; K=10, P=20; K=5, P=10 respectively. In each case, the representative and correctly/incorrectly labelled digit are examples extracted from plots appended to the end of this report showing the 10 closest inputs to each representative (for the minimum Jcluster case).

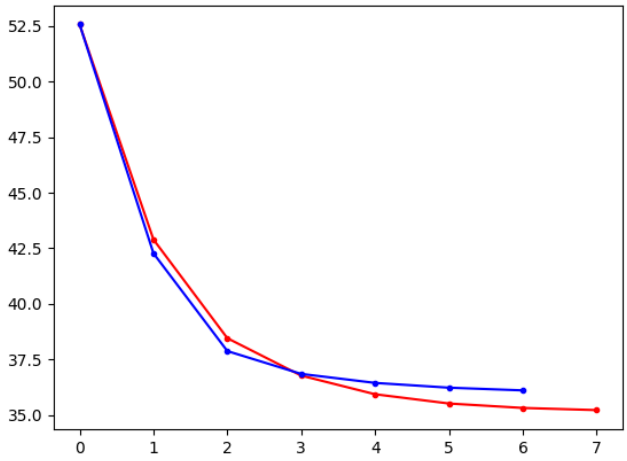

Figure 3: Jcluster K = 20, P = 30

In this case, the min and max Jcluster values are very close together. This

just means that there is little variation between input parameters in terms of their final error. This is likely because with large values of K, it becomes easier to find z's that are very close to their group due to smaller group sizes and more fine tune control over Jcluster (20 degress of freedom to minimize J vs. only 5 or 10)

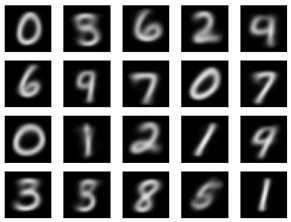

Figure 4: Z for min J

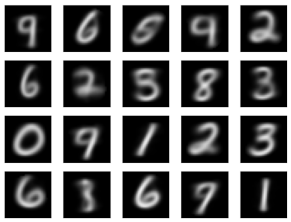

Figure 5: Z for max J

The z's that were generated using K=20 are also very crisp compares to the rest. With more z's, it is possible to create smaller clusters which means that every inputs the each z represent will have smaller variations so z will be more crisp.

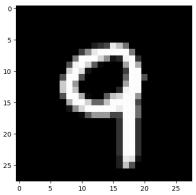

Figure 6: Correct

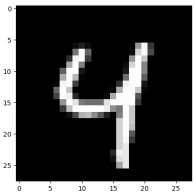

Figure 7: Incorrect

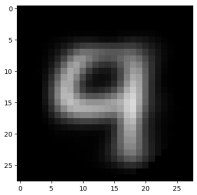

Figure 8: Representative

Here is an example a representative with two of its group members. As can also be seen in the next few pages, the incorrectly labelled digit is always a very similar digit to the correct one.

K = 20	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Predicted 6	Predicted 7	Predicted 8	Predicted 9	Total
Actual 0	30	0	0	0	0	0	0	0	0	0	30
Actual 1	0	30	0	0	0	0	0	0	0	0	30
Actual 2	0	0	20	0	0	0	0	0	0	0	20
Actual 3	0	0	0	25	0	0	0	0	0	0	25
Actual 4	0	0	0	0	0	0	0	0	0	11	11
Actual 5	0	0	0	5	0	10	0	0	0	0	15
Actual 6	0	0	0	0	0	0	20	0	0	0	20
Actual 7	0	0	0	0	0	0	0	20	0	0	20
Actual 8	0	0	0	0	0	0	0	0	10	0	10
Actual 9	0	0	0	0	0	0	0	0	0	19	19
Total	30	30	20	30	0	10	20	20	10	30	200
% Right	**1**	**1**	**1**	**1**	**0**	**0.666666667**	**1**	**1**	**1**	**1**	

Figure 9: Confusion matrix for K = 20

Here is the confusion matrix for the 10 closest points to all representatives. As can be seen, while the digits accounted for by the representatives are predicted with very minimal error (100% accuracy for 8/10 digits), we misclassify 4 100% of the time due to it not appearing as one of the representatives. This is the trouble with kmeans for this type of problem. It will become more apparent in the later examples.

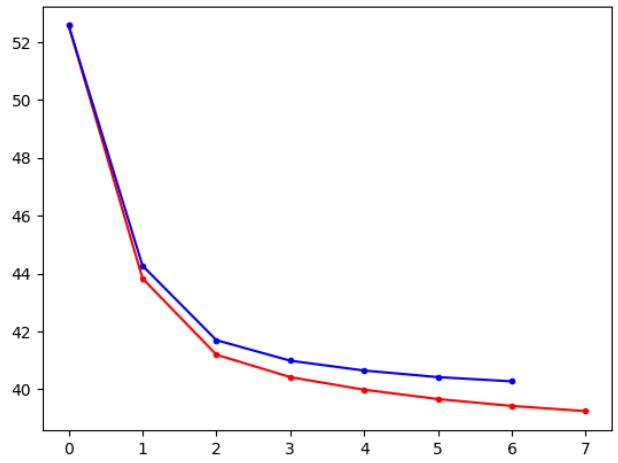

Figure 10: Jcluster K = 10, P = 20

In this case, we can begin to see the widening of the min and max Jcluster values as well as an overall increase in final values. This is happening because as the number of groups gets smaller, fine tuning Jcluster because harder so there are fewer possible sets Z that result in a certain J. Because of this, different initial conditions might produce a larger variation of final Jcluster. The overall final value increases because distances from group members to their centroid also will tend to increase because the clusters will have to be larger (include further points) with less z's.

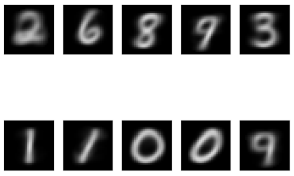

Figure 11: Z for min J

Figure 12: Z for max J

Here the z's are still fairly sharp. It seems that with K = 10, it is still possible to find centroids that have fairly little variation between them, so we do not see blurring (which would happen when the group includes multiple different digits in similar quantities instead of mostly just a single digit)

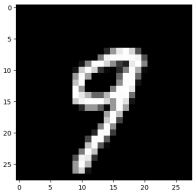

Figure 13: Correct

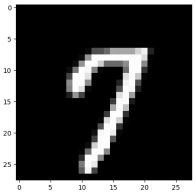

Figure 14: Incorrect

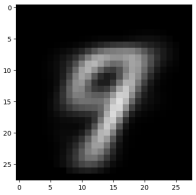

Figure 15: Representative

Similar to the previous example, the incorrectly labelled digit has a lot of similarities with the correctly labelled one. This is just a feature of how kmeans works. The inputs with similar features will be close together geometrically, so they will end up in the same group.

K = 10	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Predicted 6	Predicted 7	Predicted 8	Predicted 9	Total
Actual 0	20	0	0	0	0	0	0	0	0	0	20
Actual 1	0	20	0	0	0	0	0	0	0	0	20
Actual 2	0	0	10	0	0	0	0	0	0	0	10
Actual 3	0	0	0	10	0	0	0	0	0	0	10
Actual 4	0	0	0	0	0	0	0	0	0	1	1
Actual 5	0	0	0	0	0	0	0	0	0	0	0
Actual 6	0	0	0	0	0	0	10	0	0	0	10
Actual 7	0	0	0	0	0	0	0	0	0	4	4
Actual 8	0	0	0	0	0	0	0	0	10	0	10
Actual 9	0	0	0	0	0	0	0	0	0	15	15
Total	20	20	10	10	0	0	10	0	10	20	100
% Right	**1**	**1**	**1**	**1**	**0**	**N/A**	**1**	**0**	**1**	**1**	

Figure 16: Confusion matrix for K = 10

In the previous example, all digits appeared somewhere in the 10 closest points to the z's. In this case, K is becoming too small to achieve this. 4 and 5 do not appear as representatives, and because the other representatives are all dissimilar from 5, it doesn't even appear in any of the 10 closest points to any of the representatives.

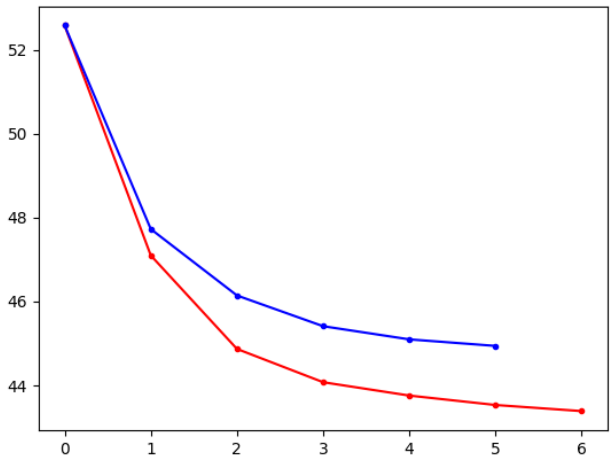

Figure 17: Jcluster K = 5, P = 10

With K = 5, we see an even bigger gap between min and max final Jcluster values for the reasons specified int the previous examples. We also see that the final value is even higher than K = 10 for the reasons listed above as well.

Figure 18: Z for min J

Figure 19: Z for max J

In this case, the lines between digits become very blurred. This makes sense. There are 10 digits, but only 5 groups. As a result, each group will likely consist of mostly more than one digit in equal quantities, so the resulting z's will look like two numbers overlayed which is exactly what we see happening.

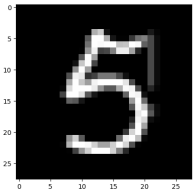

Figure 20: Correct

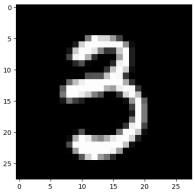

Figure 21: Incorrect

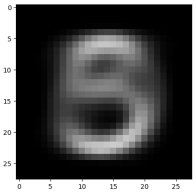

Figure 22: Representative

This case is slightly different than the previous examples. In the previous examples, the z's were still pretty sharp and still resembled different numbers very distinctly. For K = 5, since we have more digits than representatives, each representative is "forced" to represent more than one digit. As a result, we get representatives which are a mix of two digits like the example above which is a mix of a 5 and a 3 (although I thin it resembles a 5 more closely). In this

case, the incorrect digit need not be as similar to the correct digit as in previous cases. For K = 5, we are bound to get plenty incorrectly labelled digits because half of them aren't even a possible output options.

K = 5	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Predicted 6	Predicted 7	Predicted 8	Predicted 9	Total
Actual 0	10	0	0	0	0	0	0	0	0	0	10
Actual 1	0	10	0	0	0	0	0	0	0	0	10
Actual 2	0	0	0	0	0	0	0	0	0	0	0
Actual 3	0	0	0	9	0	0	0	0	0	0	9
Actual 4	0	0	0	0	0	0	0	0	0	0	0
Actual 5	0	0	0	1	0	0	0	0	0	0	1
Actual 6	0	0	0	0	0	0	10	0	0	0	10
Actual 7	0	0	0	0	0	0	0	0	0	0	0
Actual 8	0	0	0	0	0	0	0	0	0	0	0
Actual 9	0	0	0	0	0	0	0	0	0	10	10
Total	10	10	0	10	0	0	10	0	0	10	50
% Right	1	1	N/A	1	N/A	0	1	N/A	10	1	

Figure 23: Confusion matrix for K = 5

This is the case of kmeans where K < N. Here N-K classes are unreachable as can be seen in the above confusion matrix. The 2, 4, and 7 don't appear anywhere in the 10 closest points to any of the z's. This is because the only representatives we have available to select from are 0. 1. 3. 6. and 9; it makes sense that those digits to not appear.

Conclusion

In summary, I think the supervised binary classifier should be preferred over the kmeans algorithm for this problem (handwritten digit recognition) for several reasons. 1) We can set the supervised classifier to exactly classify each digit as 0 - 9, but with the kmeans algorithm, depending on our choice of k, we are not even guaranteed to receive an output that is between 0 - 9. As shown above, if K < N (where N is number of classes of input data), we aren't even able to classify N-K digits correctly from the get go. 2) Even if K $\geq$ N, we are still not guaranteed to get the N different classes we desire. As in the K = 10 example, 0, 1, and 9 had duplicate representatives. We miss out on classifying 4, 5, and 7 because of this. 3) The time it takes to run kmeans is easily at least 100x longer than training our binary classifier. This is just not desired for the resulting accuracy. 4) Finally, while the confusion matrices shown above indicate that kmeans can achieve remarkable accuracy, the test cases for this were very small (10 for each z), and they were also the most likely to be correct since they were the closest to the z that they belonged to. As a result, we cannot trust these kmeans confusion matrices to generalize to larger test cases – the error will likely be higher. As a result, I think the binary classifier should be preferred for this exercise.

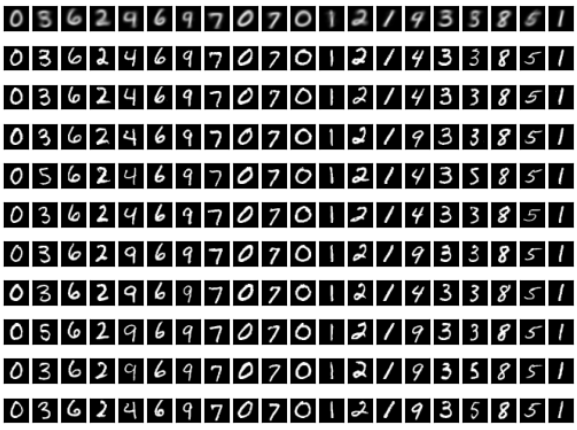

Figure 24: Closest inputs to representatives (K = 20, minimum Jcluster)

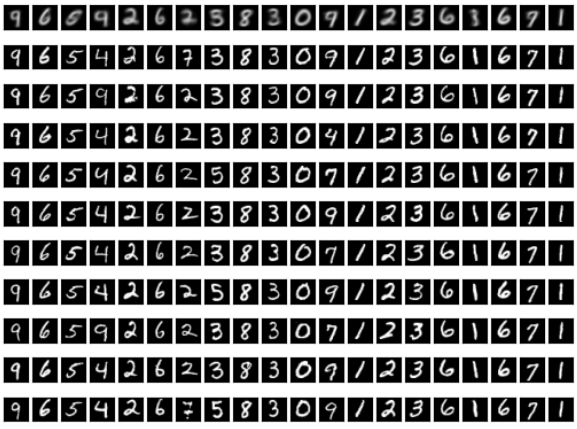

Figure 25: Closest inputs to representatives (K = 20, maximum Jcluster)

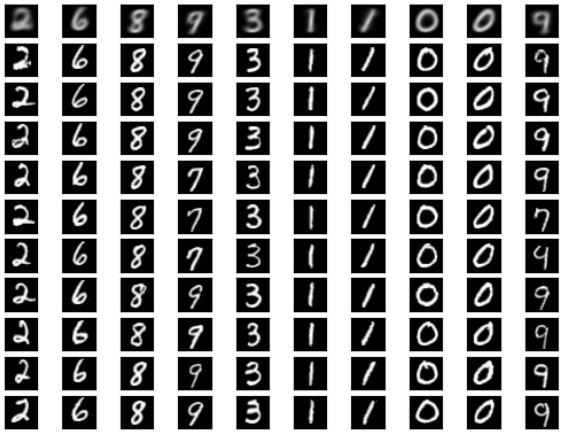

Figure 26: Closest inputs to representatives (K = 10, minimum Jcluster)

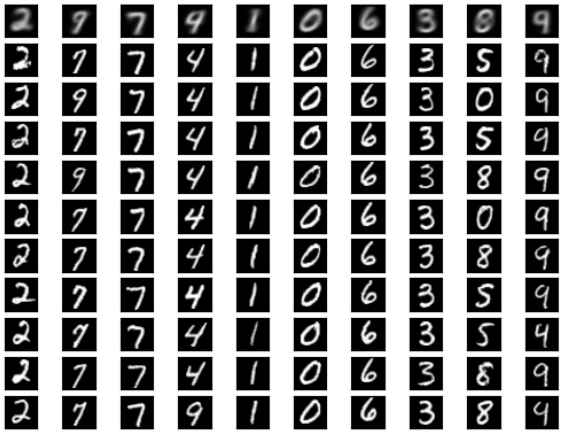

Figure 27: Closest inputs to representatives (K = 10, maximum Jcluster)

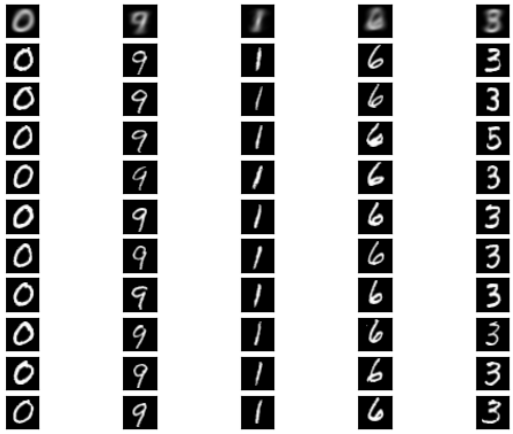

Figure 28: Closest inputs to representatives (K = 5, minimum Jcluster)

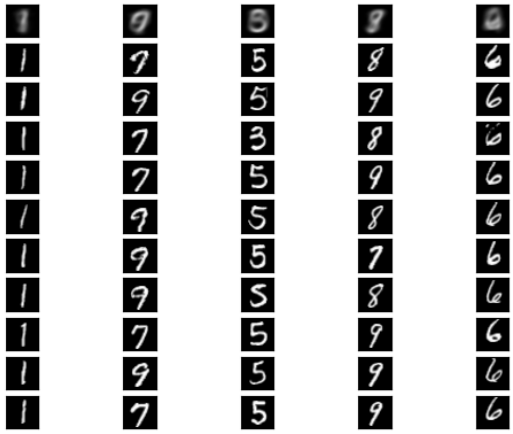

Figure 29: Closest inputs to representatives (K = 5, maximum Jcluster)