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

E12 EM Algorithm (C++/Python)

18340013 Conghao Chen

November 30, 2020

Contents

1 Iris Dataset **2**

2 EM **2**

2.1 The Gaussian Distribution 2

2.2 Mixtures of Gaussians 3

2.2.1 Introduction 3

2.2.2 About Latent Variables 4

2.3 EM for Gaussian Mixtures 6

2.4 EM Algorithm 7

3 Tasks **8**

4 Codes **8**

5 Results **12**

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

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}}\exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\} \tag{2.1.1}$$

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

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\} \tag{2.1.2}$$

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.

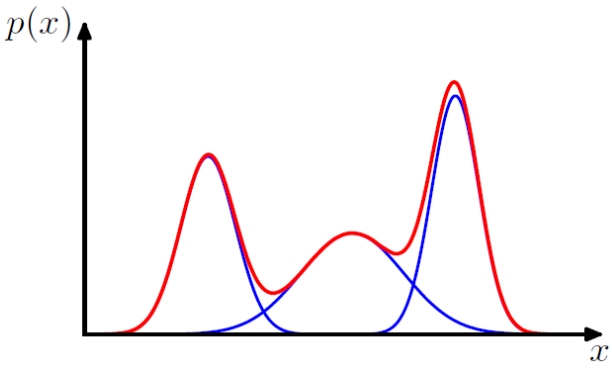


Figure 1: Example of a Gaussian mixture distribution

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

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.2.1}$$

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 (2.2.1) are called *mixing coefficients*. If we integrate both sides of (2.2.1) with respect to $\mathbf{x}$, and note that both $p(\mathbf{x})$ and the individual Gaussian components are normalized,

we obtain

$$\sum_{k=1}^{K} \pi_k = 1. \tag{2.2.2}$$

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 (2.2.2) we obtain

$$0 \leq \pi_k \leq 1. \tag{2.2.3}$$

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

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k)p(\mathbf{x}|k) \tag{2.2.4}$$

which is equivalent to (2.2.1) 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

$$\gamma_k(\mathbf{x}) = p(k|\mathbf{x}) = \frac{p(k)p(\mathbf{x}|k)}{\sum_l p(l)p(\mathbf{x}|l)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_l \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}. \tag{2.2.5}$$

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 (2.2.1) the log of the likelihood function is given by

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\} \tag{2.2.6}$$

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

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.2.7}$$

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

$$p(z_k = 1) = \pi_k \tag{2.2.8}$$

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

$$0 \leq \pi_k \leq 1 \tag{2.2.9}$$

together with

$$\sum_{k=1}^{K} \pi_k = 1 \tag{2.2.10}$$

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

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}. \tag{2.2.11}$$

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

$$p(\mathbf{x}|z_k = 1) = (\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.2.12}$$

which can also be written in the form

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} p(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}. \tag{2.2.13}$$

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

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.2.14}$$

where we have made use of (2.2.12) and (2.2.13). Thus the marginal distribution of $\mathbf{x}$ is a Gaussian mixture of the form (2.2.7). 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

$$\gamma(z_k) = p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2.2.15}$$

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

$$0 = -\sum_{n=1}^{n} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \sum_k (\mathbf{x}_n - \boldsymbol{\mu}_k) \tag{2.3.1}$$

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

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{2.3.2}$$

where we have defined

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{2.3.3}$$

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

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}} \tag{2.3.4}$$

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

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda(\sum_{k=1}^{K} \pi_k - 1) \tag{2.3.5}$$

which gives

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2.3.6}$$

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

$$\pi_k = \frac{N_k}{N} \tag{2.3.7}$$

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

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2.4.1}$$

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

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{2.4.2}$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k^{new})(\mathbf{x}_n - \boldsymbol{\mu}_k^{new})^{\mathrm{T}} \tag{2.4.3}$$

$$\pi_k^{new} = \frac{N_k}{N} \tag{2.4.4}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{2.4.5}$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\} \tag{2.4.6}$$

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

- Please classify the iris 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

```
from scipy.stats import multivariate_normal
from sklearn import preprocessing
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np

class GMM():
    def __init__(self, K, max_iter=500, error=1e-7):
        self.K = K                  # 假设由K个GMM组成
        self.max_iter = max_iter    # 最大迭代次数
        self.error = error          # 收敛误差
        self.samples = 0            # 样本数
        self.features = 0           # 特征数
        self.alpha = []             # 权重
        self.mu = []                # 均值
```

```
        self.sigma = []    # 标准差

    def para(self, data):  # 相关参数
        np.random.seed(7)
        self.mu = np.array(np.random.rand(self.K, self.features))
        self.sigma = np.array([np.eye(self.features) / self.features] * self.K)
        self.alpha = np.array([1.0 / self.K] * self.K)
        # return self.mu, self.sigma, self.alpha
        # print("initial alpha:\n{}\n".format(self.alpha))
        # print("initial mu:\n{}\n".format(self.mu))
        # print("initial sigma:\n{}\n".format(self.sigma))
        # print(self.alpha.shape, self.mu.shape, self.sigma.shape)

    def gauss(self, Y, mu, sigma):  # 多元高斯分布(假设非奇异)
        return multivariate_normal(mean=mu, cov=sigma).pdf(Y)

    def preprocess(self, data):  # 数据预处理
        self.samples = data.shape[0]
        self.features = data.shape[1]
        pre = preprocessing.MinMaxScaler()
        return pre.fit_transform(data)

    def fit(self, data):  # 拟合数据
        data = self.preprocess(data)
        self.para(data)
        weighted_probs = np.zeros((self.samples, self.K))
        for i in range(self.max_iter):
            prev_weighted_probs = weighted_probs
            weighted_probs = self.ESTEP(data)
            change = np.linalg.norm(weighted_probs - prev_weighted_probs)
            if change < self.error:
                break
            self.MSTEP(data, weighted_probs)
        return weighted_probs.argmax(axis=1)

    def ESTEP(self, data):  # E-STEP
        probs = np.zeros((self.samples, self.K))
        for i in range(self.K):
            probs[:, i] = self.gauss(data, self.mu[i, :], self.sigma[i, :, :])
        weighted_probs = np.zeros(probs.shape)
        for i in range(self.K):
```

```
            weighted_probs[:, i] = self.alpha[i] * probs[:, i]
        for i in range(self.samples):
            weighted_probs[i, :] /= np.sum(weighted_probs[i, :])
        return weighted_probs

    def MSTEP(self, data, weighted_probs):  # M-STEP
        for i in range(self.K):
            sum_probs_i = np.sum(weighted_probs[:, i])
            self.mu[i, :] = np.sum(np.multiply(data, np.mat(weighted_probs[:, i]).T),
                axis=0) / sum_probs_i
            self.sigma[i, :, :] = (data - self.mu[i, :]).T * np.multiply((data - self.mu
                [i, :]), np.mat(weighted_probs[:, i]).T) / sum_probs_i
            self.alpha[i] = sum_probs_i / data.shape[0]

        # return self.alpha, self.mu, self.sigma
        # a=np.append[self.alpha]
        # print("alpha:\n{}\n".format(self.alpha))
        # print("mu:\n{}\n".format(self.mu))
        # print("sigma:\n{}\n".format(self.sigma))


    def predict(self, data):  # 输出类别
        return self.ESTEP(data).argmax(axis=1)

    def acc(self,getresult,result):
        if len(getresult) != len(result):
            raise ValueError("Dimension don't match!")
        correct = 0
        for i in range(len(getresult)):
            if getresult[i] == result[i]:
                correct += 1
        return correct/len(getresult)


f = open('iris.txt')
data_list = f.readlines()           # 读出的是str类型
dataset = []
for data in data_list:
    data1 = data.strip('\n')    # 去掉换行符
    data2 = data1.split(',')    # 按,挑选数据
    dataset.append(data2)           # 把这一行的结果作为元素加入列表dataset
```

```
dataset = np.array(dataset)
dataset = dataset.astype(float)

result = np.array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2])
# 可视化
# mpl.rcParams['font.sans-serif'] = [u'simHei']
# mpl.rcParams['axes.unicode_minus'] = False
# plt.scatter(dataset[:,0],dataset[:,1],c = result)
# plt.title("Iris种类与前两个属性图")
# plt.show()
# plt.scatter(dataset[:,2],dataset[:,3],c = result)
# plt.title("Iris种类与后两个属性图")
# plt.show()

gmm = GMM(3)

getresult = gmm.fit(dataset)



# print("convergence alpha:\n{}\n".format(gmm.alpha))
# print("convergence mu:\n{}\n".format(gmm.mu))
# print("convergence sigma:\n{}\n".format(gmm.sigma))
# mpl.rcParams['font.sans-serif'] = [u'simHei']
# mpl.rcParams['axes.unicode_minus'] = False
# plt.scatter(dataset[:,0],dataset[:,1],c = getresult)
# plt.title("GMM预测结果与前两个属性关系图")
# plt.show()
# plt.scatter(dataset[:,2],dataset[:,3],c = getresult)
# plt.title("GMM预测结果与后两个属性关系图")
# plt.show()
print("Predict Result:\n{}\n".format(getresult))
print("True result:\n{}\n".format(result))
print("Accuracy:\n{}".format(gmm.acc(getresult,result)))
```

5 Results

注：对 `iris.txt` 做了下改动, 删除了最后一列 (即类别的那一列), 类别分别用 0、1、2 来表示, 最后直接比较两个数组元素不同个数即可.

首先是聚类后的结果:

```
Predict Result:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 2 1 1 1 2 1 2 1 2 1 1 2 2 1 1 2 1 1 2
 1 1 1 2 2 1 1 1 1 2 2 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 2 2 2 2 2 1 2 1 1 2 2
 2 2 2 2 2 2 1 1 1 2 2 1 2 2 1 2 2 2 1 1 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]

True result:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]

Accuracy:
0.82
```

代码可以输出每次迭代后的 $\boldsymbol{\gamma}$, $\boldsymbol{\mu}$ 和 $\boldsymbol{\Sigma}$, 由于数据较多, 这里只放出了初始化的数据、中间某一次迭代后的结果和收敛结果 (代码里的 $\boldsymbol{\alpha}$ 为文件里的 $\boldsymbol{\gamma}$):

```
initial alpha:
[0.33333333 0.33333333 0.33333333]

initial mu:
[[0.07630829 0.77991879 0.43840923 0.72346518]
 [0.97798951 0.53849587 0.50112046 0.07205113]
 [0.26843898 0.4998825  0.67923    0.80373904]]

initial sigma:
[[[0.25 0.   0.   0.  ]
  [0.   0.25 0.   0.  ]
  [0.   0.   0.25 0.  ]
  [0.   0.   0.   0.25]]

 [[0.25 0.   0.   0.  ]
  [0.   0.25 0.   0.  ]
  [0.   0.   0.25 0.  ]
  [0.   0.   0.   0.25]]

 [[0.25 0.   0.   0.  ]
  [0.   0.25 0.   0.  ]
  [0.   0.   0.25 0.  ]
  [0.   0.   0.   0.25]]]


alpha:
[0.33316429 0.31234811 0.3544876 ]

mu:
[[0.196182   0.59106828 0.07865801 0.05998785]
 [0.55409801 0.32415394 0.62371031 0.5620297 ]
 [0.53675051 0.39774298 0.6955105  0.73978061]]

sigma:
[[[0.00939016 0.01134912 0.00074304 0.00119856]
  [0.01134912 0.02460373 0.00080235 0.0019523 ]
  [0.00074304 0.00080235 0.00084761 0.00039487]
  [0.00119856 0.0019523  0.00039487 0.00195622]]

 [[0.04750384 0.02151403 0.03412835 0.02609117]
  [0.02151403 0.02496592 0.01348575 0.01214028]
  [0.03412835 0.01348575 0.02880491 0.0222121 ]
  [0.02609117 0.01214028 0.0222121  0.01892435]]

 [[0.0212333  0.0080203  0.01039485 0.01460958]
  [0.0080203  0.01128218 0.0044868  0.00910845]
  [0.01039485 0.0044868  0.00883729 0.01258128]
  [0.01460958 0.00910845 0.01258128 0.02700331]]]
```

```
convergence alpha:
[0.33316441 0.31239311 0.35444248]

convergence mu:
[[0.19618195 0.59106811 0.078658   0.05998786]
 [0.55408447 0.32416472 0.6237041  0.56202818]
 [0.53676041 0.39774291 0.6955253  0.73980473]]

convergence sigma:
[[[0.00939017 0.01134914 0.00074304 0.00119855]
  [0.01134914 0.0246038  0.00080236 0.0019523 ]
  [0.00074304 0.00080236 0.00084761 0.00039487]
  [0.00119855 0.0019523  0.00039487 0.00195622]]

 [[0.04749996 0.02151058 0.03412473 0.02608812]
  [0.02151058 0.02496439 0.01348367 0.01213898]
  [0.03412473 0.01348367 0.02880163 0.02220947]
  [0.02608812 0.01213898 0.02220947 0.01892242]]

 [[0.02123369 0.00802071 0.01039432 0.014609  ]
  [0.00802071 0.01128248 0.00448704 0.00910907]
  [0.01039432 0.00448704 0.00883607 0.01257954]
  [0.014609   0.00910907 0.01257954 0.02700141]]]
```

这里是可视化的结果, 由于参数都是多维矩阵加上我能力有限, 所以我只将参数中的 $\boldsymbol{\gamma}$(代码里的 $\boldsymbol{\alpha}$) 变化趋势绘制成了图来观察其收敛情况：

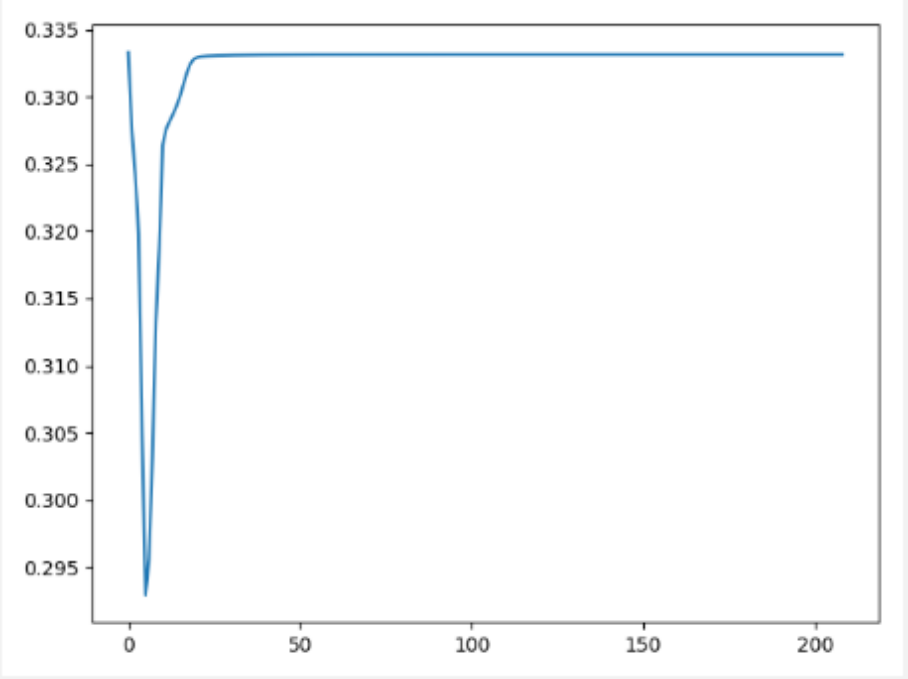


Figure 2: alpha[0]

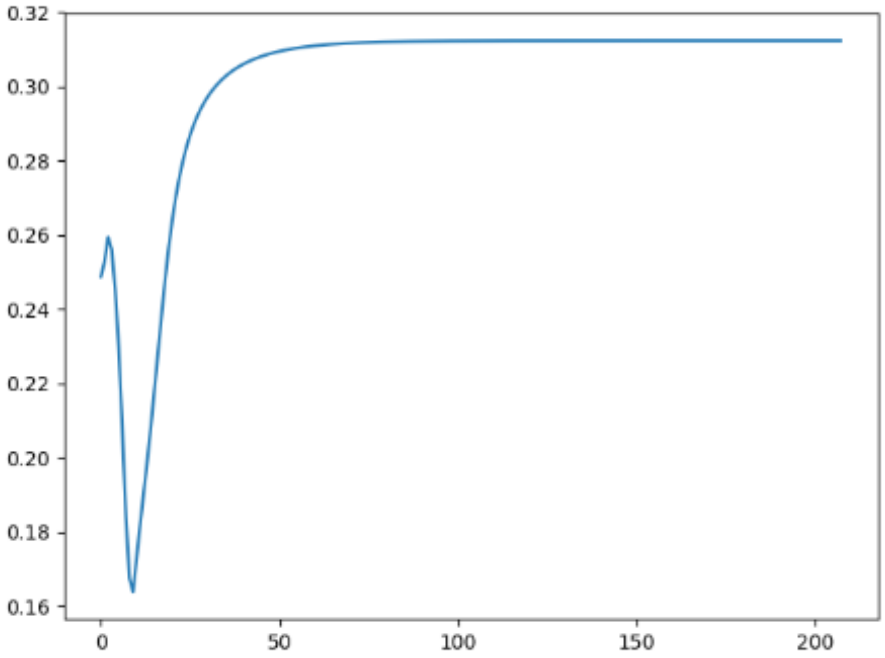


Figure 3: alpha[1]

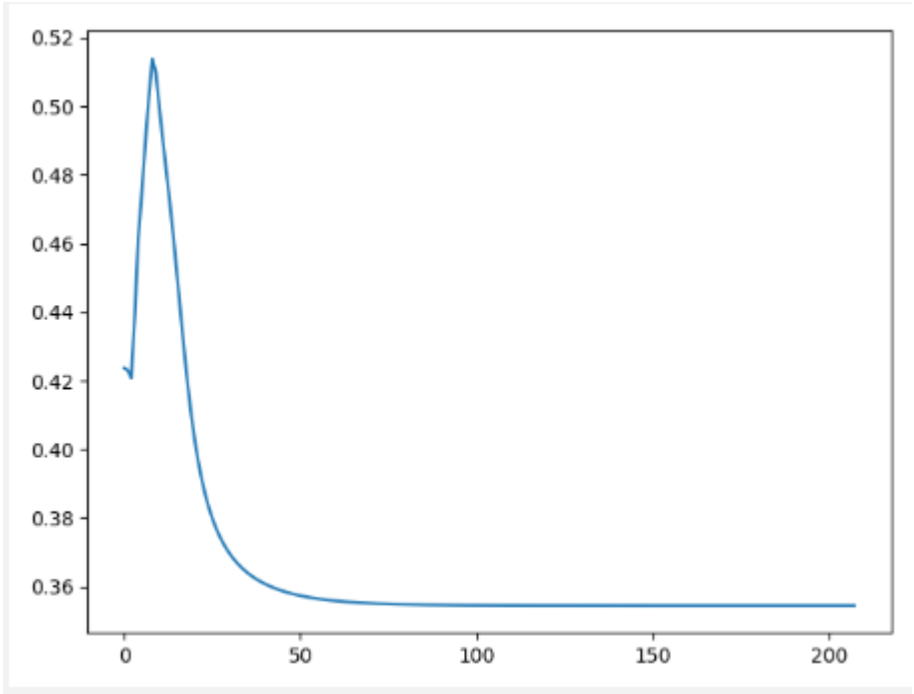
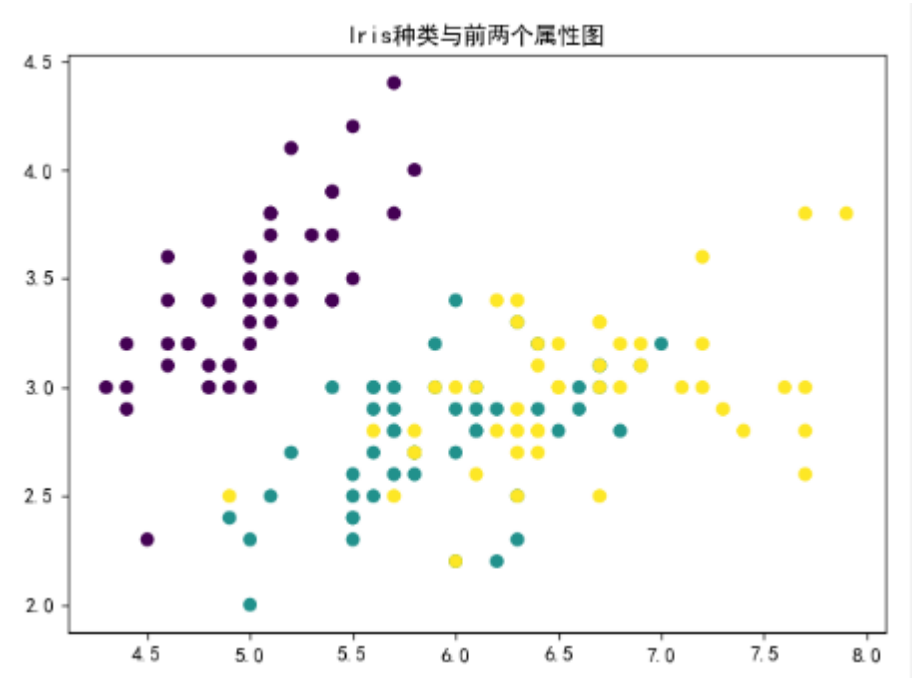
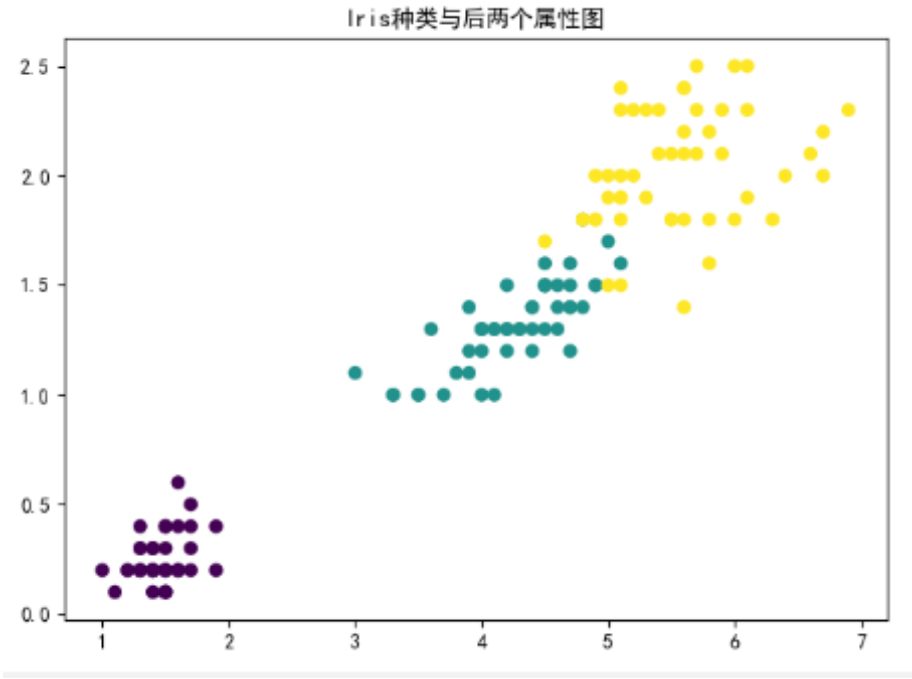


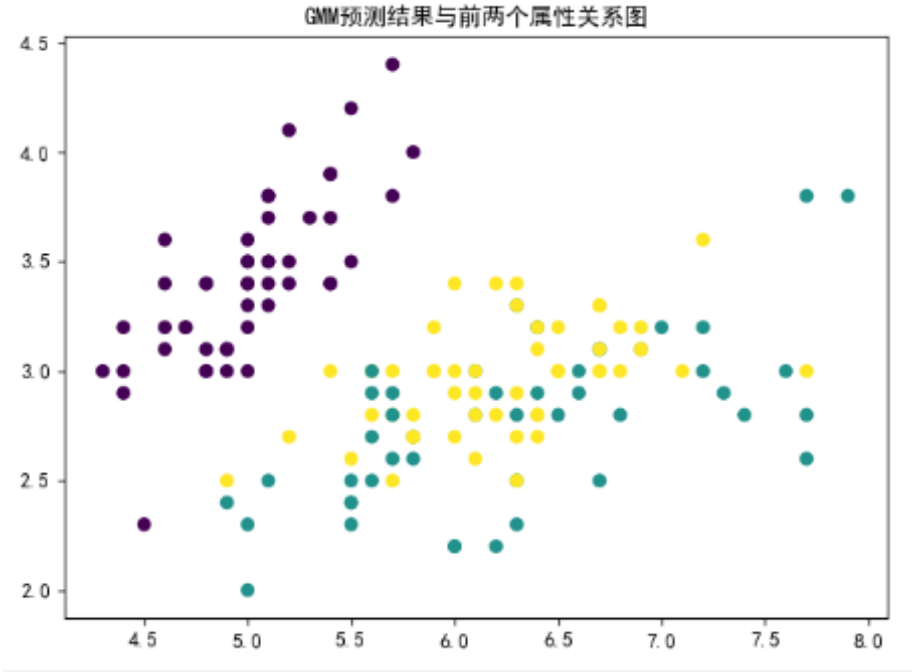
Figure 4: alpha[2]

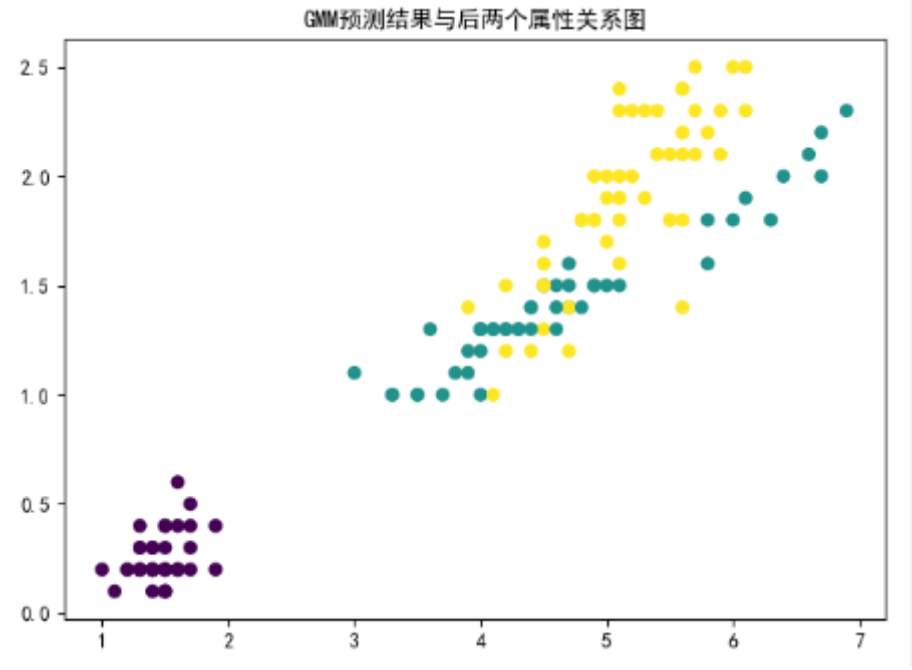
可以很直观的看到 $\boldsymbol{\alpha}(\boldsymbol{\gamma})$ 的确收敛, 虽然 $\boldsymbol{\mu}$ 和 $\boldsymbol{\Sigma}$ 没有可视化但是我通过输出的数据也可以看到收敛过程.

这几张是 `iris` 的四个属性与类别的关系图, 前两张为所给数据 (即正确分类) 分别与第 1,2 个、第 3,4 个属性的关系图；后两张为用 `GMM-EM` 预测的分类结果与第 1,2 个、第 3,4 个属性的关系图：






GMM预测结果与前两个属性关系图
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


GMM预测结果与后两个属性关系图
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
6
7