生物统计第一次编程作业——EM算法(混合二项与混合高斯)

统计81 于越 *

April 13, 2021

1 EM算法简介

本次作业主要是关于生物统计课程中所讲解的EM算法，这是生物统计中和机器学习算法相关的部分。在 EM算法中，我们希望通过学习机对于数据集的学习，可以尽可能地获取数据集背后的信息。在现实生活中，虽然很多值是可以被观测和被学习到的，但是还是有很多不完整的样本数据集存在于现实生活中，这些量就是我们常说的未知量。尽管我们确实不知道这些信息，但是我们渴望知道这些信息，因此我们就尝试着去对于这些“隐藏的”不知道的变量进行估计。这就是EM算法，通过已知信息，训练数据集的学习，去尽可能挖掘关于隐变量的信息，通过对于已知数据集的不停迭代，进而对于隐变量参数进行合适的估计与逼近。在本次作业中，我主要选取了两个主要的例子，去体现EM算法中的混合二项分布与混合高斯分布，并且对于这样的EM算法给出了自身的理解。

2 混合二项分布

2.1 问题描述

有两枚硬币，我们在不知道选择了哪一枚硬币的情况下进行抛硬币的实验，进行k组实验，每组实验抛掷n次硬币。根据已有的抛硬币实验结果，分布计算两枚硬币抛掷正面朝上的概率与选择其中一枚硬币的概率。

*学号：2183210516。关于文件夹中代码说明可参见附录

2.2 符号表

Table 1: 符号介绍

符号	相关说明
X_i	实验中硬币正面向上次数
Z	隐变量，此处为选择的硬币
π_A	选择硬币A的概率
π_B	选择硬币A的概率
n	单组实验实验次数(矩阵的列数)
k	所做的实验组数(选了几次硬币，矩阵的行数)
θ_A	硬币A正面朝上概率
θ_B	硬币B正面朝上概率
$\theta^{(old)}$	硬币未更新的老参数
$\theta^{(new)}$	硬币更新后的新参数
$\Omega(\theta, \theta^{(new)})$	目标期望值

2.3 过程推导

混合二项分布的一个典型例子是抛硬币，对于所抛的硬币进行合适的参数估计。在这里，我们假定有两枚硬币A与B，其中A正面朝上的概率用θ_A表示，B正面朝上的概率用θ_B表示。抛硬币之前我们需要对硬币进行适当地选择，用π_A表示选到硬币A的概率，用π_B表示选到硬币B的概率。但是事实上，我们很可能不知道我们所选择的是哪一枚硬币，因此我们想要尝试对着所选硬币的概率进行合适的估计。这样，在这个例子中的参数空间有4个参数:

$$\theta = \{\theta_A, \theta_B, \pi_A, \pi_B\} \tag{1}$$

我们所要做的，就是在仅仅做实验的情况下，对于实际的这个参数空间进行估计。这样我们需要通过多次反复地利用实验数据，进行迭代逼近最后的结果。

假设两枚硬币为A与B，随机变量X表示实验中硬币正面朝上次数，即(x_i).再设隐随机变量为Z，表示所选择的硬币的概率(我们事先不知道选择的是哪一枚硬币)。用n表示选择一枚硬币之后的抛掷硬币次数，用k表示选择硬币的次数(做多少组实验),这样，我们就可以得到一个$k * n$维的矩阵数据集。

令

$$P(Z = A) = \pi_A, P(Z = B) = \pi_B \tag{2}$$

用以表示先验。

与此同时，可以获得当前的似然，也就是假定选择某枚硬币之后的正面朝上次数的

概率，满足一个二项分布

$$P(X=x_i|Z=A;\theta_A)=C_n^{x_i}\theta_A^{x_i}(1-\theta_A)^{n-x_i} \tag{3}$$

$$P(X=x_i|Z=B;\theta_B)=C_n^{x_i}\theta_B^{x_i}(1-\theta_B)^{n-x_i} \tag{4}$$

而对于Z，该隐变量显然服从一个伯努利分布，即

$$P(Z=j)=\pi_j^{z_j}(1-\pi_j)^{1-z_j},\qquad j=A,B\qquad z_j=0,1 \tag{5}$$

这样，可以获得一个对数的极大似然:

$$\begin{aligned} l(\theta)&=lnP(X|\theta)=ln\Pi_{i=1}^{k}P(X=x_i|\theta)=\sum_{i=1}^{k}lnP(X=x_i|\theta)\\ &=\sum_{i=1}^{k}ln\sum_{j\in\{A,B\}}P(X=x_i,Z=j|\theta)=ln\sum_{z}P(X,Z=j|\theta)\end{aligned} \tag{6}$$

E步：

$$\begin{aligned}\Omega(\theta,\theta^{old})&=E_{Z|X,\theta^{old}}lnP(X,Z|\theta)=\sum_{j}P(Z|X,\theta^{old})lnP(X,Z|\theta)\\ &=\sum_{j\in\{A,B\}}\sum_{i}P(Z=j|X=x_i,\theta^{old})lnP(X=x_i,Z=j|\theta)=\\ &\qquad\sum_{i=1}^{k}P(Z=A|X=x_i,\theta^{old})lnP(X=x_i,Z=A|\theta)\\ &\qquad+\sum_{i=1}^{k}P(Z=B|X=x_i,\theta^{old})lnP(X=x_i,Z=B|\theta)\end{aligned} \tag{7}$$

要求得上式的E步似然函数，我们需要求一个后验分布$P(Z=j|X=x_i;\theta_j)$，这需要用到贝叶斯公式。

我们先求联合分布：

$$P(X=x_i,Z=A;\theta_A)=C_n^{x_i}\theta_A^{x_i}(1-\theta_A)^{n-x_i} \tag{8}$$

$$P(X=x_i,Z=B;\theta_B)=C_n^{x_i}\theta_B^{x_i}(1-\theta_B)^{n-x_i} \tag{9}$$

故下面使用贝叶斯公式求后验分布：

$$\begin{aligned}P(Z=A|X=x_i;\theta_A)&=\frac{P(X=x_i,Z=A;\theta_A)}{P(X=x_i|\theta)}\\ &=\frac{\pi_A C_n^{x_i}\theta_A^{x_i}(1-\theta_A)^{n-x_i}}{\pi_A C_n^{x_i}\theta_A^{x_i}(1-\theta_A)^{n-x_i}+\pi_B C_n^{x_i}\theta_B^{x_i}(1-\theta_B)^{n-x_i}}\end{aligned} \tag{10}$$

同理，

$$\begin{aligned}P(Z=B|X=x_i;\theta_B)&=\frac{P(X=x_i,Z=B;\theta_B)}{P(X=x_i|\theta)}\\ &=\frac{\pi_B C_n^{x_i}\theta_B^{x_i}(1-\theta_B)^{n-x_i}}{\pi_A C_n^{x_i}\theta_A^{x_i}(1-\theta_A)^{n-x_i}+\pi_B C_n^{x_i}\theta_B^{x_i}(1-\theta_B)^{n-x_i}}\end{aligned} \tag{11}$$

为便于推导，我们记

$$\gamma_i(A,\theta_A^{old}) = P(Z=A|X=x_i,\theta_A^{old}) \tag{12}$$

$$\gamma_i(B,\theta_B^{old}) = P(Z=B|X=x_i,\theta_B^{old}) \tag{13}$$

因为$\pi_A+\pi_B=1$，我们记$\pi_A=\pi$.

故，我们有

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{k} lnP(X=x_i|\theta) = \sum_{i=1}^{k} P(Z=j)lnP(X=x_{i|Z=j;\theta_j}) \\ &= \sum_{i=1}^{k} ln[C_n^{x_i}\theta_j^{x_i}(1-\theta_j)^{n-x_i}\pi^{z_j}(1-\pi)^{1-z_j}] \\ &= \sum_{i=1}^{k}[x_i ln\theta_j + (n-x_i)ln(1-\theta_j) + z_j ln\pi + (1-z_j)ln(1-\pi) + C] \end{aligned} \tag{14}$$

注意到

$$\gamma_i(A,\theta^{old}) + \gamma_i(B,\theta^{old}) = 1 \tag{15}$$

最后可得

$$\begin{aligned} \Omega(\theta,\theta^{old}) &= E_{Z|X,\theta^{old}} lnP(X,Z|\theta) = \sum_j P(Z|X,\theta^{old})lnP(X,Z|\theta) \\ &= \sum_{i=1}^{k}\gamma_i(A,\theta^{old})[x_i ln\theta_A + (n-x_i)ln(1-\theta_A) + ln\pi] \\ &+ \sum_{i=1}^{k}\gamma_i(B,\theta^{old})[x_i ln\theta_B + (n-x_i)ln(1-\theta_B) + ln(1-\pi)] + C_0 \\ &= \sum_{i=1}^{k} x_i[\gamma_i(A,\theta^{old})ln\theta_A + \gamma_i(B,\theta^{old})ln\theta_B] \\ &+ \sum_{i=1}^{k}(n-x_i)[\gamma_i(A,\theta^{old})ln(1-\theta_A) + \gamma_i(B,\theta^{old})ln(1-\theta_B)] \\ &+ \sum_{i=1}^{k}\gamma_i(A,\theta^{old})ln\pi + \sum_{i=1}^{k}\gamma_i(B,\theta^{old})ln(1-\pi) + C_0 \end{aligned} \tag{16}$$

M步：最后，我们对

$$\theta = \{\theta_A,\theta_B,\pi_A,\pi_B\} \tag{17}$$

求导，并且令求导结果均为0，解方程，可以得出最后结果为

$$\theta_A^{new} = \frac{\sum_{i=1}^{k} x_i\gamma_i(A,\theta^{old})}{n\sum_{i=1}^{k}\gamma_i(A,\theta^{old})} \tag{18}$$

$$\theta_B^{new} = \frac{\sum_{i=1}^{k} x_i\gamma_i(B,\theta^{old})}{n\sum_{i=1}^{k}\gamma_i(B,\theta^{old})} = \frac{\sum_{i=1}^{k} x_i[1-\gamma_i(A,\theta^{old})]}{n\sum_{i=1}^{k}[1-\gamma_i(A,\theta^{old})]} \tag{19}$$

$$\pi^{new} = \frac{\sum_{i=1}^{k} \gamma_i(A, \theta^{old})}{k} \tag{20}$$

每一步求出新的参数后，判断算法是否收敛。若收敛，则终止，不然继续迭代知道算法终止即可获得参数空间的估计。

2.4 代码实例

我们依旧利用之前的抛硬币的例子进行实验。

2.4.1 实验一:1000组实验，每组实验投币1000次

我们假定实际的参数空间取值如下：

$$\theta_A = 0.7, \theta_B = 0.45, \pi_A = 0.6, \pi_B = 0.4 \tag{21}$$

实验结果如下：

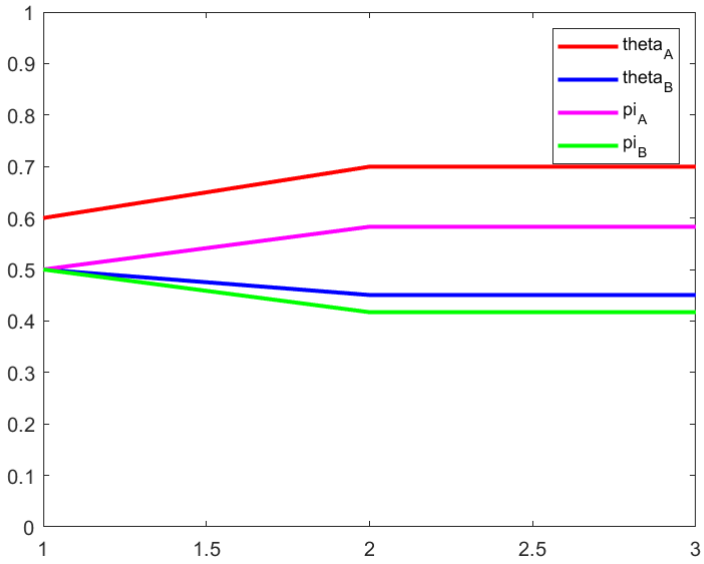


Figure 1: 实验一结果

迭代得出的参数空间结果为：

$$\theta_A = 0.6996, \theta_B = 0.4504, \pi_A = 0.5830, \pi_B = 0.4170 \tag{22}$$

可以看到迭代效果还是不错的。

接下来我们可以试着考虑一下参数选择大小接近的情况。

2.4.2 实验二：1000组实验，每组投币1000次，考虑隐变量参数大小接近情况

我们在实验二中设定的实际参数空间取值如下：

$$\theta_A = 0.8, \theta_B = 0.6, \pi_A = 0.50001, \pi_B = 0.49999 \tag{23}$$

实验结果如下：

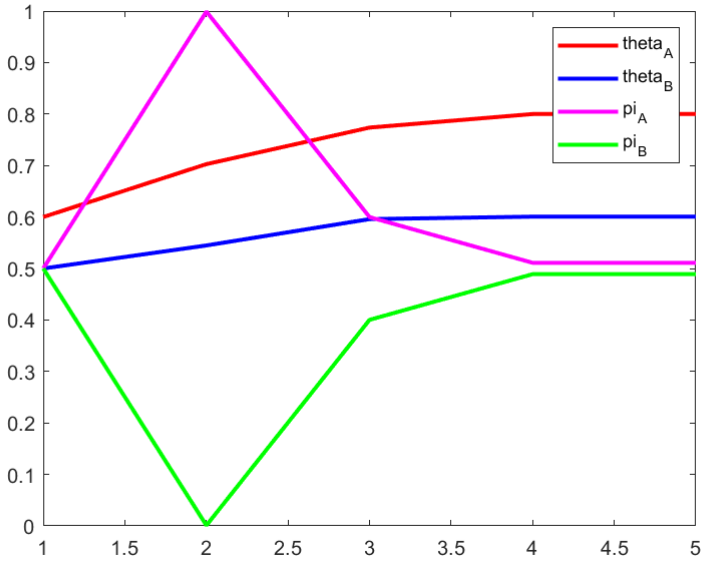


Figure 2: 实验二结果

迭代得出的参数空间结果为：

$$\theta_A = 0.8, \theta_B = 0.6, \pi_A = 0.5110, \pi_B = 0.4890 \tag{24}$$

迭代效果也是不错的。

接下来我们可以试着考虑一下比较极端的情况。

2.4.3 实验三：1000组实验，每组投币1000次，考虑极端情况

我们在实验三中设定的实际参数空间取值如下：

$$\theta_A = 0.8, \theta_B = 0.6, \pi_A = 1, \pi_B = 0 \tag{25}$$

实验结果如下：

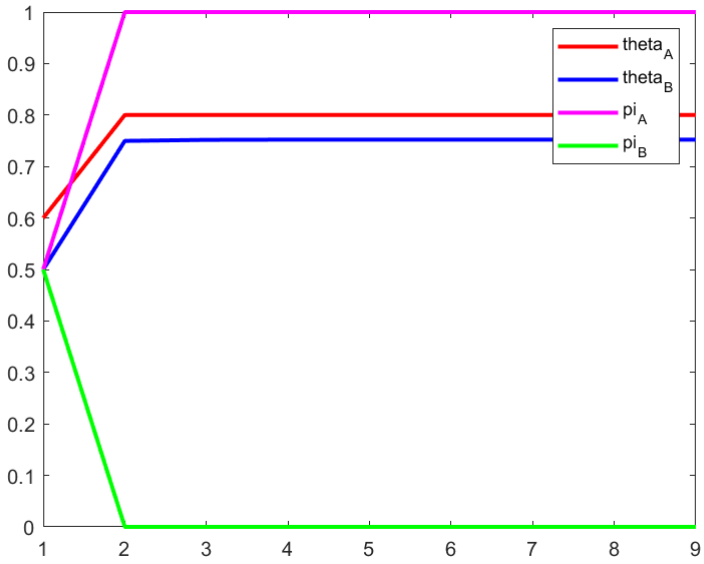


Figure 3: 实验三结果

迭代得出的参数空间结果为：

$$\theta_A = 0.8, \theta_B = 0.7523, \pi_A = 1, \pi_B = 0 \tag{26}$$

发现此时除了参数θ_B，其他的参数迭代效果都很好。这和我们设定本身就不会选择硬币B这个事实有关。因为我们为选择硬币B，学习机得不到关于硬币B的有效信息，因此关于参数θ_B，得到的结果也相当随机。

最后我们将数据矩阵缩小，观察结果。

2.4.4 实验四:10组实验，每组实验投币10次

我们假定实际的参数空间取值如下：

$$\theta_A = 0.7, \theta_B = 0.45, \pi_A = 0.6, \pi_B = 0.4 \tag{27}$$

实验结果如下：

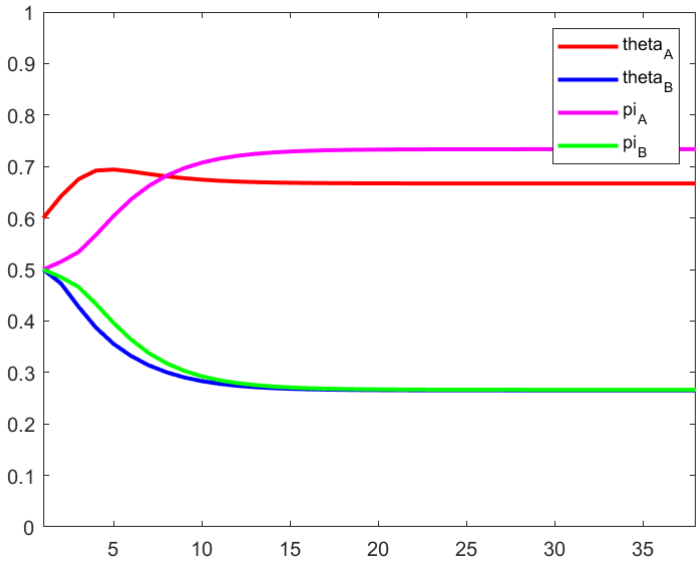


Figure 4: 实验四结果

迭代得出的参数空间结果为：

$$\theta_A = 0.6771, \theta_B = 0.2652, \pi_A = 0.7336, \pi_B = 0.2664 \tag{28}$$

显然，我们迭代参数估计的结果远没有实验一好了。这是容易理解的，因为学习机能学到的数据少了，因此掌握的信息也少，数据准确度也不如之前。

3 EM算法——混合高斯分布用以分类

3.1 问题重述

写一个EM算法，利用高斯混合分布，自行生成数据进行分类，并且运用到一个真实的数据集上。同时探讨NMI值的作用(判断分类效果好坏)。

3.2 符号表

Table 2: 符号介绍

符号	相关说明
z_{nk}	实验中数据点
Z	隐变量，此处为选择哪一个具体高斯分布
π_k	数据点选择第k个高斯分布的概率
N	实验中总数据点数
K	实验中的分类类数(高斯分布个数)
μ_k	第k个高斯分布的均值
Σ_k	第k个高斯分布的方差
$\theta^{(old)}$	EM算法未更新的老参数
$\theta^{(new)}$	EM算法更新后的新参数
$\Omega(\theta,\theta^{(new)})$	目标期望值

3.3 过程推导

类比于混合二项分布，事实上混合高斯分布的隐变量也是做某种选择，只是这里的选择是选择一个聚类。在给定一个数据集服从于某一个混合高斯分布时，我们不知道这个数据集中的点具体属于哪一个高斯分布。因此，我们采用EM算法，同样通过不断地迭代，可以具体地得出隐变量Z，也就是数据点具体属于哪一个高斯分布的结果。

首先我们有观察数据集

$$X=\{x_1,x_2,\cdots,x_N\} \tag{29}$$

观察量是独立于混合分布

$$p(x)=\sum_{i=1}^{k}\pi_k N(x|\mu_k,\Sigma_k) \tag{30}$$

其中参数空间

$$\theta=\{\pi_k,\mu_k,\Sigma_k\} \tag{31}$$

为模型参数。

考虑对数据集X的极大似然问题。

对数似然函数

$$lnP(X|\theta)=\sum_{i=1}^{N}ln(\sum_{i=1}^{k}\pi_k N(x_n|\mu_k,\Sigma_k)) \tag{32}$$

而$z_n=[z_{n1},z_{n2},\cdots,z_{nk}]$用来表示隐变量隐含$x_n$的信息。

若x_n来自第k个成分分布，则有$z_{nk}=1$，否则$z_{nk}=0$.因此有$\sum_k z_{nk}=1$.

我们用$\pi_k=P(z_{nk}=1)$表示数据点属于第k个聚类的概率。自然有$0\leq\pi_k\leq 1$，且$\sum_k z_{nk}=1$.π_k即为每个分量$N(X|\mu_k,\Sigma_k)$的权重。

那么，显然π_k服从一个多项分布，即

$$p(z_n) = \Pi_{k=1}^{k} \pi_k^{z_{nk}} \tag{33}$$

x在给定z取特定值情况下的条件分布为

$$p(x_n|z_{nk} = 1) = N(x_n|\mu_k, \Sigma_k) \tag{34}$$

则可得

$$p(x_n|z_n) = \Pi_{k=1}^{K} N(x_n|\mu_k, \Sigma_k)^{z_{nk}} = \pi_k N(x_n|\mu_k, \Sigma_k) \tag{35}$$

则x_n, z_n的联合分布为

$$\begin{aligned} p(x_n, z_n) &= p(x_n|z_n)p(z_n) = p(x_n|z_n)p(z_n|\pi) \\ &= \Pi_{k=1}^{K} (\pi_k N(x_n|\mu_k, \Sigma_k))^{z_{nk}} \end{aligned} \tag{36}$$

故x_n边缘分布为

$$p(x_n) = \sum_{z_n} p(x_n, z_n) = \sum_{z_n} (\Pi_{k=1}^{K} (\pi_k N(x_n|\mu_k, \Sigma_k))^{z_{nk}}) = \sum_{k} \pi_k N(x_n|\mu_k, \Sigma_k) \tag{37}$$

对n个独立分布数据点，完全数据似然函数为

$$p(X, Z|\therefore) = \Pi_{n=1}^{N} \Pi_{k=1}^{K} (\pi_k N(x_n|\mu_k, \Sigma_k))^{z_{nk}} \tag{38}$$

则完全数据的对数似然函数为

$$lnP(X, Z|\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} [ln\pi_k + lnN(x_n|\mu_k, \Sigma_k)] \tag{39}$$

则E步：要通过计算似然关于Z在原参数下后验分布的期望使得完整数据都得以应用。

则

$$\begin{aligned} E_{Z|X,\theta^{old}}[lnP(X, Z|\theta)] &= \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} P(z_{nk}|X, \theta^{old}) [ln\pi_k + lnN(x_n|\mu_k, \Sigma_k)] \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} E[z_{nk}|X, \theta^{old}] [ln\pi_k + lnN(x_n|\mu_k, \Sigma_k)] \end{aligned} \tag{40}$$

而

$$\begin{aligned} E[z_{nk}|X, \theta^{old}] &= P(z_{nk} = 1|x_{nk}, \theta^{old}) = \frac{P(z_{nk} = 1, x_{nk}|\theta^{old})}{P(x_{nk}|\theta^{old})} \\ &= \frac{P(x_{nk}|z_{nk} = 1, \mu_k, \Sigma_k) P(z_{nk} = 1|\pi_k)}{\sum_{j=1}^{K} P(z_{nj} = 1, x_{nk}|\pi_k, \mu_k, \Sigma_k)} = \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_k N(x_n|\mu_j, \Sigma_j)} \end{aligned} \tag{41}$$

为了便于推导，我们令$E[z_{nk}|X, \theta] = \gamma(z_{nk})$.

M步：

高维正态分布的表达式为

$$f_x(x_1,x_2,\cdots,x_k)=\frac{1}{\sqrt{(2\pi)^k|\Sigma|}}exp(-\frac{1}{2}(x-\mu)^{\mathrm{T}}\Sigma^{-1}(x-\mu)) \tag{42}$$

M步的目的是要使$E_{Z|X,\theta^{old}}[lnP(X,Z|\theta)]$最大化。

求新一步迭代的π_k可以采用lagrange乘子法。注意到有

$$\begin{aligned}\sum_{n=1}^{N}\sum_{k=1}^{K}E[z_{nk}|X,\theta^{old}]=\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})=\sum_{n=1}^{N}\sum_{k=1}^{K}\frac{\pi_k N(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j N(x_n|\mu_j,\Sigma_j)}\\ =\sum_{n=1}^{N}\frac{\sum_{k=1}^{K}\pi_k N(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j\pi_j N(x_n|\mu_j,\Sigma_j)}=\sum_{n=1}^{N}1=N\end{aligned} \tag{43}$$

利用lagrange乘子法，引入lagrange乘子λ，对π_k求导，可得

$$\frac{\sum_{n=1}^{N}\gamma(z_{nk})}{\pi_k}+\lambda=0 \tag{44}$$

则

$$\sum_{n=1}^{N}\gamma(z_{nk})+\lambda\pi_k=0,\sum_{k}(\sum_{n=1}^{N}\gamma(z_{nk})+\lambda\pi_k)=0 \tag{45}$$

而

$$\sum_{n}\sum_{k}\gamma(z_{nk})=N,\sum_{k}\pi_k=1 \tag{46}$$

故$\lambda=-N$.

可求得新一步迭代的π_k值为

$$\pi_k=\frac{\sum_{n=1}^{N}\gamma(z_{nk})}{N} \tag{47}$$

也可求得新一步迭代的期望和方差为

$$\mu_k=\frac{\sum_{n=1}^{N}\gamma(z_{nk})x_n}{\sum_{n=1}^{N}\gamma(z_{nk})} \tag{48}$$

$$\Sigma_k=\frac{\sum_{n=1}^{N}\gamma(z_{nk})(x_n-\mu)(x_n-\mu)^{\mathrm{T}}}{\sum_{n=1}^{N}\gamma(z_{nk})} \tag{49}$$

每一步求出新的参数后，判断算法是否收敛。若收敛，则终止，不然继续迭代知道算法终止即可获得参数空间的估计。也可以规定一定的迭代次数，确保分类完成后再结束迭代。

3.4 关于NMI值的推导

NMI值可以用来判定分类结果的好坏。要计算NMI值，我们需要两个结果，一个是已经分完类的各个点属于哪一分类的结果，另一个是真实的点属于哪一类的结果。这两个结果所表示向量我们用X和Y来表示。

首先，要计算NMI值，我们需要计算MI值。MI定义如下：

$$MI(X,Y)=\sum_{i=1}^{|X|}\sum_{j=1}^{|Y|}P(i,j)(\frac{P(i,j)}{P(i)P(j)}) \tag{50}$$

其中

$$P(i,j)=\frac{|X_i\cap Y_j}{N},P(i)=\frac{X_i}{N},P(j)=\frac{Y_j}{N} \tag{51}$$

再求X和Y的熵:

$$H(X)=-\sum_{i=1}^{|X|}P(i)log(P(i));H(Y)=-\sum_{i=1}^{|Y|}P(i)log(P(i)) \tag{52}$$

最后再求NMI：

$$NMI(X,Y)=\frac{2MI(X,Y)}{H(X)+H(Y)} \tag{53}$$

对于NMI值，我们可以理解为数据点集中的点分对类的概率。NMI值越接近于1，说明分类效果越好。

但是对于分类而言，我们很难去给点集中的点定义一个顺序。而要进行NMI的计算，势必要考虑进去点的相关顺序，因此我们这里默认不考虑顺序，将分类在第一类中的点默认顺序靠前，因此生成的结果向量最先为属于第一类的点，其次为属于第二类的点，最后为属于第三类的点，再计算NMI值，这样可以忽略点的顺序的影响进行分类结果好坏的判定。

我们依照上面不考虑顺序的点集进行NMI值的计算，判定当NMI值前后几乎不发生变动时则EM算法收敛，用以判断聚类的结束。

3.5 代码实例

我们考虑属于一个混合高斯分布的点，然后用EM算法不断迭代，直到可以确定数据点属于哪一个具体的高斯分布，即完成分类。一般而言，我们设置的参数初值如下：初始的π值均为$\pi=[0.3,0.5,0.2]$,初始的三个均值的横坐标为所有点横坐标最大值的3/4加上所有点横坐标最小值的1/4和所有点最大值的1/4加上所有点最小值的3/4以及所有点横坐标最大值的1/2加上所有点横坐标最小值的1/2.纵坐标均为所有点纵坐标最大值的1/2加上所有点横坐标最小值的1/2.初始协方差矩阵均设置为$[1,0;0,1]$.

3.5.1 实验一：5000个点进行分类，且数据点方差(分散度)不大

我们设置初始的参数π的向量为

$$\pi = [0.3, 0.5, 0.2] \tag{54}$$

随机生成5000个属于权重如上的某个混合高斯分布，分布点如下：

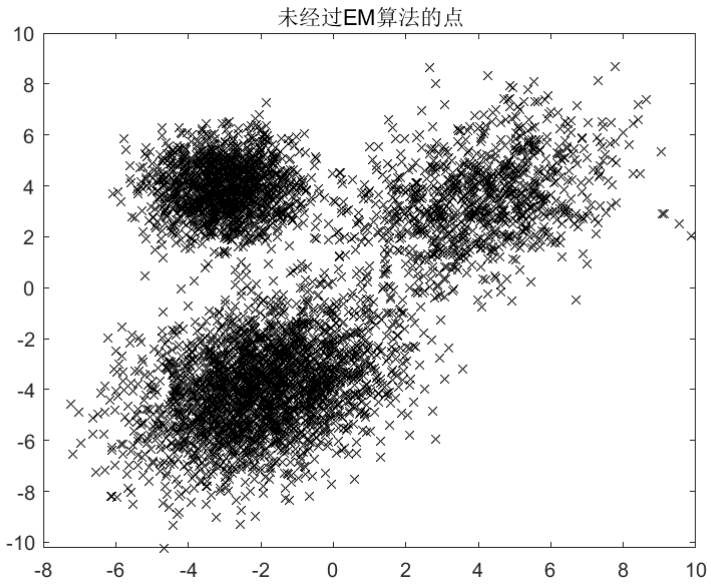


Figure 5: 实验一生成随机点

我们需要画出这些点具体所属的高斯分布与后续产生的结果进行比较。具体的高斯分布数据点如下：

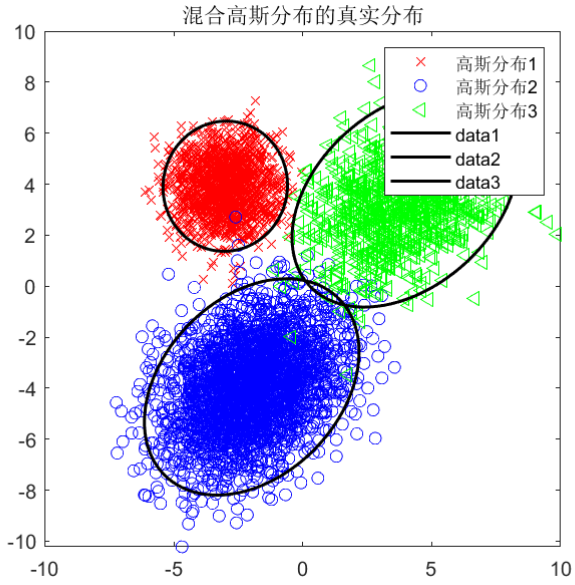


Figure 6: 实验一生成随机点真实值

经过100次的更新迭代后，我们计算出了新的参数向量π.我们对于每一个数据点的π，最大的那个分量取1，其余均为0，表示数据点具体属于哪一个高斯分布。

100次迭代后得到的数据点分类如下：

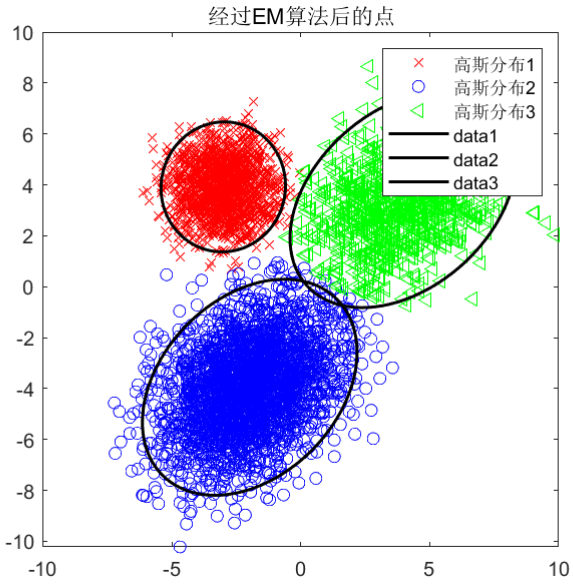
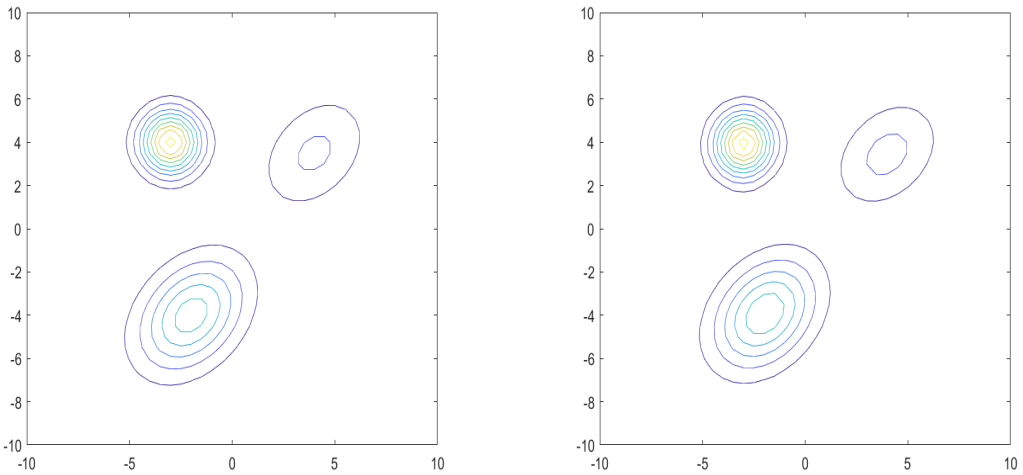


Figure 7: 实验一结果

二者对比一下感觉几乎一样。我们可以做出前后的高斯分布的等值线进行对比：



(a) 迭代前 (b) 迭代后

Figure 8: 迭代EM算法前后分布等高线对比

发现前后等高线位置和密集度近乎一样，说明分类效果很好。

我们给出了前25次迭代所得的NMI值，结果如下：

1	0
2	0.8947
3	0.9200
4	0.9622
5	0.9025
6	0.8635
7	0.8209
8	0.7887
9	0.7809
10	0.7806
11	0.8072
12	0.8652
13	0.9285
14	0.9743
15	0.9839
16	0.9832
17	0.9832
18	0.9832
19	0.9823
20	0.9823
21	0.9823
22	0.9823
23	0.9823
24	0.9823
25	0.9823

Figure 9: 实验一迭代NMI值

可以发现当迭代到约20次左右，NMI值稳定在0.9823附近，说明此时算法已收敛，且NMI值接近于1，说明分类结果很好。

3.5.2 实验二：500个点进行分类，且数据点方差(分散度)适中

我们设置初始的参数π的向量不变，仍为

$$\pi = [0.3, 0.5, 0.2] \tag{55}$$

随机生成500个属于权重如上的某个混合高斯分布，分布点如下：

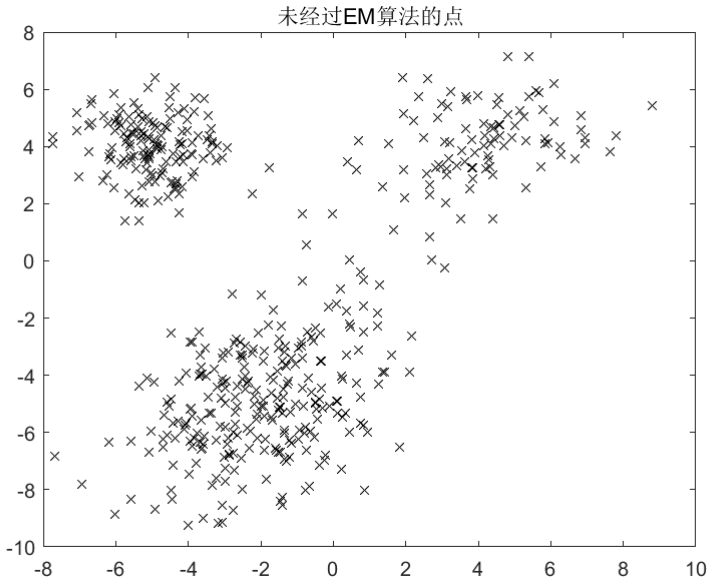


Figure 10: 实验二生成随机点

我们需要画出这些点具体所属的高斯分布与后续产生的结果进行比较。具体的高斯分布数据点如下：

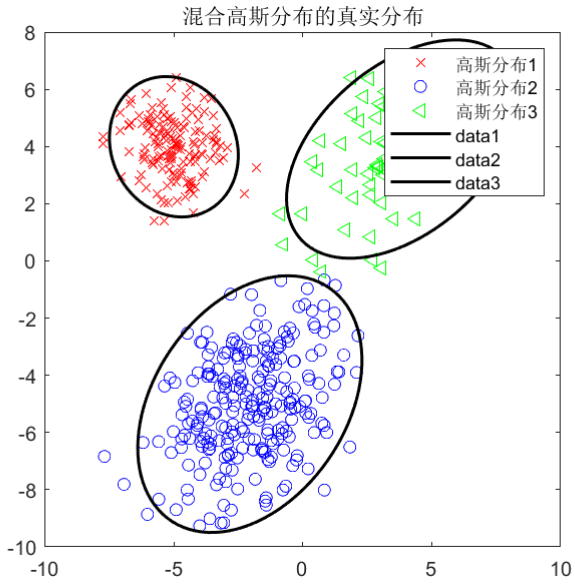


Figure 11: 实验二生成随机点真实值

经过100次的更新迭代后，我们计算出了新的参数向量π.仍旧用之前的方法确定属于哪一个聚类。

100次迭代后得到的数据点分类如下：

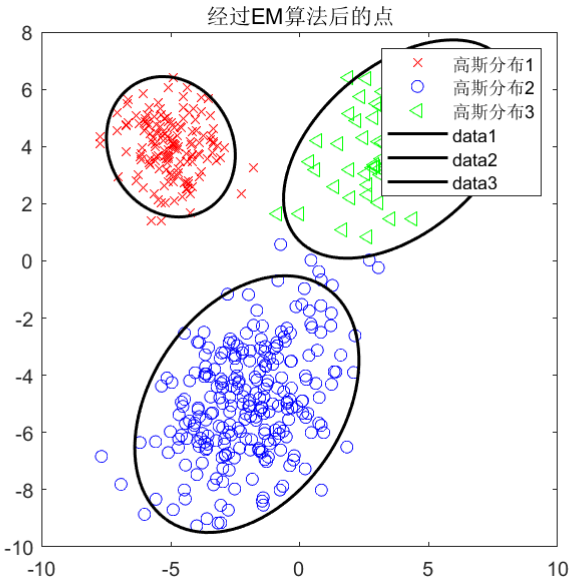
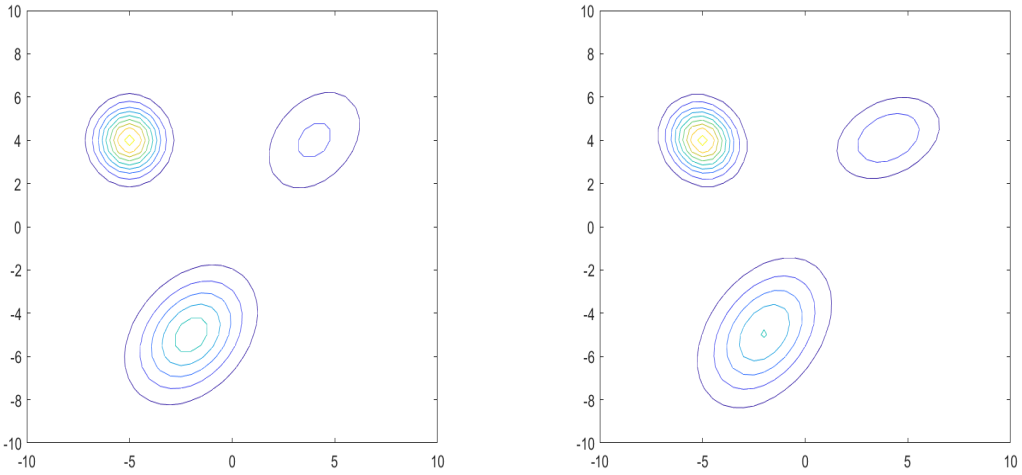


Figure 12: 实验二结果

二者对比一下感觉几乎一样。我们可以做出前后的高斯分布的等值线进行对比：



(a) 迭代前 (b) 迭代后

Figure 13: 迭代EM算法前后分布等高线对比

发现前后的位置是几乎一样的，但是分布等高线密集程度发生了一些改变，但变化不大。这个结果是完全可以的，这和数据点的减少有一定的关系。

我们给出了前25次迭代所得的NMI值，结果如下：

1	0
2	0.8257
3	0.8371
4	0.8983
5	0.9442
6	0.9670
7	0.9760
8	0.9760
9	0.9760
10	0.9760
11	0.9760
12	0.9760
13	0.9760
14	0.9760
15	0.9760
16	0.9760
17	0.9760
18	0.9760
19	0.9760
20	0.9760
21	0.9760
22	0.9760
23	0.9760
24	0.9760
25	0.9760

Figure 14: 实验二迭代NMI值

可以发现当迭代到约第8次左右，NMI值稳定在0.9760附近，说明此时算法已收敛，且NMI值接近于1，说明分类结果很好。

我们再一次减少数据点，观察是否仍旧有着这样的变化。

3.5.3 实验三：50个点进行分类，且数据点方差(分散度)较大

我们设置初始的参数π的向量不变，仍为

$$\pi = [0.3, 0.5, 0.2] \tag{56}$$

随机生成50个属于权重如上的某个混合高斯分布，分布点如下：

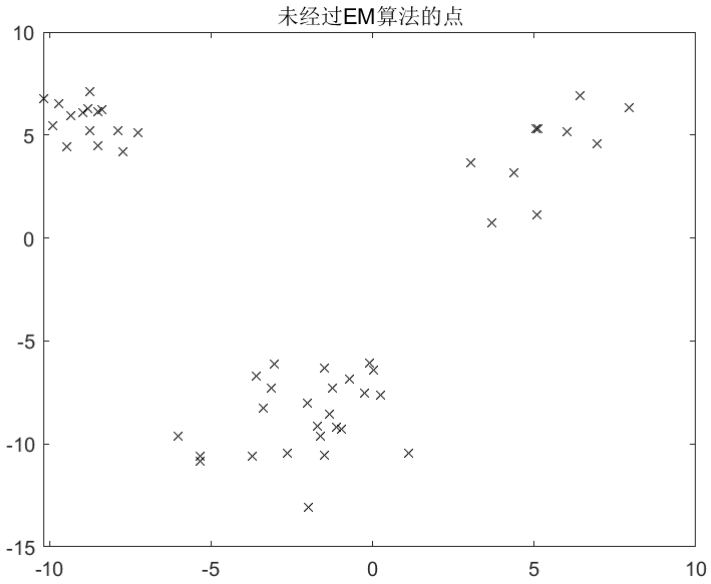


Figure 15: 实验三生成随机点

我们需要画出这些点具体所属的高斯分布与后续产生的结果进行比较。具体的高斯分布数据点如下：

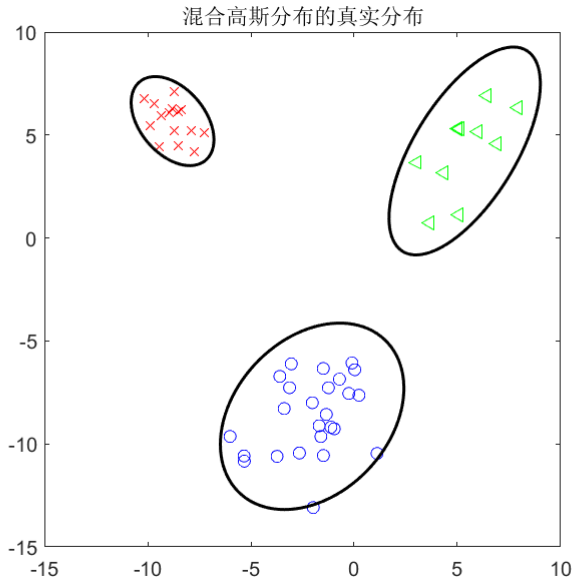


Figure 16: 实验三生成随机点真实值

经过100次的更新迭代后，我们计算出了新的参数向量π.仍旧用之前的方法确定属于哪一个聚类。

100次迭代后得到的数据点分类如下：

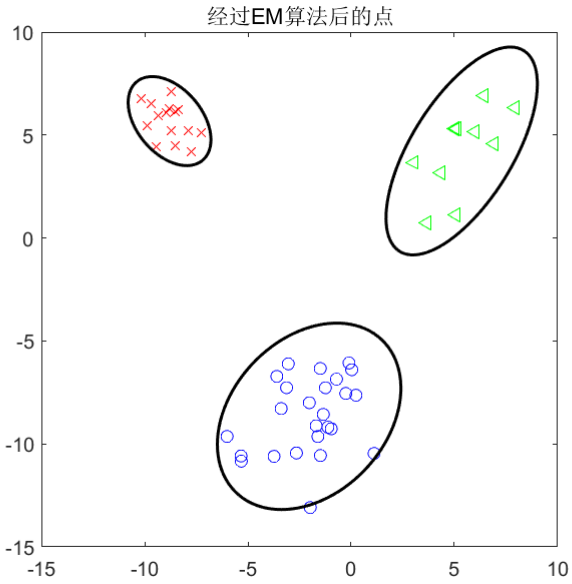
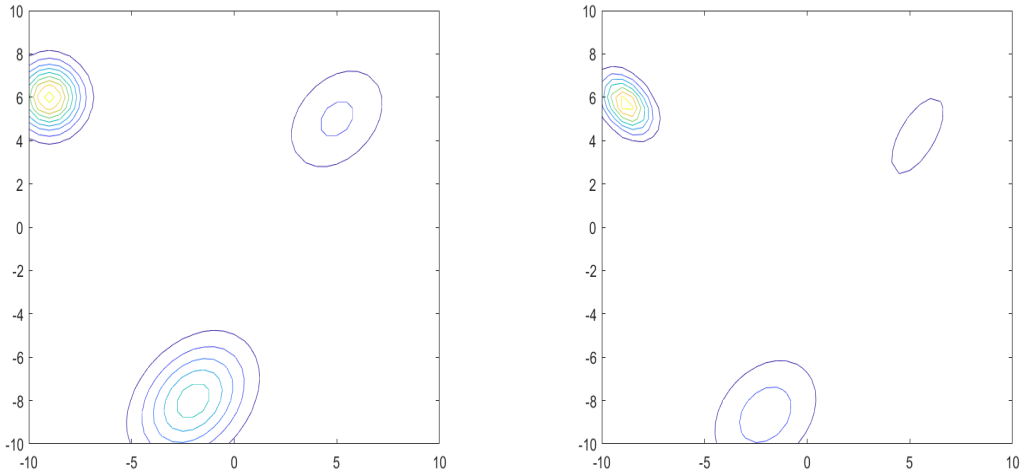


Figure 17: 实验三结果

二者对比一下还是感觉几乎一样。我们可以做出前后的高斯分布的等值线进行对比：



(a) 迭代前 (b) 迭代后

Figure 18: 迭代EM算法前后分布等高线对比

发现前后的位置是几乎一样的，但是分布等高线密集程度变化更大了。这时点的个数更少，这样的变化更明显。这间接说明了，在数据分散度较大的情况下，当数据点减少，分类的位置判别效果仍旧很好，但是密集程度不如之前。这与我们实验点集的数据量减少很多有着很大的关系。

我们给出了前25次迭代所得的NMI值，结果如下：

1	0
2	0.6889
3	0.6614
4	0.9273
5	0.9273
6	1.0000
7	1.0000
8	1.0000
9	1.0000
10	1.0000

Figure 19: 实验三迭代NMI值

可以发现当迭代到约第7次左右，NMI值稳定在1，说明此时算法已收敛，且NMI值几乎就是1，说明分类结果基本上完全正确。

3.5.4 实验四：5000个点进行分类，且数据点方差(分散度)非常小(近乎重合)

我们设置初始的参数π的向量不变，仍为

$$\pi = [0.3, 0.5, 0.2] \tag{57}$$

随机生成5000个属于权重如上的某个混合高斯分布，分布点如下：

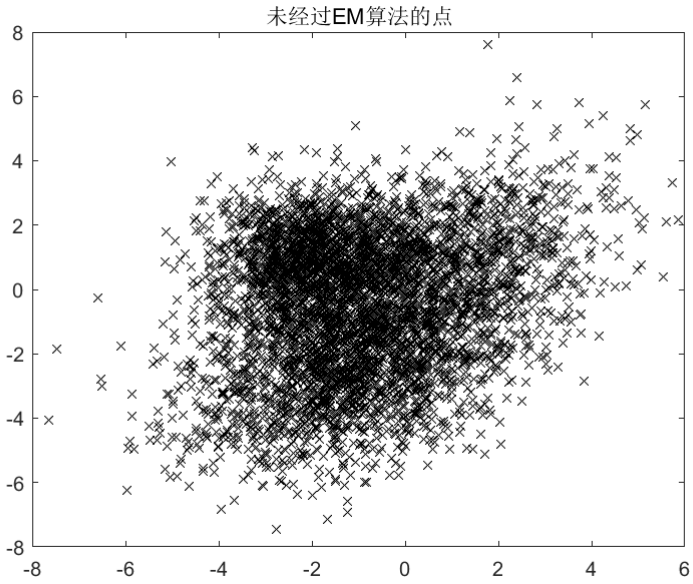


Figure 20: 实验四生成随机点

可以发现这时的数据点十分密集，挨得特别近，近乎可以看成是一个分布了。探讨这时的分类结果如何。

我们需要画出这些点具体所属的高斯分布与后续产生的结果进行比较。具体的高斯分布数据点如下：

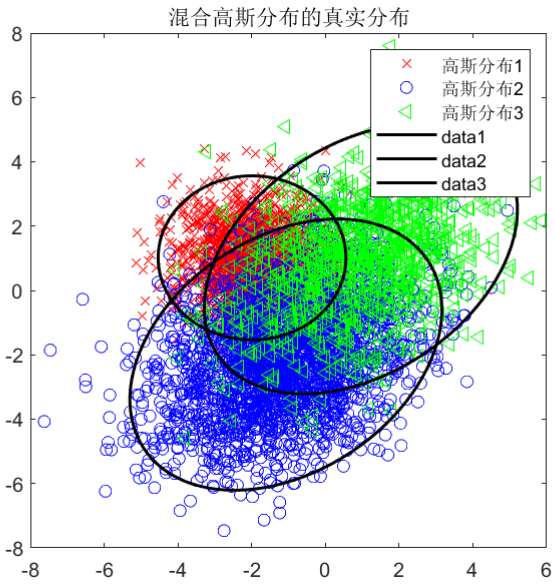


Figure 21: 实验四生成随机点真实值

发现这时要区分其实是一件非常困难的事情。那么EM算法的真实效果如何呢？

经过100次的更新迭代后，我们计算出了新的参数向量π.仍旧用之前的方法确定属于哪一个聚类。

100次迭代后得到的数据点分类如下：

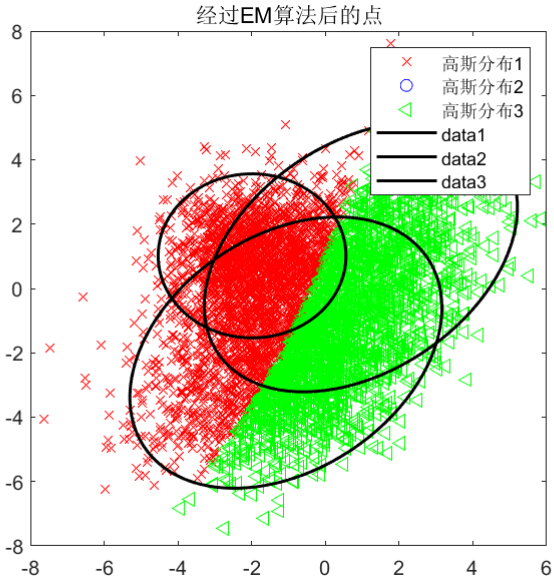
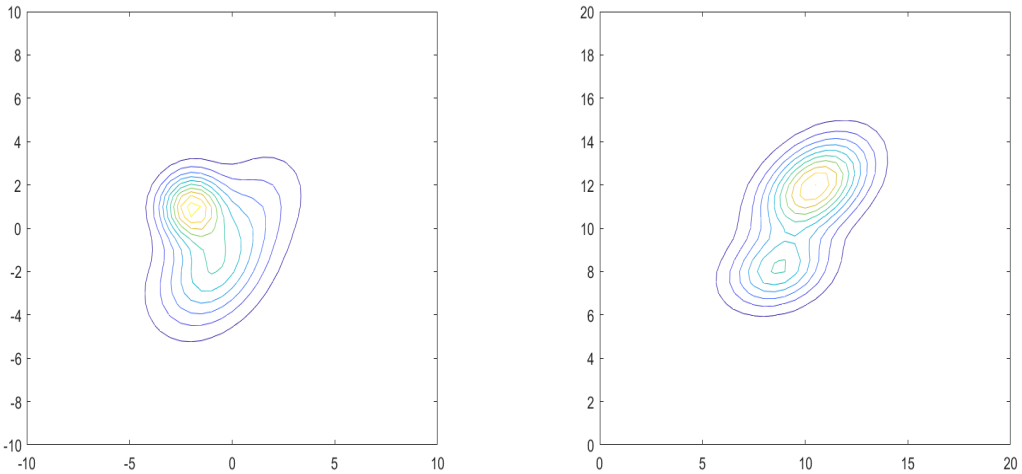


Figure 22: 实验四结果

我们可以很显然地发现，这次的分类结果相当糟糕，原来占比最大的蓝色点近乎消失，被红色点取代。绿色点很多的位置分布也出现了问题，红色点的分布范围比原先大了不少。我们画出前后的分布等值线图进一步比较：



(a) 迭代前 (b) 迭代后

Figure 23: 迭代EM算法前后分布等高线对比

对比这时的前后的等高线图，很明显地发现分布发生了很大的变化。这说明如果数据集很大，并且分布极其密集的情况下，迭代EM算法分类的效果并不是特别好。这时可以尝试别的聚类分类方法。

我们给出了前25次迭代所得的NMI值，结果如下：

1	0
2	0.4842
3	0.4753
4	0.4622
5	0.4552
6	0.4472
7	0.4449
8	0.4436
9	0.4426
10	0.4457
11	0.4520
12	0.4605
13	0.4720
14	0.4844
15	0.4969
16	0.5118
17	0.5277
18	0.5467
19	0.5744
20	0.6447
21	0.5682
22	0.5007
23	0.4506
24	0.4127
25	0.3774

Figure 24: 实验四迭代NMI值

我们给出了25次迭代的NMI值，发现此时的值仍旧变动非常大，说明算法并未收敛，且NMI值离1较远，说明分类结果比较糟糕，可以考虑其他聚类算法。

3.5.5 实验五：关于迭代次数的观察

我们设置初始的参数π的向量不变，仍为

$$\pi = [0.3, 0.5, 0.2] \tag{58}$$

为了观察迭代次数所产生的影响，我们选择500个点进行实验，将500个点分成3类。随机生成500个属于权重如上的某个混合高斯分布，分布点如下：

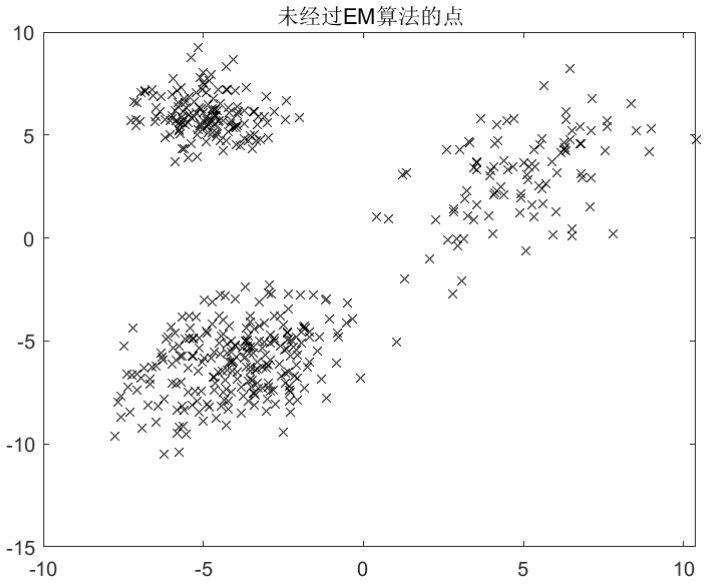


Figure 25: 实验五生成随机点

我们需要画出这些点具体所属的高斯分布与后续产生的结果进行比较。具体的高斯分布数据点如下：

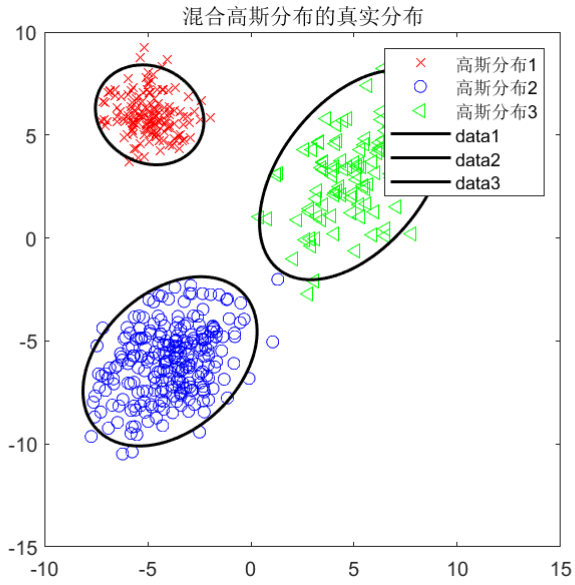
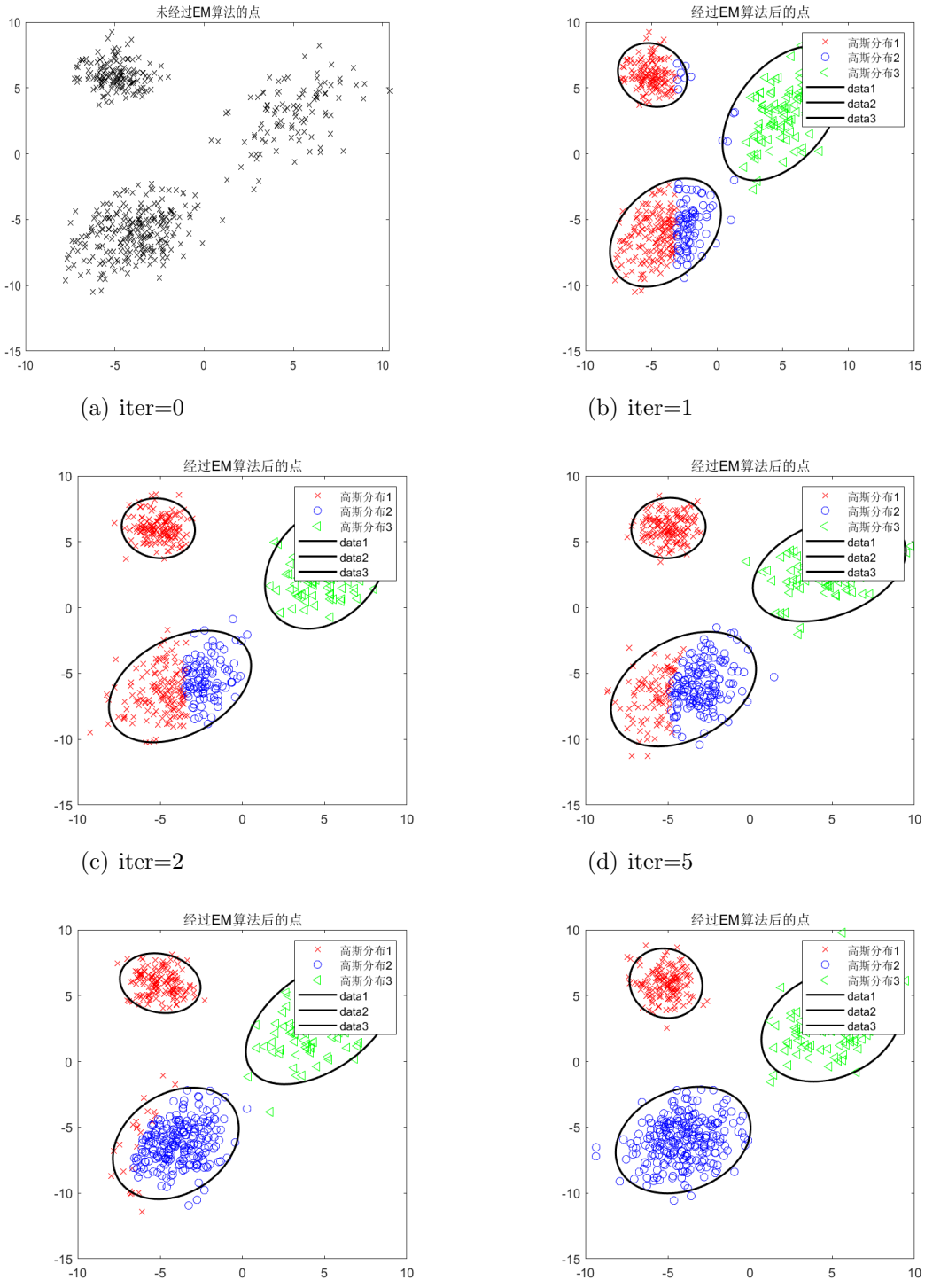


Figure 26: 实验五生成随机点真实值

我们在EM算法过程中变换迭代次数(迭代次数记为iter)，得到的分类结果如下：



(a) iter=0 (b) iter=1

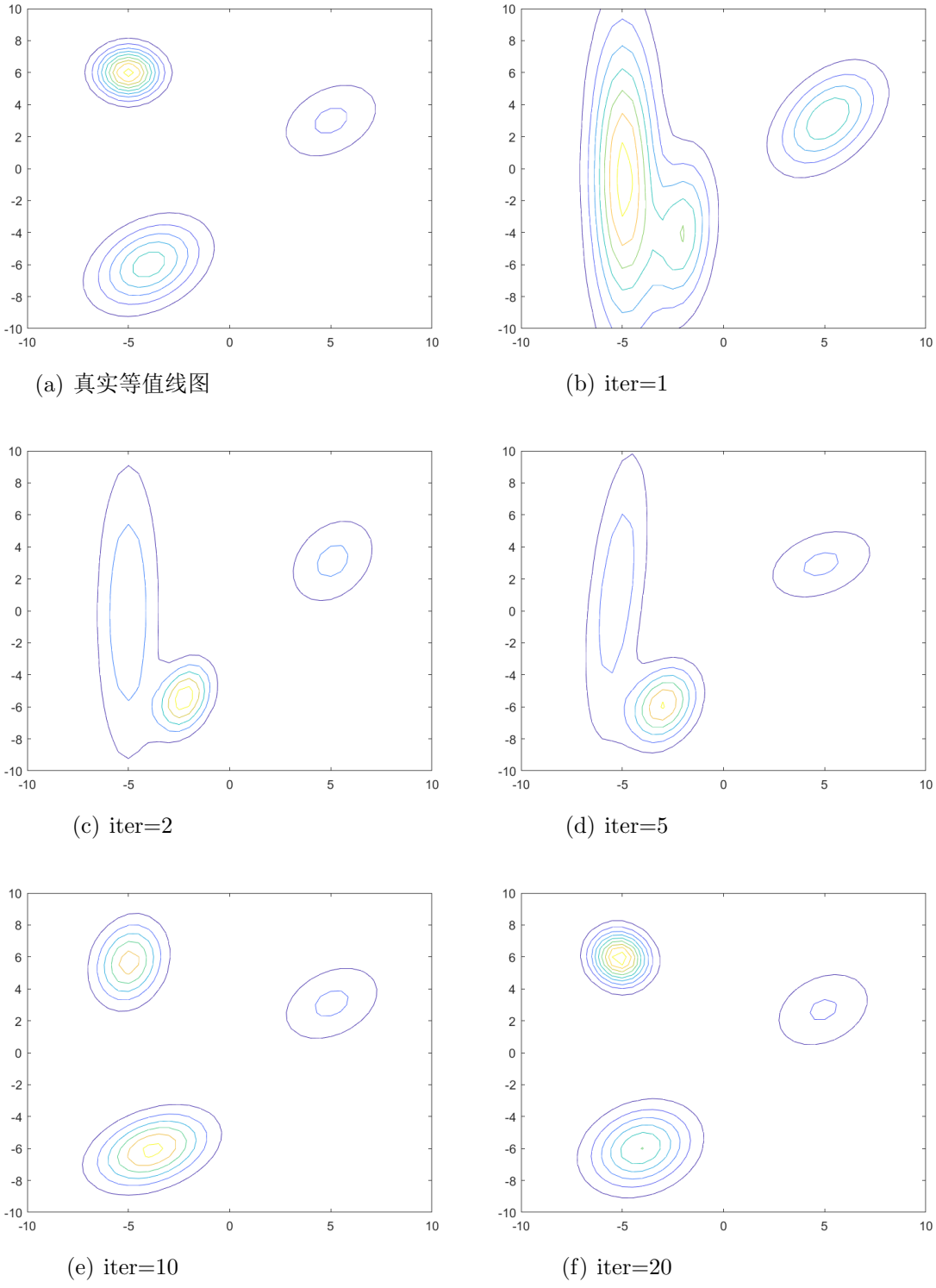
(c) iter=2 (d) iter=5

(e) iter=10 (f) iter=20

Figure 27: 迭代EM算法前后分布等高线对比

可以很显然地发现，随着迭代次数的增多，分类效果越来越好。要想完成混合高斯分布的分类，实质上只需要20步迭代足矣。

我们也可以画出随迭代次数增加的分布的等值线图：



(a) 真实等值线图 (b) iter=1

(c) iter=2 (d) iter=5

(e) iter=10 (f) iter=20

Figure 28: 迭代EM算法前后分布等高线对比

可以看到，随着迭代次数的增加，等值线图不断地向第一张图逼近。迭代次数到达20次时，等值线已经非常接近。

3.5.6 实验六：将EM算法混合高斯分类应用于真实数据

我在网络上查找了old faithful数据集，该数据集描述了位于美国黄石国家公园中的272个名为“老忠实”的天然喷泉位置的分布。得到的原数据点(天然喷泉位置分布)位置在图上绘制如下：

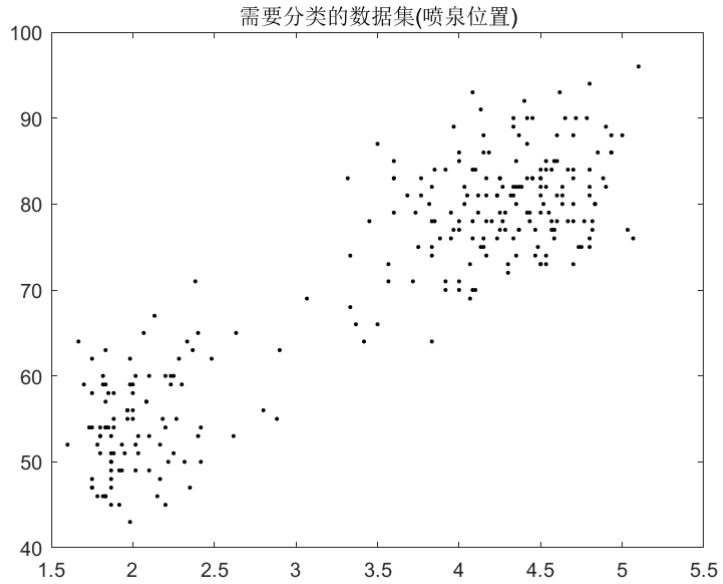
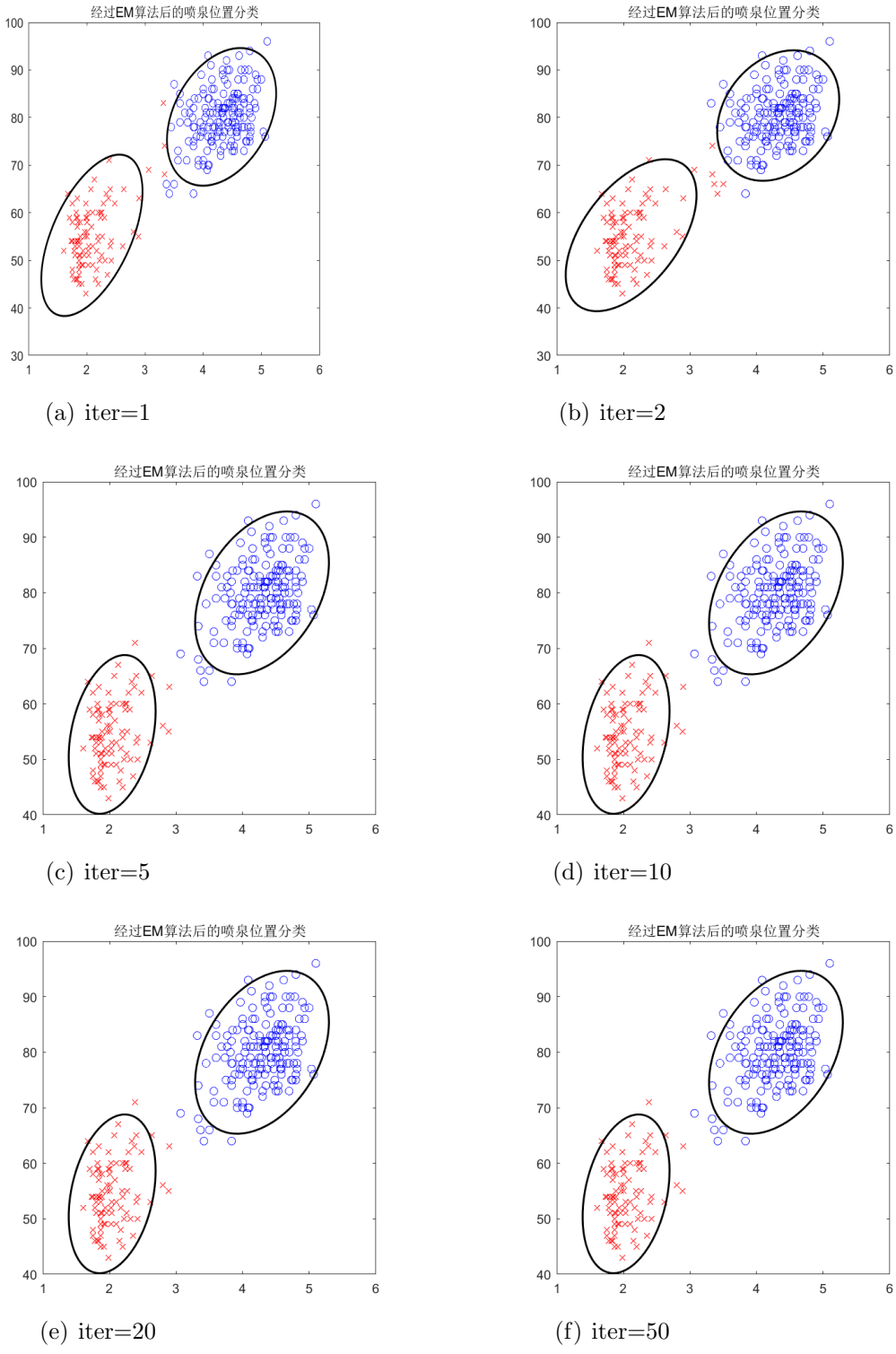


Figure 29: 实验六old faithful数据集点分布

图上的每一个黑点都代表一个天然喷泉的位置。从图上我们可以大致地认为这些温泉在位置上可以分为两类，左下角的一些数据为一类，右上角的一些数据为一类。因此，我们使用EM算法作用于这些数据集进行分类，力图将数据集分为左下角的一类与右上角的一类，共两类。

我们设置初始的权重为1/2与1/2，初始的协方差矩阵都为[1,0;0,1].我们设定初始的两个均值的横坐标为所有点横坐标最大值的3/4加上所有点横坐标最小值的1/4和所有点最大值的1/4加上所有点最小值的3/4，纵坐标均为所有点纵坐标最大值的1/2加上所有点横坐标最小值的1/2.我们经过不断迭代得到的结果如下:



(a) iter=1 (b) iter=2

(c) iter=5 (d) iter=10

(e) iter=20 (f) iter=50

Figure 30: 迭代EM算法前后分类结果

对于这个数据集而言，分类结果还是比较明显的，这和原有的数据点分布就比较分散有一定的关系。可以发现迭代5次再向后迭代结果就基本上不发生改变了。我们也如同之前设想的，将所有点集分成左下与右上两部分，分类效果看上去也不错。

4 一些思考

在我看来，EM算法和机器学习的概率解释之间的关系是十分密切的。EM算法可以理解为监督学习中的分类的一种，也可以理解为一种软聚类的方式。我们在EM算法中不断进行更新的权重本质上是对于数据点集所属哪一个分类的概率。事实上，这一点我们通过对于之前分布等值线的描绘就可以清晰地看出。而也像之前实验中所表现出来的，当点集过大并且属于不同分类的点之间的距离过近时，进行分类是十分困难的，至少分类的效果不好。个人认为这是点集本身的分布对于权重的更新产生了干扰。可以看出在实验四中生成点集时，所有的点集其实可以近似地看成是分布在一个很大的椭圆中的，也就是说，我们把这5000个数据点集给理解成一个全新的非混合的高斯分布也是没有问题的，所以当我们要不断更新权重进行分类时，这个操作就很像是被“强行加上去进行”的一种操作，这也就导致最后数据点的分类结果十分糟糕，出现了本来应该是蓝色点却变成了绿色点的情况，因为本身生成点时，这些点的聚类结果就已经被混淆了，也就最后导致了这样的结果。

要将这样的分类思想给推广到生物统计中，我的理解是一种基于特征选择所进行的分类。我们要进行分类，或者说是软分类，首要的可以考虑的方式应该是特征选择。我们选择了什么样的特征，在这里可以说是给出了什么样的初始权重。我们基于初始权重进行EM算法的迭代，进而深入挖掘新的特征，不断利用特征的结合得到我们想要的结果。例如对于基因数据或者某种种群数据，我们需要通过已有的调查的获得的信息，进一步推算这些信息背后隐含的信息，进而发现更深层次的生物规律与生物事实，这就需要EM算法大显身手，对隐藏变量进行推断。而且在EM算法中，我们有一个很重要的东西，就是关于隐变量的计算与估计。EM算法是似然方法的优良改进，表面上，EM算法将原先难以计算的极大似然问题进行了很好的转化，减少了计算的复杂度，更深层次地，是EM算法基于已有的数据点和分布做出了更为恰当的参数选择。分类应该是基于参数的，或者说基于特征。EM算法推动了特征选择的前进，甚至因为隐变量的存在，我们可以说这样的特征选择可以是在缺少观测信息的基础上实现的。我想，通过EM算法，我们也可以更深刻地理解概率和极大似然估计之间的区别，就是前者是已知分布和参数，求解事件结果出现的次数和频度；而后者是已知分布和事件结果，基于这种已有的特征，估计事件结果以最大概率出现情况下的参数。

5 总结

EM迭代算法主要采用隐变量的思想，可以应用于分类和确定隐变量参数的工作。事实上，对于EM算法如何收敛，判断收敛的标准，可以采用传统聚类的方法，即通过计算NMI值或者其他类似的互信息值来确定收敛与否。总之，EM算法是一个重要算法，通过与极大似然估计MLE理论以及贝叶斯理论的结合，实现了机器学习方面与概率统计理论的结合，为估计、学习参数提供了便利，值得进一步推广。

A 压缩包中文件说明

Task1为混合二项分布(硬币例子)。其中Codes为相关代码，image为实验中生成的图像。Report为报告总和。Codes中4个$generating_data$文件为4次实验生成数据集的代码，em_matlab为实验中EM算法主要代码，4个$test_matlab$文件为4次实验的测试代码(生成图像见image文件夹)。

Task2为混合高斯分布聚类。其中Codes为相关代码，image为实验中生成的图像。Report为报告总和。Codes中4个$generating_data$文件为前4次实验生成数据集的代码，em_matlab为实验中EM算法主要代码，4个$test_matlab$文件为前4次实验的测试代码；$old_faithful$为黄石国家公园天然喷泉分布的真实数据集，$calculate$为计算真实数据集均值与方差的代码，$test_old_faithful$为真实数据集测试代码(即实验六)。实验五代码直接在前4次实验基础上改动迭代次数即可。nmi_1为计算分类的NMI值的代码，$confiEllipse$为matlab画置信椭圆的代码。

B 相关代码

```
%混合二项分布实验：
theta_A_ori=0.6;
theta_B_ori=0.5;
pi_A_ori=0.5;
pi_B_ori=0.5;%设定初始的参数空间中的个参数
4
E_last=1;%设定初始的期望值
error=0.000001;%设定迭代终止的误差值
n=10;%设定选择一次硬币后抛掷硬币的次数
k=5;%设定选择硬币的次数，即实验组数获得的结果
    矩阵应该为
(k*阶矩阵n)
iter=1;%设定初始迭代次数为1
result=zeros(k,n);%设定初始的结果矩阵全
    为0.记硬币正面朝上为，反面朝上为
10.
choice=zeros(k,1);%设定选择硬币的向量，记为
    选择硬币
1，为选择硬币A0B.
num=zeros(1,k);%设定表示每组实验正面朝上次数
    矩阵，即抛次实验多少次正面朝上。
```

```
n
max=100;
gamma_A=zeros(1,k);
gamma_B=zeros(1,k);
theta_A=zeros(1,max);
theta_B=zeros(1,max);
pi_A=zeros(1,max);
pi_B=zeros(1,max);
theta_A(1,1)=theta_A_ori;
theta_B(1,1)=theta_B_ori;
pi_A(1,1)=pi_A_ori;
pi_B(1,1)=pi_B_ori;
for i=1:k
choice(i,1)=binornd(1,0.6,1,1);
end%设定实际上会选到硬币的概率为A0，.6即
pi_A对应的，=0.6;pi_B=0.4.
for i=1:k
if(choice(i,1)==1)
result(i,1:n)=binornd(1,0.7,1,n);
else
result(i,1:n)=binornd(1,0.4,1,n);%在这里
    设定硬币实际正面朝上概率为
A0，即.7theta_A=0.7.相应地，
theta_B=0.4.
end
end
for i=1:k
for j=1:n
if(result(i,j)==1)
num(1,i)=num(1,i)+1;
end
end
end
for t=2:max
for i=1:k
gamma_A(1,i)=pi_A(1,t-1)*(theta_A(1,t-1)
    ^(num(1,i)))
```

```
*((1-theta_A(1,t-1))^(n-num(1,i)))/(pi_A
    (1,t-1)*
(theta_A(1,t-1)^(num(1,i)))*
((1-theta_A(1,t-1))^(n-num(1,i)))+pi_B
    (1,t-1)*
(theta_B(1,t-1)^(num(1,i)))*((1-theta_B
    (1,t-1))^(n-num(1,i))));
gamma_B(1,i)=pi_B(1,t-1)*(theta_B(1,t-1)
    ^(num(1,i)))*
((1-theta_B(1,t-1))^(n-num(1,i)))/(pi_A
    (1,t-1)*
(theta_A(1,t-1)^(num(1,i)))*
((1-theta_A(1,t-1))^(n-num(1,i)))+pi_B
    (1,t-1)*
(theta_B(1,t-1)^(num(1,i)))*((1-theta_B
    (1,t-1))^(n-num(1,i))));
end
theta_A(1,t)=sum(num.*gamma_A)/(n*sum(
    gamma_A));
theta_B(1,t)=sum(num.*gamma_B)/(n*sum(
    gamma_B));
pi_A(1,t)=sum(gamma_A)/k;
pi_B(1,t)=sum(gamma_B)/k;
diff=abs(theta_A(1,t)-theta_A(1,t-1))+
abs(theta_B(1,t)-theta_B(1,t-1))+abs(
    pi_A(1,t)-pi_A(1,t-1));
if(diff<=error)
break;
else
iter=iter+1;
end
end%迭代算法EM
iter
theta_A
theta_B
pi_A
pi_B
```

```
%混合高斯分布做分类():
N=5000;
pi_real=[3/10,5/10,2/10];
mu_real=[7,12;12,7;14,15];
cov_real(:,:,1)=[1,0;0,1];
cov_real(:,:,2)=[3,1;1,3];
cov_real(:,:,3)=[3,1;1,3];
X_1=mvnrnd(mu_real(1,:),cov_real(:,:,1),
    N*pi_real(1));
X_2=mvnrnd(mu_real(2,:),cov_real(:,:,2),
    N*pi_real(2));
X_3=mvnrnd(mu_real(3,:),cov_real(:,:,3),
    N*pi_real(3));
X=[X_1;X_2;X_3];
X=X(randperm(size(X,1)),:);
x=0:0.5:20;
y=0:0.5:20;
[x y]=meshgrid(x,y);
mesh=[x(:),y(:)];
nor_real=pi_real(1)*mvnpdf(mesh,mu_real
    (1,:),cov_real(:,:,1))
+pi_real(2)*mvnpdf(mesh,mu_real(2,:),
    cov_real(:,:,2))
+pi_real(3)*mvnpdf(mesh,mu_real(3,:),
    cov_real(:,:,3));
figure(1);
plot(X_1(:,1),X_1(:,2),'rx',X_2(:,1),X_2
    (:,2),
'bo',X_3(:,1),X_3(:,2),'g<');
title('混合高斯分布的真实分布');
legend('高斯分布1','高斯分布2','高斯分布3');
hold on;
confiEllipse(X_1,0.95);
hold on;
```

```
confiEllipse(X_2,0.95);
hold on;
confiEllipse(X_3,0.95);
hold on;
figure(2);
contour(x,y,reshape(nor_real,size(x,2),
    size(y,2)));
figure(3);
surf(x,y,reshape(nor_real,size(x,2),size
    (y,2)));
figure(4);
plot(X(:,1),X(:,2),'kx');
title('未经过算法的点EM');

pi=[1/3,1/3,1/3];%初始化的取值pi
cov(:,:,1)=[1,0;0,1];
cov(:,:,2)=[1,0;0,1];
cov(:,:,3)=[1,0;0,1];
mu_y_init=(max(X(:,1))+min(X(:,1)))/2;
mu_x1_init=max(X(:,2))/4+3*min(X(:,2))
    /4;
mu_x2_init=2*max(X(:,2))/4+2*min(X(:,2))
    /4;
mu_x3_init=3*max(X(:,2))/4+min(X(:,2))
    /4;
gamma=zeros(size(X,1),length(pi));%gamma
    (i,j),是样本数
    量
i,是聚类数量j,gamma(i,j)表示样本属于聚类的概
    率，最大值表示为的聚类
iji
mu=[mu_x1_init,mu_y_init;mu_x2_init,
    mu_y_init;
mu_x3_init,mu_y_init];

%算法：EM
iter=40;
```

```
for i=1:iter
for j=1:length(pi)
gamma(:,j)=pi(j)*mvnpdf(X,mu(j,:),cov
    (:,:,j));
end
gamma=gamma./repmat(sum(gamma,2),1,size(
    gamma,2));
%建立和一样规模的矩阵，矩阵每一行
    为gammagamma(:,1)+gamma(:,2)值
pi=sum(gamma,1)./size(gamma,1);
mu=gamma'*X;
mu=mu./repmat((sum(gamma,1))',1,size(mu
    ,2));
for j=1:length(pi)
vari=repmat(gamma(:,j),1,size(X,2)).
*(X-repmat(mu(j,:),size(X,1),1));
cov(:,:,j)=(vari'*vari)/sum(gamma(:,j)
    ,1);
end
end

%估计
[c estimate]=max(gamma,[],2);

nor=pi(1)*mvnpdf(mesh,mu(1,:),cov(:,:,1)
    )+pi(2)
*mvnpdf(mesh,mu(2,:),cov(:,:,2))+pi(3)
*mvnpdf(mesh,mu(3,:),cov(:,:,3));
figure(5);
contour(x,y,reshape(nor,size(x,2),size(y
    ,2)));

one=find(estimate==1);
two=find(estimate==2);
three=find(estimate==3);

figure(6);
```

```
plot(X(one,1),X(one,2),'rx',X(two,1),X(
    two,2),'bo',
X(three,1),X(three,2),'g<');
title('经过算法后的点EM');
legend('高斯分布1','高斯分布2','高斯分布3');
hold on;
confiEllipse(X_1,0.95);
hold on;
confiEllipse(X_2,0.95);
hold on;
confiEllipse(X_3,0.95);
hold on;
```

%画置信椭圆:

```
function confiEllipse(datamatrix,p)
%print 2-demension confidence ellipse
%In:×n2 matrix, confidence probability p

data = datamatrix;
covariance = cov(data);
[eigenvec,eigenval] = eig(covariance);

[sortEigenval,index] = sort(diag(
    eigenval),'descend');
sortEigenvec = eigenvec(:,index);

largestEigenval = sortEigenval(1);
smallestEigenval = sortEigenval(end);
%求最小特征值
largestEigenvec = sortEigenvec(:,1);
%求最大特征向量（椭圆的长轴）
```

```
angle = atan2(largestEigenvec(2),
    largestEigenvec(1));
%计算轴和最大特征向量之间的角度x, [-pi,pi]

if(angle < 0)
angle = angle + 2*pi;
end

avg = mean(data); %计算两列数据的均值

%计算置信椭圆的参数
chisquareVal = sqrt(chi2inv(p,2)); %卡方
    值
thetaGrid = linspace(0,2*pi);
phi = angle; %旋转角度
X0=avg(1);
Y0=avg(2);
a=chisquareVal*sqrt(largestEigenval); %
    轮距 长
    度
b=chisquareVal*sqrt(smallestEigenval);

ellipseXR = a*cos( thetaGrid ); %作用于直
    角坐标系
ellipseYR = b*sin( thetaGrid );

R = [ cos(phi) sin(phi); -sin(phi) cos(
    phi) ];
%旋转矩阵

rEllipse = [ellipseXR;ellipseYR]' * R; %
    旋
    转
plot(rEllipse(:,1) + X0,rEllipse(:,2) +
Y0,'k-','linewidth',1.5);
axis square
```

end值的计算：

NMI

```
function NMI = nmi_1( A, B )

if length( A ) ~= length( B)
error('length(␣A␣)␣must␣==␣length(␣B)');
%判断前后两个分类结果的数据点数应相同，若不同则
    直接结束
end
total = length(A);
A_ids = unique(A);
B_ids = unique(B);%取出和中一样的元素（无重
    复）AB

% Mutual information
MI = 0;
for idA = A_ids
for idB = B_ids
idAOccur = find( A == idA );
idBOccur = find( B == idB );
idABOccur = intersect(idAOccur,idBOccur)
    ;

px = length(idAOccur)/total;
py = length(idBOccur)/total;
pxy = length(idABOccur)/total;

MI = MI + pxy*log2(pxy/(px*py)+eps);
% eps : 修正值，最小正数；此处先计算值MI

end
end
```

```
% Normalized Mutual information
Hx = 0; % Entropies
for idA = A_ids
idAOccurCount = length( find( A == idA )
    );
Hx = Hx - (idAOccurCount/total) *
log2(idAOccurCount/total + eps);
%计算所蕴含的自信息（熵）X
end
Hy = 0; % Entropies
for idB = B_ids
idBOccurCount = length( find( B == idB )
    );
Hy = Hy - (idBOccurCount/total) *
log2(idBOccurCount/total + eps);
%计算所蕴含的自信息（熵）Y
end

NMI = 2 * MI / (Hx+Hy);
end
```

```
old_faithful:
load old_faithful;
figure(1);
plot(old_faithful(:,1),old_faithful(:,2)
    ,'k.');
title('需要分类的数据集喷泉位置()');
N=size(old_faithful,1);
x=1.5:0.5:5.5;
y=40:10:100;
pi=[1/2 1/2];%设置初始的权重
cov(:,:,1)=[1,0;0,1];
cov(:,:,2)=[1,0;0,1];%设置初始方差
```

```
X=old_faithful;
mu_x1_init=max(X(:,1))/4+3*min(X(:,1))
    /4;
mu_x2_init=3*max(X(:,1))/4+min(X(:,1))
    /4;
mu_y_init=(max(X(:,2))+min(X(:,2)))/2;
gamma=zeros(size(X,1),length(pi));%gamma
    (i,j),是样本数
    量
i,是聚类数量j,gamma(i,j)表示样本属于聚类的概
    率，ij最大值表示为的聚类
i
mu=[mu_x1_init,mu_y_init;mu_x2_init,
    mu_y_init];
iter=50;
for i=1:iter
for j=1:length(pi)
gamma(:,j)=pi(j)*mvnpdf(X,mu(j,:),cov
    (:,:,j));
end
gamma=gamma./repmat(sum(gamma,2),1,size(
    gamma,2));
%建立和一样规模的矩阵，矩阵每一行
    为gammagamma(:,1)+gamma(:,2)值
pi=sum(gamma,1)./size(gamma,1);
mu=gamma'*X;
mu=mu./repmat((sum(gamma,1))',1,size(mu
    ,2));
for j=1:length(pi)
vari=repmat(gamma(:,j),1,size(X,2)).
*(X-repmat(mu(j,:),size(X,1),1));
cov(:,:,j)=(vari'*vari)/sum(gamma(:,j)
    ,1);
end
end

[c estimate]=max(gamma,[],2);
```

```
one=find(estimate==1);
two=find(estimate==2);
X_1=[X(one,1) X(one,2)];
X_2=[X(two,1) X(two,2)];

figure(2);
plot(X(one,1),X(one,2),'rx',X(two,1),X(
    two,2),'bo');
axis([1 5.5 40 100]);
title('经过算法后的喷泉位置分类EM');
hold on;
confiEllipse(X_1,0.95);
hold on;
confiEllipse(X_2,0.95);
hold on;
```