PRML ASSIGNMENT III

1 Problem Definition

1.1 Part I

In this part, you are required to construct a clustering dataset without any labels.

1.2 Part II

In this part, you are required to design Gaussian Mixture Models (GMM) to finish the unlabeled clustering task.

2 Dataset Generation

We first generate a clustering dataset without any labels that follows K Gaussian distributions. Figure 1 shows the 300-point dataset without any labels:

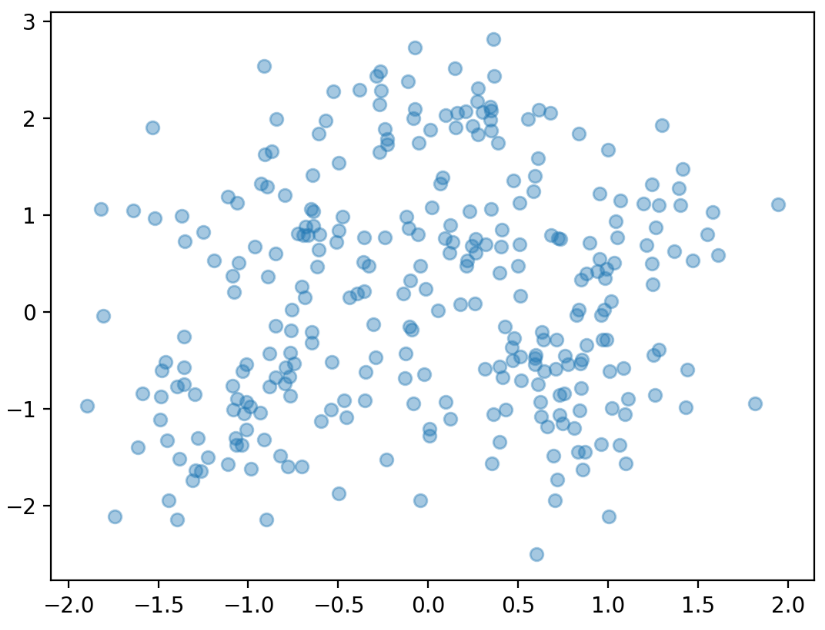

Figure 1: a clustering dataset without any labels

In fact, this dataset follows a mixture of K Gaussian distributions (Here we set K=5) as Figure 2 shows:

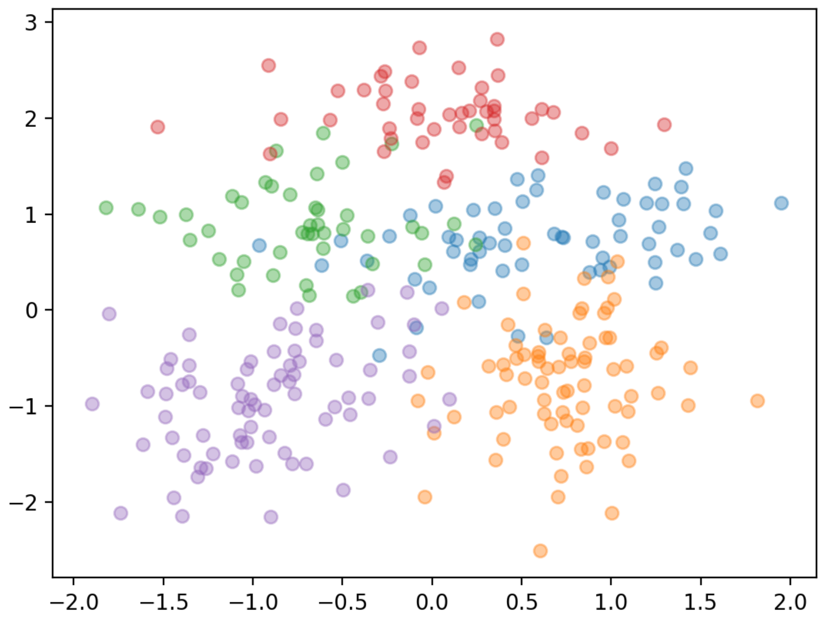

Figure 2: a clustering dataset with real labels

3 Clustering Task

Considering the dataset $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ without labels in Figure 1, our goal is to partition the dataset into some number K of clusters.

3.1 K-means algorithm

We firstly use the K-means algorithm to help the initialization of our model. The main idea is to find a set of vectors $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$ representing the centers of the clusters, such that the sum of the square-distance of each point to its closest center point will be a minimum.

The K-means algorithm is as follows:

1) Randomly initialize the center point $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$;

2) Iteratively perform the following two steps:

a) Assignment step:

$$\boldsymbol{S}_i^{(t)} = \{\boldsymbol{x}_p : \|\boldsymbol{x}_p - \boldsymbol{\mu}_i^{(t)}\|^2 \leqslant \|\boldsymbol{x}_p - \boldsymbol{\mu}_j^{(t)}\|^2 \ \forall j, 1 \le j \le K\}$$

b) Update step:

$$\boldsymbol{\mu}_i^{t+1} = \frac{1}{|\boldsymbol{S}_i^{(t)}|} \sum_{x_j \in \boldsymbol{S}_i^{(t)}} \boldsymbol{x}_j$$

Here we made some small **improvements**, we will run K-means algorithm for several times, and then choose the center point $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$ which **maximizes the minimum distance** between the center points. In this way, we are more likely to avoid the local optimal solutions.

3.2 EM algorithm applied to GMM models

Now based on the latent variable viewpoint, the posterior distribution cannot be observed in the Gaussian Mixture Models. A general technique for finding maximum likelihood estimators in latent variable models is the expectation-maximization (EM) algorithm. In our GMM models, suppose that the latent variable category z follows a multinomial distribution $\boldsymbol{\pi}$, then we can calculate the posterior probability to complete the classification task:

$$p(\boldsymbol{x}|z=k, \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)$$

Consider the problem of parameter estimation with latent variables, we can motivate the EM algorithm in the context of the Gaussian Mixture Models.

We firstly initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k.

In the **E-step**, we evaluate the posterior probability γ_{nk}:

$$\begin{aligned}\gamma_{nk} &\triangleq p(z^{(n)} = k|x^{(n)}) \\ &= \frac{p(z^{(n)})p(x^{(n)}|z^{(n)})}{p(x^{(n)})} \\ &= \frac{\pi_k \mathcal{N}(x^{(n)}|\mu_k, \sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x^{(n)}|\mu_k, \sigma_k)}\end{aligned}$$

where γ_{nk} denotes that component k is responsible for generating point $\boldsymbol{x}_n$, i.e. the posterior probability that the sample $\boldsymbol{x}_n$ belongs to the k-th Gaussian distribution.

In the **M-step**, we re-estimate the parameters $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and π_k by maximize the following function:

$$\begin{aligned}ELBO(\gamma, \mathcal{D}|\pi, \mu, \sigma) &= \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{nk} log \frac{p(x^{(n)}, z^{(n)} = k)}{\gamma_{nk}} \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{nk} \left(\frac{-(x-\mu_k)^2}{2\sigma_k^2} - log\sigma_k + log\pi_k\right) + C\end{aligned}$$

Here we set $q(z=k) = \gamma_{nk}$ in the EM algorithm, and C is a constant.

And we set $N_k = \sum_{n=1}^{K} \gamma_{nk}$, then we can get the re-esimated parameters:

$$\begin{aligned}\pi_k^{new} &= \frac{N_k}{N} \\ \boldsymbol{\mu}_k^{new} &= \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{nk} \boldsymbol{x}^{(n)} \\ \boldsymbol{\Sigma}_k^{new} &= \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{nk} (\boldsymbol{x}^{(n)} - \boldsymbol{\mu}_k^{new})(\boldsymbol{x}^{(n)} - \boldsymbol{\mu}_k^{new})^T\end{aligned}$$

So we can gradually fit the parameters of the Gaussian Mixture Model in the iterative calculation process of the EM algorithm.

4 Evaluations

4.1 Initialization

Since we will run K-means algorithm for several times to better capture the centers of components, in this experiment, we set the iteration number as 10 for K-means pre-training process, and compare the classification result with the real data distribution to get the accuracy indicator. And we find that the initialization matters when EM algorithm is applied to GMM.

Here we set our means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k as follows:

```
priors = [0.2, 0.25, 0.15, 0.15,0.25]
mu = [[0.7,0.7],[0.8,-0.7],[-0.8,0.9],[0,2],[-1,-1]]
# sigma = ... in the code
```

And the classification result of Figure 1 using EM-GMM (with K-means as initialization process) is as follows in Figure 3:

(Where the data size is 300, and the K-means-iteration is 10, EM-iteration is 50, the hyperparameter of clustering number here is equal to 5, which is the same as the real data distribution)

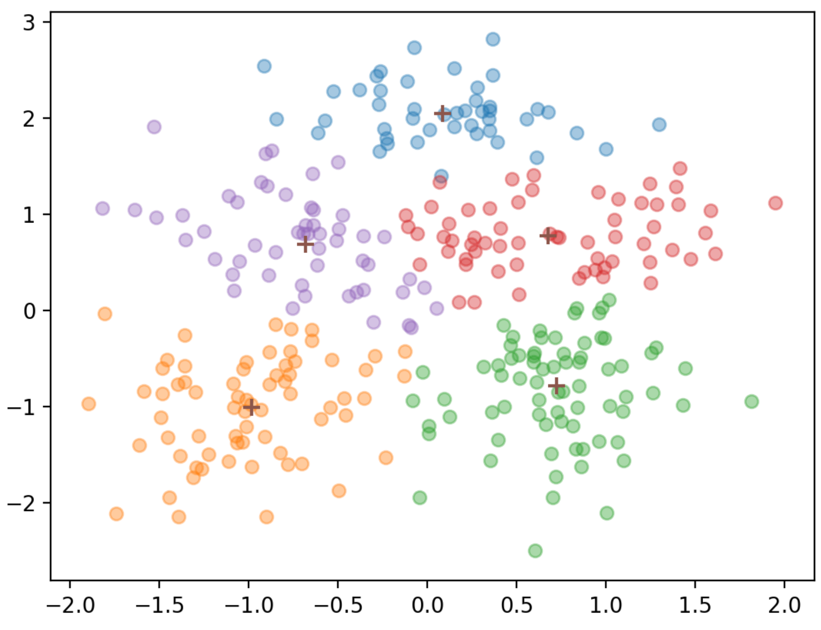

Figure 3: EM-GMM result with 300 points

Here we remove the initialization by comment these lines of code in K-means:

```
# for _ in range(kstep):
        #     label = calc_label(sample, knum, points)
        #     for i in range(knum):
        #         points[i] = np.average(sample[label==i], axis=0)
```

Then we change the data size to evaluate the performance with K-means initialization or not:

Data_size	EM without initialization	EM without initialization
300	0.88	0.84
600	0.823	0.688
1000	0.894	0.751

Table1: Accuracy of EM-GMM

Therefore, with the initialization of K-means algorithm, we can better capture the original data distribution, and then use EM-GMM to restore the true data distribution, and label the data more accurately.

4.2 K-Means vs EM

We can change the data size to evaluate the performance between K-Means and K-Means based EM:

Data_size	K-means	EM algorithm
300	0.837	0.877
600	0.828	0.847
1000	0.848	0.894

Table2: Accuracy of K-means and EM

Obviously, the EM algorithm with K-means initialization will achieve better performance than K-means only. Since the decision boundary of K-means is linear, using EM algorithm to motivate GMM can better fit the original Gaussian mixture distributions.

4.3 Larger Dataset

When the data size increases to 1000, the classification result is as Figure 5:

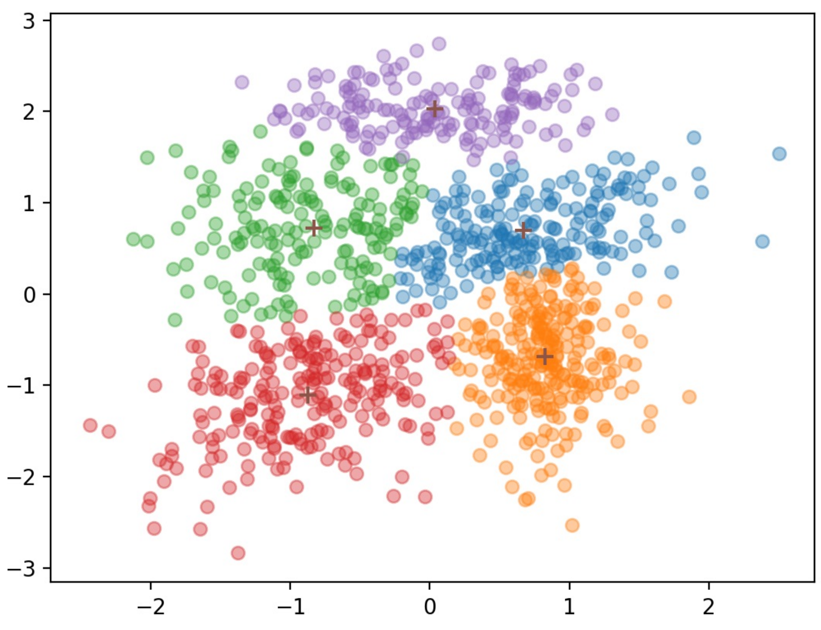

Figure 5: EM-GMM result with 1000 points

When the data size equals 1000, the accuracy still reaches a high value of 0.894 according to Table 1 or Table 2.

The performance will not decrease with the increase of data scale, and it may even increase the performance due to the denseness of the data.

Of course, the accuracy of the classification task is also impacted by the real distribution of the data itself.

5 Program Running

5.1 Generating Data

Here the subparser is gen.

```
python source.py gen -n 500 -p 1
```

-n: data size (e.g. n=500)

-p: whether to plot the dataset (1 is yes, 0 is no).

5.2 Running

Here the subparser is run.

```
python source.py run -p 1
```

-p: whether to plot the result (1 is yes, 0 is no).

-t: whether to show the accuracy (1 is yes, 0 is no, there must be a file *label.data* in the same directory.).

-k: the hyperparameter of clustring numbers (e.g. "-k 5" means k=5)

-f: specify the input file (e.g. "-f data.data").

-i: step of K-means initialization (e.g "-i 10" means kstep=10)

-e: step of EM algorithms (e.g "-e 50" means estep=50)

5.3 Default

```
python source.py
```

To run the default data.data using default settings. (There must be a file *data.data* in the same directory.)

6 References

[1] **"Neural Networks and Deep Learning" by Xipeng Qiu**

[2] **"Pattern Recognition and Machine Learning" by Bishop**