School of Data and Computer Science, Sun Yat-sen University

E12 EM Algorithm (C++/Python)

孙新梦 1834014

November 30, 2020

目录

1 Iris Dataset

Data Set Information:

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper(The use of multiple measurements in taxonomic problems) is a classic in the field and is referenced frequently to this day. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
   - Iris Setosa
   - Iris Versicolour
   - Iris Virginica

Predicted attribute: class of iris plant. This is an exceedingly simple domain. More info please refer to "iris.names" file.

2 EM

2.1 The Gaussian Distribution

The Gaussian, also known as the normal distribution, is a widely used model for the distribution of continuous variables. In the case of a single variable x, the Gaussian distribution can be written in the form

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}}\exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\} \tag{1}$$

where μ is the mean and σ^2 is the variance.

For a D-dimensional vector $\mathbf{x}$, the multivariate Gaussian distribution takes the form

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}\} \tag{2}$$

where $\boldsymbol{\mu}$ is a D-dimensional mean vector, $\boldsymbol{\Sigma}$ is a $D \times D$ covariance matrix, and $|\boldsymbol{\Sigma}|$ denotes the determinant of $|\boldsymbol{\Sigma}|$.

2.2 Mixtures of Gaussians

2.2.1 Introduction

While the Gaussian distribution has some important analytical properties, it suffers from significant limitations when it comes to modelling real data sets. Consider the example shown in Figure 1. This is known as the 'Old Faithful' data set, and comprises 272 measurements of the eruption of the Old Faithful geyser at Yel-lowstone National Park in the USA. Each measurement comprises the duration of the eruption in minutes (horizontal axis) and the time in minutes to the next eruption (vertical axis). We see that the data set forms two dominant clumps, and that a simple Gaussian distribution is unable to capture this structure, whereas a linear superposition of two Gaussians gives a better characterization of the data set.

Example of a Gaussian mixture distribution in one dimension showing three Gaussians (each scaled by a coefficient) in blue and their sum in red.

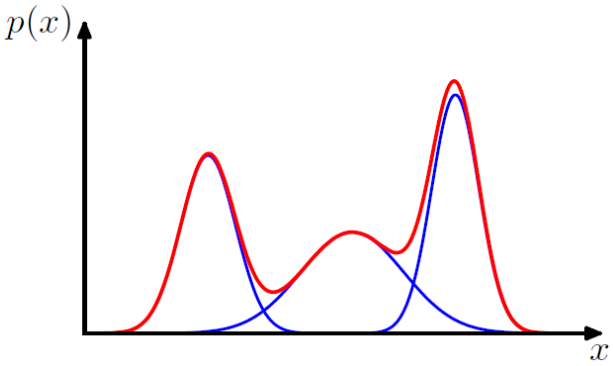


图 1: Example of a Gaussian mixture distribution

Such superpositions, formed by taking linear combinations of more basic distributions such as Gaussians, can be formulated as probabilistic models known as *mixture distributions*. In Figure 1 we see that a linear combination of Gaussians can give rise to very complex densities. By using a sufficient number of Gaussians, and by adjusting their means and covariances as well as the coefficients in the linear combination, almost any continuous density can be approximated to arbitrary accuracy.

We therefore consider a superposition of K Gaussian densities of the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{3}$$

which is called a mixture of Gaussians. Each Gaussian density $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is called a component of the mixture and has its own mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$.

The parameters π_k in (3) are called *mixing coefficients*. If we integrate both sides of (3) with respect to $\mathbf{x}$, and note that both $p(\mathbf{x})$ and the individual Gaussian components are normalized, we obtain

$$\sum_{k=1}^{K} \pi_k = 1. \tag{4}$$

Also, the requirement that $p(\mathbf{x}) \geq 0$, together with $\mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k) \geq 0$, implies $\pi_k \geq 0$ for all k. Combining this with condition (4) we obtain

$$0 \leq \pi_k \leq 1. \tag{5}$$

We therefore see that the mixing coefficients satisfy the requirements to be probabilities.

From the sum and product rules, the marginal density is given by

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k)p(\mathbf{x}|k) \tag{6}$$

which is equivalent to (3) in which we can view $\pi_k = p(k)$ as the prior probability of picking the k^{th} component, and the density $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = p(\mathbf{x}|k)$ as the probability of $\mathbf{x}$ conditioned on k. From Bayes' theorem these are given by

$$\gamma_k(\mathbf{x}) = p(k|\mathbf{x}) = \frac{p(k)p(\mathbf{x}|k)}{\sum_l p(l)p(\mathbf{x}|l)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_l \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}. \tag{7}$$

The form of the Gaussian mixture distribution is governed by the parameters π, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, where we have used the notation $\boldsymbol{\pi} = \{\pi_1, ..., \pi_K\}$, $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k\}$ and $\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_K\}$. One way to set the values of there parameters is to use maximum likelihood. From (3) the log of the likelihood function is given by

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\} \tag{8}$$

where $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$. One approach to maximizing the likelihood function is to use iterative numerical optimization techniques. Alternatively we can employ a powerful framework called expectation maximization (EM).

2.2.2 About Latent Variables

We now turn to a formulation of Gaussian mixtures in terms of discrete *latent* variables. This will provide us with a deeper insight into this important distribution, and will also serve to motivate the expectation-maximization (EM) algorithm.

Recall from (3) that the Gaussian mixture distribution can be written as a linear superposition of Gaussians in the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{9}$$

Let us introduce a K-dimensional binary random variable $\mathbf{z}$ having a 1-of-K representation in which a particular element z_k is equal to 1 and all other elements are equal to 0. The values of z_k therefore satisfy $z_k \in \{0, 1\}$ and $\Sigma_k z_k = 1$, and we see that there are K possible states for the vector $\mathbf{z}$ according to which element is nonzero. We shall define the joint distribution $p(\mathbf{x}, \mathbf{z})$ in terms of a marginal distribution $p(\mathbf{z})$ and a conditional distribution $p(\mathbf{x}|\mathbf{z})$. The marginal distribution over $\mathbf{z}$ is specified in terms of the mixing coefficients π_k, such that

$$p(z_k = 1) = \pi_k \tag{10}$$

where the parameters $\{\pi_k\}$ must satisfy

$$0 \leq \pi_k \leq 1 \tag{11}$$

together with

$$\sum_{k=1}^{K} \pi_k = 1 \tag{12}$$

in order to be valid probabilities. Because $\mathbf{z}$ uses a 1-of-K representation, we can also write this distribution in the form

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}. \tag{13}$$

Similarly, the conditional distribution of $\mathbf{x}$ given a particular value for $\mathbf{z}$ is a Gaussian

$$p(\mathbf{x}|z_k = 1) = (\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{14}$$

which can also be written in the form

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} p(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}. \tag{15}$$

The joint distribution is given by $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$, and the marginal distribution of $\mathbf{x}$ is then obtained by summing the joint distribution over all possible states of $\mathbf{z}$ to give

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{16}$$

where we have made use of (14) and (15). Thus the marginal distribution of $\mathbf{x}$ is a Gaussian mixture of the form (9). If we have several observations $\mathbf{x_1}, ..., \mathbf{x_N}$, then, because we have represented the marginal distribution in the form $p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z})$, it follows that for every observed data point $\mathbf{x}_n$ there is a corresponding latent variable $\mathbf{z}_n$.

We have therefore found an equivalent formulation of the Gaussian mixture involving an explicit latent variable. It might seem that we have not gained much by doing so. However, we are now able to work with the joint distribution $p(\mathbf{x}, \mathbf{z})$ instead of the marginal distribution $p(\mathbf{x})$, and this will lead to significant simplifications, most notably through the introduction of the expectation-maximization (EM) algorithm.

Another quantity that will play an important role is the conditional probability of $\mathbf{z}$ given $\mathbf{x}$. We shall use $\gamma(z_k)$to denote $p(z_k = 1|\mathbf{x})$, whose value can be found using Bayes' theorem

$$\gamma(z_k) = p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{17}$$

We shall view π_k as the prior probability of $z_k = 1$, and the quantity $\gamma(z_k)$ as the corresponding posterior probability once we have observed $\mathbf{x}$. As we shall see later, $\gamma(z_k)$ can also be viewed as the responsibility that component k takes for 'explaining' the observation $\mathbf{x}$.

2.3 EM for Gaussian Mixtures

Initially, we shall motivate the EM algorithm by giving a relatively informal treatment in the context of the Gaussian mixture model.

Let us begin by writing down the conditions that must be satisfied at a maximum of the likelihood function. Setting the derivatives of $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to the means $\boldsymbol{\mu}_k$ of the Gaussian components to zero, we obtain

$$0 = -\sum_{n=1}^{n} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \sum_k (\mathbf{x}_n - \boldsymbol{\mu}_k) \tag{18}$$

Multiplying by $\boldsymbol{\Sigma}_k^{-1}$ (which we assume to be nonsingular) and rearranging we obtain

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{19}$$

where we have defined

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{20}$$

We can interpret N_k as the effective number of points assigned to cluster k. Note carefully the form of this solution. We see that the mean $\boldsymbol{\mu}_k$ for the k^{th} Gaussian component is obtained by taking a weighted mean of all of the points in the data set, in which the weighting factor for data point $\mathbf{x}_n$ is given by the posterior probability $\gamma(z_{nk})$ that component k was responsible for generating $\mathbf{x}_n$.

If we set the derivative of $\ln(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to $\boldsymbol{\Sigma}_k$ to zero, and follow a similar line of reasoning, making use of the result for the maximum likelihood for the covariance matrix of a single Gaussian, we obtain

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}} \tag{21}$$

which has the same form as the corresponding result for a single Gaussian fitted to the data set, but again with each data point weighted by the corresponding posterior probability and with the denominator given by the effective number of points associated with the corresponding component.

Finally, we maximize $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to the mixing coefficients π_k. Here we must take account of the constraint $\sum_{k=1}^{K} \pi_k = 1$. This can be achieved using a Lagrange multiplier and maximizing the following quantity

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda(\sum_{k=1}^{K} \pi_k - 1) \tag{22}$$

which gives

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{23}$$

where again we see the appearance of the responsibilities. If we now multiply both sides by π_k and sum over k making use of the constraint $\sum_{k=1}^{K} \pi_k = 1$, we find $\lambda = -N$. Using this to eliminate λ and rearranging we obtain

$$\pi_k = \frac{N_k}{N} \tag{24}$$

so that the mixing coefficient for the k^{th} component is given by the average responsibility which that component takes for explaining the data points.

2.4 EM Algorithm

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients).

1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.

2. **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{25}$$

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n \tag{26}$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k^{new})(\mathbf{x}_n - \boldsymbol{\mu}_k^{new})^{\mathrm{T}} \tag{27}$$

$$\pi_k^{new} = \frac{N_k}{N} \tag{28}$$

where

$$N_k = \sum_{n=1}^{N}\gamma(z_{nk}). \tag{29}$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N}\ln\{\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\} \tag{30}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

3 Tasks

- Assume that score vectors of teams in the same class are normally distributed, we can thus adopt the Gaussian mixture model. Please classify the teams into 3 classes by using EM algorithm. If necessary, you can refer to page 430-439 in the book `Pattern Recognition and Machine Learning.pdf` and the website `https://blog.csdn.net/jinping_shi/article/details/59613054` which is a Chinese translation.

- You should show the values of these parameters: $\boldsymbol{\gamma}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. You can also visualize the values of these parameters to see the convergence of EM algorithm. Give the results and simple analysis in your report.

- Please submit a file named `E12_YourNumber.pdf` and send it to `ai_2020@foxmail.com`

4 Codes and Results

Code

```
1 """
2     File :EM.py
```

```
3       Name孙新梦:
4       ID:18340149
5       TASK用:算法对数据集进行分类，EMIris输出γ， μ， Σ
6
7   """
8   import math
9   import numpy as np
10  from copy import deepcopy as cp
11
12
13  def get_init(data, k, dim):
14      N = len(data)
15
16      mask = np.random.permutation(N)[:k]
17      miu = cp(data[mask])
18      sigma = np.array([np.zeros((dim, dim)) for i in range(k)])
19      for i in range(k):
20          for j in range(dim):
21              sigma[i, j, j] = np.random.rand()
22      pi = np.ones(k) / k
23      return pi, miu, sigma
24
25  # 高斯函数的模拟，输入一个数据行，输出对应的分布
26  def gaussion(inputs, miu, sigma):
27      D = len(miu)
28
29      # 获得按位的减法
30      m = [inputs[i] - miu[i] for i in range(D)]
31      minus = np.array(m)
32
33      return 1 / math.sqrt(math.pow(2 * math.pi, D) * np.linalg.det(
            sigma)) * math.exp(
34          -0.5 * (minus.reshape(1, D)).dot(np.linalg.inv(sigma)).dot
```

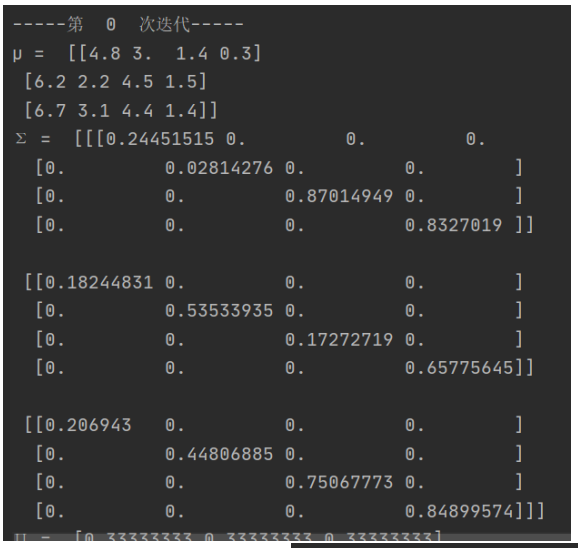
```
            (minus.reshape(D, 1)))

# 分类算法EM
def EM(data, K, dim,lable):
    # 初始化1.
    pi, miu, sigma = get_init(data, K, dim)

    N = len(data)
    # N*的一个矩阵——参数Kshape行数表示哪一个样例，列数表示是哪一类的概率,
    gamma = np.ndarray((N, K))

    # 开始迭代2.
    for epoch in range(10):
        print("\n第——— ", epoch, " 次迭代———")
        print μ(" = ", miu)
        print Σ(" = ", sigma)
        print Π(" = ", pi)

        # E step
        for n in range(N):
            temp_sum = 0
            for k in range(K):
                # 计算val
                val = pi[k] * gaussion(data[n], miu[k], sigma[k])
                gamma[n, k] = val
                temp_sum += val
            gamma[n] /= temp_sum


        # M step
        Nk = gamma.sum(0)
        pi = Nk / N
        for k in range(K):
```

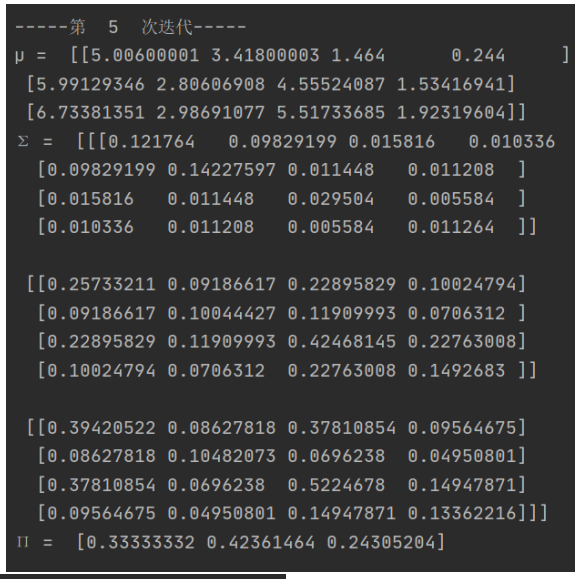
```
67              temp_sum = np.zeros(miu[k].shape)
68              for n in range(N):
69                  temp_sum += gamma[n, k] * data[n]
70              miu[k] = temp_sum / Nk[k]
71              temp_sum = np.zeros(sigma[k].shape)
72              for n in range(N):
73                  temp_sum += gamma[n, k] * ((data[n] - miu[k]).
                        reshape(data[n].size, 1)).dot(
74                          (data[n] - miu[k]).reshape(1, data[n].size))
75              sigma[k] = temp_sum / Nk[k]
76
77
78      # 输出分类并计算准确率
79      correct = 0
80      header_dict = {'Iris-setosa':0,'Iris-versicolor':1,'Iris-
            virginica':2}
81
82      print("\n\n分类**********\n")
83      for n in range(N):
84          the_class = np.argmax(gamma[n])
85          print第(" ", n, " 条，分类为：第 ", the_class, " 类")
86          if n != N-1:
87              if the_class == header_dict[lable[n][:-1]]:
88                  correct += 1
89          else:
90              if the_class == header_dict[lable[n]]:
91                  correct += 1
92      print("\n准确率**********\n",correct/N)
93
94
95
96
97
```

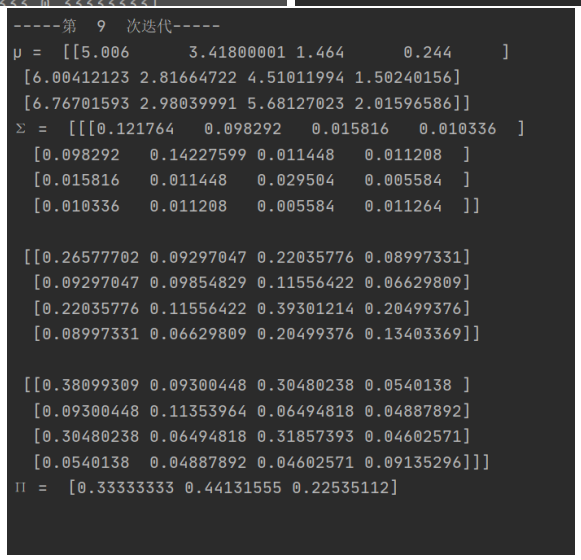
```
98  # 读取数据
99  def readData():
100     file_path = 'iris.data'
101     data = []
102     label = []
103     with open(file_path, 'r') as f:
104         for line in f.readlines():
105             line = line.split(',')
106             data.append(list(map(lambda x: float(x), line[:-1])))
107             label.append(line[-1])
108     return data, label
109
110
111 if __name__ == '__main__':
112     K = 3
113     data, lable = readData()
114     data = np.array(data)
115
116     # dimension = 4
117     EM(data, K, 4,lable)
```

result 本次实验结果输出分为三个部分：

1.输出迭代过程的μ, σ, π 经过10次迭代后算法收敛，可以看到在我们随机选出 μ 上，几个矩阵都

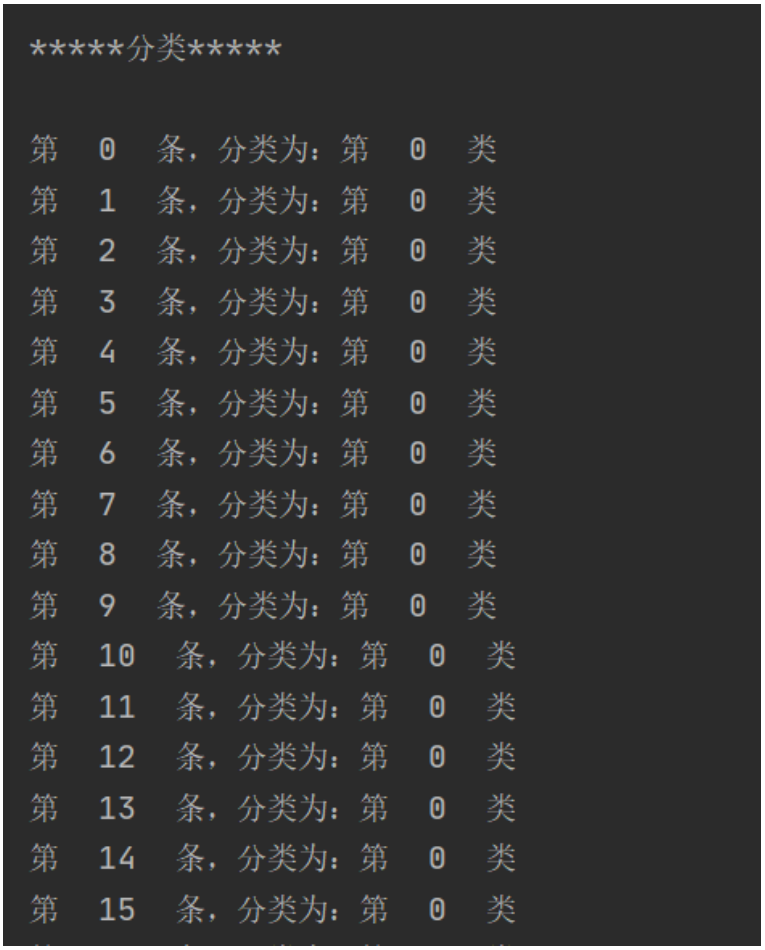






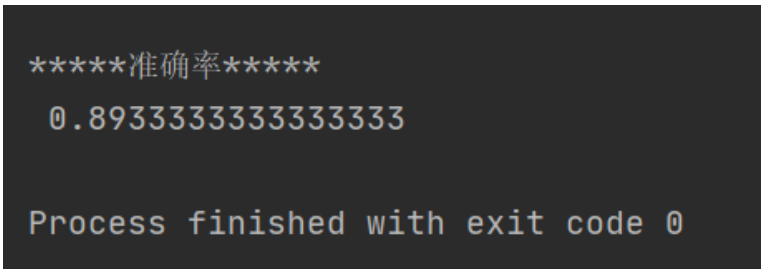
变化越来越小逐渐收敛。

2.输出150条数据的分类结果 在迭代完成之后，根据我们gamma矩阵对应位置最大的那个分类来



做分类，把分类输出。

3.输出准确率 与原先数据集的标签对比之后，计算准确率为百分之89左右。不过准确率会随着



我们随机生成的 μ 值改变而变化，开始的时候我们随机认为三类均值（中心）是哪些，然后根据和均值的距离来逐步调整分类，再调整均值（中心），经过这样几步的E和M，我们的数据能够根据较为准确的中心来计算在划分为这个类的前提下的后验概率，比较三者能够得出最可能是哪一类。

5 感想和分析

本次实验是EM分类算法的实现：通过迭代进行极大似然估计的优化算法，首先随机指定类别的中心，其中E步骤是根据已有的类别中心进行极大似然估计，计算出来每个数据属于三类的概率，接下来M步骤根据分出的类别进行新一轮的中心计算。

算法对初始值很敏感，收敛的优劣很大程度上取决于初始对于 μ 的选择。