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

E13 EM Algorithm

17341015 Hongzheng Chen

December 5, 2019

Contents

1 Chinese Football Dataset

The following Chinese Football Dataset has recored the performance of 16 AFC football teams between 2005 and 2018.

```
Country 2006WorldCup 2010WorldCup 2014WorldCup 2018WorldCup 2007AsianCup 2011AsianCup 2015AsianCup
China   50  50  50  40  9   9   5
Japan   28  9   29  15  4   1   5
South_Korea 17 15  27  19  3   3   2
Iran      25  40  28  18  5   5   5
Saudi_Arabia    28  40  50  26  2   9   9
Iraq      50  50  40  40  1   5   4
Qatar   50  40  40  40  9   5   9
United_Arab_Emirates    50  40  50  40  9   9   3
Uzbekistan  40  40  40  40  5   4   9
Thailand    50  50  50  40  9   17  17
Vietnam 50 50  50  50  5   17  17
Oman    50  50  40  50  9   17  9
Bahrain 40  40  50  50  9   9   9
North_Korea 40 32  50  50  17  9   9
Indonesia   50  50  50  50  9   17  17
Australia   16  21  30  30  9   2   1
```

The scoring rules are below:

- For the FIFA World Cup, teams score the same with their rankings if they enter the World Cup; teams score 50 for failing to entering the Asia Top Ten; teams score 40 for entering the Asia Top Ten but not entering the World Cup.
- For the AFC Asian Cup, teams score the same with their rankings if they finally enter the top four; teams score 5 for entering the top eight but not the top four, and 9 for entering the top sixteen but not top eight; teams score 17 for not passing the group stages.

We aim at classifying the above 16 teams into 3 classes according to their performance: the first-class, the second-class and the third-class. In our opinion, teams of Australia, Iran, South Korea and Japan belong to the first-class, while the Chinese football team belongs to the third-class.

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

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

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

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

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.

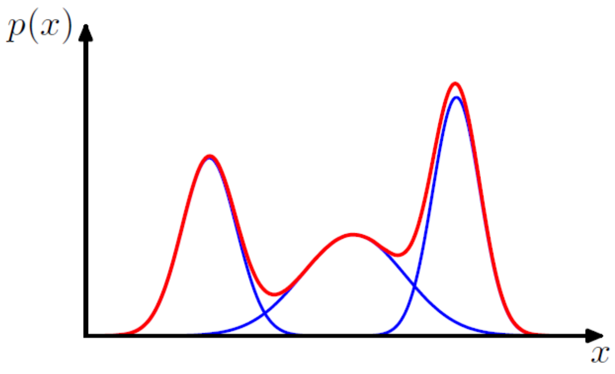


Figure 1: Example of a Gaussian mixture distribution

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 $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k) = p(\mathbf{x}|k)$ as the probability of $\mathbf{x}$ conditioned on k. From Bayes' theorem these are given by

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

The form of the Gaussian mixture distribution is governed by the parameters π, $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$, where we have used the notation $\boldsymbol{\pi} = \{\pi_1, ..., \pi_K\}$, $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k\}$ and $\mathbf{\Sigma} = \{\mathbf{\Sigma}_1, ..., \mathbf{\Sigma}_K\}$. 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

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

where $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$. 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

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

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

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

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

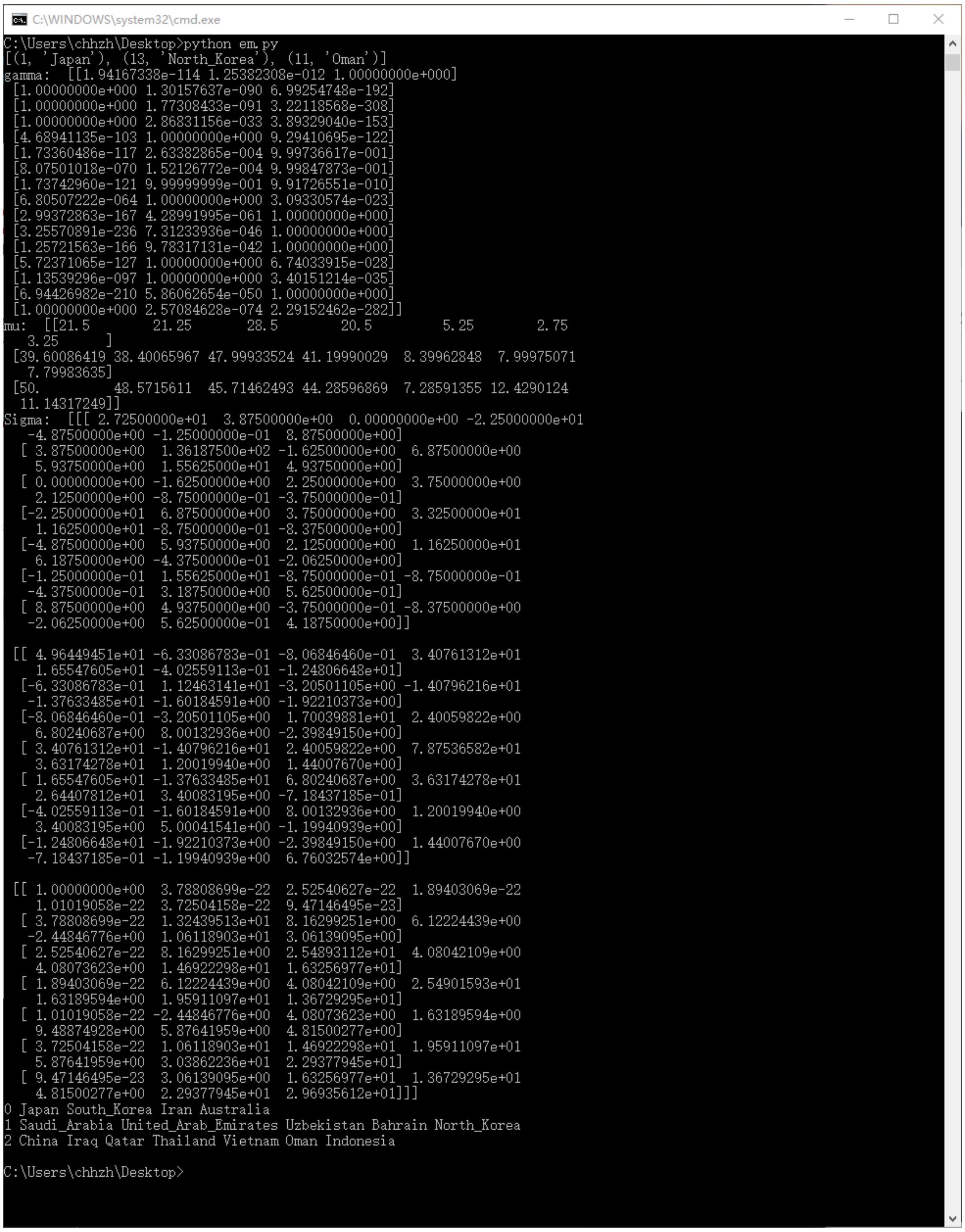
- Assume that score vectors of teams in the same class are normally distributed, we can thus adopt the Gaussian mixture model. Please classify the teams into 3 classes by using EM algorithm. If necessary, you can refer to page 430-439 in the book `Pattern Recognition and Machine Learning.pdf` and the 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 $\boldsymbol{\Sigma}$. If necessary, you can plot the clustering results. Note that γ is essential for classifying.
- Please submit a file named `E13_YourNumber.pdf` and send it to `ai_201901@foxmail.com`

4 Codes and Results

The classification results are listed below.

0	Japan South_Korea Iran Australia
1	Saudi_Arabia United_Arab_Emirates Uzbekistan Bahrain North_Korea
2	China Iraq Qatar Thailand Vietnam Oman Indonesia

And the $\boldsymbol{\gamma}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are shown in Fig. 2.



```
C:\Users\chhzh\Desktop>python em.py
[(1, 'Japan'), (13, 'North_Korea'), (11, 'Oman')]
gamma:  [[1.94167338e-114 1.25382308e-012 1.00000000e+000]
 [1.00000000e+000 1.30157637e-090 6.99254748e-192]
 [1.00000000e+000 1.77308433e-091 3.22118568e-308]
 [1.00000000e+000 2.86831156e-033 3.89329040e-153]
 [4.68941135e-103 1.00000000e+000 9.29410695e-122]
 [1.73360486e-117 2.63382865e-004 9.99736617e-001]
 [8.07501018e-070 1.52126772e-004 9.99847873e-001]
 [1.73742960e-121 9.99999999e-001 9.91726551e-010]
 [6.80507222e-064 1.00000000e+000 3.09330574e-023]
 [2.99372863e-167 4.28991995e-061 1.00000000e+000]
 [3.25570891e-236 7.31233936e-046 1.00000000e+000]
 [1.25721563e-166 9.78317131e-042 1.00000000e+000]
 [5.72371065e-127 1.00000000e+000 6.74033915e-028]
 [1.13539296e-097 1.00000000e+000 3.40151214e-035]
 [6.94426982e-210 5.86062654e-050 1.00000000e+000]
 [1.00000000e+000 2.57084628e-074 2.29152462e-282]]
mu:  [[21.5        21.25       28.5        20.5         5.25        2.75
   3.25      ]
 [39.60086419 38.40065967 47.99933524 41.19990029  8.39962848  7.99975071
   7.79983635]
 [50.         48.5715611  45.71462493 44.28596869  7.28591355 12.4290124
  11.14317249]]
Sigma:  [[[ 2.72500000e+01  3.87500000e+00  0.00000000e+00 -2.25000000e+01
   -4.87500000e+00 -1.25000000e-01  8.87500000e+00]
  [ 3.87500000e+00  1.36187500e+02 -1.62500000e+00  6.87500000e+00
    5.93750000e+00  1.55625000e+01  4.93750000e+00]
  [ 0.00000000e+00 -1.62500000e+00  2.25000000e+00  3.75000000e+00
    2.12500000e+00 -8.75000000e-01 -3.75000000e-01]
  [-2.25000000e+01  6.87500000e+00  3.75000000e+00  3.32500000e+01
    1.16250000e+01 -8.75000000e-01 -8.37500000e+00]
  [-4.87500000e+00  5.93750000e+00  2.12500000e+00  1.16250000e+01
    6.18750000e+00 -4.37500000e-01 -2.06250000e+00]
  [-1.25000000e-01  1.55625000e+01 -8.75000000e-01 -8.75000000e-01
   -4.37500000e-01  3.18750000e+00  5.62500000e-01]
  [ 8.87500000e+00  4.93750000e+00 -3.75000000e-01 -8.37500000e+00
   -2.06250000e+00  5.62500000e-01  4.18750000e+00]]

 [[ 4.96449451e+01 -6.33086783e-01 -8.06846460e-01  3.40761312e+01
    1.65547605e+01 -4.02559113e-01 -1.24806648e+01]
  [-6.33086783e-01  1.12463141e+01 -3.20501105e+00 -1.40796216e+01
   -1.37633485e+01 -1.60184591e+00 -1.92210373e+00]
  [-8.06846460e-01 -3.20501105e+00  1.70039881e+01  2.40059822e+00
    6.80240687e+00  8.00132936e+00 -2.39849150e+00]
  [ 3.40761312e+01 -1.40796216e+01  2.40059822e+00  7.87536582e+01
    3.63174278e+01  1.20019940e+00  1.44007670e+00]
  [ 1.65547605e+01 -1.37633485e+01  6.80240687e+00  3.63174278e+01
    2.64407812e+01  3.40083195e+00 -7.18437185e-01]
  [-4.02559113e-01 -1.60184591e+00  8.00132936e+00  1.20019940e+00
    3.40083195e+00  5.00041541e+00 -1.19940939e+00]
  [-1.24806648e+01 -1.92210373e+00 -2.39849150e+00  1.44007670e+00
   -7.18437185e-01 -1.19940939e+00  6.76032574e+00]]

 [[ 1.00000000e+00  3.78808699e-22  2.52540627e-22  1.89403069e-22
    1.01019058e-22  3.72504158e-22  9.47146495e-23]
  [ 3.78808699e-22  1.32439513e+01  8.16299251e+00  6.12224439e+00
   -2.44846776e+00  1.06118903e+01  3.06139095e+00]
  [ 2.52540627e-22  8.16299251e+00  2.54893112e+01  4.08042109e+00
    4.08073623e+00  1.46922298e+01  1.63256977e+01]
  [ 1.89403069e-22  6.12224439e+00  4.08042109e+00  2.54901593e+01
    1.63189594e+00  1.95911097e+01  1.36729295e+01]
  [ 1.01019058e-22 -2.44846776e+00  4.08073623e+00  1.63189594e+00
    9.48874928e+00  5.87641959e+00  4.81500277e+00]
  [ 3.72504158e-22  1.06118903e+01  1.46922298e+01  1.95911097e+01
    5.87641959e+00  3.03862236e+01  2.29377945e+01]
  [ 9.47146495e-23  3.06139095e+00  1.63256977e+01  1.36729295e+01
    4.81500277e+00  2.29377945e+01  2.96935612e+01]]]
0 Japan South_Korea Iran Australia
1 Saudi_Arabia United_Arab_Emirates Uzbekistan Bahrain North_Korea
2 China Iraq Qatar Thailand Vietnam Oman Indonesia

C:\Users\chhzh\Desktop>
```

Figure 2: Values of $\boldsymbol{\gamma}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$

The classification plot in 2D is shown in Fig. 3 (only the first two dimensions are selected).

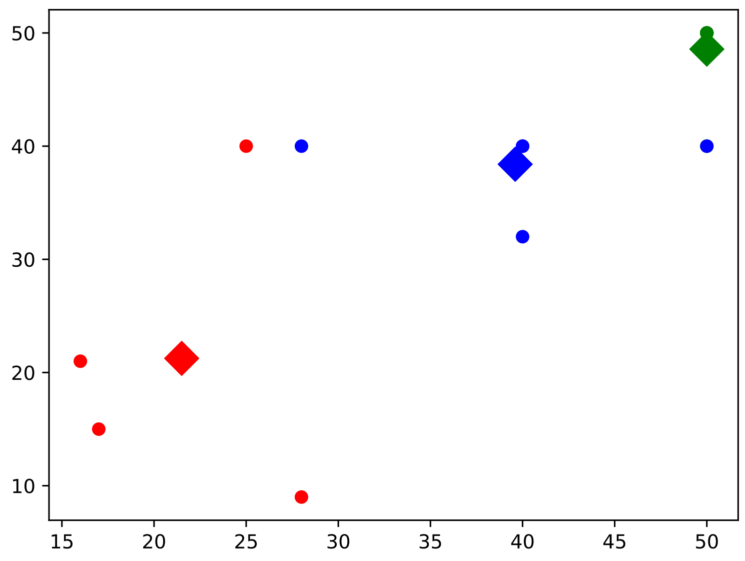


Figure 3: classification plot

The following code `em.py` is generated by jupyter notebook `em.ipynb`.

```
# coding: utf-8

# In[ ]:


import numpy as np
import pandas as pd
import sys

dataset = pd.read_csv("football.txt",sep=",")
dataset.head()


# In[ ]:


def prob(x, mu, sigma):
    """
    Calculate the Gaussian distribution

    N(x|\mu,\Sigma)=\frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}}\exp\{-\frac{1}{2}(x-\mu)^T}\Sigma
        ↪ ^{-1}(x-\mu)\}

    Notice the input x here should be a 1-d vector
    """
    if x.ndim != 1 or mu.ndim != 1:
        raise RuntimeError("Dimension error!")
    # print(x,mu,sigma)
    D = x.shape[0]
    expOn = - 1 / 2 * np.matmul(np.matmul((x - mu).T,np.linalg.inv(sigma)),x - mu)
    divBy = np.power(2 * np.pi, D / 2) * np.sqrt(np.linalg.det(sigma))
    return np.exp(expOn) / divBy

def EM(dataMat, n_components=3, maxIter=100):
    """
    Expectation-Maximization (EM) algorithm

    This implementation has been extended to support different number of components
    """
    n_samples, D = np.shape(dataMat) # n_samples=16 D=7

    # 1. Initialize Gaussian parameters
    pi_k = np.ones(n_components) / n_components # mixing coefficients
    # randomly select k(n_components) samples as the mean of each class
    # choices = np.random.choice(n_samples,n_components)
    # predefined mean of each class
    choices = [1,13,11]
    print([(i,dataset["Country"][i]) for i in choices])
    mu_k = np.array([dataMat[i,:] for i in choices]).reshape(n_components,D) # k * D
    # mu_k = [dataMat[5, :], dataMat[21, :], dataMat[26, :]]
    sigma_k = [np.eye(D) for x in range(n_components)] # k * D * D

    gamma_k = np.zeros((n_samples, n_components)) # n * k
    # Iterate for maxIter times
    for i in range(maxIter):
        """
        2. E step
        \gamma(z_{nk}) = \frac{\pi_k prob(x_n|\mu_k,\Sigma_k)}{sum_pi_mul}
        sum_pi_mul = \sum_{j=1}^K \pi_j prob(x_n|\mu_j,\Sigma_j)
        """
        for n in range(n_samples):
            sum_pi_mul = 0 # denominator
            for k in range(n_components):
                gamma_k[n, k] = pi_k[k] * prob(dataMat[n, :], mu_k[k], sigma_k[k])
                sum_pi_mul += gamma_k[n, k]
            # normalization
            for k in range(n_components):
                gamma_k[n, k] /= sum_pi_mul
        # summarize gamma_k along different samples
        N_k = np.sum(gamma_k, axis=0)

        """
        3. M step
        \mu_k^{new} = \frac{1}{N_k}\sum_{n=1}^N\gamma(z_{nk})x_n
        \Sigma_k^{new} = \frac{1}{N_k}\sum_{n=1}^N\gamma(z_{nk})(x_n-\mu_k^{new})(x_n-\mu_k
            ↪ ^{new})^T
        \pi_k^{new} = \frac{N_k}{N}

        N_k=\sum_{n=1}^N\gamma(z_{nk})
        """
        for k in range(n_components):
            # Calculate \mu_k
            mu_k[k] = np.zeros(D,dtype=np.float64)
            for n in range(n_samples):
                mu_k[k] += gamma_k[n, k] * dataMat[n, :]
            mu_k[k] /= N_k[k]
            # Calculate \Sigma_k
            sigma_k[k] = np.zeros((D, D),dtype=np.float64)
            for n in range(n_samples):
                sigma_k[k] += gamma_k[n, k] * np.matmul((dataMat[n, :] - mu_k[k]).reshape
                    ↪ (1,-1).T, (dataMat[n, :] - mu_k[k]).reshape(1,-1)) # be careful of
                    ↪ outer product!
            sigma_k[k] /= N_k[k]

            # Calculate new mixing coefficient
            pi_k[k] = N_k[k] / n_samples

        sigma_k += np.eye(D)

    print("gamma: ",gamma_k)
    print("mu: ",mu_k)
    print("Sigma: ",sigma_k)

    return gamma_k


# In[ ]:


def gaussianCluster(dataMat, n_components, max_iter):
    n_samples, D = np.shape(dataMat)
    centroids = np.zeros((n_components, D))
    gamma = EM(dataMat,n_components,max_iter)

    # get the cluster result
    clusterAssign = np.zeros((n_samples, 2))
    for n in range(n_samples):
        clusterAssign[n, :] = np.argmax(gamma[n, :]), np.amax(gamma[n, :])

    # calculate the final results
    for k in range(n_components):
        pointsInCluster = dataMat[np.nonzero(clusterAssign[:, 0] == k)[0]]
        centroids[k, :] = np.mean(pointsInCluster, axis=0)
    return centroids, clusterAssign[:,0]


# In[ ]:


from sklearn import mixture

def sklearn_em(data,n_components,max_iter):
    clst = mixture.GaussianMixture(n_components=n_components,max_iter=max_iter,
        ↪ covariance_type="full")
    clst.fit(data)
    predicted_labels = clst.predict(data)
    return clst.means_, predicted_labels


# In[ ]:
```

```
import matplotlib.pyplot as plt

def showCluster(dataset, k, centroids, clusterAssment):
    numSamples, dim = dataset.shape

    mark = ['or', 'ob', 'og', 'ok', '^r', '+r', 'sr', 'dr', '<r', 'pr']
    if k > len(mark):
        print("Sorry! Your k is too large!")
        return 1

    # draw all samples
    for i in range(numSamples):
        markIndex = int(clusterAssment[i])
        plt.plot(dataset[i, 0], dataset[i, 1], mark[markIndex])

    mark = ['Dr', 'Db', 'Dg', 'Dk', '^b', '+b', 'sb', 'db', '<b', 'pb']
    # draw the centroids
    for i in range(k):
        plt.plot(centroids[i, 0], centroids[i, 1], mark[i], markersize=12)

    plt.show()


# In[ ]:


n_components = 3
data = dataset[dataset.columns.values[dataset.columns.values != "Country"]].to_numpy()
# centroids, labels = sklearn_em(data,n_components,100)
centroids, labels = gaussianCluster(data.astype(np.float64),n_components,100)
showCluster(data, n_components, centroids, labels)
res = {0:[],1:[],2:[]}
for i,label in enumerate(labels):
    res[label].append(dataset["Country"][i])
for key in res:
    print(key,*res[key])
```