Homework for Artificial Intelligence Problem Solving and Practice

Author: Kailing Wang(Email)
Shanghai Jiao Tong University

2022.6.15

***Abstract*—In this homework, I used different self-written unsupervised clustering methods and compared them each to decide which works better to do clustering on the given four datasets.**

***Index Terms*—Clustering, Dimension Reduction, K-means, Expectation Maximization, Gaussian Mixture Model, Davis-Bouldin Index, Dunn Index, Multiple Dimensional Scaling, Primary Component Analysis, K Nearest Neighborhood, Isometric Mapping**

I. Algorithm and Code

I've realized a package-like python script file with name Lo_clustering_with_dr.py, which is now open source on github at openFiles. **I don't think it necessary to explain the algorithm in detail(for that's just copying from book)**, but here I have to explain the specific feature of my code, and special method used during coding. My code is 100% original, with 0% copied or online reference. It may not be perfect, but it contains my weeks of hard work.

I realized class KMeans, Gaussian Mixture, Clustering Assessment, MDS, PCA and Isomap along with other tool functions. All the derivation process are from book Machine Learning[1]. I wrote very detailed explanation and usage example in the code file.

A. *Tool functions*

1) Distance measurement: Default distances used are Euclidean Distance in the form of function 1

$$d_{ed}(x,y)=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\cdots+(x_n-y_n)^2} \tag{1}$$

Of course, when computing, the square root is unnecessary when we only need to compare, so I wrote square root as an option and defined default cmp function.

I also realized Minkowski Distance 2

$$d_{mk}(x,y)=\sqrt{\sum_{u=1}^{n}|x_u-y_u|} \tag{2}$$

Users can select p for Minkowski and use it as the distance inside KMeans or Dimension Reduction.

*2) **Probability**:* Standard gaussian function 3 is realized, and the function support vector-like input.

$$p(\boldsymbol{x})=\frac{1}{(2\pi)^{\frac{n}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}}e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{3}$$

*3) **Assessment**:* The assessment methods are Davis-Bouldin Index 8 (the lower, the better) and Dunn Index 9 (the higher, the better)

$$\operatorname{avg}(C)=\frac{2}{|C|\left(|C|-1\right)}\sum_{1\leqslant i\leq j\leqslant|C|}\operatorname{dist}\left(x_i,x_j\right) \tag{4}$$

$$\operatorname{diam}(C)=max_{1\leqslant i\leq j\leqslant|C|}\operatorname{dist}\left(x_i,x_j\right) \tag{5}$$

$$d_{min}\left(C_i,C_j\right)=min_{x_i\in C_i,x_j\in C_j}\operatorname{dist}\left(x_i,x_j\right) \tag{6}$$

$$d_{cen}\left(C_i,C_j\right)=\operatorname{dist}\left(\mu_i,\mu_j\right) \tag{7}$$

$$\mathrm{DBI}=\frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\operatorname{avg}\left(C_i\right)+\operatorname{avg}\left(C_j\right)}{d_{\text{cen}}\left(C_i,C_j\right)}\right) \tag{8}$$

$$\mathrm{DI}=\min_{1\leqslant i\leqslant k}\left\{\min_{j\neq i}\left(\frac{d_{\min}\left(C_i,C_j\right)}{\max_{1\leqslant l\leqslant k}\operatorname{diam}\left(C_l\right)}\right)\right\} \tag{9}$$

Both indexes are internal, for there isn't a standard for this homework.

*4) **Other**:* Some of the functions for vectors are realized, for I didn't know we could use numpy. Vector add and decide and vector.T.vector were realized. Other tools include data normalize, dijkstra.

B. *Clustering*

*1) **KMeans**:* My KMeans allows user to decide initialization seed or directly give the initial centroids. The parameter tolerance is to stop the iteration when the maximum movement of the centroid is lower than tol, and the parameter stop means maximum iteration round.

My KMeans finds initial centroids by Gaussian fit each dimension of the dataset and randomly choose a value within range $[\mu-\sigma,\mu+\sigma)$ as the coordinate of this dimension for one centroid. This method is not as good as KMeans++, but it is original.

After comparing, my KMeans works as fine as the one in sklearn under most tested seeds. As I can decide the

initial centroid and support any distance function, my KMeans sometimes work better. In this home work, my KMeans converges after only 1 rounds for problem 1 and 4 rounds for problem 2.

*2) **GMM**:* It's really hard to code such complex matrix calculation as a new numpy learner, so the code is a bit ugly. My GMM include all the features of KMeans. As for initialization, random initialize sometimes work badly. I recommend using the default method: initialize with KMeans to get clusters, and then calculate mean position and covariance matrix.

The update formulas are as follows 10, 11:

$$\begin{aligned} p_{\mathcal{M}}\left(z_j = i \mid \boldsymbol{x}_j\right) &= \frac{P\left(z_j = i\right) \cdot p_{\mathcal{M}}\left(\boldsymbol{x}_j \mid z_j = i\right)}{p_{\mathcal{M}}\left(\boldsymbol{x}_j\right)} \\ &= \frac{\alpha_i \cdot p\left(\boldsymbol{x}_j \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)}{\sum_{l=1}^{k} \alpha_l \cdot p\left(\boldsymbol{x}_j \mid \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\right)} \end{aligned} \tag{10}$$

$$\begin{aligned} \boldsymbol{\mu}_i' &= \frac{\sum_{j=1}^{m} \gamma_{ji} \boldsymbol{x}_j}{\sum_{j=1}^{m} \gamma_{ji}} \\ \boldsymbol{\Sigma}_i' &= \frac{\sum_{j=1}^{m} \gamma_{ji} \left(\boldsymbol{x}_j - \boldsymbol{\mu}_i'\right) \left(\boldsymbol{x}_j - \boldsymbol{\mu}_i'\right)^{\mathrm{T}}}{\sum_{j=1}^{m} \gamma_{ji}} \\ \alpha_i' &= \frac{\sum_{j=1}^{m} \gamma_{ji}}{m} \end{aligned} \tag{11}$$

It usually converge after more but less than 150 rounds in less than 1 minute.

C. Dimensional Reduction

*1) **MDS**:* Multiple Dimensional Scaling, easy to realize with numpy.linalg, could convert data of any dimension to certain dimension fast.

*2) **PCA**:* Primary Component Analysis. Similar to MDS, use bigger eigenvalues and vectors to reduce dimension.

*3) **Isomap**:* Calculate distance with dijkstra on a connected graph generated by KNN, and then user can shoose MDS or PCA to reduce dimension.

The former two method runs rather fast, but because dijkstra is slow and k need to be large in KNN inside Isomap for sometimes data is not very sparse, Isomap usually takes a while.

II. Problem 1

The distribution of dataset 1 is shown in Fig. 1. and the problem is to divide it into to clusters.

There is no obvious boundary. So it's actually hard to decide how to do clustering.

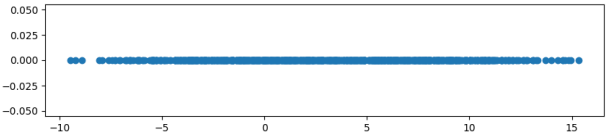

Fig. 1. Distribution of dataset 1

A. Method 1: Simple average

Suggest it is a simple linearly separable problem, and the distribution then we just need to calculate the mean or median of the dataset. I chose to devide by mean, and the result is Fig. 2.

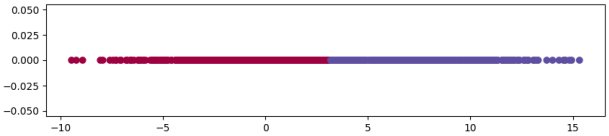

Fig. 2. Clustering using method 1 on dataset 1

B. Method 2: KMeans

I tried KMeans myself, and the result is like Fig. 3. Actually the result looks just like dividing by mean valve.

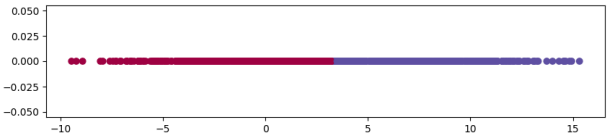

Fig. 3. Clustering using method 2 on dataset 1

C. Method 3: Kernel function+KMeans

Anyway, the dataset doesn't seem to be separable at all. However, since no feature is known, I have no way to design a kernel function, so I look into the distance between points, and found that the points on both ends seem to part farther from points in the middle area. I draw Fig. 4. to describe this feature, based on which I designed function 12:

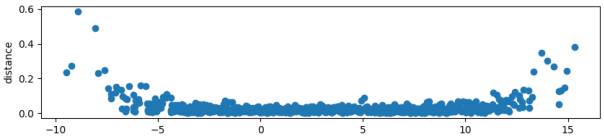

Fig. 4. Distance between a point and its neighbors

$$f(x) = -\left(x - r\right)^4 \tag{12}$$

where r is the radius of data defined as 13

$$r = x_{max} - x_{min} \quad for \quad x \in data \tag{13}$$

Fig. 5. shows the result.

It seems that I've perfectly separates points on both ends from the central ones.

D. Method 4: Expectation Maximization(GMM)

I tried to draw dataset 1 with a histogram Fig. 6

This figure reminds me of Gaussian Distribution. So I realized GMM and tried. See Fig 7, in which I also plotted gaussian probability

Here KMeans converged after 1 round, and GMM 23 rounds. I chose this as the final result.

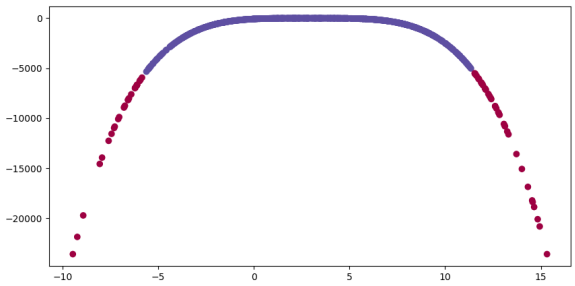

Fig. 5. Clustering using method 3 on dataset 1

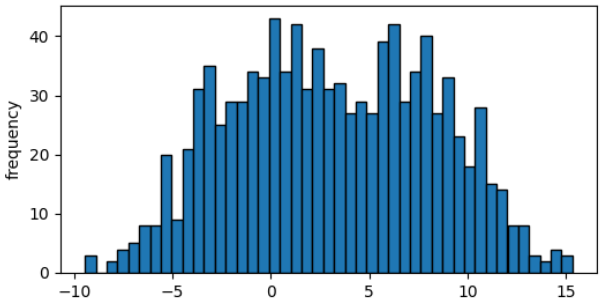

Fig. 6. Histogram of dataset 1

III. Problem 2

The dataset is like Fig. 8.

Obviously we can recognize four or five clusters.

A. The best K(Clusters for KMeans)

Fig 9 shows results using KMeans for cluster number from 3 to 6.

To further decide the best K for KMeans, I calculate DBI and DunnI for K from 2 to 10, as shown in Fig. 10.

Both index decide that 4 cluster is the best choice.

B. The best C(components for GMM)

I repeated this process using GMM, and the index (in Fig. 10.) also shows C should be 4.

Fig. 11. compares the results of the two. As we can see, the boundaries of KMeans are strict lines, and looks curt, while those of GMM looks more smooth. Actually when C = 5 the result looks really beautiful(all the pictures are attached with code file). I feel pity 5 is not the best.

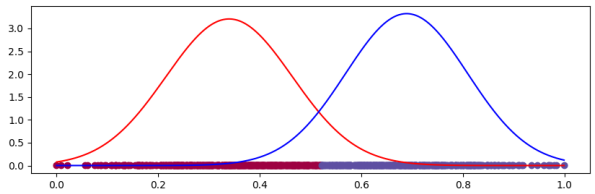

Fig. 7. Method 4 on dataset 1 and gaussian

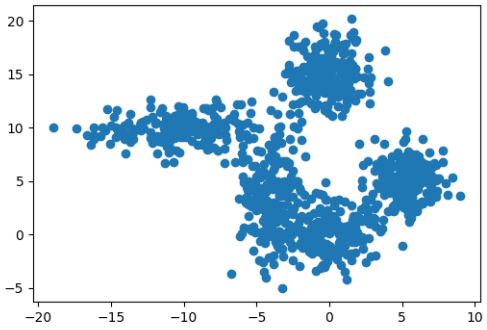

Fig. 8. Distribution of dataset 2

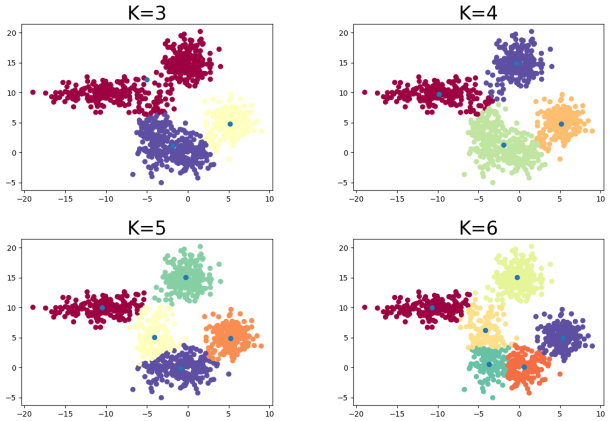

Fig. 9. K = 3, 4, 5, 6 on dataset 2

After thinking twice, I decide to put Fig. 12. here. I suppose the TA generated 5 clusters, but the data was just not very well that the index showed 4 is better. Anyway, I have to choose 4 cluster with GMM as the final result.

IV. Problem 3

A. Dimension Reduction

Fig. 13. is the result of MDS and PCA reducing dataset 3 to 2 dimension. One of the clusters is obvious, but the other two are close.

B. Clustering

I tried to use different combination of method do clustering. Some one said in the QQ chat group that he directly applied KMeans without dimensional reduction, so I made a simple comparison. I tried MDS+KMeans, MDS+GMM, PCA+KMeans, PCA+GMM, KMeans-only and GMM-only. I won't put all the graph here, for there are already too much graphs.

Fig. 14. shows result of MDS+KMeans and KMeans-only. They are likely the same. So what's the point in dimension

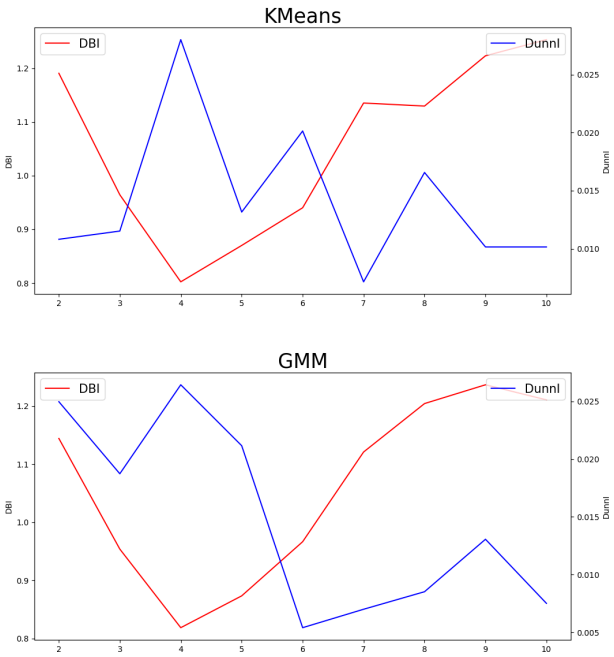

Fig. 10. DBI and DI for KMeans and GMM on dataset 2

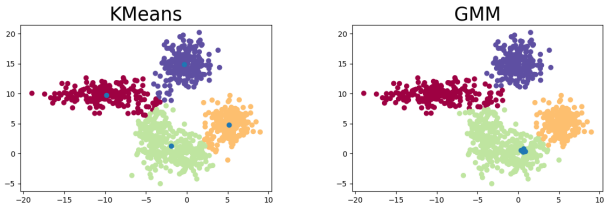

Fig. 11. Results from KMeans and GMM on dataset 2

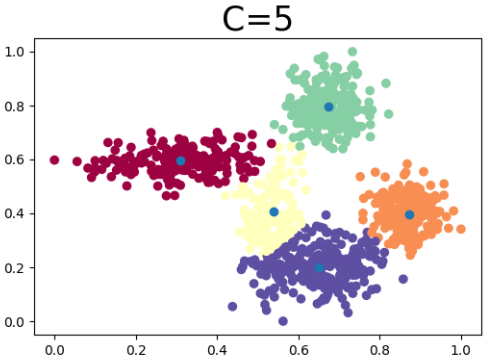

Fig. 12. GMM C=5 on dataset 2

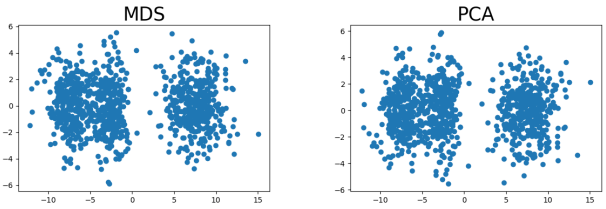

Fig. 13. Results of MDS and PCA on dataset 3

reduction? I can't understand. The dataset wasn't properly constructed.

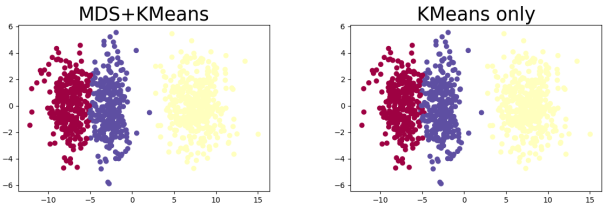

Fig. 14. MDS+KMeans and KMeans-only on dataset 3

But I'm not satisfied. there are two clusters very close. I wonder if some of the distance was ignored during reduction. Then I tried to reduce the data to 5 dimension and apply GMM, and find Fig. 15.

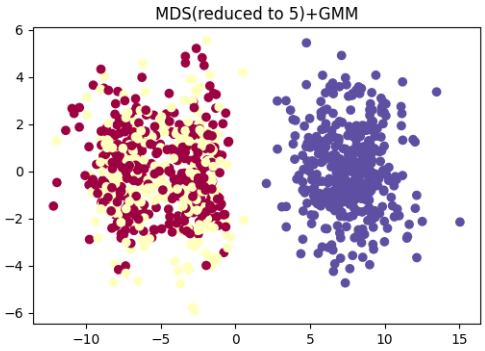

Fig. 15. Results of MDS and PCA on dataset 3

C. Final solution

I tried to apply Isomap to reduce dimension, but I find that the k in KNN used in Isomap need to be over 200 to get the k-distance graph connected, which means there must be a huge gap between the data, and the data is not sparse at all! So I gave up coding Kernelized PCA.

I thought of a method: I could only apply Isomap only on the closer two clusters. I used KMeans to separate the obvious cluster, and then use Isomap to reduce dimension of the other part of the dataset. However, the result was like Fig. 16. There must be something in the higher dimension, but I can't figure out a way to visualize it. At last, I used PCA to reduce the data

to 5 dimension and then apply Isomap. Then I use KMeans to do clustering. The result is as Fig. 17

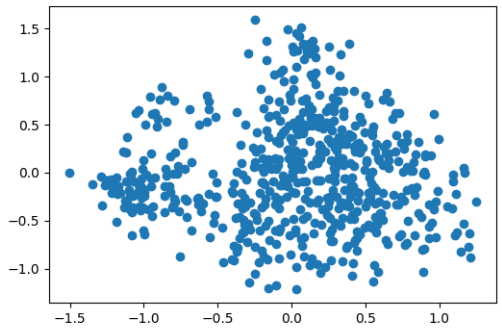

Fig. 16. Isomap on dataset 3

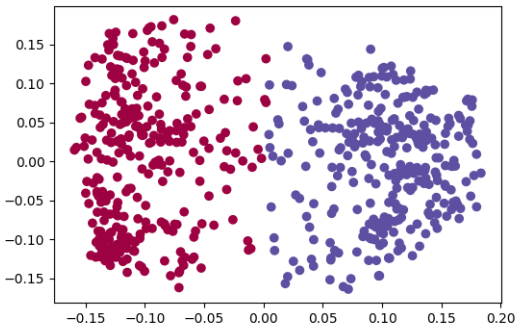

Fig. 17. PCA-Isomap and KMeans on dataset 3

In fact, I simply cut the reduced-to 2 dataset 3 with line x = 1. I chose k=10 for KNN in Isomap. The final result is Fig. 18. and the final label is Fig. 18.

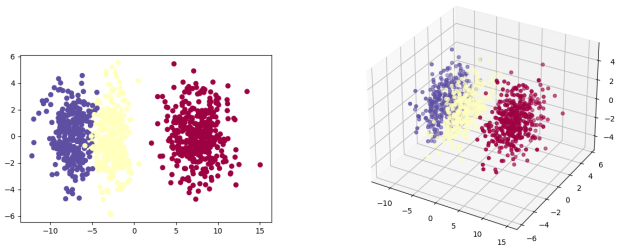

Fig. 18. Final result on dataset 3

V. Problem 4

This problem is nothing more than the combination of problem 2 and problem 3. Fig. 19. shows reducing dimension of dataset 4 using MDS and PCA. 2000 points was chosen to plot to look clearer. Under a dataset of 10000, MDS takes much longer for it concerns distance calculation. Isomap includes

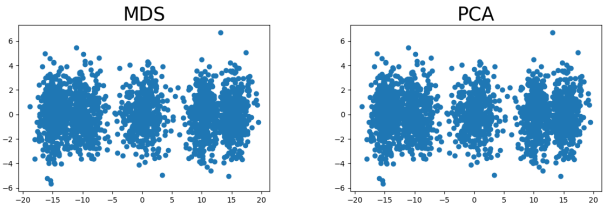

Fig. 19. Dimension reduction on dataset 4

dijkstra, so it needs around a week to finish calculation because for dijkstra is $O(n^3)$.

During plotting, I realized the result from MDS and PCA are completely the same, and I also find all the other 127 dimension are useless. The deciding factor is the first dimension only. Anyway, I still calculated DBI and DunnI for k form 2 to 10. PCA was faster, so I used PCA to reduce the dataset to dimension 3, 5 and 10 and applied KMeans and GMM. This time, DBI was lowest at 2, and DunnI highest at 3. However, at least 5 obvious clusters are recognized.

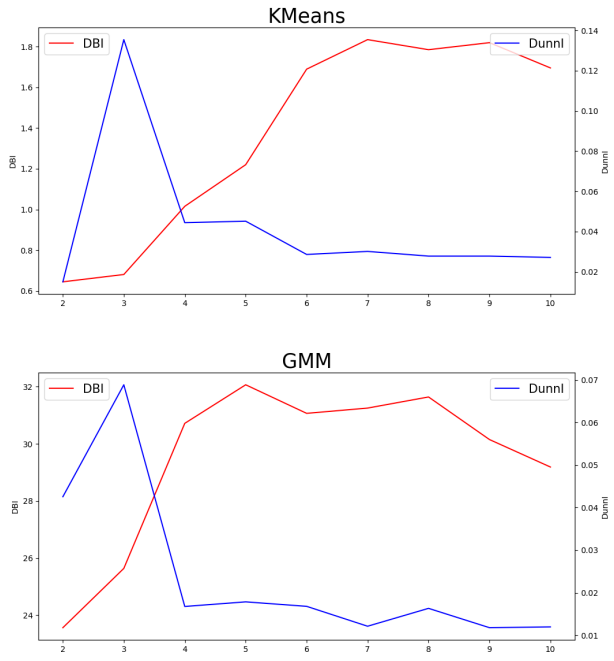

Fig. 20. DBI and DI on dataset 4(reduced to 5)

I also have to show you the result of GMM(C=10) Fig. 21. It's absurd, but beautiful.

Next I tried directly apply KMeans directly, and the index and results are about the same of PCA and KMeans. This also provide evidence for the dataset is poorly constructed.

I tried and tried, but I can hardly give a reason for choosing k = 5. Square error was considered, but the elbow is ambiguous to determine.

Finally, I choose PCA(to 3) and KMeans(k = 5) to give the final label Fig. 22

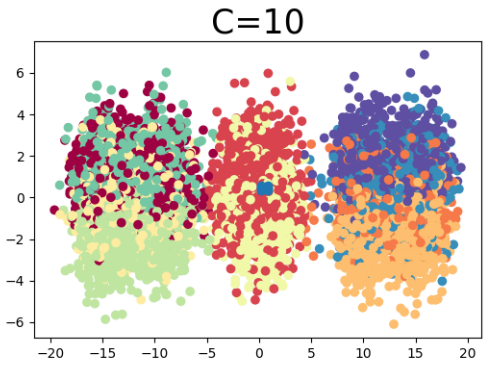

Fig. 21. PCA and GMM(C=10) on dataset 4

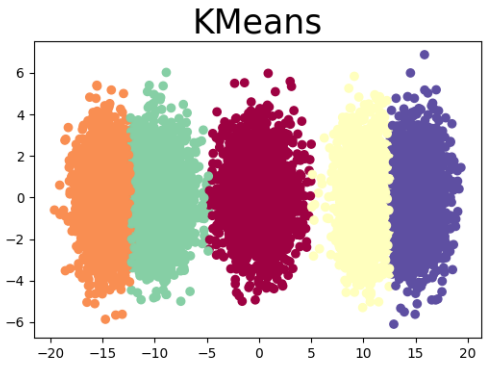

Fig. 22. Final result on dataset 4

ACKNOWLEDGMENT

Here I sincerely thank Professor Cewu Lu and Lixin Yang and other TA, who open the door of machine learning and deep learning for me.

REFERENCES

[1] Machine Learning. Zhou Zhihua. Tsinghua University Press. 2016.