Experiment 1: Application and analysis of cluster algorithms

I. Experiment Introduction

Given a dataset of 48200 samples and 784 features for each sample, we need to implement 3 cluster algorithms and write a report to analyze the clustering results and performance. In this experiment, I chose to realize the following 3 cluster algorithms: C-Means Clustering (Competitive learning version), Fuzzy C-Means (FCM) and K-Nearest Neighbor (KNN).

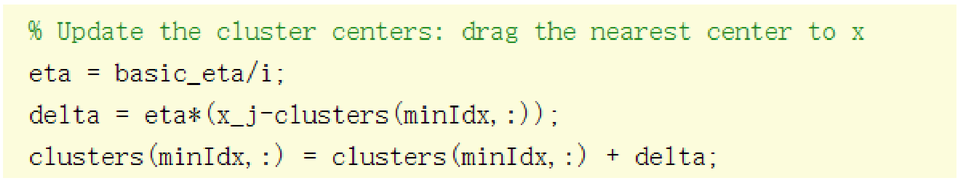
II. Analysis of cluster algorithms

2.1 C-Means Clustering

Here we briefly describe the procedure of this algorithm:

1) Randomly choose *n* patterns as the initial cluster centers;
2) Set the unclassified patterns (samples) to one of the *n* clusters according to certain criterion, such as minimum distance criterion. Thus we get the new clusters in iteration 1.
3) Recalculate the centers of clusters.
4) Repeat 2) and 3) until the centers do not change or change in acceptable margins.

The difference of ordinary C-Means and we used algorithm here is that in the iterative clustering procedure, we use the stochastic gradient descent method to change the cluster centers with a small change instead of recalculate all the features in one cluster.



```
% Update the cluster centers: drag the nearest center to x
eta = basic_eta/i;
delta = eta*(x_j-clusters(minIdx,:));
clusters(minIdx,:) = clusters(minIdx,:) + delta;
```

Fig.1 the difference of 3) in our code

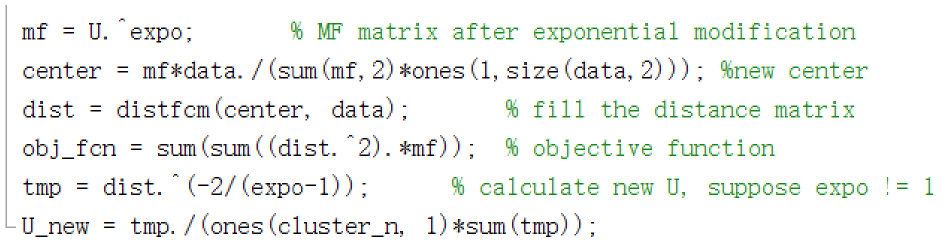
2.2 Fuzzy C-Means Clustering (FCM)

We still briefly introduce the procedure of FCM:

1) Randomly choose *n* patterns as the initial cluster centers and parameter *b*;

2) Calculate the fuzzy membership function for unclassified samples. Here we do not directly label the input samples to one certain cluster, instead, each sample has its own fuzzy membership function to the clusters.
3) Recalculate the centers of clusters.
4) Repeat 2) and 3) until the centers do not change or change in acceptable margins.

The greatest difference between FCM and C-Means is that FCM introduces the concept of "fuzzy membership function" in 2), therefore every input sample does not belong to one certain cluster. This is the so called "soft clustering" and we can implement "de-fuzzy" operation according to the maximum membership criterion if needed.



```
mf = U.^expo;       % MF matrix after exponential modification
center = mf*data./(sum(mf,2)*ones(1,size(data,2))); %new center
dist = distfcm(center, data);       % fill the distance matrix
obj_fcn = sum(sum((dist.^2).*mf));  % objective function
tmp = dist.^(-2/(expo-1));      % calculate new U, suppose expo != 1
U_new = tmp./(ones(cluster_n, 1)*sum(tmp));
```

Fig.2 Main calculations

2.3 K-Nearest Neighbor Clustering (KNN)

KNN is a supervised learning algorithm. The main idea of this algorithm is that for a given input sample, find its nearest *k* patterns and set it to be in the cluster with most patterns in these *k* patterns as shown below.

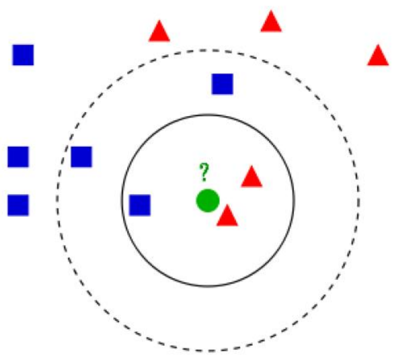


Fig.3 Illustration of KNN

Note that in all above 3 algorithms we use Euclidean distance in different criteria to determine the clusters.

III. Results and evaluation

3.1 Dimensionality Reduction: PCA

Since each sample has 784 features, which is a quite large number for cluster algorithms to deal with, we have to implement PCA to reduce the dimensions of features. This procedure can be easily done by one line of code *[coeff,score,latent] = pca(data)*.

We find that the top 10 features preserve 50.51% of the overall information, the top 42 features preserve 80.26%, the top 85 features preserve 90.05%, the top 152 features preserve 95.02%. This is calculated by *ratio = cumsum(latent)/sum(latent)* where *latent* is the eigenvalues for 784 features calculated in *[coeff,score,latent] = pca(data)*.

3.2 Comparison and evaluation of clustering results

Note that the following 3D scatter pictures are just illustration of the clustering results. It does not mean that we only choose 3 features for clustering!!! We analyzed the clustering results of 6 to 8 clusters and here we just show the clustering figures of 8-cluster case. For more pictures, please check the /Figures folder.

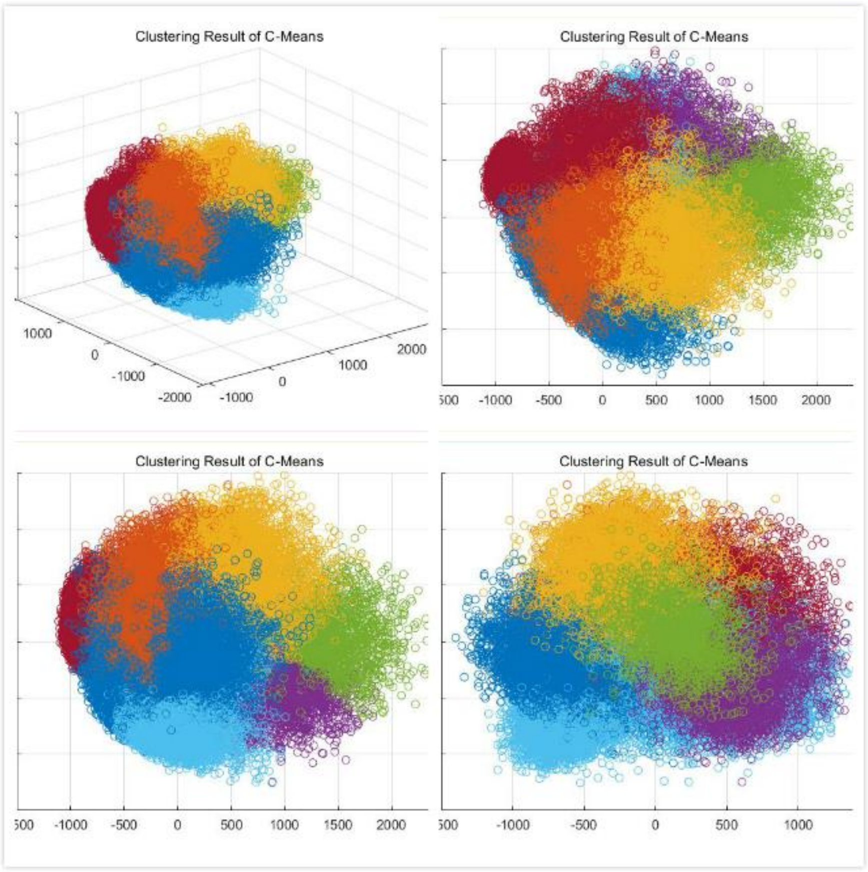


Fig.4 Clustering results of C-Means

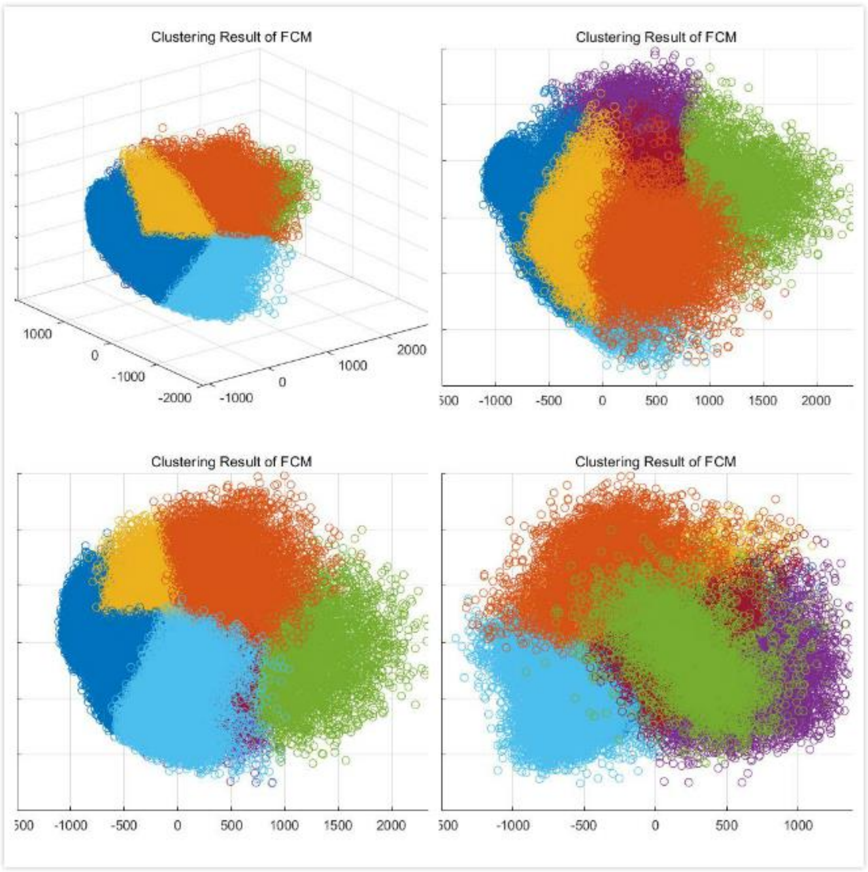


Fig.5 Clustering results of FCM

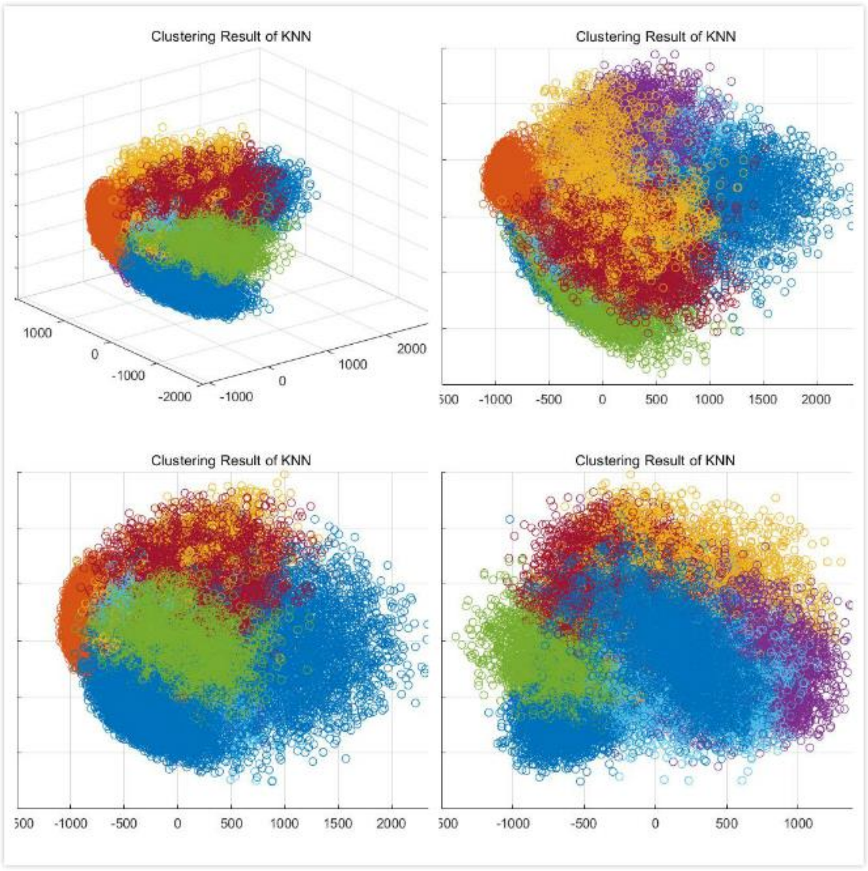
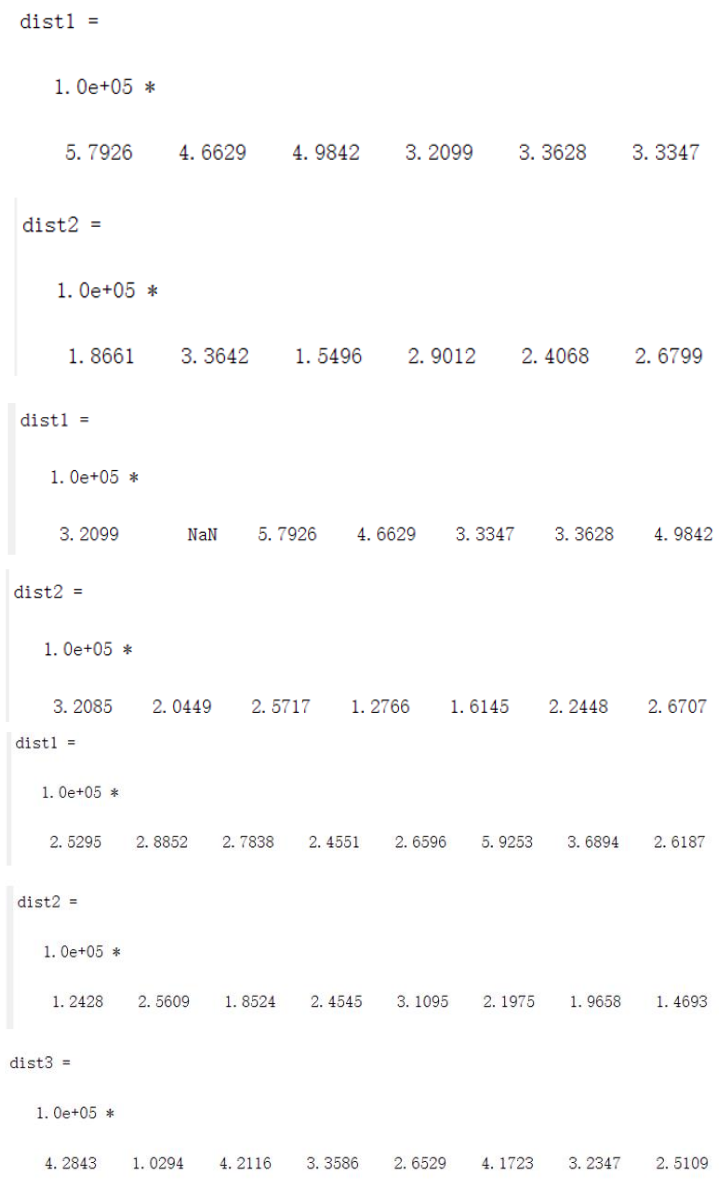


Fig.6 Clustering results of KNN

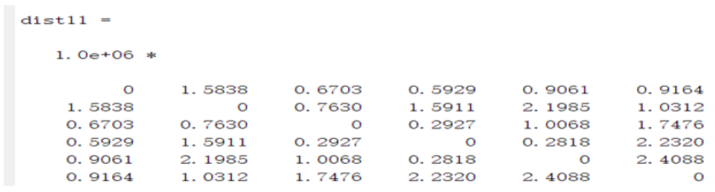
From Fig.4 to Fig.6 we can observe the clustering results in 3D view. In each algorithm we set the initial cluster number as 6/7/8. Generally, the 3 algorithms show roughly alike results. But we cannot ignore there exists much difference.

Then we need to investigate the ***within-cluster distance*** and ***between-cluster distance*** of 6/7/8-cluster cases for further analysis.

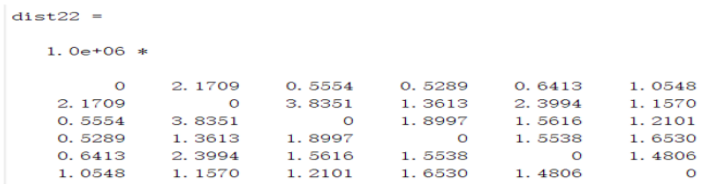


```
dist1 =

   1.0e+05 *

    5.7926    4.6629    4.9842    3.2099    3.3628    3.3347

dist2 =

   1.0e+05 *

    1.8661    3.3642    1.5496    2.9012    2.4068    2.6799

dist1 =

   1.0e+05 *

    3.2099       NaN    5.7926    4.6629    3.3347    3.3628    4.9842

dist2 =

   1.0e+05 *

    3.2085    2.0449    2.5717    1.2766    1.6145    2.2448    2.6707

dist1 =

   1.0e+05 *

    2.5295    2.8852    2.7838    2.4551    2.6596    5.9253    3.6894    2.6187

dist2 =

   1.0e+05 *

    1.2428    2.5609    1.8524    2.4545    3.1095    2.1975    1.9658    1.4693

dist3 =

   1.0e+05 *

    4.2843    1.0294    4.2116    3.3586    2.6529    4.1723    3.2347    2.5109
```

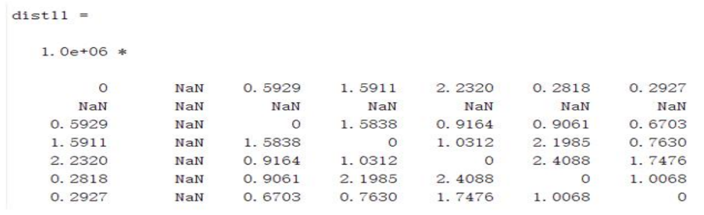
Fig.7 Within-cluster distance in 3 algorithms



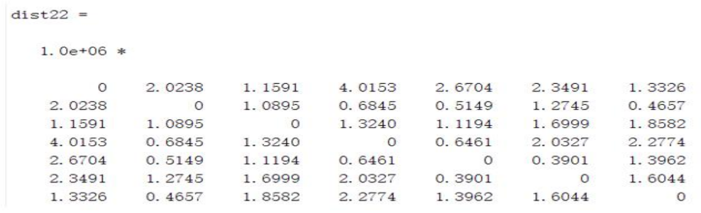
```
dist11 =

   1. 0e+06 *

         0    1. 5838    0. 6703    0. 5929    0. 9061    0. 9164
   1. 5838         0    0. 7630    1. 5911    2. 1985    1. 0312
   0. 6703    0. 7630         0    0. 2927    1. 0068    1. 7476
   0. 5929    1. 5911    0. 2927         0    0. 2818    2. 2320
   0. 9061    2. 1985    1. 0068    0. 2818         0    2. 4088
   0. 9164    1. 0312    1. 7476    2. 2320    2. 4088         0
```



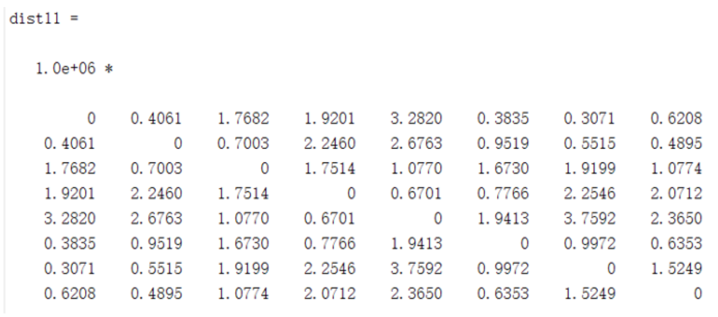
```
dist22 =

   1. 0e+06 *

         0    2. 1709    0. 5554    0. 5289    0. 6413    1. 0548
   2. 1709         0    3. 8351    1. 3613    2. 3994    1. 1570
   0. 5554    3. 8351         0    1. 8997    1. 5616    1. 2101
   0. 5289    1. 3613    1. 8997         0    1. 5538    1. 6530
   0. 6413    2. 3994    1. 5616    1. 5538         0    1. 4806
   1. 0548    1. 1570    1. 2101    1. 6530    1. 4806         0
```



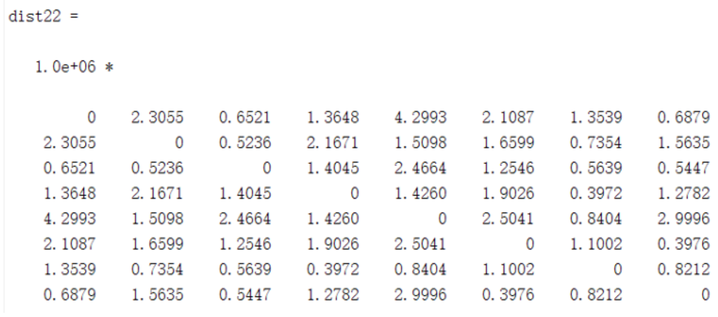
```
dist11 =

   1. 0e+06 *

         0        NaN    0. 5929    1. 5911    2. 2320    0. 2818    0. 2927
       NaN        NaN        NaN        NaN        NaN        NaN        NaN
   0. 5929        NaN         0    1. 5838    0. 9164    0. 9061    0. 6703
   1. 5911        NaN    1. 5838         0    1. 0312    2. 1985    0. 7630
   2. 2320        NaN    0. 9164    1. 0312         0    2. 4088    1. 7476
   0. 2818        NaN    0. 9061    2. 1985    2. 4088         0    1. 0068
   0. 2927        NaN    0. 6703    0. 7630    1. 7476    1. 0068         0
```



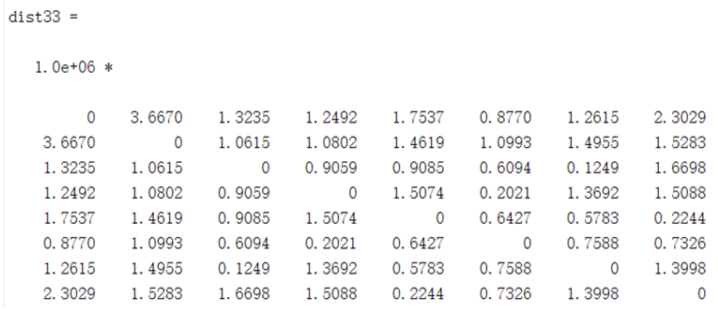
```
dist22 =

   1. 0e+06 *

         0    2. 0238    1. 1591    4. 0153    2. 6704    2. 3491    1. 3326
   2. 0238         0    1. 0895    0. 6845    0. 5149    1. 2745    0. 4657
   1. 1591    1. 0895         0    1. 3240    1. 1194    1. 6999    1. 8582
   4. 0153    0. 6845    1. 3240         0    0. 6461    2. 0327    2. 2774
   2. 6704    0. 5149    1. 1194    0. 6461         0    0. 3901    1. 3962
   2. 3491    1. 2745    1. 6999    2. 0327    0. 3901         0    1. 6044
   1. 3326    0. 4657    1. 8582    2. 2774    1. 3962    1. 6044         0
```



```
dist11 =

   1. 0e+06 *

         0    0. 4061    1. 7682    1. 9201    3. 2820    0. 3835    0. 3071    0. 6208
   0. 4061         0    0. 7003    2. 2460    2. 6763    0. 9519    0. 5515    0. 4895
   1. 7682    0. 7003         0    1. 7514    1. 0770    1. 6730    1. 9199    1. 0774
   1. 9201    2. 2460    1. 7514         0    0. 6701    0. 7766    2. 2546    2. 0712
   3. 2820    2. 6763    1. 0770    0. 6701         0    1. 9413    3. 7592    2. 3650
   0. 3835    0. 9519    1. 6730    0. 7766    1. 9413         0    0. 9972    0. 6353
   0. 3071    0. 5515    1. 9199    2. 2546    3. 7592    0. 9972         0    1. 5249
   0. 6208    0. 4895    1. 0774    2. 0712    2. 3650    0. 6353    1. 5249         0
```



```
dist22 =

   1. 0e+06 *

         0    2. 3055    0. 6521    1. 3648    4. 2993    2. 1087    1. 3539    0. 6879
   2. 3055         0    0. 5236    2. 1671    1. 5098    1. 6599    0. 7354    1. 5635
   0. 6521    0. 5236         0    1. 4045    2. 4664    1. 2546    0. 5639    0. 5447
   1. 3648    2. 1671    1. 4045         0    1. 4260    1. 9026    0. 3972    1. 2782
   4. 2993    1. 5098    2. 4664    1. 4260         0    2. 5041    0. 8404    2. 9996
   2. 1087    1. 6599    1. 2546    1. 9026    2. 5041         0    1. 1002    0. 3976
   1. 3539    0. 7354    0. 5639    0. 3972    0. 8404    1. 1002         0    0. 8212
   0. 6879    1. 5635    0. 5447    1. 2782    2. 9996    0. 3976    0. 8212         0
```



```
dist33 =

   1. 0e+06 *

         0    3. 6670    1. 3235    1. 2492    1. 7537    0. 8770    1. 2615    2. 3029
   3. 6670         0    1. 0615    1. 0802    1. 4619    1. 0993    1. 4955    1. 5283
   1. 3235    1. 0615         0    0. 9059    0. 9085    0. 6094    0. 1249    1. 6698
   1. 2492    1. 0802    0. 9059         0    1. 5074    0. 2021    1. 3692    1. 5088
   1. 7537    1. 4619    0. 9085    1. 5074         0    0. 6427    0. 5783    0. 2244
   0. 8770    1. 0993    0. 6094    0. 2021    0. 6427         0    0. 7588    0. 7326
   1. 2615    1. 4955    0. 1249    1. 3692    0. 5783    0. 7588         0    1. 3998
   2. 3029    1. 5283    1. 6698    1. 5088    0. 2244    0. 7326    1. 3998         0
```

Fig.8 Between-cluster distance in 3 algorithms

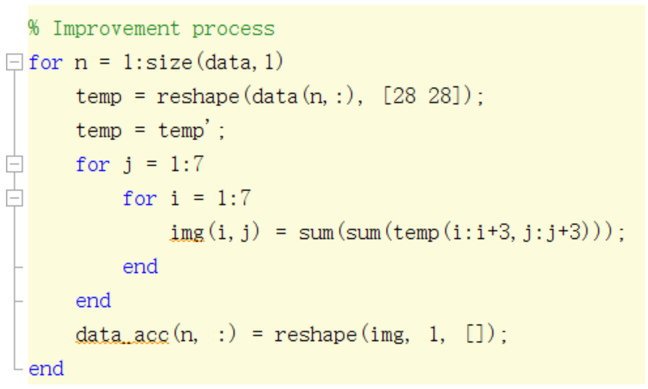
Here we could obviously see that in 7-cluster case of C-Means algorithm, there is a bad clustering which results in a NaN distance. As for 8-cluster case, from Fig.7 and Fig.8 we can find from the un-normalized data that the within-cluster distances are in 2~3 * 10^5 scale and between-cluster distances are in 3~30 * 10^5, 6~40* 10^5, 8~40* 10^5 scales. This means that all algorithms show not too bad performance for clustering task. In all 3 algorithms, KNN shows the best performance because it is a supervised learning algorithm and has much advantages in such task with many features.

But still there remain several problems:

1) Number of features. Theoretically, choosing more features should cover more information and therefor show better clustering performance than situation with fewer features. But trails show contradictory results, which is quite confusing. For example, when 50 features were chosen, FCM algorithm outputs all samples in 2 clusters instead of 8.
2) Parameters of algorithm. FCM introduced the fuzzy membership function and obviously speed up the calculation than C-Means. But the objective function always show larger than C-Means and show worse clustering performance than C-means.

IV. Improvements

While coding for KNN algorithm, I found that the given data is MNIST handwritten figures, 784 features construct a 28*28-pixel figure. Therefore, we can implement some tricks to deal with the number of features. I tried a convolution-like procedure and reduced the data to 49 features.



```
% Improvement process
for n = 1:size(data, 1)
    temp = reshape(data(n, :), [28 28]);
    temp = temp';
    for j = 1:7
        for i = 1:7
            img(i, j) = sum(sum(temp(i:i+3, j:j+3)));
        end
    end
    data_acc(n, :) = reshape(img, 1, []);
end
```

Fig.9 Dimensionality Reduction

There is another advantage for such trick. We find that the top 7 features can cover 99% information. This many further improve the clustering performance.

Looking back to the clustering results, obviously we can see that the original data is very condensed in given feature space and hard to find satisfied clustering results. Since given data are pictures, we can do more before clustering, such as binarization, filtering, and other more sophisticated measures to choose more appropriate features.

V. Conclusions

In this experiment, I implemented C-Means, FCM and KNN algorithms to find the clusters for given 48200 * 784 data. KNN shows the best clustering result because we checked the data and split training dataset and test dataset for it while the other two algorithms also show not too bad clustering results.

While coding, I gained deeper recognition of the three algorithms. After implementation and analysis, I revised the conceptions and formula of many clustering criteria. This experiment does help me consolidate what I have learned in class.

But still, there remain several aspects to be ameliorated. The most important experience I learned here is that, before implementing the algorithm, we need to do more investigation about the data. How to choose appropriate features matters a lot in later procedures. The choice of parameters for certain algorithm is also worth studying.

In general, this experiment is not as simple as I thought at the very beginning. I still found missing points of knowledge and learned much in this experiment.