Matthew O’Brien
CS445

Homework 5 K-Mean Clusters

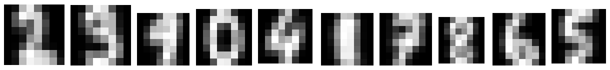
Introduction:

The purpose of this project was to use K mean clustering algorithm to classify the OptDigits data from the UCI ML repository. The data was split into two pieces for training and testing of the data. This was given within the assignment’s zip folder. Two different K values were used for the project, the first was 10 clusters, and the second was 30 clusters. Each of these experimental values had 5 randomly generated cluster sets, and the one with the smallest SSE was selected for each trial. The 30 cluster set outperformed the 10 cluster, with the 10 cluster set getting 79.35% accuracy on the test data, and the 30 cluster set getting 91.88% accuracy.

Experiment 1:

Experiment one used 10 clusters for the K means algorithm. The idea for this experiment was to allow for a cluster for each of the expected integer values, and see if the algorithm was able to assignment all of the integers to their correct cluster set. The following labels were generated for the clusters and their images:

These are the corresponding labels for each cluster: [2. 3. 1. 0. 4. 1. 7. 8. 6. 5.]



The images for this project were extremely small and rather difficult to expand in size without distorting the image too much. With the exception of the 3 cluster being a “1” and looking more like a 4 or 9, the rest of them seemed to generate a rather accurate image for their expected class. It is worth mentioning that with this trial, there was no cluster for the class “9”, which seemed to be absorbed into the “3” class. Considering the similarities to 3 and 9, it would make sense for this sort of grouping to happen under randomly generated clusters. Among other tests ran out of pure interest, there were many times when the “6” and “5” classes would end up mixed, as well as the “9” and “4”. Even with these faults the following values were determined for this experiment:

This is the accuracy for the training data: 80.6173162438

This is the accuracy for the testing data: 79.3544796884

This is the best SSS: 113740.919346

This is the best SSE: 2478785.06037

The mean Entropy is: 0.852485041836

It is quite impressive how well the algorithm did on the testing data without even being able to classify any of the “9” classes into the correct cluster. Though, there is clearly a very high entropy for this classification set. As well as a large SSE value, and a smaller SSS value. This is likely due to the fact that the experiment is missing an entire group of “9”s. Without that explicit grouping, it would make sense that the clusters are too close together to be able to classify all the data. The last piece of information confusion matrix generated for this experiment set. It reads from 0 to 9, top left to right, top to bottom. The expected value of “9” is clearly 0 for this instance, and the algorithm classifies them as “3” as stated before.

The confusion matrix:

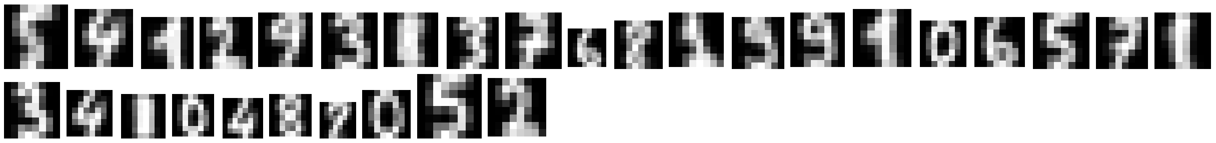
```
[[176   0   0   0   2   0   0   0   0   0]

 [  0 156  21   1   0   1   3   0   0   0]

 [  1   4 149   9   0   0   0   4  10   0]

 [  0   1   0 163   0   2   0   9   8   0]

 [  0   8   0   0 161   0   0   7   5   0]

 [  0   1   0  31   1 148   1   0   0   0]

 [  1   4   0   0   0   0 175   0   1   0]

 [  0   8   0   0   1   0   0 167   3   0]

 [  0  29   1  10   0   1   1   1 131   0]

 [  0  26   0 145   0   3   0   3   3   0]]
```

In summary, this experimental value of 10 for K is not ideal given the many different ways a person is able to write or symbolize the same digit. There are too many features of each digit that can be similar to others. With some other testing runs with 10 for K, at times the algorithm was able to make 10 different classifications for each digit, but it was not very common. I chose to represent experiment 1 with the given information because it shows the lack of flexibility with the given features. There would need to be more clusters to accurately represent the many different ways each digit can be shown in data set.

Experiment 2:

The second experiment performed better than the previous as stated in the introduction. This experiment used 30 different clusters to represent the OptDigit data. The following labels for each of the clusters and their images were generated:

[5. 4. 1. 2. 9. 3. 1. 3. 7. 6. 8. 1. 9. 9. 1. 0. 6. 5. 7. 1. 3. 4. 1. 0. 4. 8. 7. 0. 5. 2.]



Right away there is a very clear difference in the data set. This time, each of the possible digits has a represented cluster, and many have duplicates. The images generated were again very small, but it is possible to see that each of the clusters match quite well with their corresponding images. Looking at cluster 27, or the “7” class, it seems that it is possible that the program could have confused “7” and “9” based on the shading, but it is possible that it is accounting for the two ways of writing a “7”, with and without the dash through the middle.

The following data was determined through the experiment:

This is the accuracy for the training data: 93.1205859273

This is the accuracy for the testing data: 92.3205342237

This is the best SSS: 1328439.82156

This is the best SSE: 1819036.06255

The mean Entropy is: 0.320179519594

The final accuracy on the testing data was 92.32%, or over 10% greater than the 10 K set from the previous experiment. This jump is more than likely due to the extra clusters allowing for the different ways of representing each of the digits. The best SSS is clearly larger than the previous experiment, but that could be due to the increased number of clusters. That would give a much larger summation of distances between clusters, simply due to the number of clusters. And that would also explain the SSE for this data set. The summation of the distances for SSE is going to be larger simply due to the increased number of clusters. The second biggest indication that this K value was better than the previous one comes from the mean entropy of this experiment. A much lower mean entropy of 0.32 was determined. This shows a smaller level of disorder to each of the clusters, suggesting that the clusters were indeed gathering around the same classes. The finally piece of information for this experiment is the confusion matrix:

```
[[177   0   0   0   1   0   0   0   0   0]
 [  0 178   0   0   0   0   0   0   1   3]
 [  1   6 159   1   0   0   0   4   6   0]
 [  0   0   2 166   0   1   0   5   5   4]
 [  0   5   0   0 173   0   0   3   0   0]
 [  0   0   0   1   1 172   0   0   0   8]
 [  2   3   0   0   1   1 172   0   2   0]
 [  0   0   0   0   1   0   0 169   2   7]
 [  0  29   0   2   0   2   1   1 130   9]
 [  1   3   0   4   0   2   0   5   2 163]]
```

This matrix was generated in the same fashion as experiment one, and shows a clear picture of how much better 30 clusters to 10 is for this data set. With the exception of the “1” and “8” getting mixed, the algorithm was extremely accurate at classifying the data into the correct class. Looking at the cluster’s images, the 7th and the 23rd clusters show a distorted image of what could be a “1” or an “8”. Given the many different ways people write or represent different digits, it’s does make sense for this type of error to pop up within randomly generated centroids.

Conclusion:

Overall, it is very clear that the 30 cluster set outperformed the 10 cluster set from the two experiments. Looking at the difference between the accuracies and entropies, the 30 cluster set is a much better option for this sort of data set. It would be interesting to see if more clusters could solve the issues that the second experiment was having between the “8” and “1”. As well as a possibly better way to actually determine the centroid’s starting attributes. The few extra times I ran the second experiment, the accuracy was always over 90% and the entropy was roughly 0.3, but there was normally sort misclassification between two similar digits. Even with the issues of the second experiment, it was still clearly the better option for this data set.