Assignment04

October 10, 2018

- Name : Joonyoung-Choi
- Student ID: 20112096
- Description: import k-means algorithm to MNIST test data set
- github: https://github.com/mydream757/Computer_Vision

1. Import liabraries and define class

- import needed libraries and define the global variables

```
In [1]: import matplotlib.pyplot as plt
        import random
        import numpy as np
        size_row = 28
        size_col = 28
```

```
In [2]: #normalize data
        def normalize(data):
            normalized = (data - min(data)) / (max(data)-min(data))

            return normalized

        #calculate distance
        def distance(x,y):
            d = (x - y)**2
            s = np.sum(d)

            return s
```

- choose initial centroids from given data set.

```
In [3]: #get initial centroids from data set
        def chooseCentroid(list_image,list_label,numOfk):
            list_centroid =np.empty((size_row*size_col, numOfk), dtype=float)
            list_centroidLabel = np.empty(numOfk,dtype=int)

            rand_num = random.randint(0,len(list_label)-1)
            for i in range(numOfk):
                random.seed()
```

```
            while list_label[rand_num] in list_centroidLabel:
                rand_num = random.randint(0,len(list_label)-1)
            list_centroidLabel[i] = list_label[rand_num]
            list_centroid[:,i] = list_image[:,rand_num]

        return [list_centroid, list_centroidLabel]
```

- compute accuaracy when k = 10.

```
In [20]: #when k=10, compute accuracy
         def computeAccuracy(list_label,list_clabel):
             num = len(list_label)
             count = 0
             for i in range(num):
                 if list_label[i]==list_clabel[i]:
                     count +=1

             accuracy = (count/num)*100
             return accuracy
```

- compute energy per iteration. $E = \sum_{k=1}^{K} \sum_{n=1}^{N} r_{nk} \| X_n - c_k \|^2$

```
In [5]: def computeEnergy(list_image, list_centroid,list_clabel,list_centroidLabel):
            energy = 0
            for i in range(len(list_clabel)):
                for k in range(len(list_centroidLabel)):
                    if list_clabel[i] == list_centroidLabel[k]:
                        energy += distance(list_image[:,i],list_centroid[:,k])
            return energy
```

- change label on images's label list.

```
In [6]: def assignLabel(list_image,list_clabel,list_centroid,list_centroidLabel):
            for i in range(len(list_clabel)):
                result = distance(list_image[:,i],list_centroid[:,0])
                list_clabel[i] = list_centroidLabel[0]
                for k in range(len(list_centroidLabel)):
                    tmp = distance(list_image[:,i],list_centroid[:,k])
                    if result > tmp:
                        result = tmp
                        list_clabel[i] = list_centroidLabel[k]
```

- this is used for checking the loop condition

```
In [7]: def makeLabelList(list_clabel):
            list = np.empty(len(list_clabel),dtype=int)
            for i in range(len(list_clabel)):
                list[i] = list_clabel[i]
            return list
```

- compute and return the centroids.

```
In [8]: def computeCentroid(list_image,list_clabel,list_centroidLabel):
            im_average  = np.zeros((size_row * size_col, len(list_clabel)), dtype=float)
            im_count    = np.zeros(len(list_clabel), dtype=int)

            for i in range(len(list_clabel)):
                for k in range(len(list_centroidLabel)):
                    if list_clabel[i] == list_centroidLabel[k]:
                        im_average[:, k] += list_image[:, i]
                        im_count[k] += 1

            for i in range(len(list_centroidLabel)):
                im_average[:, i] /= im_count[i]

            return im_average
```

- check the loop condition by label lists.

```
In [9]: def conditionCheck(before_label,after_label):
            for i in range(len(before_label)):
                if before_label[i] != after_label[i]:
                    return True
            return False
```

2. Main function of K-means algorithm.

- this is the main function of k-means algorithm.

```
In [22]: # start of k = 2, end of k = 10
         def k_means(k):
             count = 0

             #get data from .csv file
             file_data = "mnist_test.csv"
             handle_file = open(file_data, "r")
             data = handle_file.readlines()
             num_image = len(data)
             handle_file.close()
             #generate lists of image and (true) labels
             list_image  = np.empty((size_row * size_col, num_image), dtype=float)
             list_label  = np.empty(num_image, dtype=int)

             # split data into list
             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

    #generate list of centroid (clustering)label
    list_clabel = np.empty(num_image, dtype=int)
    #choose initial centroids and generate centroid label list
    list_centroid,list_centroidLabel = chooseCentroid(list_image,list_label,k)

    #initial centroids show
    f1 = plt.figure()
    for i in range(k):

        label       = list_centroidLabel[i]
        im_vector   = list_centroid[:, i]
        im_matrix   = im_vector.reshape((size_row, size_col))

        plt.subplot(1, 10, i+1)
        plt.title(label)
        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()
    #get label list for condition check
    before = makeLabelList(list_clabel)
    #assign labels
    assignLabel(list_image,list_clabel,list_centroid,list_centroidLabel)
    #get label list for condition check
    after = makeLabelList(list_clabel)

    iterCount = 0
    #if before list and after list is same, stop loop
    while conditionCheck(before,after)==True :
        #update before list
        before = after
        #compute new centroids from changed labels
        list_centroid = computeCentroid(list_image,list_clabel,list_centroidLabel)
        #by the new centroids, change the labels
        assignLabel(list_image,list_clabel,list_centroid,list_centroidLabel)
        after = makeLabelList(list_clabel)

        #compute energy at this iteration
        energy = computeEnergy(list_image, list_centroid,list_clabel,list_centroidLabel
        iterCount += 1
        print("Iter: ",iterCount," & Energy: ",energy)
```

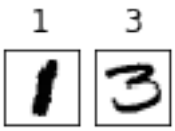
```
    im_average  = np.zeros((size_row * size_col, k), dtype=float)
    im_count    = np.zeros(k, dtype=int)

    f2 = plt.figure(2)
    for i in range(len(list_centroidLabel)):

        plt.subplot(1, 10, i+1)
        plt.title(list_centroidLabel[i])
        plt.imshow(list_centroid[:,i].reshape((size_row, size_col)), cmap='Greys')

        frame   = plt.gca()
        frame.axes.get_xaxis().set_visible(False)
        frame.axes.get_yaxis().set_visible(False)
    plt.show()
    #if k=10, compute accuracy and show
    if k==10:
        accuracy = computeAccuracy(list_label,list_clabel)
        print("accuracy: ",accuracy)
```

3. Results

This program doesn't make centroids. this just choose random units from data set.

- the start of k is '2'

```
In [11]: k_means(2)
```

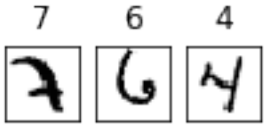



```
Iter:  1  & Energy:  498803.87242124154
Iter:  2  & Energy:  495760.2432725557
Iter:  3  & Energy:  494454.45303971885
Iter:  4  & Energy:  493809.8769836099
Iter:  5  & Energy:  493432.1199801324
Iter:  6  & Energy:  493241.46199379634
Iter:  7  & Energy:  493153.3454539614
Iter:  8  & Energy:  493121.1677338114
Iter:  9  & Energy:  493113.5369395657
Iter:  10  & Energy:  493108.2841998108
Iter:  11  & Energy:  493104.95829513005
Iter:  12  & Energy:  493103.94312557427
```

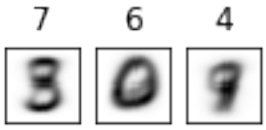
```
Iter:  13  & Energy:  493103.5465340409
Iter:  14  & Energy:  493103.2746975684
Iter:  15  & Energy:  493103.21124978334
Iter:  16  & Energy:  493103.18680798414
```

The result is hard to recognize.

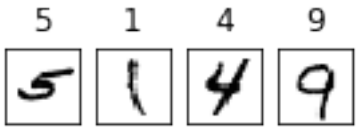
```
In [12]: k_means(3)
```




```
Iter:  1  & Energy:  485780.63758017117
Iter:  2  & Energy:  476037.1417951543
Iter:  3  & Energy:  472984.177666077
Iter:  4  & Energy:  472183.1075374662
Iter:  5  & Energy:  472023.83686686074
Iter:  6  & Energy:  471981.32366304693
Iter:  7  & Energy:  471968.27806577255
Iter:  8  & Energy:  471960.35407858784
Iter:  9  & Energy:  471952.9335702368
Iter:  10  & Energy:  471949.8696635349
Iter:  11  & Energy:  471948.8386487071
Iter:  12  & Energy:  471947.76788445236
Iter:  13  & Energy:  471946.37670752616
Iter:  14  & Energy:  471945.2041845023
Iter:  15  & Energy:  471944.7417612958
Iter:  16  & Energy:  471944.58707147324
Iter:  17  & Energy:  471944.2223717432
Iter:  18  & Energy:  471943.8715299087
Iter:  19  & Energy:  471943.73988533195
Iter:  20  & Energy:  471943.6994516185
```

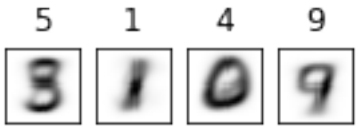


In [13]: k_means(4)

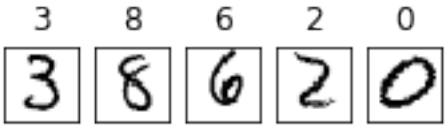



```
Iter:  1  & Energy:  481727.8392561482
Iter:  2  & Energy:  474239.33441937383
Iter:  3  & Energy:  469770.02642840584
Iter:  4  & Energy:  465593.0518069285
Iter:  5  & Energy:  461437.2003824289
Iter:  6  & Energy:  458400.50692892005
Iter:  7  & Energy:  456892.09847017436
Iter:  8  & Energy:  456129.94845004234
Iter:  9  & Energy:  455743.3761687409
Iter:  10  & Energy:  455539.64769744856
Iter:  11  & Energy:  455402.4680453792
Iter:  12  & Energy:  455317.279036559
Iter:  13  & Energy:  455252.1844353519
Iter:  14  & Energy:  455190.10174534033
Iter:  15  & Energy:  455118.8722669243
Iter:  16  & Energy:  455056.521989718
Iter:  17  & Energy:  454972.8694049361
Iter:  18  & Energy:  454888.629373247
Iter:  19  & Energy:  454759.9746287354
Iter:  20  & Energy:  454615.2415386454
Iter:  21  & Energy:  454406.7186908886
Iter:  22  & Energy:  454114.5546604468
Iter:  23  & Energy:  453764.08640630497
Iter:  24  & Energy:  453302.305393386
Iter:  25  & Energy:  452838.9025268404
Iter:  26  & Energy:  452357.8637899243
Iter:  27  & Energy:  451864.7798693255
Iter:  28  & Energy:  451264.7453440553
Iter:  29  & Energy:  450548.9513122416
```

```
Iter:  30  & Energy:  449909.2743073259
Iter:  31  & Energy:  449522.1814886338
Iter:  32  & Energy:  449239.49553563434
Iter:  33  & Energy:  449005.70455337357
Iter:  34  & Energy:  448817.71661245625
Iter:  35  & Energy:  448683.3967577466
Iter:  36  & Energy:  448564.75311446434
Iter:  37  & Energy:  448468.6774593563
Iter:  38  & Energy:  448382.5698384887
Iter:  39  & Energy:  448307.70841899863
Iter:  40  & Energy:  448269.64632189117
Iter:  41  & Energy:  448253.08645943354
Iter:  42  & Energy:  448245.0618997325
Iter:  43  & Energy:  448240.4251388208
Iter:  44  & Energy:  448236.15298169485
Iter:  45  & Energy:  448233.3459005238
Iter:  46  & Energy:  448232.1049640369
Iter:  47  & Energy:  448231.2190989348
Iter:  48  & Energy:  448230.8212532189
Iter:  49  & Energy:  448230.42017002124
Iter:  50  & Energy:  448230.35746381065
```

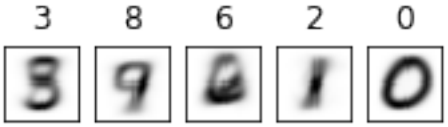


In [14]: k_means(5)



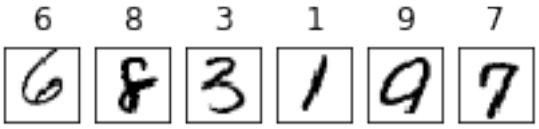

```
Iter:  1  & Energy:  453197.66078474117
Iter:  2  & Energy:  446265.7194169439
Iter:  3  & Energy:  444230.9073569672
Iter:  4  & Energy:  442612.07622038363
Iter:  5  & Energy:  440964.1153880761
Iter:  6  & Energy:  439470.88211030595
```

```
Iter:  7  & Energy:  438225.63015757146
Iter:  8  & Energy:  437337.75816643296
Iter:  9  & Energy:  436688.71033597447
Iter:  10  & Energy:  436159.9052618775
Iter:  11  & Energy:  435641.5394147954
Iter:  12  & Energy:  435117.39681533613
Iter:  13  & Energy:  434554.3283538235
Iter:  14  & Energy:  433944.0730698408
Iter:  15  & Energy:  433405.9377133465
Iter:  16  & Energy:  432954.21529475506
Iter:  17  & Energy:  432657.05978515156
Iter:  18  & Energy:  432466.818635154
Iter:  19  & Energy:  432355.9924399566
Iter:  20  & Energy:  432314.2142752114
Iter:  21  & Energy:  432299.2821100238
Iter:  22  & Energy:  432289.73754276754
Iter:  23  & Energy:  432286.6908574218
Iter:  24  & Energy:  432284.78904103744
Iter:  25  & Energy:  432284.0118664811
Iter:  26  & Energy:  432283.78224865976
```



Although k is much bigger than k=2, stil we can't tell that is good clustering.

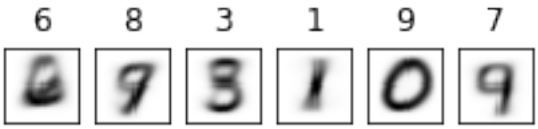
```
In [15]: k_means(6)
```



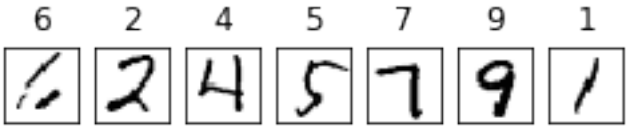

```
Iter:  1  & Energy:  431379.0122974147
Iter:  2  & Energy:  425618.6185393748
Iter:  3  & Energy:  424244.89724548993
Iter:  4  & Energy:  423752.48981145857
Iter:  5  & Energy:  423517.8109436064
Iter:  6  & Energy:  423355.26348031557
```

```
Iter:  7  & Energy:  423227.30034251814
Iter:  8  & Energy:  423139.8804074069
Iter:  9  & Energy:  423063.47435879736
Iter:  10  & Energy:  422984.41465285065
Iter:  11  & Energy:  422908.5174731895
Iter:  12  & Energy:  422848.50616584916
Iter:  13  & Energy:  422812.73647642514
Iter:  14  & Energy:  422774.52731582266
Iter:  15  & Energy:  422730.7907967787
Iter:  16  & Energy:  422687.99748942856
Iter:  17  & Energy:  422644.32777261286
Iter:  18  & Energy:  422591.2258305145
Iter:  19  & Energy:  422526.80158147076
Iter:  20  & Energy:  422441.04449438327
Iter:  21  & Energy:  422344.102608684
Iter:  22  & Energy:  422234.5377569585
Iter:  23  & Energy:  422089.3211384859
Iter:  24  & Energy:  421897.5099984761
Iter:  25  & Energy:  421676.31428844994
Iter:  26  & Energy:  421467.23002460133
Iter:  27  & Energy:  421263.1078435813
Iter:  28  & Energy:  421084.4608453833
Iter:  29  & Energy:  420881.58198872505
Iter:  30  & Energy:  420718.8714659764
Iter:  31  & Energy:  420589.63486175856
Iter:  32  & Energy:  420485.65652524825
Iter:  33  & Energy:  420414.398049958
Iter:  34  & Energy:  420373.32669203944
Iter:  35  & Energy:  420347.41632441845
Iter:  36  & Energy:  420329.8558256299
Iter:  37  & Energy:  420311.16893904214
Iter:  38  & Energy:  420289.9765509849
Iter:  39  & Energy:  420260.00371501624
Iter:  40  & Energy:  420239.8340217912
Iter:  41  & Energy:  420232.53758871206
Iter:  42  & Energy:  420226.88355067367
Iter:  43  & Energy:  420219.02250626514
Iter:  44  & Energy:  420207.9090639535
Iter:  45  & Energy:  420200.03531349363
Iter:  46  & Energy:  420197.91582974955
Iter:  47  & Energy:  420196.98615320446
Iter:  48  & Energy:  420195.9330326874
Iter:  49  & Energy:  420194.5656519569
Iter:  50  & Energy:  420193.65143438417
Iter:  51  & Energy:  420192.8091382402
Iter:  52  & Energy:  420192.1164325567
Iter:  53  & Energy:  420191.2939235608
Iter:  54  & Energy:  420190.4834683685
```

```
Iter:  55  & Energy:  420190.30117701634
Iter:  56  & Energy:  420190.22504515736
Iter:  57  & Energy:  420190.1852396935
```



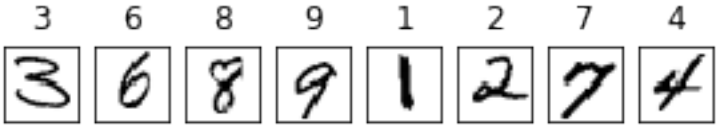

```
In [16]: k_means(7)
```




```
Iter:  1  & Energy:  441369.4732066819
Iter:  2  & Energy:  430590.95385029045
Iter:  3  & Energy:  426530.03753264964
Iter:  4  & Energy:  422787.3988567071
Iter:  5  & Energy:  419595.959160055
Iter:  6  & Energy:  418125.66564312205
Iter:  7  & Energy:  417519.337893581
Iter:  8  & Energy:  417034.1671827589
Iter:  9  & Energy:  416367.20244639757
Iter:  10  & Energy:  415390.3364617552
Iter:  11  & Energy:  413958.48226858664
Iter:  12  & Energy:  412729.41475541407
Iter:  13  & Energy:  412114.1458297913
Iter:  14  & Energy:  411776.4641795949
Iter:  15  & Energy:  411577.2932465257
Iter:  16  & Energy:  411466.02563373494
Iter:  17  & Energy:  411404.09125579963
Iter:  18  & Energy:  411366.4787701266
Iter:  19  & Energy:  411343.4035881605
Iter:  20  & Energy:  411318.1085922824
Iter:  21  & Energy:  411297.0132717604
Iter:  22  & Energy:  411282.90431939415
Iter:  23  & Energy:  411270.47103614983
Iter:  24  & Energy:  411259.43747556297
```

```
Iter:  25  & Energy:  411254.07458903955
Iter:  26  & Energy:  411249.0900103287
Iter:  27  & Energy:  411244.12060546735
Iter:  28  & Energy:  411242.03652042744
Iter:  29  & Energy:  411241.2111621121
Iter:  30  & Energy:  411241.05116966355
Iter:  31  & Energy:  411240.74154556467
Iter:  32  & Energy:  411240.5520004364
Iter:  33  & Energy:  411240.2922864607
Iter:  34  & Energy:  411240.0417442266
Iter:  35  & Energy:  411239.73474382557
Iter:  36  & Energy:  411239.3699139464
Iter:  37  & Energy:  411239.27923616953
Iter:  38  & Energy:  411238.9545820408
Iter:  39  & Energy:  411238.3717134266
Iter:  40  & Energy:  411237.87823344115
Iter:  41  & Energy:  411237.7279217084
```


```
In [17]: k_means(8)
```

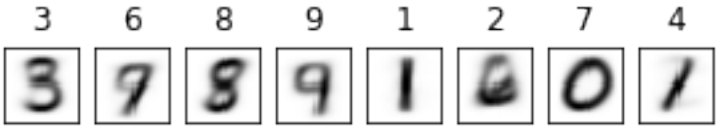



```
Iter:  1  & Energy:  435353.5136607683
Iter:  2  & Energy:  421416.73136179853
Iter:  3  & Energy:  416324.43984678737
Iter:  4  & Energy:  412889.85338557087
Iter:  5  & Energy:  411002.996657156
Iter:  6  & Energy:  410268.32103964617
Iter:  7  & Energy:  409870.9451822136
Iter:  8  & Energy:  409587.28157724295
Iter:  9  & Energy:  409360.3368975232
Iter:  10  & Energy:  409218.2293921035
```

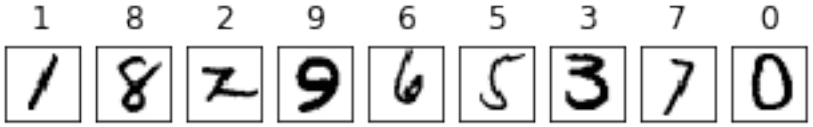
```
Iter:  11  & Energy:  409100.65052645904
Iter:  12  & Energy:  409035.2407205957
Iter:  13  & Energy:  408971.99855951633
Iter:  14  & Energy:  408918.46616139886
Iter:  15  & Energy:  408885.6007815107
Iter:  16  & Energy:  408860.0990456538
Iter:  17  & Energy:  408847.8202893133
Iter:  18  & Energy:  408837.60913278285
Iter:  19  & Energy:  408825.7661663122
Iter:  20  & Energy:  408818.527654588
Iter:  21  & Energy:  408813.24382852565
Iter:  22  & Energy:  408807.88594092143
Iter:  23  & Energy:  408803.4774250088
Iter:  24  & Energy:  408801.58770126896
Iter:  25  & Energy:  408798.6367895905
Iter:  26  & Energy:  408794.21586127364
Iter:  27  & Energy:  408791.4390598762
Iter:  28  & Energy:  408788.79106831155
Iter:  29  & Energy:  408786.9970537205
Iter:  30  & Energy:  408785.89995080204
Iter:  31  & Energy:  408785.31936322304
Iter:  32  & Energy:  408785.08401190036
Iter:  33  & Energy:  408784.80169865105
Iter:  34  & Energy:  408784.6785890563
Iter:  35  & Energy:  408784.29963269277
Iter:  36  & Energy:  408783.60515048646
Iter:  37  & Energy:  408782.7109854351
Iter:  38  & Energy:  408781.22929252905
Iter:  39  & Energy:  408779.0832250611
Iter:  40  & Energy:  408776.9115618437
Iter:  41  & Energy:  408775.9617690922
Iter:  42  & Energy:  408773.92375755636
Iter:  43  & Energy:  408771.69729569374
Iter:  44  & Energy:  408769.478334363
Iter:  45  & Energy:  408766.944422556
Iter:  46  & Energy:  408765.7016358568
Iter:  47  & Energy:  408763.2348312776
Iter:  48  & Energy:  408761.6593697846
Iter:  49  & Energy:  408758.9402701809
Iter:  50  & Energy:  408756.42620111426
Iter:  51  & Energy:  408753.4333461075
Iter:  52  & Energy:  408751.5329590999
Iter:  53  & Energy:  408750.1276669587
Iter:  54  & Energy:  408748.60096542776
Iter:  55  & Energy:  408746.19307869236
Iter:  56  & Energy:  408744.1651067176
Iter:  57  & Energy:  408740.75355233945
Iter:  58  & Energy:  408734.27083953825
```

```
Iter:  59  & Energy:  408723.759331248
Iter:  60  & Energy:  408709.9767522182
Iter:  61  & Energy:  408698.6778302477
Iter:  62  & Energy:  408682.36123914004
Iter:  63  & Energy:  408666.2817814076
Iter:  64  & Energy:  408651.6017917632
Iter:  65  & Energy:  408635.43337183504
Iter:  66  & Energy:  408619.94043446257
Iter:  67  & Energy:  408604.0070365668
Iter:  68  & Energy:  408585.56882174395
Iter:  69  & Energy:  408574.96485296416
Iter:  70  & Energy:  408564.063690689
Iter:  71  & Energy:  408544.33654424857
Iter:  72  & Energy:  408515.022102769
Iter:  73  & Energy:  408488.87362522533
Iter:  74  & Energy:  408466.89331840543
Iter:  75  & Energy:  408444.7442344371
Iter:  76  & Energy:  408411.5817329532
Iter:  77  & Energy:  408369.97674737155
Iter:  78  & Energy:  408319.4165187494
Iter:  79  & Energy:  408248.97555275273
Iter:  80  & Energy:  408157.11908535444
Iter:  81  & Energy:  408022.135401098
Iter:  82  & Energy:  407806.6681820317
Iter:  83  & Energy:  407437.6373745418
Iter:  84  & Energy:  406962.76025135175
Iter:  85  & Energy:  406429.93445692636
Iter:  86  & Energy:  405971.60115159996
Iter:  87  & Energy:  405607.3603626096
Iter:  88  & Energy:  405353.92752955813
Iter:  89  & Energy:  405179.7117030071
Iter:  90  & Energy:  405047.91435775766
Iter:  91  & Energy:  404934.027329193
Iter:  92  & Energy:  404835.47782939865
Iter:  93  & Energy:  404736.8084199897
Iter:  94  & Energy:  404647.94568412943
Iter:  95  & Energy:  404584.3083044918
Iter:  96  & Energy:  404538.74025206367
Iter:  97  & Energy:  404512.86205342395
Iter:  98  & Energy:  404493.991104704
Iter:  99  & Energy:  404479.68391067965
Iter:  100  & Energy:  404465.21434880147
Iter:  101  & Energy:  404453.1344142958
Iter:  102  & Energy:  404434.61081575986
Iter:  103  & Energy:  404421.7504506172
Iter:  104  & Energy:  404410.37958203565
Iter:  105  & Energy:  404398.4671355112
Iter:  106  & Energy:  404385.55707105657
```

```
Iter:  107  & Energy:  404378.98517144506
Iter:  108  & Energy:  404373.02770081995
Iter:  109  & Energy:  404370.1765296238
Iter:  110  & Energy:  404368.7206963198
Iter:  111  & Energy:  404367.4311231186
Iter:  112  & Energy:  404366.5440401771
Iter:  113  & Energy:  404366.15788515855
```

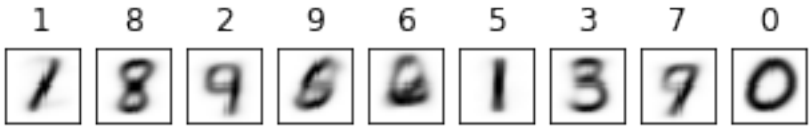



```
In [18]: k_means(9)
```

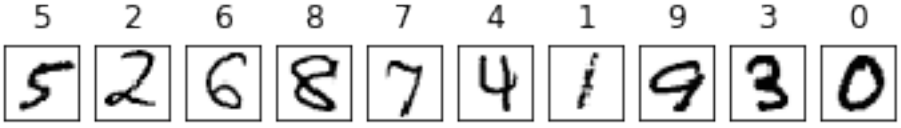



```
Iter:  1  & Energy:  435254.2938407262
Iter:  2  & Energy:  417567.3829882291
Iter:  3  & Energy:  409280.31659279566
Iter:  4  & Energy:  405133.3841348033
Iter:  5  & Energy:  402582.10386770434
Iter:  6  & Energy:  401026.6189378817
Iter:  7  & Energy:  400207.7963733028
Iter:  8  & Energy:  399795.9147393042
Iter:  9  & Energy:  399552.64746705664
Iter:  10  & Energy:  399416.3135800789
Iter:  11  & Energy:  399305.9142934744
Iter:  12  & Energy:  399189.1715944581
Iter:  13  & Energy:  399081.94865815906
Iter:  14  & Energy:  399006.2679102212
Iter:  15  & Energy:  398929.50741755246
Iter:  16  & Energy:  398849.4682366529
Iter:  17  & Energy:  398775.19058541243
Iter:  18  & Energy:  398677.2218882676
Iter:  19  & Energy:  398569.864685256
Iter:  20  & Energy:  398476.6827148266
```

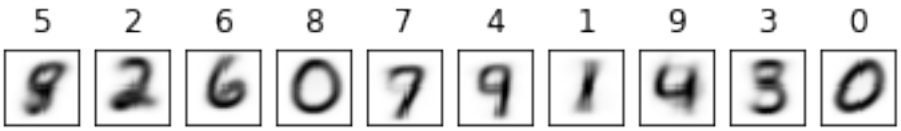
```
Iter:  21  & Energy:  398430.6747422861
Iter:  22  & Energy:  398372.6808863982
Iter:  23  & Energy:  398325.86835893145
Iter:  24  & Energy:  398291.1620323096
Iter:  25  & Energy:  398248.310659186
Iter:  26  & Energy:  398214.2598737541
Iter:  27  & Energy:  398196.76948410587
Iter:  28  & Energy:  398179.21957830095
Iter:  29  & Energy:  398158.81319830375
Iter:  30  & Energy:  398141.3761107723
Iter:  31  & Energy:  398126.0906984229
Iter:  32  & Energy:  398110.57573174837
Iter:  33  & Energy:  398098.18315663264
Iter:  34  & Energy:  398084.61483802
Iter:  35  & Energy:  398067.4798506326
Iter:  36  & Energy:  398058.14781728695
Iter:  37  & Energy:  398049.958900537
Iter:  38  & Energy:  398044.9939086398
Iter:  39  & Energy:  398040.90057572885
Iter:  40  & Energy:  398038.2078164279
Iter:  41  & Energy:  398036.34854380874
Iter:  42  & Energy:  398035.55237764167
Iter:  43  & Energy:  398035.0265375434
Iter:  44  & Energy:  398034.39529870177
Iter:  45  & Energy:  398033.9654092028
Iter:  46  & Energy:  398033.73263118777
Iter:  47  & Energy:  398033.6269752825
Iter:  48  & Energy:  398033.42728179926
```




```
In [21]: k_means(10)
```



```
Iter:  1  & Energy:  414926.41958721884
Iter:  2  & Energy:  400704.1492578523
Iter:  3  & Energy:  396418.15073701565
Iter:  4  & Energy:  394834.9146623979
Iter:  5  & Energy:  394240.5191531714
Iter:  6  & Energy:  393958.30335304205
Iter:  7  & Energy:  393806.06708153273
Iter:  8  & Energy:  393702.1317819628
Iter:  9  & Energy:  393632.73039570544
Iter:  10  & Energy:  393561.69299367775
Iter:  11  & Energy:  393488.86455910874
Iter:  12  & Energy:  393405.0391005762
Iter:  13  & Energy:  393330.9303187796
Iter:  14  & Energy:  393254.3062498745
Iter:  15  & Energy:  393198.83931796224
Iter:  16  & Energy:  393168.174602458
Iter:  17  & Energy:  393142.0512975813
Iter:  18  & Energy:  393124.0924257433
Iter:  19  & Energy:  393113.11100499326
Iter:  20  & Energy:  393103.8578557945
Iter:  21  & Energy:  393092.3789304392
Iter:  22  & Energy:  393079.8420527198
Iter:  23  & Energy:  393069.14346800256
Iter:  24  & Energy:  393061.26345311996
Iter:  25  & Energy:  393055.21803359553
Iter:  26  & Energy:  393051.59542185516
Iter:  27  & Energy:  393047.868539233
Iter:  28  & Energy:  393046.1044569399
Iter:  29  & Energy:  393044.3902525685
Iter:  30  & Energy:  393042.3063495353
Iter:  31  & Energy:  393038.9733463976
Iter:  32  & Energy:  393035.682935363
Iter:  33  & Energy:  393033.8506278274
Iter:  34  & Energy:  393032.6401998789
Iter:  35  & Energy:  393031.8662295271
Iter:  36  & Energy:  393031.38854988833
Iter:  37  & Energy:  393031.0949284747
Iter:  38  & Energy:  393030.95726803143
```



```
accuracy:  51.61
```

More and more k, the final energy is lower. But the result is disappointing. About this result(accuracy is 51.61), I think that the initial value is important and recognizing image is very hard. Maybe I need some other solution for this.