assignment04

October 11, 2018

1 This script demonstrates MNIST clustering

2 Name : Ji-Su Lee

3 Student ID : 20141718

4 github link : https://github.com/Jisu-Lee/HII

5 import packages for plotting graphs and manipulating data:

```
In [63]: import numpy as np
         import matplotlib.pyplot as plt
         import random
         import matplotlib.cm as cm
```

6 declare variables to get input

```
In [64]: file_data   = "mnist_test.csv"
         handle_file = open(file_data, "r")
         data        = handle_file.readlines()
         handle_file.close()

         size_row    = 28    # height of the image
         size_col    = 28    # width of the image

         num_image   = len(data)
         count       = 0     # count for the number of images
```

7 define function to compute distance between two vectors

```
In [66]: def distance(x, y):

             d = (x - y) ** 2
             s = np.sum(d)
             r = np.sqrt(s)
```

```
            return(r)
```

8 normalize the values of the input data to be [0, 1]

```
In [67]: def normalize(data):

            data_normalized = (data - min(data)) / (max(data) - min(data))

            return(data_normalized)
```

9 function to make distance list

```
In [68]: def makeDistList(nCluster, dataList, cenList):
            distList = []
            for i in range(len(dataList)):
                cenDist = []
                for j in range(nCluster):
                    cenDist.append(distance(dataList[i], cenList[j]))
                distList.append(cenDist)
            return distList
```

10 define function to initialise label

```
In [70]: def initialiseLabel(nCluster, nData):
            a = []
            for i in range(nData):
                a.append(random.randint(1, nCluster)-1)

            return a
```

11 define function to compute centroid

```
In [71]: def computeCentroid(nCluster, nData, dataList, labelList):
            addData = []
            numLabel = []
            res = []
            #initialise list [sum_of_data, num_of_data]
            addData = np.zeros((nCluster, len(dataList[0])), dtype=float)
            numLabel = np.zeros(nCluster, dtype=int)
            for i in range(nData):
                numLabel[labelList[i]] += 1
                addData[labelList[i]]= np.add(addData[labelList[i]], dataList[i])
            # get centroid by calculating mean value
            for i in range(nCluster):
                if(numLabel[i] == 0):
```

```
                res.append(dataList[0])
            else:
                res.append(np.divide(addData[i],numLabel[i]))
        return res
```

12 define fuction to assign label

```
In [73]: def assignLabel(nAllPoints, distList):
            res = []
            for i in range(nAllPoints):
                res.append(0)
            for i in range(nAllPoints):
                res[i] = distList[i].index(np.amin(distList[i]))
            return res
```

13 define function to compute energy

```
In [95]:  # param : number of all points, list of points, list of centroid for each clusters,
          # list of label for each points
         def computeEnergy(nAllPoints, dataList, cenList, labelList):
            energy = 0
            for i in range(nAllPoints):
                energy += np.square(distance(dataList[i], \
                                             cenList[labelList[i]]))
            energy = float(energy)/nAllPoints
            return energy
```

14 define function to compute accuracy

```
In [77]: def computeAccuracy(nCluster, nData, real_labels, cluster_labels):
            res = []
            acc = 0
            for i in range(nCluster):
                res.append([])
            for i in range(nData):
                res[cluster_labels[i]].append(real_labels[i])

            for i in range(nCluster):
                freq = np.bincount(res[i]).max()
                acc += freq/float(len(res[i]))
            acc = acc/float(nCluster)

            return acc
```

15 make a matrix each column of which represents an images in a vector form

```
In [79]: list_image  = np.empty((size_row * size_col, num_image), dtype=float)
         list_label  = np.empty(num_image, dtype=int)

         for line in data:

             line_data   = line.split(',')
             label       = line_data[0]
             im_vector   = np.asfarray(line_data[1:])
             im_vector   = normalize(im_vector)

             list_label[count]       = label
             list_image[:, count]    = im_vector

             count += 1
```

16 calculates initial centroid

```
In [85]: nc = 10  #k = 10
         nd = num_image
         e = []
         acc = []
         data_list = np.transpose(list_image)
         test_label = initialiseLabel(nc, nd)
         cen_list = computeCentroid(nc, nd, data_list, test_label)
         dist_list = makeDistList(nc, data_list, cen_list)
         test_label = assignLabel(nd, dist_list)
         e.append(computeEnergy(nd, data_list, cen_list, test_label))
         acc.append(computeAccuracy(nc, nd, list_label, test_label))
```

17 shows initial centroid

```
In [86]: f1 = plt.figure(1)
         print("Initial Centroid")

         # shows centroid
         for i in range(nc):
             im_vector   = cen_list[i]
             im_matrix   = im_vector.reshape((size_row, size_col))

             plt.subplot(10, 15, i+1)
         #    plt.title(label, y=1.08)
             plt.imshow(im_matrix, cmap='Greys', interpolation='None')
```

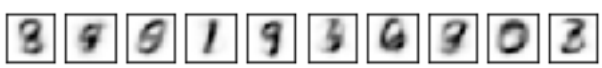
```
            frame   = plt.gca()
            frame.axes.get_xaxis().set_visible(False)
            frame.axes.get_yaxis().set_visible(False)

        plt.show()
```

```
Initial Centroid
```

18 shows centroids for each attempt. Last one is the final centroid

```
In [89]: for i in range(10):
            cen_list = computeCentroid(nc, nd, data_list, test_label)
            dist_list = makeDistList(nc, data_list, cen_list)
            test_label = assignLabel(nd, dist_list)
            e.append(computeEnergy(nd, data_list, cen_list, test_label))
            acc.append(computeAccuracy(nc, nd, list_label, test_label))
            f1 = plt.figure(1)

            print("Centroid - ", i+1)

            # shows centroid
            for i in range(nc):
                im_vector   = cen_list[i]
                im_matrix   = im_vector.reshape((size_row, size_col))

                plt.subplot(10, 15, i+1)
                plt.imshow(im_matrix, cmap='Greys', interpolation='None')

                frame   = plt.gca()
                frame.axes.get_xaxis().set_visible(False)
                frame.axes.get_yaxis().set_visible(False)

            plt.show()
```

```
Centroid -  1
```



Centroid - 2

Centroid - 3

Centroid - 4

Centroid - 5

Centroid - 6

```
Centroid -  7
```

```
Centroid -  8
```

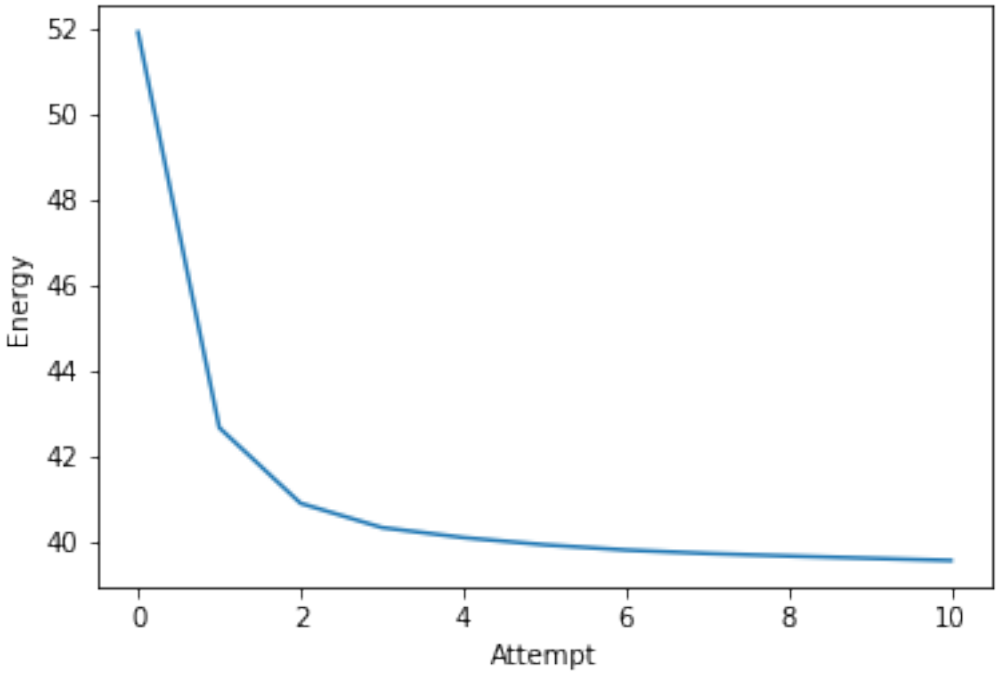
```
Centroid -  9
```

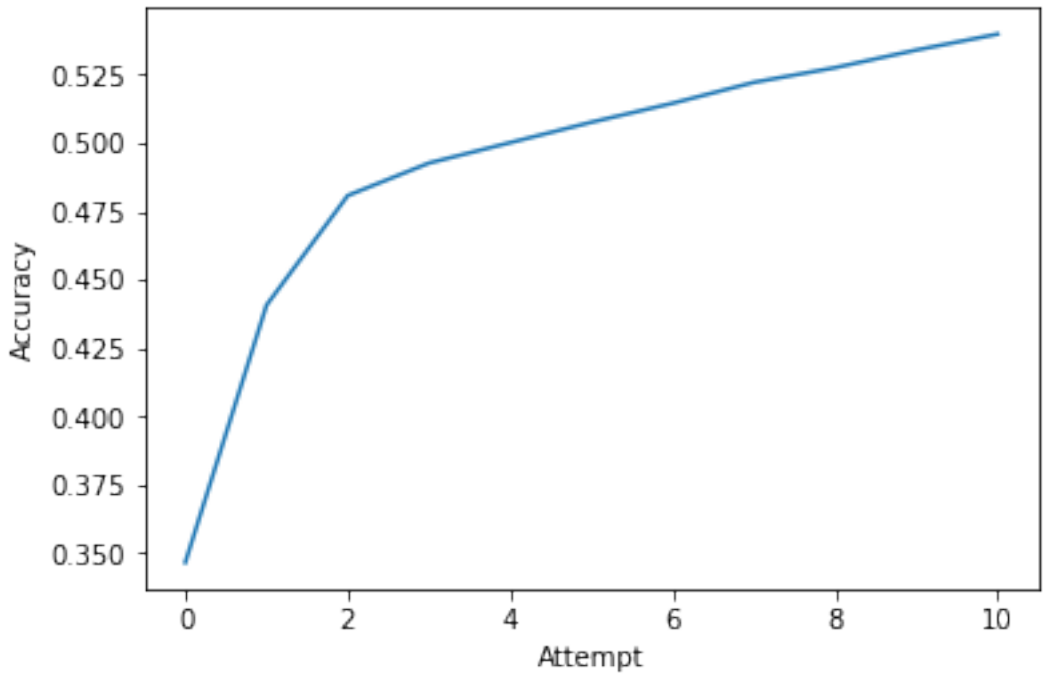
```
Centroid -  10
```

19 shows Energy graph and Accuracy graph

```
In [92]: #energy graph
         x = np.arange(0,11)
         plt.plot(x,e)
         plt.xlabel('Attempt')
         plt.ylabel('Energy')
```

```
plt.show()
#accuracy graph
x = np.arange(0,11)
plt.plot(x,acc)
plt.xlabel('Attempt')
plt.ylabel('Accuracy')
plt.show()
```





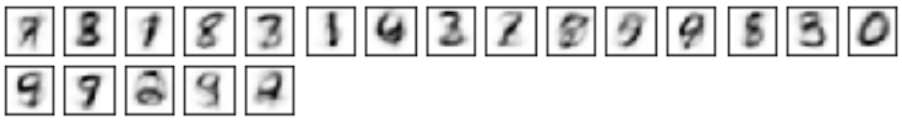
20 Let's try with k = 20

```
In [93]: nc = 20  #k = 10
         nd = num_image
         e = []
         acc = []
         data_list = np.transpose(list_image)
         test_label = initialiseLabel(nc, nd)

         for i in range(10):
             cen_list = computeCentroid(nc, nd, data_list, test_label)
             dist_list = makeDistList(nc, data_list, cen_list)
             test_label = assignLabel(nd, dist_list)
             e.append(computeEnergy(nd, data_list, cen_list, test_label))
             acc.append(computeAccuracy(nc, nd, list_label, test_label))
             f1 = plt.figure(1)

             print("Centroid - ", i)

             # shows centroid
             for i in range(nc):
                 im_vector   = cen_list[i]
                 im_matrix   = im_vector.reshape((size_row, size_col))
```

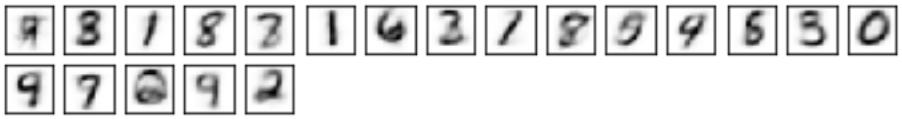
```
                plt.subplot(10, 15, i+1)
                plt.imshow(im_matrix, cmap='Greys', interpolation='None')

                frame   = plt.gca()
                frame.axes.get_xaxis().set_visible(False)
                frame.axes.get_yaxis().set_visible(False)

            plt.show()

        #energy graph
        x = np.arange(0,10)
        plt.plot(x,e)
        plt.xlabel('Attempt')
        plt.ylabel('Energy')
        plt.show()
        #accuracy graph
        x = np.arange(0,10)
        plt.plot(x,acc)
        plt.xlabel('Attempt')
        plt.ylabel('Accuracy')
        plt.show()
```

```
Centroid -  0
```

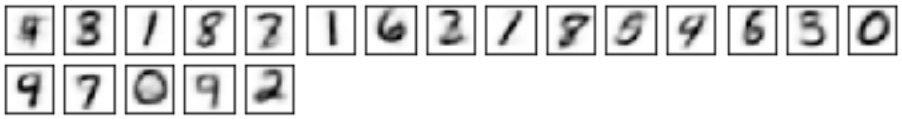
```
Centroid -  1
```



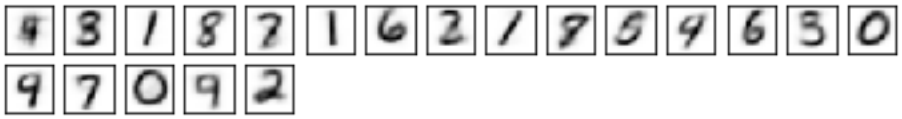
```
Centroid -  2
```



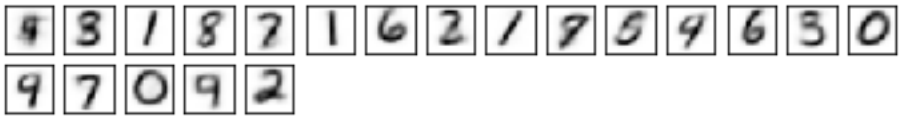
Centroid - 3



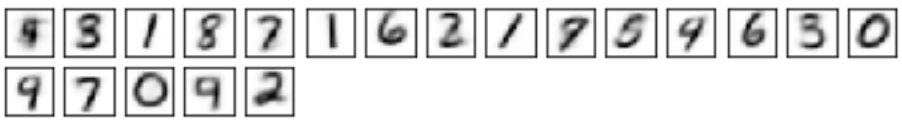
Centroid - 4



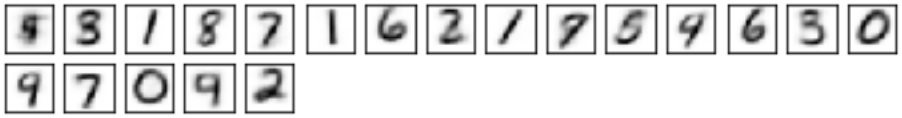
Centroid - 5



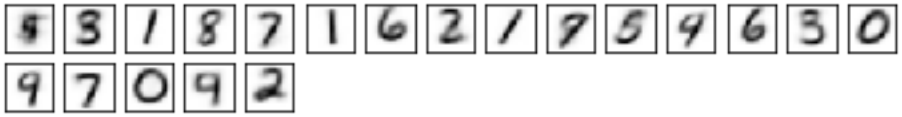
Centroid - 6



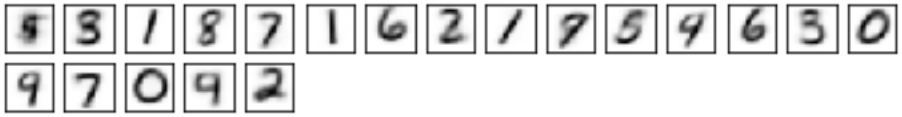
Centroid - 7

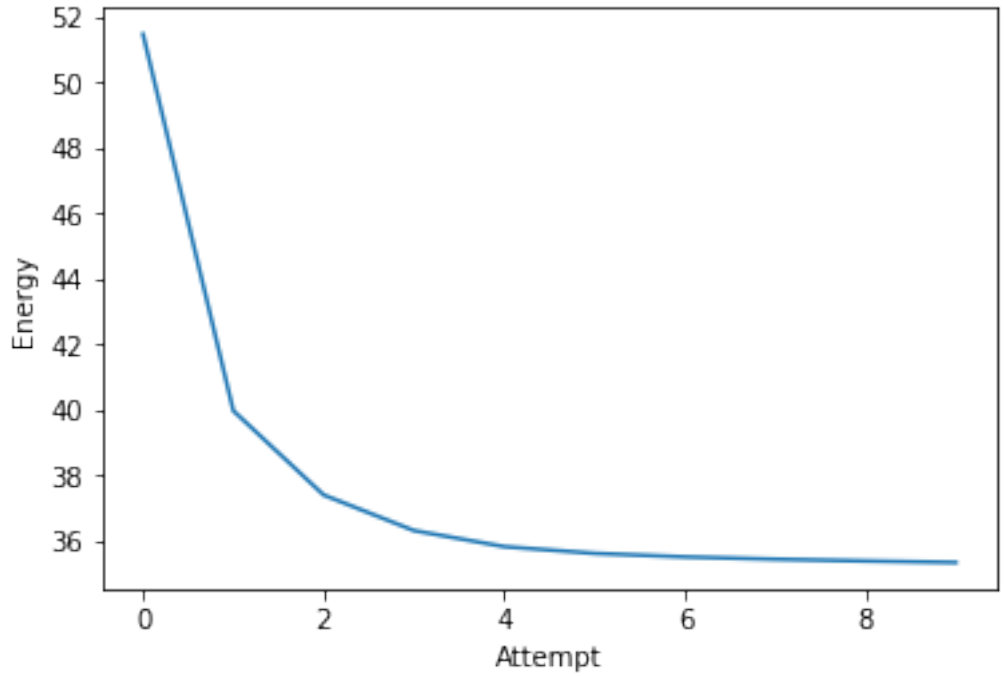


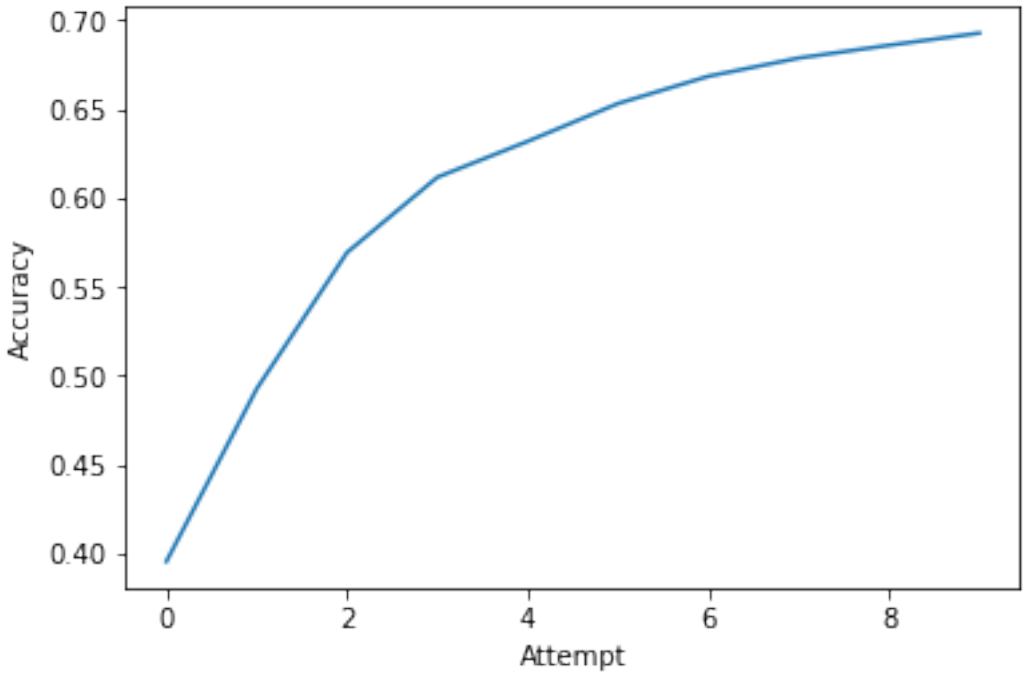
Centroid - 8



Centroid - 9




52
50
48
46
44
42
40
38
36
Energy
0
2
4
6
8
Attempt


0.70
0.65
0.60
0.55
0.50
0.45
0.40
Accuracy
0
2
4
6
8
Attempt

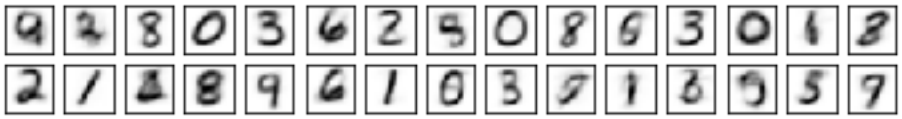
21 How about k = 30

```
In [94]: nc = 30  #k = 10
         nd = num_image
         e = []
         acc = []
         data_list = np.transpose(list_image)
         test_label = initialiseLabel(nc, nd)

         for i in range(10):
             cen_list = computeCentroid(nc, nd, data_list, test_label)
             dist_list = makeDistList(nc, data_list, cen_list)
             test_label = assignLabel(nd, dist_list)
             e.append(computeEnergy(nd, data_list, cen_list, test_label))
             acc.append(computeAccuracy(nc, nd, list_label, test_label))
             f1 = plt.figure(1)

             print("Centroid - ", i)

             # shows centroid
             for i in range(nc):
                 im_vector   = cen_list[i]
                 im_matrix   = im_vector.reshape((size_row, size_col))

                 plt.subplot(10, 15, i+1)
                 plt.imshow(im_matrix, cmap='Greys', interpolation='None')

                 frame   = plt.gca()
                 frame.axes.get_xaxis().set_visible(False)
                 frame.axes.get_yaxis().set_visible(False)

             plt.show()

         #energy graph
         x = np.arange(0,10)
         plt.plot(x,e)
         plt.xlabel('Attempt')
         plt.ylabel('Energy')
         plt.show()
         #accuracy graph
         x = np.arange(0,10)
         plt.plot(x,acc)
         plt.xlabel('Attempt')
         plt.ylabel('Accuracy')
         plt.show()
```

```
Centroid -  0
```

Centroid - 1

Centroid - 2



Centroid - 3

Centroid - 4

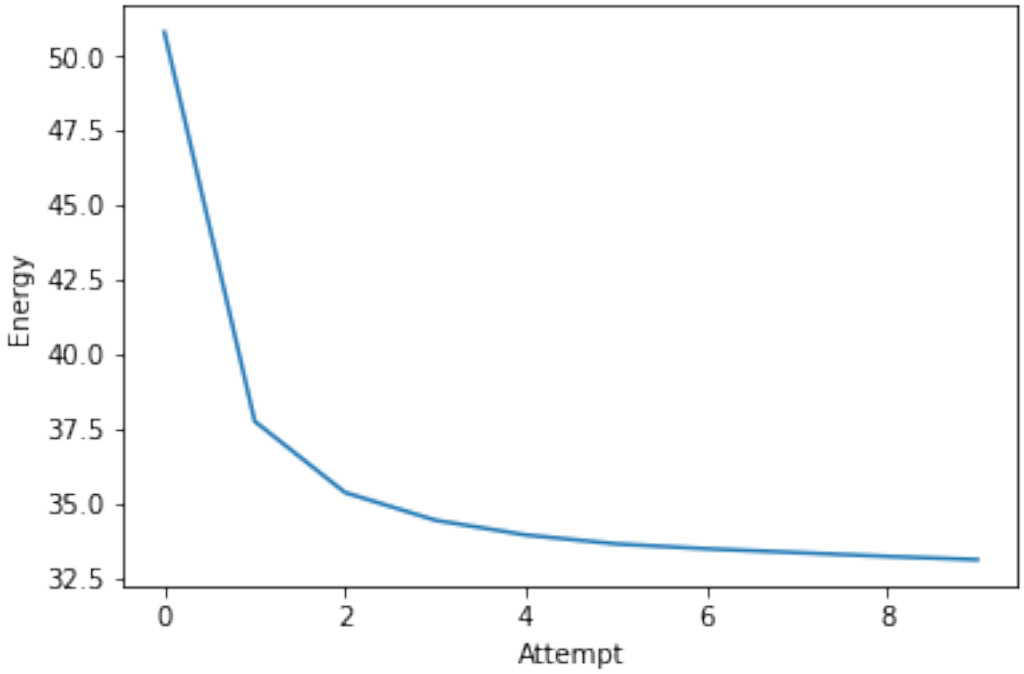
Centroid - 5

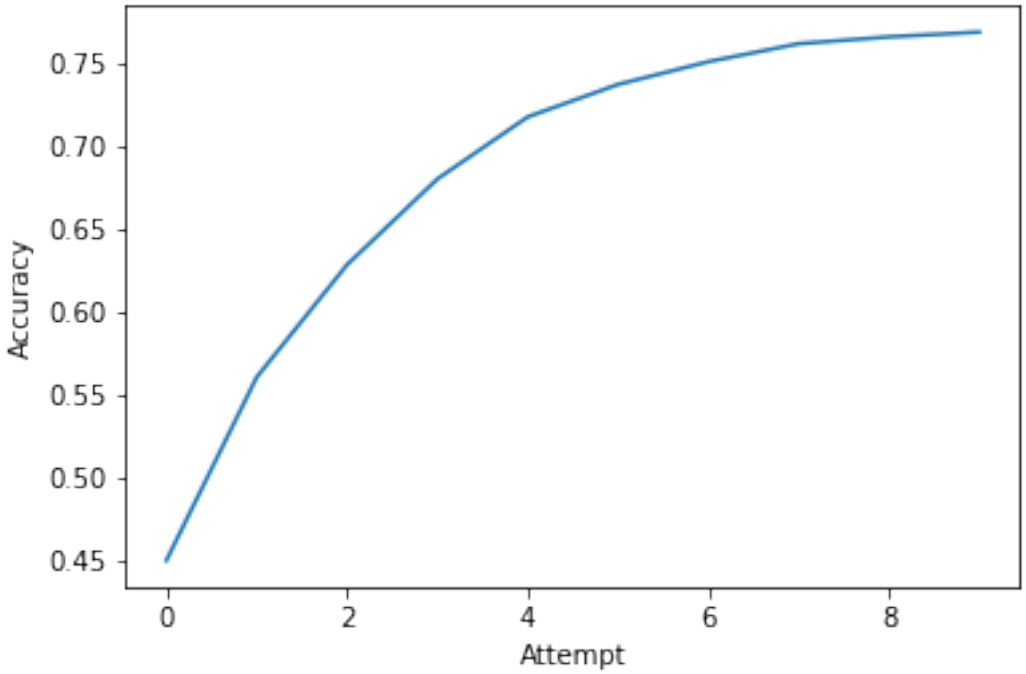
Centroid - 6

Centroid - 7

Centroid - 8

```
Centroid -  9
```




0.75
0.70
0.65
0.60
0.55
0.50
0.45
Accuracy
0
2
4
6
8
Attempt