CMSC 678
Introduction to Machine Learning
Homework 4

Q1. Randomly sample k = 10 instances, use them as the initial cluster centroids, and run the algorithm to convergence. For each cluster, find the most common digit in that cluster and count the number of instances in the cluster that are different from the most common one. Sum that counts over all of the clusters.

PFB the results which were developed after running the code.

In [9]:
```
training_data = [line.split() for line in open('mnist_raw_data.txt').readlines()]
training_labels=[int(line.strip('\n')) for line in open('mnist_labels.txt','r').readlines()]
train_data= np.array(training_data, dtype=np.uint8)
kmeans = KMeans(n_clusters=10,max_iter=200)
iter_avg=[None for _ in range(10)]
incorrect_avg=[None for _ in range(10)]
#Part2 of the HW
for i in range(10):
    print("Runtime - ",i+1)
    iter_avg[i],incorrect_avg[i]=kmeans.form_cluster(train_data,training_labels)
print("Average number of iterations to converge : ",sum(iter_avg)//10)
print("Average number of incorrects labels : ",sum(incorrect_avg)//10)
kmeans.plot_imgs(train_data,25,True)
```

```
Runtime -  1

Epoch:  0 ==> Difference between centroids:  7644.3681884116495

Epoch:  1 ==> Difference between centroids:  4568.933266287454

Epoch:  2 ==> Difference between centroids:  915.1747318771413

Epoch:  3 ==> Difference between centroids:  495.2558724613098

Epoch:  4 ==> Difference between centroids:  381.20256077227896

Epoch:  5 ==> Difference between centroids:  406.3570124617401

Epoch:  6 ==> Difference between centroids:  391.14344140267264

Epoch:  7 ==> Difference between centroids:  221.32083675659098

Epoch:  8 ==> Difference between centroids:  122.1181658384007
```

In [4]:
```
print(sum(iter_avg)//10)
```

In [13]:
```
incorrect_all=[0 for _ in range(10)]
for i in range(10):
    print("Cluster Label : ",kmeans.clusters_info[i][0])
    print("Number of Accurate Labels : ",kmeans.clusters_info[i][1])
    print("Total number of samples : ",kmeans.clusters_info[i][2])
    print("Incorrect Labels : ",kmeans.clusters_info[i][2] - kmeans.clusters_info[i][1])
    print("Cluster Accuracy : ",kmeans.clusters_info[i][3])
    print("-----------------------------------------------")
    incorrect_all.append(kmeans.clusters_info[i][2] - kmeans.clusters_info[i][1])
print('Accuracy:',kmeans.accuracy)
print('Sum over all the Clusters:',sum(incorrect_all))
```

```
Cluster Label :  8
Number of Accurate Labels :  628
Total number of samples :  988
Incorrect Labels :  360
Cluster Accuracy :  0.6356275303643725
-----------------------------------------------
Cluster Label :  6
Number of Accurate Labels :  629
Total number of samples :  1223
Incorrect Labels :  594
Cluster Accuracy :  0.5143090760425184
-----------------------------------------------
Cluster Label :  3
Number of Accurate Labels :  736
Total number of samples :  1273
Incorrect Labels :  537
Cluster Accuracy :  0.5781618224666143
-----------------------------------------------
Cluster Label :  4
Number of Accurate Labels :  540
Total number of samples :  1449
Incorrect Labels :  909
Cluster Accuracy :  0.37267080745341613
-----------------------------------------------

-----------------------------------------------
Cluster Label :  0
Number of Accurate Labels :  460
Total number of samples :  569
Incorrect Labels :  109
Cluster Accuracy :  0.8084358523725835
-----------------------------------------------
Cluster Label :  1
Number of Accurate Labels :  652
Total number of samples :  811
Incorrect Labels :  159
Cluster Accuracy :  0.8039457459926017
-----------------------------------------------
Cluster Label :  1
Number of Accurate Labels :  473
Total number of samples :  793
Incorrect Labels :  320
Cluster Accuracy :  0.5964691046658259
-----------------------------------------------
Cluster Label :  0
Number of Accurate Labels :  408
Total number of samples :  466
Incorrect Labels :  58
Cluster Accuracy :  0.8755364806866953
-----------------------------------------------
Cluster Label :  7
Number of Accurate Labels :  649
Total number of samples :  1402
Incorrect Labels :  753
Cluster Accuracy :  0.4629101283880171
-----------------------------------------------
                -
-----------------------------------------------
Cluster Label :  5
Number of Accurate Labels :  328
Total number of samples :  1026
Incorrect Labels :  698
Cluster Accuracy :  0.31968810916179335
-----------------------------------------------
Accuracy: 0.5967754657594438
Sum over all the Clusters: 4497
```

In [ ]:

Q2. Repeat the above step 10 times in total and report the average number of iterations to convergence and the average number of instances that are in the wrong cluster.

PFB the results of the above mentioned question. Also, please check *ML-HW4-Kmeans.ipynb* for all the runtimes.

```
#Part2 of the HW
for i in range(10):
    print("Runtime - ",i+1)
    iter_avg[i],incorrect_avg[i]=kmeans.form_cluster(train_data,training_labels)
print("Average number of iterations to converge : ",sum(iter_avg)//10)
print("Average number of incorrects labels : ",sum(incorrect_avg)//10)
kmeans.plot_imgs(train_data,25,True)
```

```
Epoch:  50 ==> Difference between centroids:  2.83098528218758

Epoch:  51 ==> Difference between centroids:  3.0258500891832494

Epoch:  52 ==> Difference between centroids:  0.0029867663554642146

Epoch:  53 ==> Difference between centroids:  3.130956349887746e-06

Epoch:  54 ==> Difference between centroids:  3.3881600165802998e-09
Converged! ==>  3.746678872753727e-12
Accuracy: 0.6487537068691785
Runtime -  10

Epoch:  0 ==> Difference between centroids:  8304.76086350474

Epoch:  1 ==> Difference between centroids:  5091.741465242482

Epoch:  2 ==> Difference between centroids:  981.4713811089273

Epoch:  3 ==> Difference between centroids:  588.3434335276602
```

```
#Part2 of the HW
for i in range(10):
    print("Runtime - ",i+1)
    iter_avg[i],incorrect_avg[i]=kmeans.form_cluster(train_data,training_labels)
print("Average number of iterations to converge : ",sum(iter_avg)//10)
print("Average number of incorrects labels : ",sum(incorrect_avg)//10)
kmeans.plot_imgs(train_data,25,True)
```

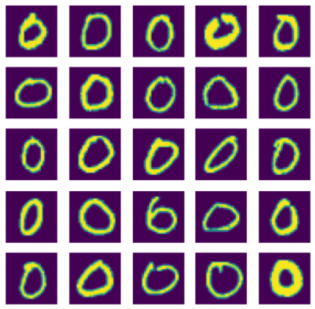
```
Epoch:  43 ==> Difference between centroids:  32.7357160556007

Epoch:  44 ==> Difference between centroids:  21.579857895840885

Epoch:  45 ==> Difference between centroids:  10.474908034202102

Epoch:  46 ==> Difference between centroids:  11.523188911199826

Epoch:  47 ==> Difference between centroids:  5.1967308704479995

Epoch:  48 ==> Difference between centroids:  0.008529518120409705

Epoch:  49 ==> Difference between centroids:  1.4738391985618857e-05

Epoch:  50 ==> Difference between centroids:  2.5772819831652963e-08
Converged! ==>  4.518421359565164e-11
Accuracy: 0.5967754657594438
Average number of iterations to converge :  47
Average number of incorrects labels :  304
```

Q3. Run the algorithm with k = 5. Look at the clusters and see if there are digits that tend to get grouped together. What are they and explain why you think they are grouped into the same cluster.

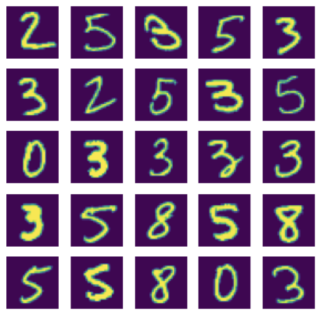
PFB the screenshots where k=5. There were 5 centroids i.e {0,3,9,6,1}. We can see label 6 was grouped with centroid label 0 as the pixels in the image closely represent the pixels in 0. The euclidean distance is less for an image that represents 6 with centroid 0. Similarly, {5,8} closely represent centroid 3, {4,7} closely represent centroid 9, {2} closely represent centroid 6 and {7} are close enough with centroid 1.

```
In [5]: for key,data in list(kmeans.clusters['data'].items()):
            print('Cluster:',key,'Label:',kmeans.clusters_labels[key])
            kmeans.plot_imgs(data[:min(25,data.shape[0])],min(25,data.shape[0]))
```

```
Cluster: 0 Label: 0
```



```
Cluster: 1 Label: 3
```



```
Cluster: 2 Label: 9
```

Cluster: 3 Label: 6

Cluster: 4 Label: 1

In [8]:
```
incorrect_all=[0 for _ in range(5)]
for i in range(5):
    print("Cluster Label : ",kmeans.clusters_info[i][0])
    print("Number of Accurate Labels : ",kmeans.clusters_info[i][1])
    print("Total number of samples : ",kmeans.clusters_info[i][2])
    print("Incorrect Labels : ",kmeans.clusters_info[i][2] - kmeans.clusters_info[i][1])
    print("Cluster Accuracy : ",kmeans.clusters_info[i][3])
    print("-------------------------------------------")
    incorrect_all.append(kmeans.clusters_info[i][2] - kmeans.clusters_info[i][1])
print('Accuracy:',kmeans.accuracy)
print('Sum over all the Clusters:',sum(incorrect_all))
```

```
Cluster Label :  0
Number of Accurate Labels :  849
Total number of samples :  944
Incorrect Labels :  95
Cluster Accuracy :  0.899364406779661
-------------------------------------------
Cluster Label :  3
Number of Accurate Labels :  865
Total number of samples :  2217
Incorrect Labels :  1352
Cluster Accuracy :  0.3901668921966622
-------------------------------------------
Cluster Label :  9
Number of Accurate Labels :  881
Total number of samples :  2852
Incorrect Labels :  1971
Cluster Accuracy :  0.30890603085553997
-------------------------------------------
Cluster Label :  6
Number of Accurate Labels :  803
Total number of samples :  1710
Incorrect Labels :  907
Cluster Accuracy :  0.4695906432748538
-------------------------------------------
Total number of samples :  944
Incorrect Labels :  95
Cluster Accuracy :  0.899364406779661
-------------------------------------------
Cluster Label :  3
Number of Accurate Labels :  865
Total number of samples :  2217
Incorrect Labels :  1352
Cluster Accuracy :  0.3901668921966622
-------------------------------------------
Cluster Label :  9
Number of Accurate Labels :  881
Total number of samples :  2852
Incorrect Labels :  1971
Cluster Accuracy :  0.30890603085553997
-------------------------------------------
Cluster Label :  6
Number of Accurate Labels :  803
Total number of samples :  1710
Incorrect Labels :  907
Cluster Accuracy :  0.4695906432748538
-------------------------------------------
Cluster Label :  1
Number of Accurate Labels :  1126
Total number of samples :  2277
Incorrect Labels :  1151
Cluster Accuracy :  0.4945103205972771
-------------------------------------------
Accuracy: 0.5125076587407988
Sum over all the Clusters: 5476
```

Q4 Finally, run the algorithm 10 times again with k = 10 and report the same information as above (iterations to convergence and number of wrongly clustered instances). But this time do not choose random instances for the cluster centroids. Randomly choose an instance that represents each of the digits and use them as the centroids. That is, one of the centroids will be a randomly chosen 0, another will be a randomly chosen 1, and so on. Do you observe any difference in the performance statistics? Why or why not?

Here you can see that I've taken instances from training data instead of random.

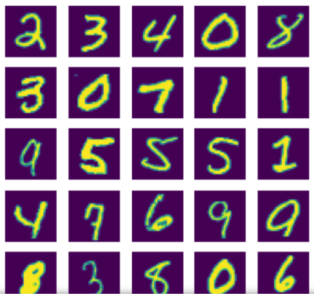
```
17
18      def initialize_centroids(self):
19  #         np.random.seed(np.random.randint(0, 100000))
20
21          random.choice(training_data)
22          self.centroids = []
23          for i in range(self.n_clusters):
24              rand_index = np.random.choice(range(len(self.fit_data)))
25              self.centroids.append(self.fit_data[rand_index])
26
```

```
In [2]: training_data = [line.split() for line in open('mnist_raw_data.txt').readlines()]
        training_labels=[int(line.strip('\n')) for line in open('mnist_labels.txt','r').readlines()]
        train_data= np.array(training_data, dtype=np.uint8)
        kmeans = KMeans(n_clusters=10,max_iter=200)
        iter_avg=[None for _ in range(10)]
        incorrect_avg=[None for _ in range(10)]
        #Part2 of the HW
        for i in range(10):
            print("Runtime - ",i+1)
            iter_avg[i],incorrect_avg[i]=kmeans.form_cluster(train_data,training_labels)
        print("Average number of iterations to converge : ",sum(iter_avg)//10)
        print("Average number of incorrects labels : ",sum(incorrect_avg)//10)
        kmeans.plot_imgs(train_data,25,True)
```

```
Epoch:  37 ==> Difference between centroids:  4.800442693949891e-06

Epoch:  38 ==> Difference between centroids:  5.9055795017860484e-09
Converged! ==>  7.330314129785505e-12
Accuracy: 0.6522635095403352
Average number of iterations to converge :  74
Average number of incorrects labels :  603
```

In [2]:
```
training_data = [line.split() for line in open('mnist_raw_data.txt').readlines()]
training_labels=[int(line.strip('\n')) for line in open('mnist_labels.txt','r').readlines()]
train_data= np.array(training_data, dtype=np.uint8)
kmeans = KMeans(n_clusters=10,max_iter=200)
iter_avg=[None for _ in range(10)]
incorrect_avg=[None for _ in range(10)]
#Part2 of the HW
for i in range(10):
    print("Runtime - ",i+1)
    iter_avg[i],incorrect_avg[i]=kmeans.form_cluster(train_data,training_labels)
print("Average number of iterations to converge : ",sum(iter_avg)//10)
print("Average number of incorrects labels : ",sum(incorrect_avg)//10)
kmeans.plot_imgs(train_data,25,True)
```

```
Accuracy: 0.6653992719154764
Average number of iterations to converge :  60
Average number of incorrects labels :  529
```



In [ ]:
```
print(sum(iter_avg)//10)
```

As we can see the number of iterations is less when the random instances are selected as centroid. Below is the snippet how the performance metrics looks when the centroids are selected randomly based on training data instances.

In [5]:
```
incorrect_all=[0 for _ in range(10)]
for i in range(10):
    print("Cluster Label : ",kmeans.clusters_info[i][0])
    print("Number of Accurate Labels : ",kmeans.clusters_info[i][1])
    print("Total number of samples : ",kmeans.clusters_info[i][2])
    print("Incorrect Labels : ",kmeans.clusters_info[i][2] - kmeans.clusters_info[i][1])
    print("Cluster Accuracy : ",kmeans.clusters_info[i][3])
    print("-----------------------------------------")
    incorrect_all.append(kmeans.clusters_info[i][2] - kmeans.clusters_info[i][1])
print('Accuracy:',kmeans.accuracy)
print('Sum over all the Clusters:',sum(incorrect_all))
```

```
Cluster Label :  5
Number of Accurate Labels :  259
Total number of samples :  853
Incorrect Labels :  594
Cluster Accuracy :  0.30363423212192264
-----------------------------------------
Cluster Label :  0
Number of Accurate Labels :  711
Total number of samples :  764
Incorrect Labels :  53
Cluster Accuracy :  0.930628272251309
-----------------------------------------
Cluster Label :  7
Number of Accurate Labels :  710
Total number of samples :  787
Incorrect Labels :  77
Cluster Accuracy :  0.9021601016518425
-----------------------------------------
Cluster Label :  8
Number of Accurate Labels :  622
Total number of samples :  922
Incorrect Labels :  300
Cluster Accuracy :  0.6746203904555315
-----------------------------------------
```


```
Cluster Accuracy :  0.6746203904555315
-----------------------------------------
```


```
Cluster Label :  3
Number of Accurate Labels :  700
Total number of samples :  1230
Incorrect Labels :  530
Cluster Accuracy :  0.5691056910569106
-----------------------------------------
Cluster Label :  4
Number of Accurate Labels :  475
Total number of samples :  1103
Incorrect Labels :  628
Cluster Accuracy :  0.43064369900271987
-----------------------------------------
Cluster Label :  2
Number of Accurate Labels :  724
Total number of samples :  818
Incorrect Labels :  94
Cluster Accuracy :  0.8850855745721271
-----------------------------------------
Cluster Label :  6
Number of Accurate Labels :  654
Total number of samples :  721
Incorrect Labels :  67
Cluster Accuracy :  0.9070735090152566
-----------------------------------------
```


```
Cluster Label :  2
Number of Accurate Labels :  724
Total number of samples :  818
Incorrect Labels :  94
Cluster Accuracy :  0.8850855745721271
-----------------------------------------
Cluster Label :  6
Number of Accurate Labels :  654
Total number of samples :  721
Incorrect Labels :  67
Cluster Accuracy :  0.9070735090152566
-----------------------------------------
```


```
Cluster Label :  1
Number of Accurate Labels :  1102
Total number of samples :  1635
Incorrect Labels :  533
Cluster Accuracy :  0.674006116207951
-----------------------------------------
Cluster Label :  4
Number of Accurate Labels :  440
Total number of samples :  1167
Incorrect Labels :  727
Cluster Accuracy :  0.37703513281919454
-----------------------------------------
Accuracy: 0.6653992719154764
Sum over all the Clusters: 3603
```

References

https://medium.datadriveninvestor.com/k-means-clustering-for-imagery-analysis-56c9976f16b6
https://medium.com/@joel_34096/k-means-clustering-for-image-classification-a648f28bdc47
https://johnloeber.com/docs/kmeans.html
https://sbrouil.github.io/ml-sandbox/MNIST+Kaggle+Digit+clusterization+using+KMeans.html