机器学习第三次作业

统计 81　　宋程　　2184312719

目录

注：本文仅仅只是根据已做实验结果的分析，并没有大量实验保证统计学意义（如 EM 算法和 K-Means 算法的比较），但本文的重点也不在于对于实验结果的分析，而是对于概率角度理解模型的思考。除此之外本文也仅仅只是部分学习过程中思考，可能思路不够清晰，也可能有许多不严谨的地方，期待能有更多的理解也期待与本文读者和老师更多的交流。

一．聚类算法

1 引言

混合高斯分布本质是融合几个单高斯分布，获得一个更加复杂的分布，从而我们可以根据该分布产生更复杂的样本或逼近真实分布。理论上，如果某个混合高斯模型融合的高斯模型个数足够多，它们之间的权重以及各自的参数设定得足够合理，这个混合模型可以拟合任意分布的样本，因此高斯混合分布可以看成是分布的万有逼近器。在现实中有许多数据都可以通过混合高斯分布进行逼近，并进行聚类。

使用 EM 算法可以通过迭代的方法计算混合高斯模型的相应参数并确定权重，从而到聚类的目的。而 K-Means 算法可以理解为退化的 EM 算法，但在此报告中我们不这么理解两个模型，我们将可以理解为 EM 算法始终使用概率进行度量与聚类而 K-means 算法我们理解为其从传统的机器学习角度来考虑聚类问题，两者的主要区分在于软聚类与硬聚类，我们将视线放在概率角度理解两种算法或许会有更多的收获

我们将使用随机生成混合高斯分布数据，并利用 EM 算法计算相应参数。

2 符号说明

符号	说明
x_i	第 i 个样本的数据
π_{j}	从总体中随机取出一个数据点，该数据点属于第 k 类的概率。
θ_{j}	第 k 类的高斯分布的参数。
z_{ij}	第 i 个数据点的类型，当第 i 个数据点属于第 k 类时为 1，否则为 0。
ϕ	高斯分布的概率分布函数。

注：1. 当符号没有下标时，表示相应的向量表述。

2. 当符号为大写字母时表示相应随机变量，小写时表示相应的样本数据。

3. i = 1，2，・・・number_data　　j = 1，2，・・・ k

4. 带上标符号表示迭代次数为上标时，该符号代表的参数值。

3 EM 算法推导

目标:

$$\hat{\theta} = \underset{\theta,\pi}{\operatorname{argmax}} \log P(X;\theta,\pi) = \underset{\theta,\pi}{\operatorname{argmax}} \log(\prod_i P(X_i = x_i;\theta,\pi)$$
$$= \underset{\theta,\pi}{\operatorname{argmax}} \sum_i \log P(x_i;\theta,\pi) = \underset{\theta,\pi}{\operatorname{argmax}} \sum_i \log \sum_{z_i} P(X_i = x_i, Z_i = z_i;\theta,\pi)$$

步骤:

①随机化或根据专业知识等先验选取相应初始参数，记 ite=0（迭代次数），转步二。

②E-step：

$$E_z(\ln P(X,Z;\theta,\pi)) = \sum_Z \ln(P(X,Z;\theta,\pi))P(Z \mid X;\theta^{(ite)},\pi^{(ite)})$$
$$= \sum_Z \sum_i Ln(P(X_i,Z;\theta,\pi))P(Z \mid X_i;\theta^{(ite)},\pi^{(ite)})$$

已知：

$$P(X_i = x_i \mid Z_i = z_i;\theta,\pi) = \sum_{j=1}^{k} z_{ij}\phi(x_i)$$
$$P(Z_{ik} = 1;\pi) = \pi_k$$
$$P(X_i = x_i, Z_i = z_i;\theta,\pi) = P(Z_i = z_i;\theta,\pi)P(X_i = x_i \mid Z_i = z_i;\theta,\pi)$$
$$P(Z_i = z_i \mid X_i = x_i;\theta,\pi) = \frac{P(X_i = x_i, Z_i = z_i;\theta,\pi)}{P(X_i = x_i;\theta,\pi)}$$
$$= \frac{P(X_i = x_i, Z_i = z_i;\theta,\pi)}{P(X_i = x_i, Z_i = 0;\theta,\pi) + P(X_i = x_i, Z_i = 1;\theta,\pi)}$$

将已知代入 $E_z(\ln P(X,Z;\theta,\pi))$，转步 3。

③M-step

$$\theta^{(ite+1)} = \arg\max_{\theta,\pi} E_{z;\theta^{(ite)}}(\ln P(X,Z;\theta))$$
$$= \arg\max_{\theta,\pi} \sum_i \sum_j \ln(\pi_j \phi(X_i;\theta_j)) \frac{\pi_j^{(ite)}\phi(X_i;\theta_j^{(ite)})}{\sum_j \pi_j^{(ite)}\phi(X_i;\theta_j^{(ite)})}$$

$\frac{\pi_j^{(ite)}\phi(X_i;\theta_j^{(ite)})}{\sum_j \pi_j^{(ite)}\phi(X_i;\theta_j^{(ite)})}$ 记为 $\gamma_{ij}^{(ite)}$

$$\theta^{(ite+1)} = \arg\max_{\theta,\pi} E_{z;\theta^{(ite)}}(\ln P(X,Z;\theta))$$
$$= \arg\max_{\theta,\pi} \sum_i \sum_j \ln(\pi_j \phi(X_i;\theta_j))\gamma_{ij}^{(ite)}$$

对上式化简,可将π_B用π_A表示，并求极大值：

$$\underset{\theta,\pi,\lambda}{\operatorname{argmax}}\, g(\theta,\pi,\lambda) = \sum_i \sum_j \ln(\pi_j \phi(X_i;\theta_j))\gamma_{ij}^{(ite)} + \lambda(\sum_j \pi_j - 1)$$

$$\frac{\partial g}{\partial \pi} = 0$$

$$\frac{\partial g}{\partial \theta} = 0$$

$$\frac{\partial g}{\partial \lambda} = 0$$

可得：

$$\pi_j^{(ite+1)} = \frac{\sum_i \gamma_{ij}^{(ite)}}{\sum_j \sum_i \gamma_{ij}^{(ite)}}$$

$$\mu_j^{(ite+1)} = \frac{\sum_i \gamma_{ij}^{(ite)} x_i}{\sum_i \gamma_{iA}^{(ite)}}$$

$$\Sigma_j^{(ite+1)} = \frac{\sum_i \gamma_{iB}^{(ite)} (x_i - \mu^{(ite)})(x_i - \mu^{(ite)})^T}{\sum_i \gamma_{ij}^{(ite)}}$$

计算出相应的参数，ite=ite+1，转步四。

④计算是否满足收敛，若是输出参数，若否转步 2。

4 实验

我们根据参数生成数据集并使用 NMI（标准化互信息）作为收敛判断的准则进行如下实验：

4.1 生成数据

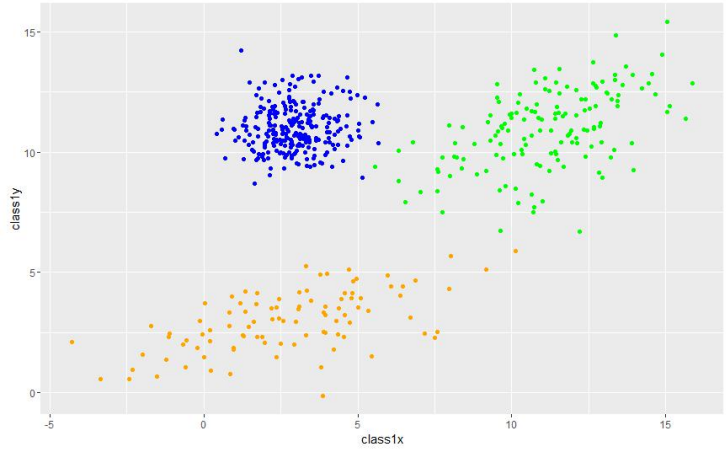
在 r 中设定参数：

```
Sigma1 <- matrix(c(10,3,3,2),2,2)
Sigma2 <- matrix(c(5,2,2,3),2,2)
Sigma3 <- matrix(c(1,0,0,1),2,2)

mu1=matrix(c(3,3))
mu2=matrix(c(11,11))
mu3=matrix(c(3,11))
num_data=500

pi1=0.2
pi2=0.3
pi3=1-pi1-pi2
```

根据如上参数生成 500 个数据点：



4.2 EM 算法

我们首先使用 EM 算法观察聚类效果。

4.2.1 迭代结果及参数

$$\pi_1 = 0.191951 \qquad \pi_2 = 0.2934275 \qquad \pi_3 = 0.5146215$$

$$\mu_1 = \begin{bmatrix} 2.771872 \\ 2.988539 \end{bmatrix} \qquad \mu_2 = \begin{bmatrix} 11.13351 \\ 10.95592 \end{bmatrix} \qquad \mu_1 = \begin{bmatrix} 2.946446 \\ 10.995315 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 8.046127 & 2.139173 \\ 2.139173 & 1.594005 \end{bmatrix}$$

$$\Sigma_2 = \begin{bmatrix} 4.279949 & 1.781623 \\ 1.781623 & 2.693934 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 1.01759224 & 0.05251702 \\ 0.05251702 & 0.87708998 \end{bmatrix}$$

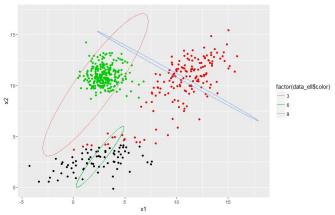
NMI=0.9984428

得到如下迭代结果：

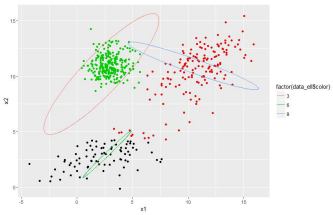
4.2.2 迭代参数二倍标准差椭圆

在这部分我们根据参数画出置信椭圆（即该部分的椭圆由迭代参数计算得到）：

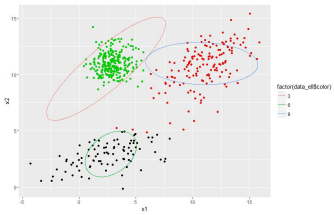
Iteration=1



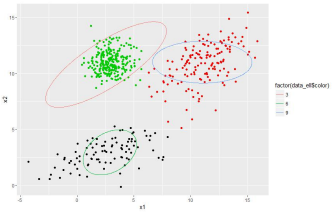
Iteration=2



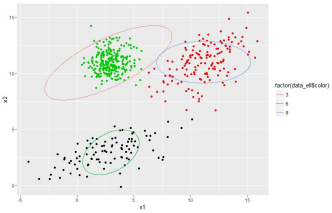
Iteration=3



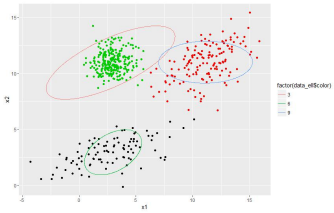
Iteration=4



Iteration=6



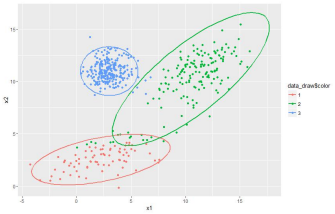
Iteration=7



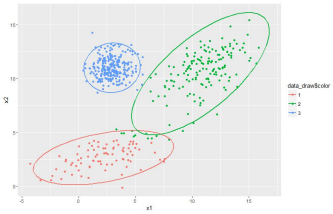
4.2.3 分类椭圆

此部分的置信椭圆通过对分类点计算协方差得到：

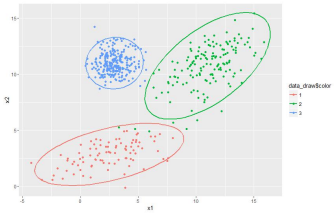
Iteration=1



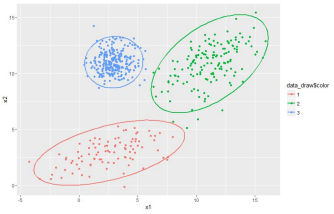
Iteration=2



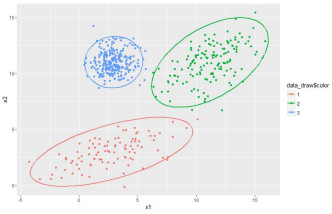
Iteration=3



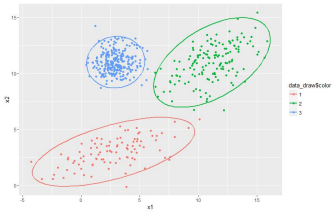
Iteration=5



Iteration=6

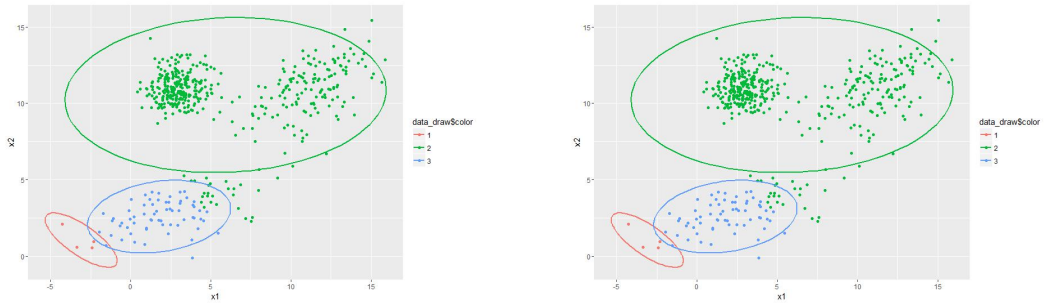


Iteration=7



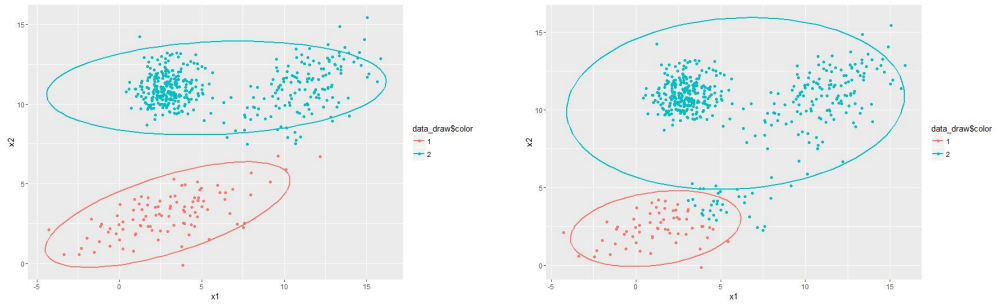
4.3 K-Means 算法

下面我们使用 K-means 算法对数据进行聚类：



NMI 为 0.5411593，且迭代次数多于 EM 算法。

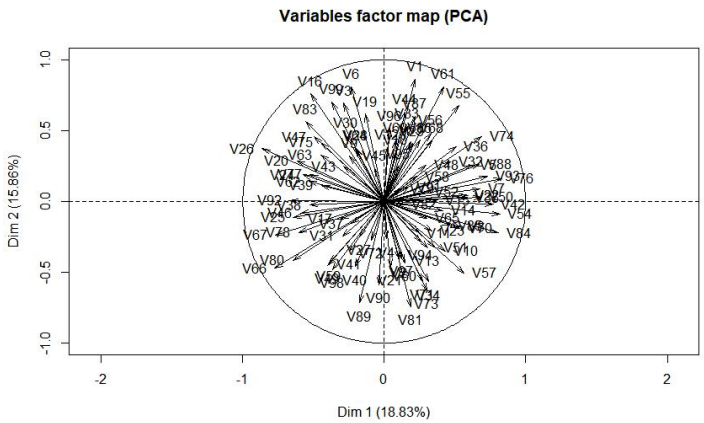
4.4 分类类别为 2



左图为 EM 算法，NMI 为 0.693329；右图为 K-Means 算法结果，NMI 为 0.463188。

4.5 真实数据迭代

我们选取甲型流感对人浆细胞样树突状细胞（pDC）基因表达的影响的数据，选取其中前 100 个表达的基因进行 PCA 降维，得到如下分析：



可视化特征值：

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	18.641064	18.829358	18.82936
Dim.2	15.705959	15.864605	34.69396
Dim.3	13.905536	14.045996	48.73996
Dim.4	12.113784	12.236146	60.97610
Dim.5	9.899507	9.999502	70.97561
Dim.6	9.533847	9.630148	80.60575
Dim.7	7.153840	7.226101	87.83186
Dim.8	6.481665	6.547136	94.37899
Dim.9	5.564798	5.621008	100.00000

我们选取数据的前选取前五个维度进行聚类分析，最终得到如下结果:
实际分类：c（1 2 1 2 1 2 1 2 2）
分析结果：c（2 2 1 2 1 2 1 1 1 1）

我们可以看到分类的准确率只有 70%，而且这还是我们不断调整初值的结果，在这个过程中我们发现，使用 K-Means 聚类对选取的初值有极高的要求，不合适的初值选择很容易造成分类的错误，且由于基因表达数据的衡量指标与变化幅度等数据原因更加凸显了这一缺点，因此处理生物数据，提前进行更多的数据分析也是很有必要的。

5 实验结果分析及思考

5.1 实验结果分析

在 4 中，我们分别使用了传统 EM 算法和 K-means 算法对生成的数据集进行了聚类，我们可以看到不论 k=3 还是 k=2，EM 算法都较 K-Means 算法由更好的表现，且在 k=2 时 EM 算法将部分点画在了置信椭圆外，如果我们将该部分点视为异常点，那么在这种情况下显然 EM 算法受到的影响较 K-means 更小。而这或许就是概率的作用，我们将在思考部分进一步阐述对于概率对聚类算法的影响。

5.2 思考

实际上由于 EM 算法的协方差矩阵可变且分类方法属于软聚类，在数据集生成如 5.1 中所示的数据情况下，EM 算法有更好的表现，其分类情况更加贴合实际，更为重要的是我们可以看到在 k=2 两种算法有一个很明显的区别，EM 算法将部分点置于置信椭圆外，而 K-means 方法将部分 a 类点放在了 b 类的置信椭圆中，我们认为造成这种现象的原因是 EM 算法从概率的角度思考问题，对于部分点可能是介于两个不同类中间的，而 K-Means 算法将每个点划分为非 A 即 B，该方法更容易导致一些错分，也不利于实际问题的分析，从个人角度来看，以概率的方法分析问题就类似于假设检验中的 p 值，而不是仅仅在确定的检验水平下简单的设置拒绝域，这样的方法更加贴合实际，也更有利于在专业知识以及先验的帮助下获得更好的分析结果。我们可能在现实中碰到一些比较模糊的问题，比如

一个人是否健康，而在这样的问题下，使用概率的去度量更加适用，也更能为诊断提供依据。而这也是我们所认为从概率分析问题的一大好处。

5.3 其他分析

在经典的EM算法中，我们可以发现如果我们将不同类的参数设置为相同的，且将权重设为等概率的，那么迭代结果将不会发生改变，这是因为在这样的设置下先验为样本来自同一总体。

我们使用EM算法很多时候求出的可能仅仅是极优值，受到先验与迭代结果的影响，因此适用EM算法和K-means都局限于先验的选择（包括聚类的类数和高斯模型的参数），因此使用K-means等聚类方法相应的专业知识和先验也是很重要的。

实际数据的分析中也凸显了先验选择与数据处理的重要性。

二．神经网路分类问题的概率理解

在许多神经网络模型中，许多神经元使用的激活函数为 sigma 函数，而该函数的取值在[0,1]之间，我们更易于利用该函数从概率角度分析神经网络模型。我们可以将每个权重看成一个概率，而与该神经元连接的浅层神经元（离输入端更近的神经元）类似于 EM 算法中 X 即数据，而权重则为参数，在这样的解释下，我们对权重的调整就可以理解为在给定先验下（随机化或预训练等方法得到的权重矩阵），根据已知的样本数据求出最佳的参数来让分类正确的概率尽可能大。

或许我们对卷积神经网络（CNN）进行分析会有更好的理解。对于进行卷积之后的神经元，我们往往会将其理解为某种特征，比如对动物的分类，某一个卷积后的神经元的作用可以被理解为是否有较尖锐的嘴，而如果我们从概率的角度我们可以将该神经元的作用理解为样本图片中有较尖锐的嘴的概率，而多种特征的概率共同反应最终分类正确的概率。

1. 通过对物理世界的观察，我们理解图像识别中的某个隐状态或者说特征一定是局部像素就可以确认的，但是我们不知道这个局部有多大，所以只能一点点的搜索，即从一个很小的窗口开始，观察所有的这样的小窗口是否可以转换为一个隐状态。
2. 由于窗口足够小，所以特征类别可以足够少，每个小窗口的观察结果只输出类别信息，而不企图学习可以泛化的规律，即不去寻找映射函数和映射后的z向量，只寻找一个典型状态z的概率，这个典型状态z仅仅是因为样本聚集在某个特定空间附近，即满足特定分布。

正如上述来源知乎的分析，我们可以认为神经网络在寻找一个特定的分布，从而使我们的分布更加贴合实际情况，在这样的模型下，我们不断利用子模型对局部进行分析，提取更为复杂的特征，最后将我们得到的子模型进行组合，用概率的组合来计算概率得到神经网络模型。

我们不仅可以通过概率的角度来分析神经网络，同时也可以通过概率来构建模型，不同的分布、参数计算方式（极大似然、最大后验）也对应于不同的损失函数，且自然的引入并解释了正则项，但使用概率的方法，我们将目标由寻找分类尽可能准确的模型转向使得分类正确概率大的分布，不仅使得模型的可解释性更强，同时也利用现实问题的解释（总结部分）。

三. 总结

我们使用概率的角度来分析机器学习，不仅能拓展模型的可解释性，同时也能获得更加适合现实的模型。

1. 前文总结

在第一部分中我们比较了 K-means 算法和 EM 算法，解释了从概率角度构建模型获得的性能提升，也分析了两者结果的原因（我们认为造成这种现象的原因是 EM 算法从概率的角度思考问题，对于部分点可能是介于两个不同类中间的，而 K-Means 算法将每个点划分为非 A 即 B，该方法更容易导致一些错分，也不利于实际问题的分析，从个人角度来看，以概率的方法分析问题就类似于假设检验中的 p 值，而不是仅仅在确定的检验水平下简单的设置拒绝域，这样的方法更加贴合实际，也更有利于在专业知识以及先验的帮助下获得更好的分析结果。我们可能在现实中碰到一些比较模糊的问题，比如一个人是否健康，而在这样的问题下，使用概率的去度量更加适用，也更能为诊断提供依据。而这也是我们所认为从概率分析问题的一大好处。

在神经网络模型中，我们不断利用子模型对局部进行分析，提取更为复杂的特征，计算相应特诊的概率，最后将子模型进行组合，用概率的组合来计算概率得到神经网络模型。我们始终想要得到的只是更加贴合特定数据的分布，从而保证分类性能与泛化能力。

2. 思考

我们可以看到从现实的角度来看许多时候分类并不是准确的，就如同黑白中间会有灰色一般，仅从数据进行分析得到的结果未必是好的，而这时候先验和专业知识所能发挥的作用就会凸显出来，仅仅从数据去构建模型，我们得到的仅仅是一个在现有数据情况下的一个分类器，就如同一个能分类各种动物的 CNN 网络却可能无法找出横线和竖线的区别，而从概率的角度来看，我们能得到的是一个计算分布的模型，我们可以利用这个分布去解释生活中的更多现象。

或许我们也可以从这个角度来分析数据驱动与模型驱动的区别，模型驱动就像概率角度理解机器学习，我们的目的是从更加贴近现实，更能发挥专业知识的作用的角度去构建模型，这并不意味着模型使难度的提升，而是再构建模型时更多的保证模型的稳定与可分性，或许在很多时候数据驱动的模型可以通过大量的数据来分析结果获得较好的模型，即使再搭建模型时或许也不需要专业知识，但是我们无从解释对错的模型我们又怎么能说明其正确性呢，而使用模型驱动固然需要更加复杂的模型思考，但这并不意味着放弃数据，也不意味着更复杂的模型，就像使 EM 算法我们同样能通过提升样本容量来获得更加贴近现实的分布，使用

模型驱动可以帮助我们理解模型，但更加重要的是我们可以从专业知识的角度去分析模型的优劣，当有一天模型不再适用的时候我们能够及时的发现，也能够及时的修正。就像在 4.5 中对实际处理数据的聚类一样，简单的拿到数据并使用数据直接分析往往难以得到很好的效果，数据处理、模型解释、分析改进，我们要做的远远不止处理数据。我们不认为机器学习能取代人类，因为我们始终相信模型的构建需要的不仅仅只是数据。

四．代码

代码软件 R，版本信息：Version:1.0 StartHTML:0000000107 EndHTML:0000001013 StartFragment:0000000127 EndFragment:0000000995

```
#生成数据
library(MASS)
library(ggplot2)
library(mvtnorm)
library(ellipse)

Sigma1 <- matrix(c(10,3,3,2),2,2)
Sigma2 <- matrix(c(5,2,2,3),2,2)
Sigma3 <- matrix(c(1,0,0,1),2,2)

mu1=matrix(c(3,3))
mu2=matrix(c(11,11))
mu3=matrix(c(3,11))
num_data=500

pi1=0.2
pi2=0.3
pi3=1-pi1-pi2

set.seed(123)
num_data_vec=rmultinom(1,num_data,c(pi1,pi2,pi3))
set.seed(123)
data1=mvrnorm(num_data_vec[1], mu1, Sigma1)
```

```
set.seed(123)
data2=mvrnorm(num_data_vec[2], mu2, Sigma2)
set.seed(123)
data3=mvrnorm(num_data_vec[3], mu3, Sigma3)
standard=c(rep(1,num_data_vec[1]),rep(2,num_data_vec[2]),rep(3,num_data_vec[2]))#sta_classif
y
data=rbind(data1,data2,data3)
#画图

data1=as.data.frame(data1)
colnames(data1)=c("class1x","class1y")
data2=as.data.frame(data2)
colnames(data2)=c("class2x","class2y")
data3=as.data.frame(data3)
colnames(data3)=c("class3x","class3y")
ggplot() +
  geom_point(data =data1,mapping = aes(x = class1x, y = class1y),color = 'orange')+
  geom_point(data =data2,mapping = aes(x = class2x, y = class2y),color = 'green')+
  geom_point(data =data3,mapping = aes(x = class3x, y = class3y),color = 'blue')

ggplot() +
  geom_point(data =data1,mapping = aes(x = class1x, y = class1y),color = 'orange')



#em
#计算 gamma 值
gamma_value=function(number_data,k,data,parameter_pi,parameter_mu,parameter_sigma)
{
  number_data=num_data
  list1=matrix(rep(0,time=number_data),nrow = number_data)
  gamma_val=matrix(rep(0,number_data*k),nrow=number_data)

  for (i in 1:number_data) {

    for (j in 1:k) {
      data_ij=data[i,]
```

```
list1[i]=list1[i]+parameter_pi[,,j]*dmvnorm(as.vector(data[i,]),as.matrix(parameter_mu[,,j])
,
                                                as.matrix(parameter_sigma[,,j]))

    }

    for (j in 1:k) {
      data_ij=data[i,]
      gamma_val[i,j]=parameter_pi[,,j]*dmvnorm(data_ij,parameter_mu[,,j],
parameter_sigma[,,j])

      gamma_val[i,j]=(gamma_val[i,j])/(list1[i])
    }

  }
  return(gamma_val)
}

gamma_value1=function(number_data,k,data,parameter_mu)
{
  number_data=num_data
  list1=matrix(rep(0,time=number_data),nrow = number_data)
  gamma_val=matrix(rep(0,number_data*k),nrow=number_data)

  for (i in 1:number_data) {

    for (j in 1:k) {
      data_ij=data[i,]
      gamma_val[i,j]=(data_ij-t(parameter_mu[,,j]))%*%t((data_ij-t(parameter_mu[,,j])))

    }

  }
  return(gamma_val)
}

#计算分类
```

```
Cla=function(number_data,gamma_val)
{
  cla=rep(0,number_data)
  for (i in 1:number_data) {
    cla[i]=which.max(gamma_val[i,])
  }
  return(cla)

}
Cla1=function(number_data,gamma_val)
{
  cla=rep(0,number_data)
  for (i in 1:number_data) {
    cla[i]=which.min(gamma_val[i,])
  }
  return(cla)

}
#计算 NMI
Cov=function(number_data,k,cla,standard)
{
  pro_cla_sta=matrix(rep(0,k*3),nrow = k)#联合概率密度
  MI_cla_sta=0
  H_cla=0
  H_sta=0
  for(i in 1:number_data)
  {
    pro_cla_sta[cla[i],standard[i]]=pro_cla_sta[cla[i],standard[i]]+1
  }
  pro_cla_sta=pro_cla_sta*(1/number_data)
  for (i in 1:k) {
    H_cla=H_cla-sum(pro_cla_sta[i,])*log(sum(pro_cla_sta[i,]))
    H_sta=H_sta-sum(pro_cla_sta[,i])*log(sum(pro_cla_sta[,i]))
    for (j in 1:k) {
      if(pro_cla_sta[i,j]!=0)
      {
```

```
MI_cla_sta=MI_cla_sta+pro_cla_sta[i,j]*log(pro_cla_sta[i,j]/(sum(pro_cla_sta[i,])*sum(pro_cla_sta[,j])))
      }
    }

  }
  NMI=2*MI_cla_sta/(H_cla+H_sta)
  print(NMI)
  return(NMI)

}
#给定协方差阵获取椭圆#来源：https://zhuanlan.zhihu.com/p/65934683
ellipse.simple <- function(ell_s1, ell_s2, ell_c){
  a <- ell_s1*ell_c
  b <- ell_s2*ell_c
  x <- seq(from = -a, to = a, length.out = 400)
  points <- data.frame(
    x1 = c(-x, x),
    x2 = NA

  )
  points$x2[1:400] <- sqrt(((a*b)^2-(b*x)^2)/a^2)
  points$x2[401:800] <- -sqrt(((a*b)^2-(b*x)^2)/a^2)
  return(points)
}
ellipse.general <- function(ell_mu, ell_sig){

  #ell_C=2
  lambda <- diag(eigen(ell_sig)$values)  # 特征值
  P <- eigen(ell_sig)$vectors            # 特征向量
  Y <- ellipse.simple(ell_s1 = sqrt(lambda[1,1]), ell_s2 = sqrt(lambda[2,2]), 2)  # 中心在原点，没有倾斜的椭圆的坐标
  X <- t(P%*%t(Y) + rep(ell_mu,800))  # 对坐标旋转位移
  X <- as.data.frame(X)
  colnames(X) <- c('x1', 'x2')
  return(X)
}
```

```
#根据协方差阵画图
draw_po=function(number_data,k,data,cla,ell_mu1,ell_sigma)
{
  if(FALSE)
  {
    num_class=rep(0,k)
    data_class=c()
    for (j in 1:k) {
      for (i in 1:number_data) {

        if(cla[i]==j)
        {
          data_class=append(data_class,data[i,])
          num_class[j]=num_class[j]+1
        }
      }

    }
    data_class=matrix(data_class,nrow = 2)
    #data_class=data.frame(data_class)
    #rownames(data_class)=c("x1","x2")
    p=ggplot()
    num_p=1
    #data_class_draw_list=list()
    for (i in 1:3) {
      #data_class_draw=data.frame(data_class[,num_p:(num_p+num_class[i]-1)])
      #rownames(data_class_draw)=c("x1","x2")
      #data_class_draw_list=append(data_class_draw_list,data_class_draw)
      p=p +
        geom_point(data = data.frame(data_class[,num_p:(num_p+num_class[i]-1)],row.names
=c("x1","x2")),
                   mapping = aes(x = x1, y = x2),
                   color="blue")
      num_p=num_p+num_class[i]

    }
```

```
  }
  data_draw=data.frame(matrix(c(data,cla),ncol=3))
  colnames(data_draw)=c("x1","x2","color")
  p=ggplot()
  p=p +
    geom_point(data = data_draw,
               mapping = aes(x = x1, y = x2),
               color=data_draw$color)
  data_ell=c()
  data_ell_color=c()

  for (i in 1:k) {

data_ell=append(data_ell,ellipse.general(as.matrix(ell_mu1[,,i]),as.matrix(ell_sigma[,,i])))
    data_ell_color=append(data_ell_color,c(rep(i,800)))

  }
  data_ell=matrix(unlist(data_ell),ncol = 2)
  data_ell=data.frame(data_ell,data_ell_color*3)
  #data_ell_color=factor(data_ell_color*2)
  colnames(data_ell)=c("y3","y4","color")
  p=p+
    geom_path(data = data_ell, mapping = aes(x = y3, y = y4,color =factor(data_ell$color)))
  print(p)
  return(p)
}

draw_po3=function(number_data,k,data,cla,ell_mu1,ell_sigma)
{
  data_draw=data.frame(matrix(c(data,cla),ncol=3))
  colnames(data_draw)=c("x1","x2","color")
  p=ggplot()
  p=p +
    geom_point(data = data_draw,
               mapping = aes(x = x1, y = x2),
               color=data_draw$color)
  data_ell=c()
```

```
  data_ell_color=c()

  for (i in 1:k) {
    data_ell=append(data_ell,ellipse(ell_sigma[1,2,i],
                                     c(ell_sigma[1,1,i],ell_sigma[2,2,i]),
                                     ell_mu1[,,i]))
    data_ell_color=append(data_ell_color,c(rep(i,800)))

  }
  data_ell=matrix(unlist(data_ell),ncol = 2)
  data_ell=data.frame(data_ell,data_ell_color*3)
  #data_ell_color=factor(data_ell_color*2)
  colnames(data_ell)=c("y3","y4","color")
  p=p+
    geom_path(data = data_ell, mapping = aes(x = y3, y = y4,color =factor(data_ell$color)))
  print(p)
  return(p)
}

#根据分类点画图
draw_po2=function(number_data,k,data,cla,ell_mu1,ell_sigma)
{
  data_draw=data.frame(data,factor(cla))
  colnames(data_draw)=c("x1","x2","color")
  p=ggplot()
  p=p +
    geom_point(data = data_draw,
               mapping = aes(x = x1, y = x2,color=data_draw$color))
  data_ell=data.frame()
  for(g in levels(data_draw$color)){
    data_ell = rbind(data_ell,
cbind(as.data.frame(with(data_draw[factor(data_draw$color)==g,],ellipse(cor(x1, x2),

scale=c(sd(x1),sd(x2)),

centre=c(mean(x1),mean(x2))))),color=g))
  }
```

```
  colnames(data_ell)=c("x1","x2","color")
  p=p+geom_path(data=data_ell, aes(x=x1, y=x2,color=color), size=1, linetype=1)
  print(p)
}

EM=function(number_data,k,data,data_dim,parameter_pi,parameter_mu,parameter_sigma,standard)
{

  mu_cov=rep(0,k)
  par_cov=0
  error=0.0001
  num_ite=0#迭代次数
  cov1=rep(0,1000)
  for (l in 1:1000) {


    #更新 gamma
    gamma_val=gamma_value(number_data,k,data,parameter_pi,parameter_mu,parameter_sigma)

    #更新 pi
    pi_new=rep(0,k)
    mu_new=matrix(rep(0,data_dim*k),nrow = data_dim)



    for (j in 1:k) {
      sigma_new=list(rep(1,data_dim*data_dim))
      sigma_matrix=matrix(rep(0,data_dim*data_dim),nrow=data_dim)
      for (i in 1:num_data) {
        pi_new[j]=pi_new[j]+gamma_val[i,j]#gamma 按行求和
        mu_new[,j]=mu_new[,j]+gamma_val[i,j]*t(data[i,])
        sigma_matrix=sigma_matrix+
          gamma_val[i,j]*
          (((as.matrix(data[i,]))-as.matrix(parameter_mu[,,j]))%*%
             (data[i,]-t(as.matrix(parameter_mu[,,j]))))
      }


      parameter_pi[,,j]=pi_new[j]/number_data
```

```
      parameter_mu[,,j]=mu_new[,j]/pi_new[j]
      parameter_sigma[,,j]=sigma_matrix/pi_new[j]
      sigma_matrix=matrix(rep(0,data_dim*data_dim),nrow=data_dim)
    }
    cla=Cla(number_data,gamma_val)
    cov1[l+1]=Cov(number_data,k,cla,standard)
    num_ite=num_ite+1
    draw_po(number_data,k,data,cla,parameter_mu,parameter_sigma)
    draw_po2(number_data,k,data,cla,parameter_mu,parameter_sigma)
    if(cov1[l+1]-cov1[l]<error)
    {
      print(num_ite)
      param=list(parameter_pi,parameter_mu,parameter_sigma)
      return(param)
    }
  }
}

KM=function(number_data,k,data,data_dim,parameter_pi,parameter_mu,parameter_sigma,standard)
{

  mu_cov=rep(0,k)
  par_cov=0
  error=0.0001
  num_ite=0#迭代次数
  cov1=rep(0,1000)
  for (l in 1:1000) {


    #更新 gamma
    gamma_val=gamma_value1(number_data,k,data,parameter_mu)

    #更新 pi
    pi_new=rep(0,k)
    mu_new=matrix(rep(0,data_dim*k),nrow = data_dim)

    cla=Cla1(number_data,gamma_val)
```

```
    mu_new=matrix(rep(0,k*data_dim),nrow = data_dim)
    mu_num=rep(0,k)
    for(j in 1:k)
    {
      for(i in 1:num_data)
      {
        if(cla[i]==j)
        {
          mu_new[,j]=mu_new[,j]+t(data[i])
          mu_num[j]=mu_num[j]+1
        }

      }

      parameter_mu[,,j]=mu_new[,j]/mu_num[j]
    }


    cov1[l+1]=Cov(number_data,k,cla,standard)
    num_ite=num_ite+1
    #draw_po3(number_data,k,data,cla,parameter_mu,parameter_sigma)
    draw_po2(number_data,k,data,cla,parameter_mu,parameter_sigma)
    if(abs(cov1[l+1]-cov1[l])<error)
    {
      print(num_ite)
      param=list(parameter_pi,parameter_mu,parameter_sigma)
      return(param)
    }
  }
}

data_dim=2#数据维数
pi1_initial=0.3
pi2_initial=0.3
pi3_initial=1-pi1_initial-pi2_initial
Sigma1_initial <- matrix(c(5,1.5,1.5,1),2,2)
```

```
Sigma2_initial <- matrix(c(10,4,4,6),2,2)
Sigma3_initial <- matrix(c(2,0,0,2),2,2)
Unit_Sigama=matrix(c(1,0,0,1),2,2)

mu1_initial=matrix(c(2,2))
mu2_initial=matrix(c(5,5))
mu3_initial=matrix(c(4,9))

parameter_initial=list(pi1_initial,mu1_initial,Sigma1_initial,pi2_initial,mu2_initial,Sigma2
_initial,
                       pi3_initial,mu3_initial,Sigma3_initial)#参数
parameter_pi=array(c(pi1_initial,pi2_initial,pi3_initial),dim=c(1,1,3))
parameter_mu=array(c(mu1_initial,mu2_initial,mu3_initial),dim = c(2,1,3))
parameter_sigma=array(c(Sigma1_initial,Sigma2_initial,Sigma3_initial),dim = c(2,2,3))

parameter_sigma1=array(c(Unit_Sigama,Unit_Sigama,Unit_Sigama),dim = c(2,2,3))

parameter_mu=array(c(mu1_initial,mu2_initial),dim = c(2,1,2))
parameter_pi=array(c(pi1_initial,pi2_initial,pi3_initial),dim=c(1,1,2))
parameter_sigma=array(c(Sigma1_initial,Sigma2_initial),dim = c(2,2,2))

parameter_sigma1=array(c(Unit_Sigama,Unit_Sigama),dim = c(2,2,2))
k=2#类数量
#parameter_fin=EM(num_data,k,data,data_dim,parameter_pi,parameter_mu,parameter_sigma,standar
d)
parameter_fin=KM(num_data,k,data,data_dim,parameter_pi,parameter_mu,parameter_sigma1,standar
d)
print(parameter_fin)
```