Assignment04_20133096_HyunjaeLee

April 3, 2019

''' [K-means clustering]

1. Apply K-means clustering to MNIST training dataset with different K = 5, 10, 15, 20 and present the following results for each K.
2. Visualize K centroid images for each category.
3. Plot the training energy per optimization iteration.
4. Plot the training accuracy per optimization iteration.
5. Plot the testing accuracy per optimization iteration.

- (training energy) is computed on the training dataset.
- (training accuracy) is computed on the training dataset.
- (testing accuracy) is computed on the testing dataset. '''

[energy]
$\sum\{k = 1\}\hat{}K \mid x\_i - c\{k\_i\}\blacksquare 2 where k_i$ denotes the category of x_i, and c_{k_i} denotes the centroid of category x_i.
[accuracy]
$\sum_{k=1}^{K} m_k \frac{}{N where N}$ denotes the total number of data, and m_k denotes the number of data with majority for category k.
First, I defined varaibles

```
<li> average[number] : when central point compute is done, it will update</li><li> Energy , Accu
```

```
In [254]: #20133096 Hyunjae Lee
          import matplotlib.pyplot as plt
          import numpy as np
          import random
          import itertools

          %matplotlib inline

          # Setup variables
          numOfClusters = [5,10,15,20] # K
          size_row        = 28    # height of the image
          size_col          = 28    # width of the image
          dimension = size_row * size_row # dimension
          count = 0
```

```
        # avg for centroid of each cluster
        average5 = np.zeros((dimension, numOfClusters[0]), dtype=float)
        avgrage10 = np.zeros((dimension, numOfClusters[1]), dtype=float)
        avgrage15 = np.zeros((dimension, numOfClusters[2]), dtype=float)
        avgrage20 = np.zeros((dimension, numOfClusters[3]), dtype=float)
        average5_test = np.zeros((dimension, numOfClusters[0]), dtype=float)
        avgrage10_test = np.zeros((dimension, numOfClusters[1]), dtype=float)
        avgrage15_test = np.zeros((dimension, numOfClusters[2]), dtype=float)
        avgrage20_test = np.zeros((dimension, numOfClusters[3]), dtype=float)

        # Array for plotting
        Energy = [] # energy of each iteration
        Accuracy = [] # accuracy of each iteration

        # some variables
        iteration = 0
        threshold = 0.005
```

Second, I defined functions that I will use

```
<li> firstLabel(numOfClusters, num_image): it will label randomly for the first time</li>
<li> assignLabel(list_distance, list_random,num_image): it will update labels</li>
```

Distance(numOfClusters,list_distance, avg): it will calculate distance between an image vector and a set of central points

centroid(Clusters,num_image,list_random,list_image): it will update central points

FunctionEnergy(avg, num_image, list_random, list_image): it will update energy values

```
<li> FunctionAccuracy(Clusters, num_image, list_random, list_label): it will update accuracy val
```

record(avg, cluster, num_image, list_random, list_image, list_label): it will save energy and accuracy values

```
<li> plot(Cluster, accracy, average, iteration, Ener): it will plot three different graphs </li>
```

```
In [255]: # read data from .csv file
          def readData(nameOfFile):
              file_data = nameOfFile + ".csv"
              handle_file = open(file_data, "r")
              data = handle_file.readlines()
              handle_file.close()

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

              # make a matrix each column of which represents an images in a vector form
              list_image  = np.empty((size_row * size_col, num_image), dtype=float)
              list_label  = np.empty(num_image, dtype=int)
```

```
    # list_image : it contains a series of images (784 , 60000) in case of train_csv
    # list_label : each index indicates its label (0 ~ 9)
    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

    return list_image, list_label, num_image


# normalize the values of the input data to be [0, 1]
def normalize(data):

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

    return(data_normalized)

# Initialize first labels
def firstLabel(numOfClusters, num_image):
    return np.random.randint(numOfClusters, size = num_image)

# Distance between lists
def Distance(numOfClusters,list_distance, avg, num_image):
    # L2 Norm
    for k in range(numOfClusters):
        for img in range(num_image):
            list_distance[k][img] = sum((list_image[:,img] - avg[:,k])**2)

def assignLabel(list_distance, list_random,num_image):
    for img in range(num_image):
        list_random[img] = np.argmin(list_distance[:,img])

def centroid(Clusters,num_image,list_random,list_image):
    #num = np.zeros((Clusters))
    returnvalue = np.zeros((dimension, Clusters))

    for k in range(Clusters):
        cnt = 0
        for i in range(num_image):
            if(list_random[i] == k):
```

```
                returnvalue[:, k] += list_image[:, i]
                cnt += 1
        returnvalue[:, k] /= cnt

    return returnvalue

def FunctionEnergy(avg, num_image, list_random, list_image):
    energy = 0

    for i in range(num_image):
        energy += sum((avg[:,list_random[i]] - list_image[:,i])**2)

    energy /= num_image

    return energy

def FunctionAccuracy(Clusters, num_image, list_random, list_label):

    answer = 0
    num = np.zeros(Clusters)

    # count how many elements are in the same cluster
    for k in range(Clusters):

        list_accuracy = []
        maxnumber = 0
        compare = 0
        for i in range(num_image):
            if(list_random[i] == k):
                num[k] += 1
                list_accuracy.append(list_label[i])

        for a, v in itertools.groupby(sorted(list_accuracy)):

            compare = len((list(v)))
            if (compare > maxnumber):
                maxnumber = compare
        answer += maxnumber
    print("answer is " + str(answer))
    return answer/num_image

def record(avg, cluster, num_image, list_random, list_image, list_label,energy,accurac
    eng = FunctionEnergy(avg, num_image, list_random, list_image)
    energy.append(eng)

    acc = FunctionAccuracy(cluster,num_image, list_random, list_label)
    accuracy.append(acc)
```

```
def plot(Cluster, accuracy, average, iteration, energy, test):


    if test == False:
    # Averaged Image
        plt.figure(1)

        for i in range(Cluster):
            plt.subplot(2, 10, i + 1)

            plt.imshow(average[:, i].reshape((size_row, size_col)), cmap='Greys', inte

            frame = plt.gca()
            frame.axes.get_xaxis().set_visible(False)
            frame.axes.get_yaxis().set_visible(False)
        plt.title("K = " + str(Cluster))
        plt.show()

    # Energy
        plt.figure(3)
        #x = np.arange(iteration+1)
        plt.plot(energy,"b")
        plt.title("Energy graph when K = " + str(Cluster))
        plt.grid(True)
        plt.show()


    # delete assigned zero values that happend when it was initialized
    for i in range(len(accuracy)):
        if(accuracy[i] == 0):
            accuracy.remove(0.0)
    # Accuracy

    plt.figure(4)
    #x = np.arange(len(accuracy))
    plt.plot(accuracy,"b")
    if(test==True):
        plt.title("[TEST]Accuracy graph when K = " + str(Cluster))
    else:
        plt.title("Accuracy graph when K = " + str(Cluster))

    plt.grid(True)
    plt.show()
```

Third, I defined several arrays to contain values depending on K (from K-means)

```
In [256]: # start Program #
          list_image_test, list_label_test, num_image_test = readData("mnist_test")
```

```
list_image, list_label, num_image = readData("mnist_train")
Energy = []
Accracy = []

Energe_test = []
Accracy_test = []

# lists for initializing first labels
list_random5 = firstLabel(numOfClusters[0], num_image)
list_random10 = firstLabel(numOfClusters[1], num_image)
list_random15 = firstLabel(numOfClusters[2], num_image)
list_random20 = firstLabel(numOfClusters[3], num_image)
list_random5_test = firstLabel(numOfClusters[0], num_image_test)
list_random10_test = firstLabel(numOfClusters[1], num_image_test)
list_random15_test = firstLabel(numOfClusters[2], num_image_test)
list_random20_test = firstLabel(numOfClusters[3], num_image_test)


# lists for containg distance depending on K
list_distance5 = np.zeros((numOfClusters[0], num_image))
list_distance10 = np.zeros((numOfClusters[1], num_image))
list_distance15 = np.zeros((numOfClusters[2], num_image))
list_distance20 = np.zeros((numOfClusters[3], num_image))
list_distance5_test = np.zeros((numOfClusters[0], num_image_test))
list_distance10_test = np.zeros((numOfClusters[1], num_image_test))
list_distance15_test = np.zeros((numOfClusters[2], num_image_test))
list_distance20_test = np.zeros((numOfClusters[3], num_image_test))

# average for each cluster
average5 = centroid(numOfClusters[0], num_image, list_random5,list_image)
average10 = centroid(numOfClusters[1], num_image, list_random10,list_image)
average15 = centroid(numOfClusters[2], num_image, list_random15,list_image)
average20 = centroid(numOfClusters[3], num_image, list_random20,list_image)
average5_test = centroid(numOfClusters[0], num_image_test, list_random5_test,list_imag
average10_test = centroid(numOfClusters[1], num_image_test, list_random10_test,list_im
average15_test = centroid(numOfClusters[2], num_image_test, list_random15_test,list_im
average20_test = centroid(numOfClusters[3], num_image_test, list_random20_test,list_im
```

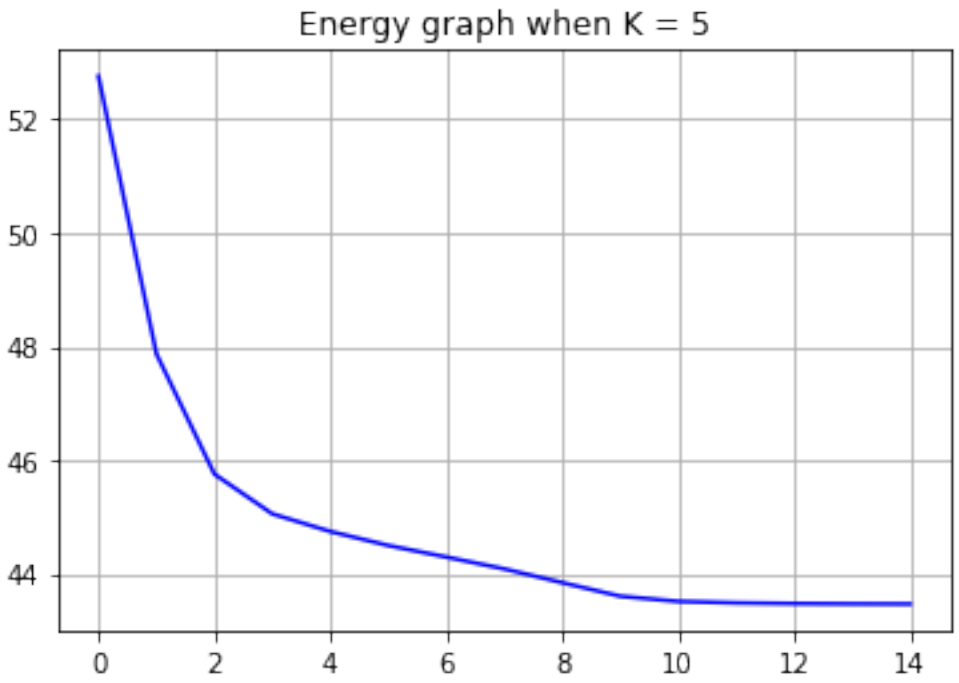
Then, Im running program
K = 5

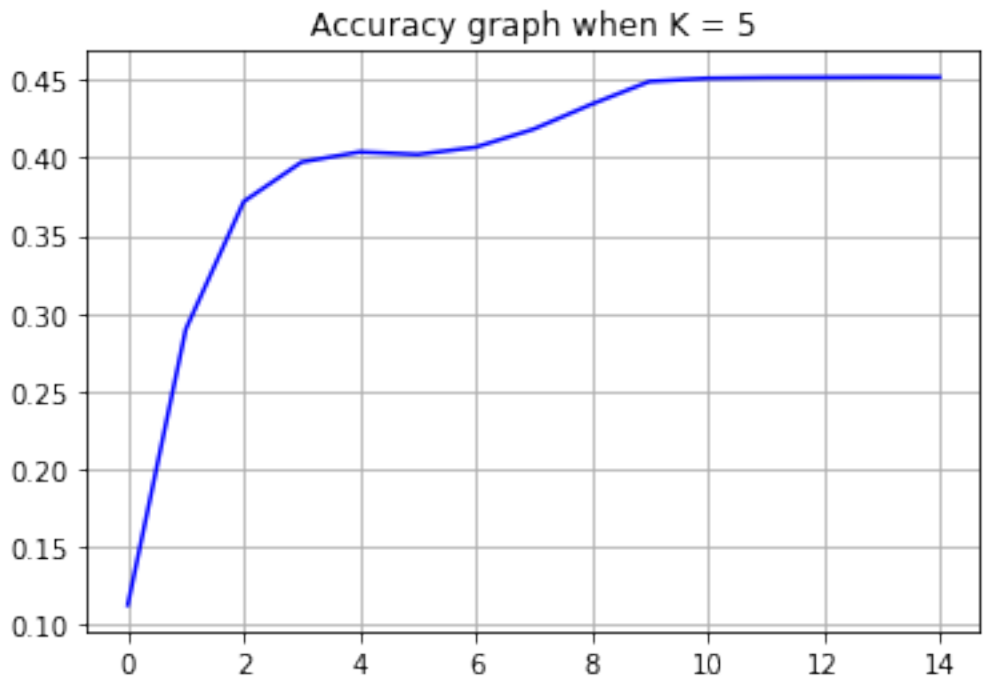
```
In [242]: iteration = 0
          iteration_test = 0
          Energy = []
          Accuracy = []

          Energy_test = []
          Accuracy_test = []
```

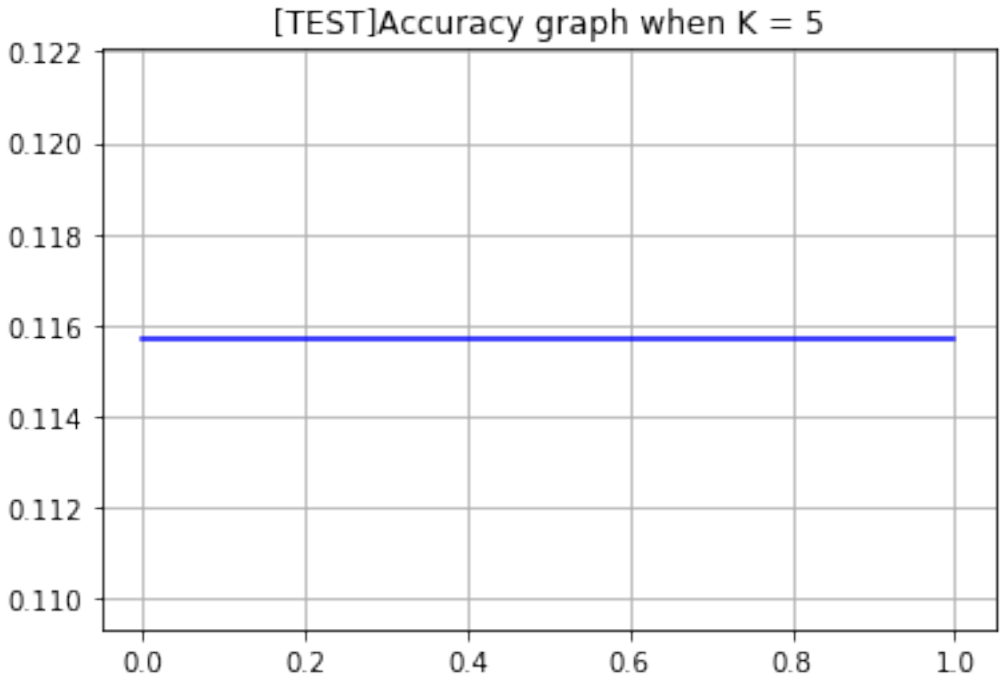
```
        record(average5_test,numOfClusters[0], num_image_test, list_random5_test, list_image_t
        record(average5,numOfClusters[0], num_image, list_random5, list_image,list_label,Energ

        while True:
            # k = 5
            Distance(numOfClusters[0],list_distance5, average5,num_image)
            assignLabel(list_distance5, list_random5,num_image)
            average5 = centroid(numOfClusters[0], num_image, list_random5,list_image)
            record(average5,numOfClusters[0], num_image, list_random5, list_image,list_label,E
            iteration += 1

            if Energy[-2] - Energy[-1] < threshold:
                print('done')
                break

        while True:
            # k = 5 # TEST
            Distance(numOfClusters[0],list_distance5_test, average5_test,num_image_test)
            assignLabel(list_distance5_test, list_random5,num_image_test)
            average5_test = centroid(numOfClusters[0], num_image_test, list_random5_test,list_
            record(average5_test,numOfClusters[0], num_image_test, list_random5_test, list_ima
            iteration_test += 1

            if Energy_test[-2] - Energy_test[-1] < threshold:
                print('done')
                break
```

```
answer is 1157
answer is 6742
answer is 17378
answer is 22311
answer is 23837
answer is 24225
answer is 24128
answer is 24408
answer is 25106
answer is 26068
answer is 26947
answer is 27069
answer is 27097
answer is 27105
answer is 27114
answer is 27111
done
answer is 1157
done
```

In [243]:
```
plot(numOfClusters[0], Accuracy, average5, iteration, Energy, False)
plot(numOfClusters[0], Accuracy_test, average5_test, iteration_test, Energy_test, True
```




Accuracy graph when K = 5
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0
2
4
6
8
10
12
14


[TEST]Accuracy graph when K = 5
0.122
0.120
0.118
0.116
0.114
0.112
0.110
0.0
0.2
0.4
0.6
0.8
1.0

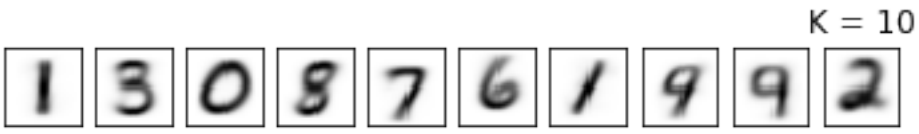
K = 10

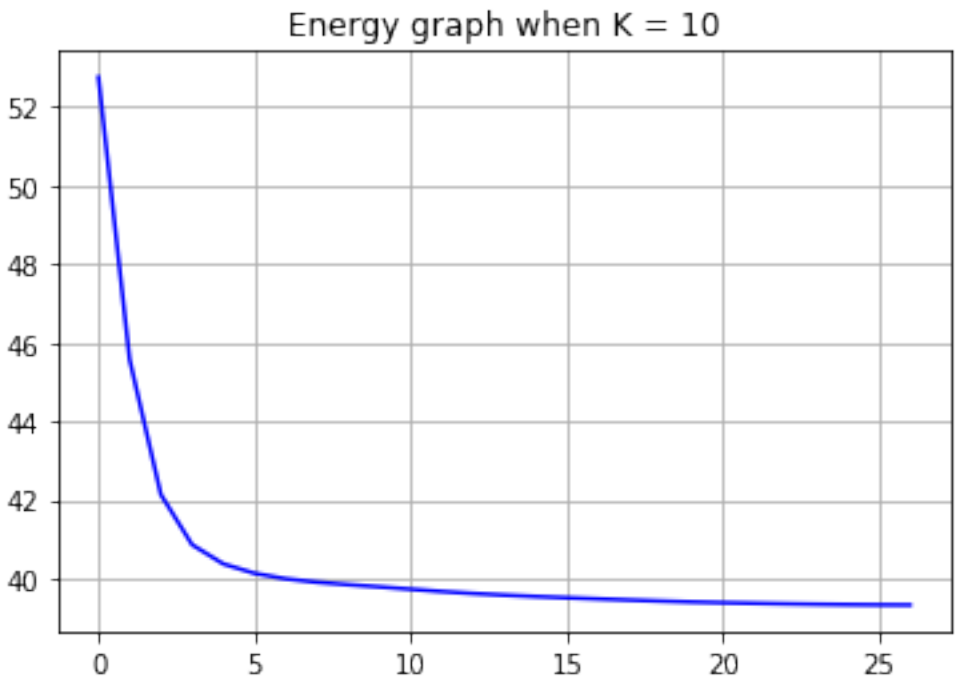
In [229]:
```
iteration = 0
iteration_test = 0
Energy = []
Accuracy = []

Energy_test = []
Accuracy_test = []

record(average10_test,numOfClusters[1], num_image_test, list_random10_test, list_image
record(average10,numOfClusters[1], num_image, list_random10, list_image,list_label,Ene

while True:
    # k = 10
    Distance(numOfClusters[1],list_distance10, average10,num_image)
    assignLabel(list_distance10, list_random10,num_image)
    average10 = centroid(numOfClusters[1], num_image, list_random10,list_image)
    record(average10,numOfClusters[1], num_image, list_random10, list_image,list_label
    iteration += 1

    if Energy[-2] - Energy[-1] < threshold:
        print('done')
        break

while True:
    # k = 10 # TEST
    Distance(numOfClusters[1],list_distance10_test, average10_test,num_image_test)
    assignLabel(list_distance10_test, list_random10,num_image_test)
    average10_test = centroid(numOfClusters[1], num_image_test, list_random10_test,lis
    record(average10_test,numOfClusters[1], num_image_test, list_random10_test, list_i
    iteration_test += 1

    if Energy_test[-2] - Energy_test[-1] < threshold:
        print('done')
        break
```

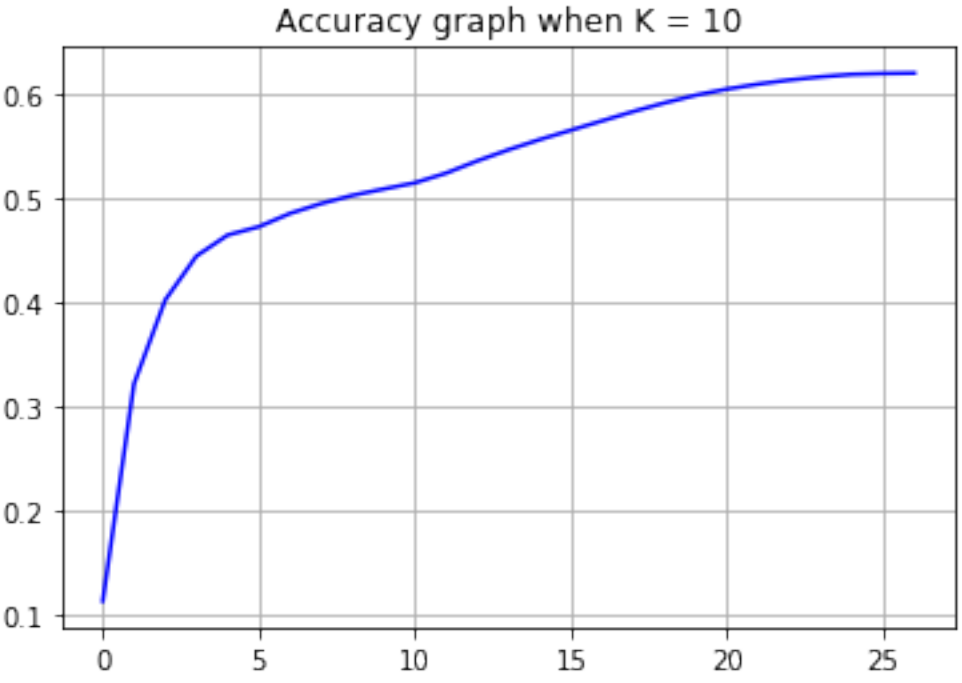
```
answer is 1145
answer is 6760
answer is 19261
answer is 24097
answer is 26629
answer is 27837
answer is 28316
answer is 29083
answer is 29653
answer is 30123
answer is 30489
answer is 30845
answer is 31404
```

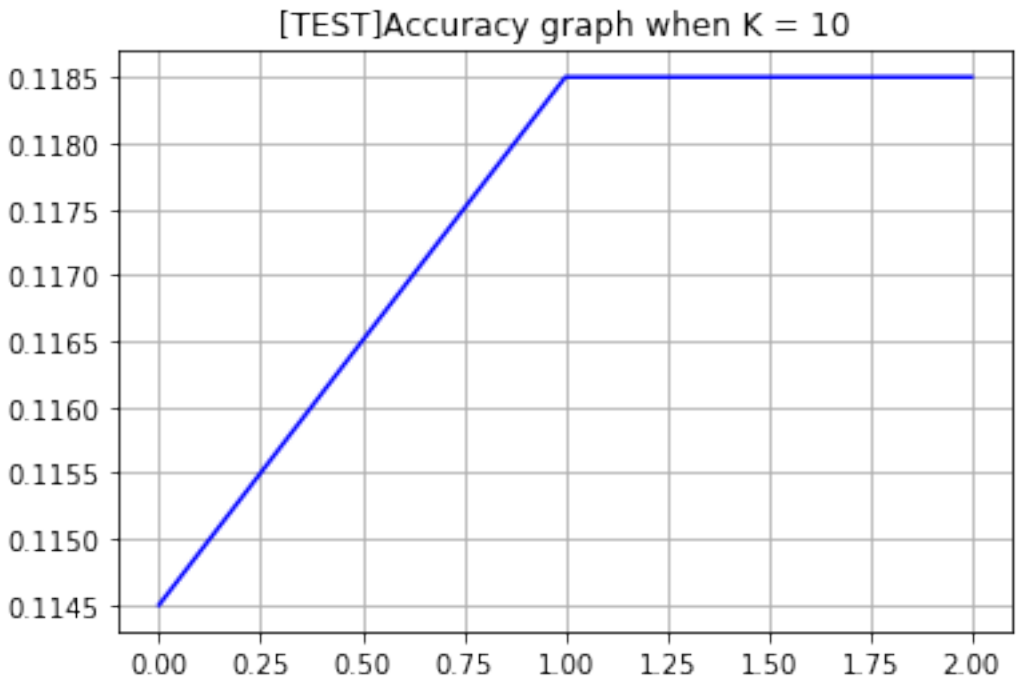
```
answer is 32111
answer is 32752
answer is 33338
answer is 33878
answer is 34417
answer is 34957
answer is 35451
answer is 35895
answer is 36246
answer is 36532
answer is 36769
answer is 36952
answer is 37091
answer is 37141
answer is 37165
done
answer is 1185
answer is 1185
done
```

In [231]:
```
plot(numOfClusters[1], Accuracy, average10, iteration, Energy, False)
plot(numOfClusters[1], Accuracy_test, average10_test, iteration_test, Energy_test, Tru
```




Energy graph when K = 10
52
50
48
46
44
42
40
0
5
10
15
20
25


Accuracy graph when K = 10
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
15
20
25



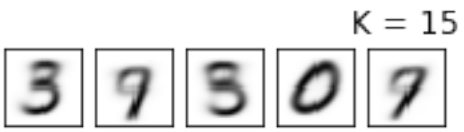
K = 15

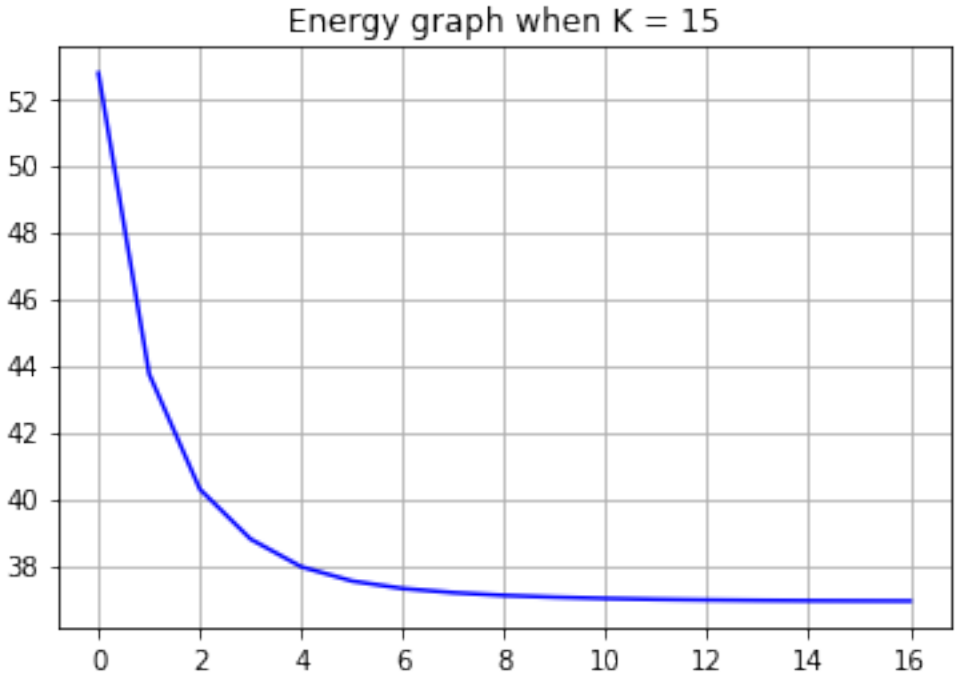
```
In [244]: iteration = 0
          iteration_test = 0
          Energy = []
          Accuracy = []

          Energy_test = []
          Accuracy_test = []

          record(average15_test,numOfClusters[2], num_image_test, list_random15_test, list_image
          record(average15,numOfClusters[2], num_image, list_random15, list_image,list_label,Ene

          while True:
              # k = 10
              Distance(numOfClusters[2],list_distance15, average15,num_image)
              assignLabel(list_distance15, list_random15,num_image)
              average15 = centroid(numOfClusters[2], num_image, list_random15,list_image)
              record(average15,numOfClusters[2], num_image, list_random15, list_image,list_label
              iteration += 1

              if Energy[-2] - Energy[-1] < threshold:
                  print('done')
                  break

          while True:
```

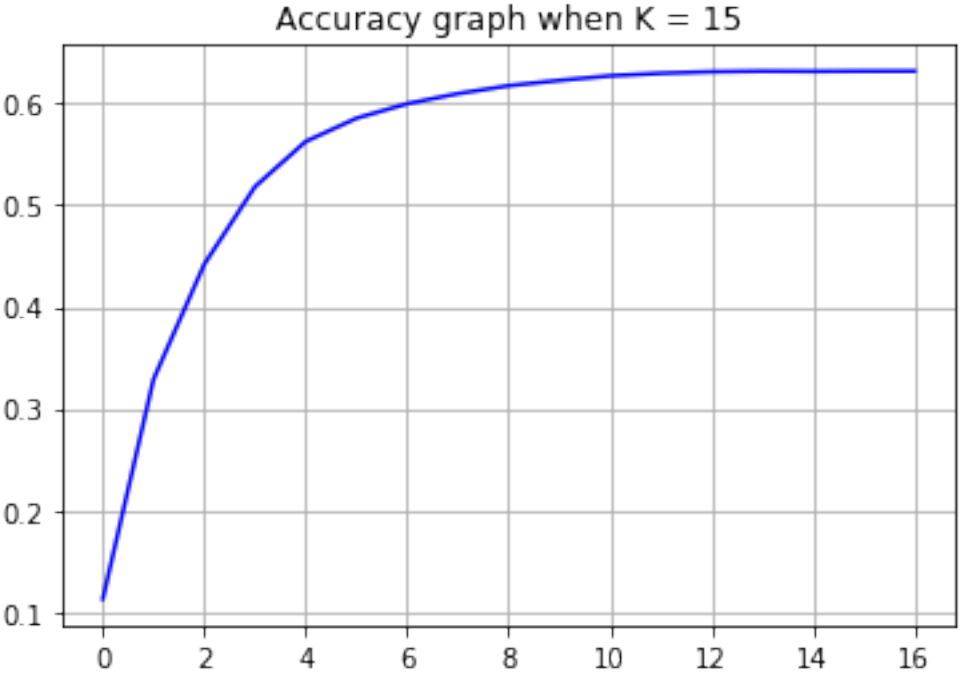
```
            # k = 10 # TEST
            Distance(numOfClusters[2],list_distance15_test, average15_test,num_image_test)
            assignLabel(list_distance15_test, list_random15,num_image_test)
            average15_test = centroid(numOfClusters[2], num_image_test, list_random15_test,lis
            record(average15_test,numOfClusters[2], num_image_test, list_random15_test, list_i
            iteration_test += 1

            if Energy_test[-2] - Energy_test[-1] < threshold:
                print('done')
                break
```

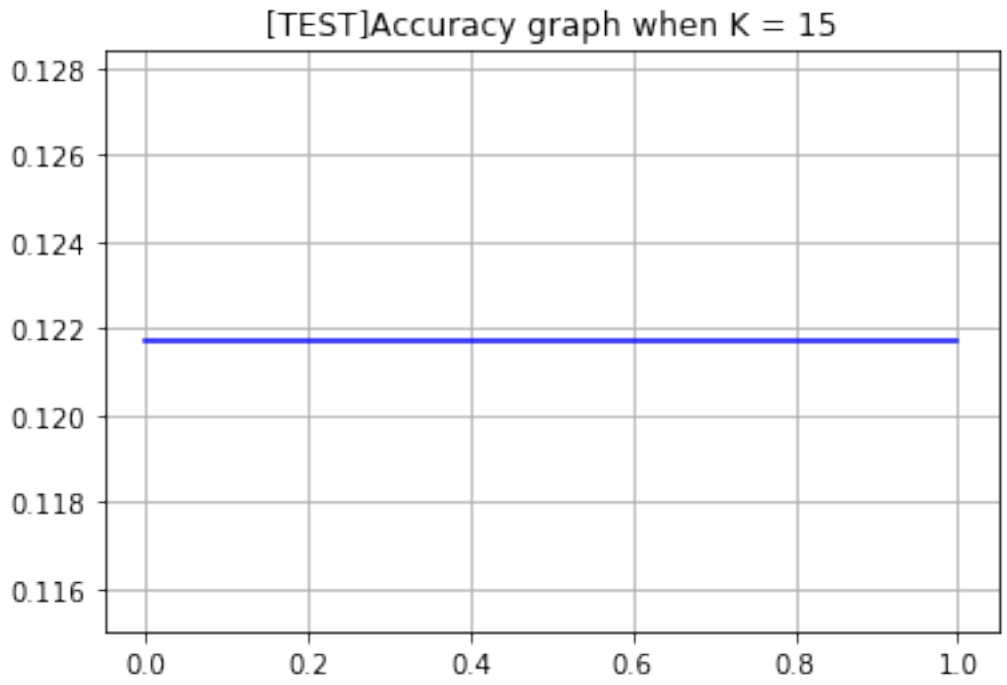
```
answer is 1217
answer is 6761
answer is 19724
answer is 26515
answer is 31095
answer is 33759
answer is 35132
answer is 35988
answer is 36583
answer is 37051
answer is 37364
answer is 37634
answer is 37784
answer is 37875
answer is 37910
answer is 37897
answer is 37912
answer is 37914
done
answer is 1217
done
```

In [245]:
```
plot(numOfClusters[2], Accuracy, average15, iteration, Energy, False)
plot(numOfClusters[2], Accuracy_test, average15_test, iteration_test, Energy_test, Tru
```




Energy graph when K = 15
52
50
48
46
44
42
40
38
0
2
4
6
8
10
12
14
16


Accuracy graph when K = 15
0.6
0.5
0.4
0.3
0.2
0.1
0
2
4
6
8
10
12
14
16



K = 20

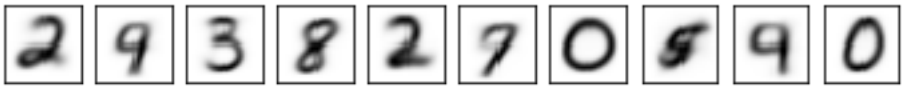
```
In [257]: iteration = 0
          iteration_test = 0
          Energy = []
          Accuracy = []

          Energy_test = []
          Accuracy_test = []

          record(average20_test,numOfClusters[3], num_image_test, list_random20_test, list_image
          record(average20,numOfClusters[3], num_image, list_random20, list_image,list_label,Ene

          while True:
              # k = 10
              Distance(numOfClusters[3],list_distance20, average20,num_image)
              assignLabel(list_distance20, list_random20,num_image)
              average20 = centroid(numOfClusters[3], num_image, list_random20,list_image)
              record(average20,numOfClusters[3], num_image, list_random20, list_image,list_label
              iteration += 1

              if Energy[-2] - Energy[-1] < threshold:
                  print('done')
                  break

          while True:
```

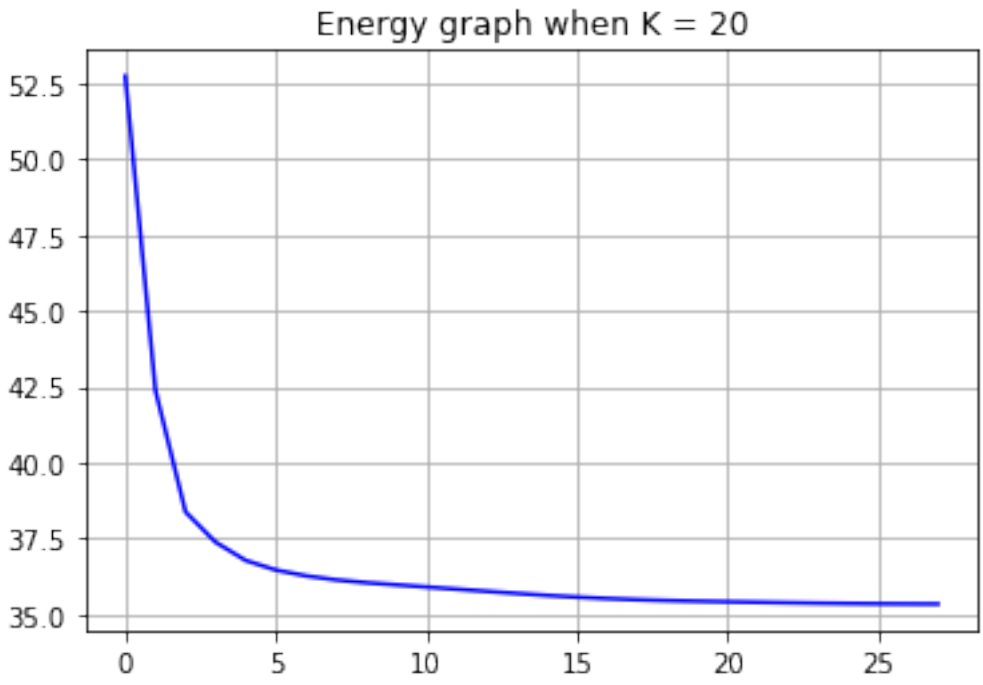
```
            # k = 10 # TEST
            Distance(numOfClusters[3],list_distance20_test, average20_test,num_image_test)
            assignLabel(list_distance20_test, list_random20,num_image_test)
            average20_test = centroid(numOfClusters[3], num_image_test, list_random20_test,lis
            record(average20_test,numOfClusters[3], num_image_test, list_random20_test, list_i
            iteration_test += 1

            if Energy_test[-2] - Energy_test[-1] < threshold:
                print('done')
                break
```

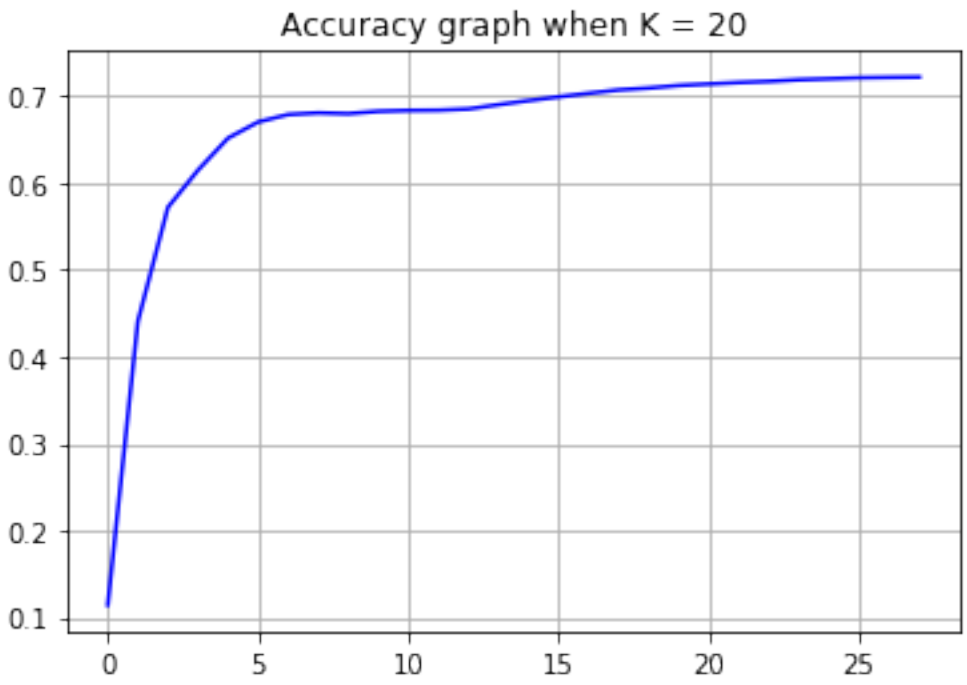
```
answer is 1252
answer is 6870
answer is 26446
answer is 34330
answer is 36897
answer is 39110
answer is 40224
answer is 40740
answer is 40852
answer is 40799
answer is 40963
answer is 41021
answer is 41038
answer is 41125
answer is 41420
answer is 41702
answer is 41961
answer is 42206
answer is 42441
answer is 42570
answer is 42746
answer is 42850
answer is 42951
answer is 43030
answer is 43145
answer is 43209
answer is 43278
answer is 43306
answer is 43325
done
answer is 1252
done
```

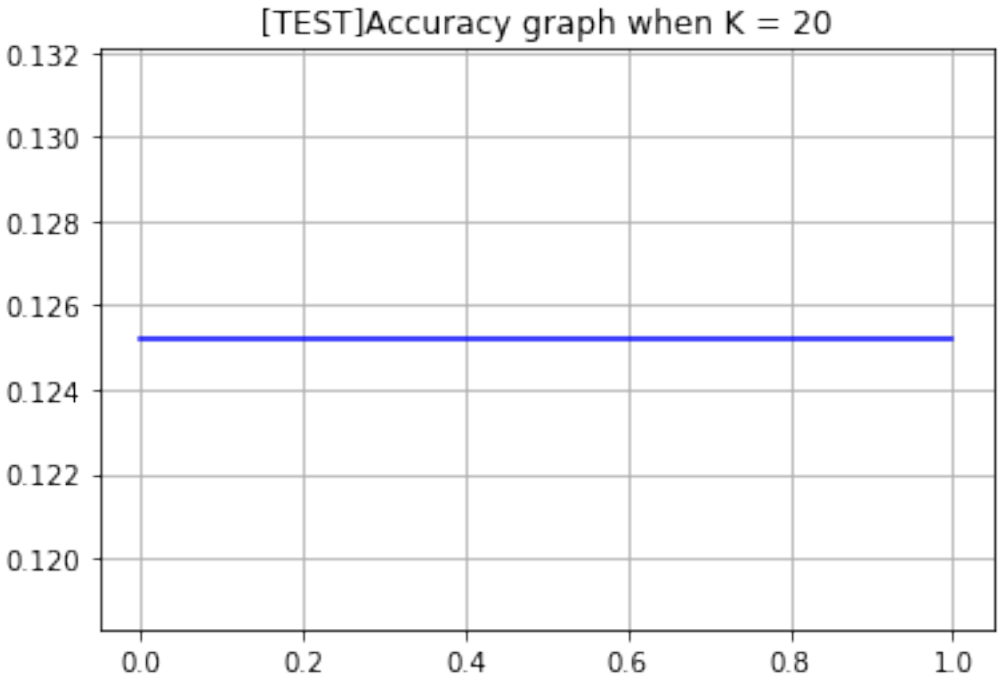
```
In [258]: plot(numOfClusters[3], Accuracy, average20, iteration, Energy, False)
          plot(numOfClusters[3], Accuracy_test, average20_test, iteration_test, Energy_test, Tru
```

K = 20




Energy graph when K = 20
52.5
50.0
47.5
45.0
42.5
40.0
37.5
35.0
0
5
10
15
20
25





Conclusion
I couldn't get acceptable results for test file, although I did test several times.