机器学习第四次作业

统计 81　宋程　2184312719

February 28th, 2021

目录

1 引入

我们在之前已经学习了 EM 算法以及机器学习的概率理解，且在上次的作业中已经进行了 EM 算法、K-means 算法的比较，并且应用了 PCA 降维后对实际数据进行了分类，因此我们在此次中将重点转向对无监督学习其实是“有监督的学习”的理解与阐述。

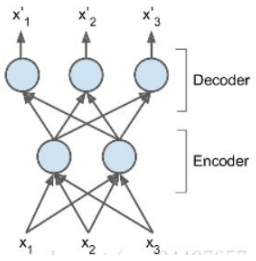
无监督学习主要分为聚类和降维,其相较于监督学习最大的特点在于没有使用标签来进行训练。我们可以通过现实中的学习来类比了解无监督学习，当我们听到许多首歌时，我们可以把他们划成几类，但可能不同的人分类方法又是不同的，有的人可能会根据曲风来进行区分，有的人可能会根据歌词或语言来进行区分，而不同的区分方法也对应着不一样的分类标准，而无监督学习在对数据进行分类时，正是基于一个由算法确定的准则，如常见的 k-means 算法对应的准则为数据点与确定的中心点的欧式距离，而不同的距离以及类别相似度的方法也对应了不同的分类方法、数据的信息多少的判断方法则决定了降维的方法（如 PCA 以方差的大小来度量数据的信息量）。

2 为什么无监督学习是有监督的

2.1 自编码器

我们上面说到实际上无监督的学习也是有监督的,我们可以先通过一个以数据特征为标签的算法——自编码器来进入我们的内容。

自编码器是一种能够通过无监督学习，学到输入数据高效表示的人工神经网络。数据从输入层到隐藏层的过程称为编码（codings），隐藏层的神经元个数则对应于编码之后的数据维度，其维度一般小于输入数据，使得自编码器可用于降维，由于自编码器的输出与输入是相同的，因此数据从隐藏层到输出层的过程称为解码，自编码器就是一个以自身特征为标签，输入等于输出，通过编码再解码来获得数据降维效果的人工神经网络。尽管其以神经网络这样一种有监督学习算法的形式呈现，但由于其使用的输出就是输入，因此其还是可以被认为是无监督学习的一部分,我们可以通过自编码器来分析无监督学习为什么其实是有监督的。



自编码器示意图

（source：https://blog.csdn.net/qq_24407657/article/details/82499677）

我们可以看到对于一个自编码器而言，当我们得到了一组数据后，我们可以如果确定决策函数、表现度量和优化算法即可进行对自编码器的训练，自编码器的决策函数集可以是多种多样的，如线性函数对应了线性降维，我们此处可以暂不考虑决策函数集，表现度量也可以是多种多样的，但核心思想仍然是保证输入与输出的差距尽可能的小，因此我们此处以欧氏距离作为此处的表现度量进行分析，同样自编码的优化算法也不是固定的，我们在此处使用神经网络最常用的梯度下降作为优化算法进行考虑：

自编码器：

$$\text{训练数据集:}\{\mathrm{x}_i, y_i\}$$
$$\text{表现度量:}\sum_{\mathrm{i}}(\mathrm{x}_i - \hat{x}_i)^2$$
$$\text{优化算法:反向传播}$$

我们也可以认为上述过程是最大化后验概率：

$$\mathrm{p}(\hat{X} \mid X;\omega)$$

其中 $\hat{X}, \hat{x}_i$ 均表示相对应的以 x 作为输入的输出值。

我们通过上述描述可以看出，自编码器通过保证输入与输出的距离尽可能的小来完成降维，其过程正是求解最优的网络权重，而其实 k-means 等一系列非监督学习算法也有着类似的表现形式. 我们需要注意的是不论是概率的角度的表现度量还是机器学习角度的表现度量，两者并没有什么区别，因此我们下面的表现度量仅写出我们认为较为直观、易于理解的一个。

2.2 k-means

我们也将 k-means 看成自编码器的形式，对于输入的数据可能是多维的，我们的目的是寻找三个中心，使得所有数据点与其最近的中心的距离的平方和最小，即最小化：

$$\text{表现度量:}\sum_{\mathrm{i}}(\min_{\mathrm{j}}(\mathrm{x}_i - \mathrm{u}_{\mathrm{j}}))^2$$

在这个过程中，我们不再像之前一样改变神经网络的权重，而是通过改变中心 μ_{j} 来达到最小化上述表现度量的目的。

2.3 EM 算法

同样 EM 算法也可以写成上述形式，k-means 是一种 EM 算法的协方差为单位阵、且将软聚类改为硬聚类的特殊情形。其目的是最大化：

$$P(X \mid \theta)$$

在这个过程我们通过改变分布的比例来使得原数据的分布概率最大，我们需要著定义的是 EM 算法往往被认为是生成模型，而自编码器一般被认为是判别模型（成模型是所有变量的全概率模型，而判别模型是在给定观测变量值前提下目标变量条件概率模型。因此生成模型能够用于模拟（即生成）模型中任意变量的分布情况，而判别模型只能根据观测变量得到目标变量的采样。），因此 EM 算法往往能更好表现数据的分布情况。

2.4 PCA

PCA 同样是在给定数据特征的情况进行以自身为标签的无监督学习，其表现度量可以表示为类似于自编码器的形式，但相较于自编码器其对权重、激活函数有了更多的限制，如各主方向正交等。

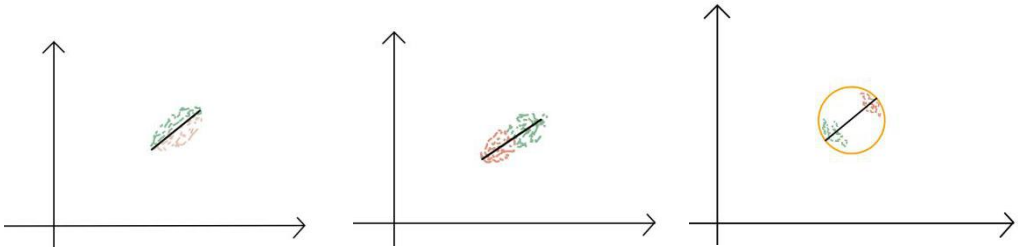
其表现度量为最大化投影方差：

$$\sum_{i}(w^T x_i)^2$$

但其实也是在使得投影距离最小，无论是哪一种表现度量，其实都是在比较数据本身的特征以及经过主成分分析降维后的数据，使得在某种度量之下两者差异较小。

2.4.1 PCA 的一些其他思考

我们通常使用 PCA 降维，并且可以认为其在最大化投影方差。但在进行 PCA 之前我们需要考虑是否需要将数据进行归一化，因为我们需要判断数据的方差是否对整体是有用的信息，我们可以通过一个较为直观的方法来对上述问题进行分析，并对 PCA 处理的是投影方差获得一个直观的理解。



我们可以看到在上述三幅图中，不同颜色的点代表不同的类，黑色的线代表坐标轴和主方向，我们可以看到在图一中，如果我们不进行降维，数据降维后将无法区分，这是因为主方向上的方差可能是由于量纲等原因引起的并没有反映出数据类别上的差异，而在图二中，主方向上的方差正是由类别差异引起的，因此不需要进行归一化，而在图三中，我们发现归一化后的数据如果进行降维并分类，我们考虑的可能是使得类间方差大的方向（如果数据本身能反应类别差异）。

通过上述过程我们可以看出 PCA 实际是在利用方差的差异来完成降维的，而降维可以减少之后分类的计算量，在应对高维数据计算困难、数据特征多维度低、可视化要求等方面有着重要的意义。

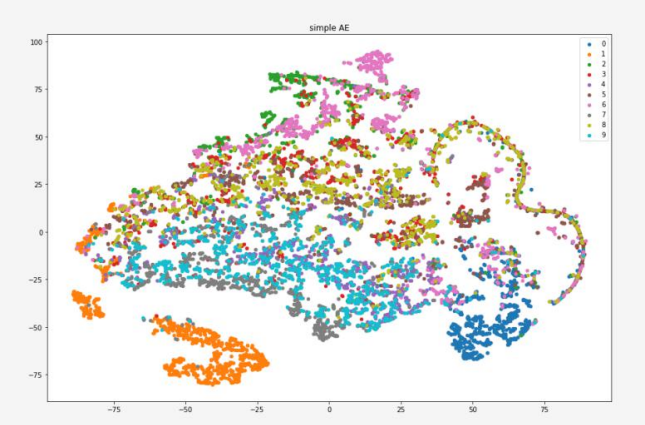
2.5 总结

通过上述分析我们可以发现，无监督学习实际上是利用数据本身的结构（如距离等）来对数据进行区分，以每个训练数据在整个数据中的结构等特征作为衡量标准，根据算法进行聚类或降维的过程。经典的非监督学习往往有一套非常固定的数学上的价值评判标准，这些我们在上述过程中已经说明了，因此我们既可以从无监督学习本身实现过程的角度来理解无监督学习，也可以通过其与有监督学习的相同共通之处来理解无监督学习。

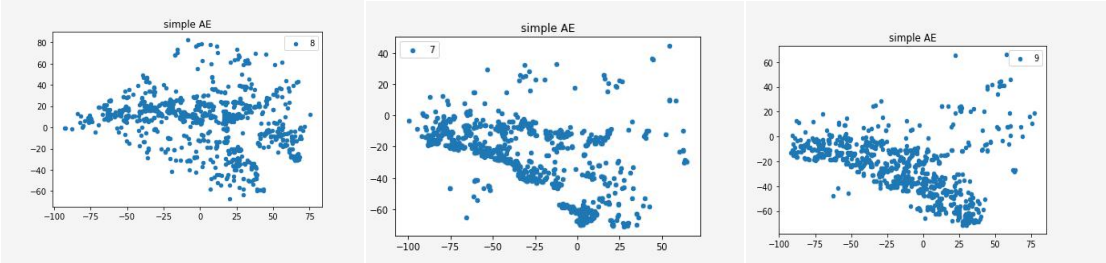
3 实验

3.1 自编码器

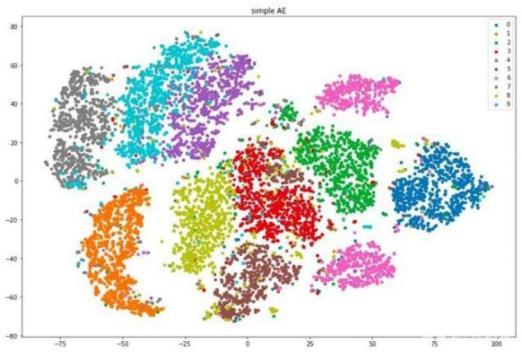
我们将手写数据集（MINIST）的数据进行降维并可视化，我们将 MINIST 的图片转换为 784 维的向量，并使用自编码器将数据降到 32 维度，然后使用 tSNE 方法将其降到两维，以便可视化，我们得到下图：



我们可以看到部分数字的数据点分布如下：



我么可以看到尽管有部分数字被较好的区分了，但也有许多数字重合了，因此尽管自编码器在低维表示下保留了许多信息，但同时也失去了许多信息，对于 MINIST 这样较为复杂的数据集获得二维的表示是较为困难的，但同时我们也要注意到，无论是自编码器还是 tSNE 其获得都是局部极优值，其结果受到初值、学习率等的影响，可能也会得到如下图的结果：



从这个角度我们也可以看出 PCA 与自编码器的一些区别，尽管自编码器可以得到非线性的降维表示，但是其获得的一般是局部极优的结果，而 PCA 一般能获得全局极优，因此 PCA 在高维时计算量可能会较大。

3.2 EM 算法与 k-means

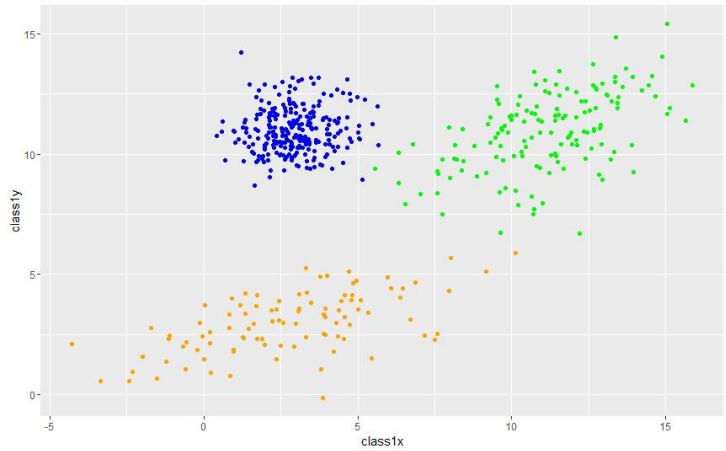
设定参数：

```
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
```

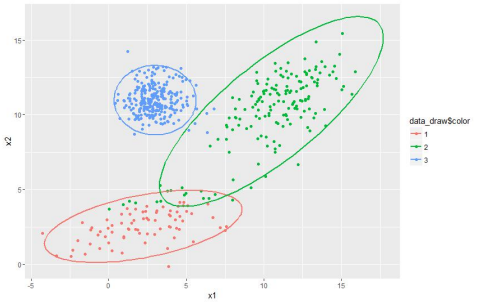
根据如上参数生成 500 个数据点：



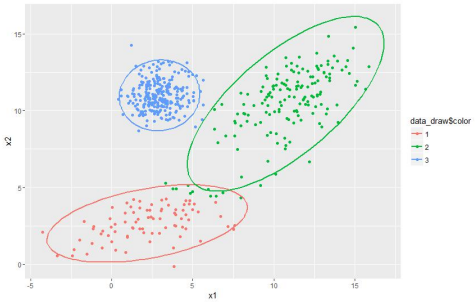
生成数据分布图

3.2.1 EM 算法

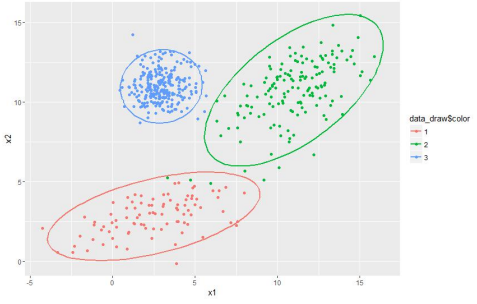
Iteration=1



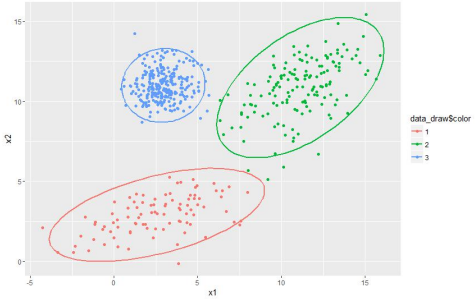
Iteration=2



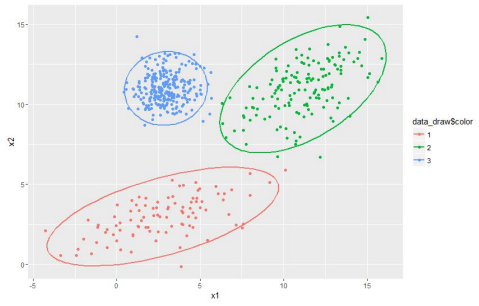
Iteration=3



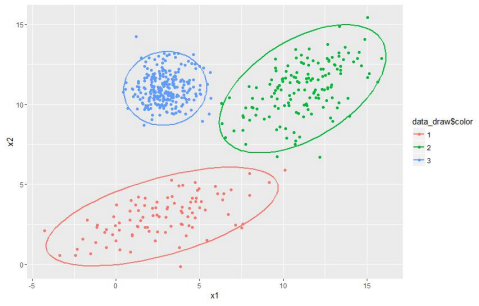
Iteration=4



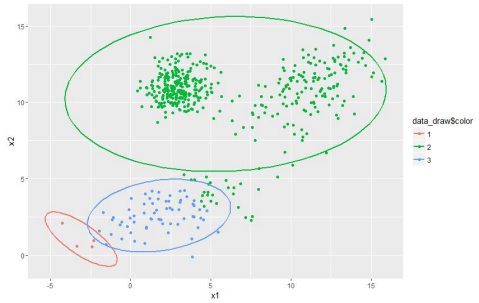
Iteration=6

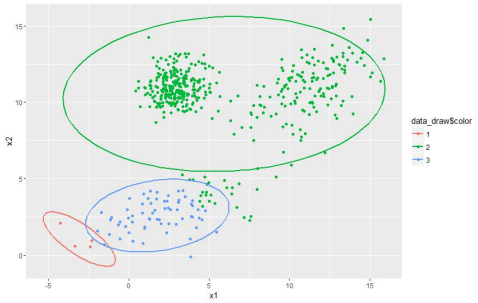


Iteration=7



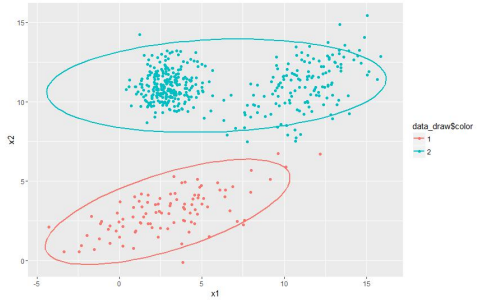
3.2.2 K-means

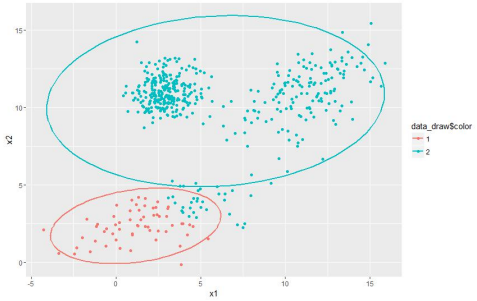




使用 KMEANS 的 NMI 为 0.5411593，且 NMI 不断波动。

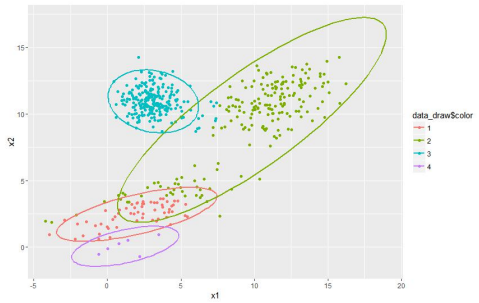
3.2.3 k=2

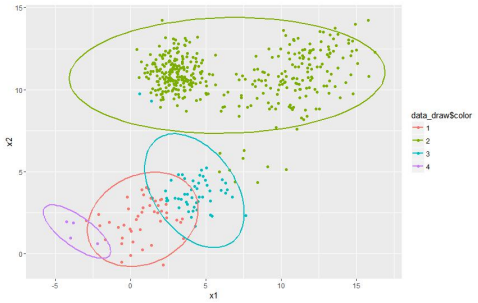




左图为 EM 算法，NMI 为 0.693329； 右图为 K-Means 算法结果，NMI 为 0.463188。

3.2.4 k=4





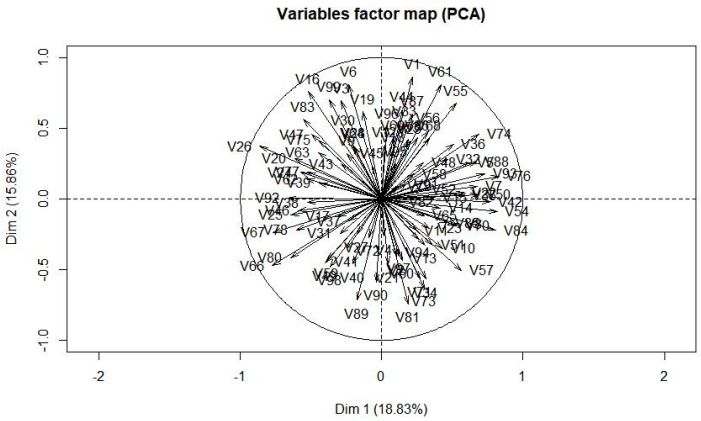
左图为 EM 算法；

右图为 K-Means 算法结果。

4 案列分析

4.1 生物数据的降维与聚类

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：



相应的特征值：

	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（k-means），最终得到如下结果:

实际分类：c（1 2 1 2 1 2 1 2 1 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较高的要求，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。

4.2 手写数据集（MINIST）

我们在上面已经使用自编码器进行了降维并得到了一个不错的可视化结果，我们可以将其与卷积神经网络的结果做一个粗略的比较，我们使用 Le-Net5 卷积神经网络这一监督学习方法对 MINIST 进行分类，在十次训练中平均花费时间为 69.87s，精确度为 98.52%，且训练精度非常稳定，而使用自编码器的降维过程 10batch 需要花费的时间就在 60s 左右，因此哦我们可以看出实际上在大多数时候有监督学习往往能得到比无监督学习更好的结果，当由于标签的成本、降维带来的计算出成本的降低、可视化的要求等原因，无监督学习也在发挥着其不可或缺的作用。

5 总结

通过上述分析与实验，我们认为无监督学习的本质是以数据本身特征作为标签，根据数据本身结构等特点进行的“有监督的学习”，其同样有决策函数与表现度量。对于无监督学习，大多数时候其都是保证输出是输入的某种特定的表示（如低维表示等），在算法定义的度量下，寻找输入与输出差距的最小值，并尝试泛化，其同样可以通过引入正则项的方法来抑制过拟合现象的发生（自编码器与神经网络的关系），其同样有着多种多样的优化方法。

值得一提的是，我们通过实验和对机器学习领域的了解，在大多数时候，如果数据具有标签，有监督学习往往会表现出比无监督学习更好的性能，但是无监督学习有着其特定的应用场景，其在降维减少计算量、提升数据可视化性能、应对无标签数据等有着不可替代的作用，我们也注意到，由于无监督学习的性能限制、有监督学习标签的要求，以及生活中许多实际情况的原因，半监督或弱监督学习愈发受到人们的重视，而研究无监督的学习方法或许也能促进有监督、半监督学习的发展。

无监督学习不仅仅在本质上是有监督的学习，同时无监督学习与有监督学习一起使用的场合也并不罕见，两类学习方法的进步可能也会推动着另一种方法的发展，机器学习的各个领域本身就是不可分割的总体，无监督学习与有监督学习之间有着许多的区别，但理解他们的共同之处或许能给我们带来更多的启发。

6 代码

r

Kmeans、EM 及实际应用

```r
#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
pi4=0.1
pi3=1-pi1-pi2-pi4
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_classify
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_si
gma)
  {
    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(paramete
r_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,]
```

```
77
gamma_val[i,j]=(data_ij-t(parameter_mu[,,j]))%*%t((data_ij-t(parameter_mu[,,j])))
78
79      }
80
81    }
82    return(gamma_val)
83  }
84  #计算分类
85  Cla=function(number_data,gamma_val)
86  {
87    cla=rep(0,number_data)
88    for (i in 1:number_data) {
89      cla[i]=which.max(gamma_val[i,])
90    }
91    return(cla)
92
93  }
94  Cla1=function(number_data,gamma_val)
95  {
96    cla=rep(0,number_data)
97    for (i in 1:number_data) {
98      cla[i]=which.min(gamma_val[i,])
99    }
100   return(cla)
101
102 }
103 #计算 NMI
104 Cov=function(number_data,k,cla,standard)
105 {
106   pro_cla_sta=matrix(rep(0,k*3),nrow = k)#联合概率密度
107   MI_cla_sta=0
108   H_cla=0
109   H_sta=0
110   for(i in 1:number_data)
111   {
112     pro_cla_sta[cla[i],standard[i]]=pro_cla_sta[cla[i],standard[i]]+1
113   }
114   pro_cla_sta=pro_cla_sta*(1/number_data)
115   for (i in 1:3)
116   {
117     H_sta=H_sta-sum(pro_cla_sta[,i])*log(sum(pro_cla_sta[,i]))
118   }
119   for (i in 1:k) {
```

```
    H_cla=H_cla-sum(pro_cla_sta[i,])*log(sum(pro_cla_sta[i,]))

    for (j in 1:3) {
      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$color)))
245   print(p)
246   return(p)
247 }
248 #根据分类点画图
249 draw_po2=function(number_data,k,data,cla,ell_mu1,ell_sigma)
250 {
251   data_draw=data.frame(data,factor(cla))
252   colnames(data_draw)=c("x1","x2","color")
253   p=ggplot()
254   p=p +
255     geom_point(data = data_draw,
256                mapping = aes(x = x1, y = x2,color=data_draw$color))
257   data_ell=data.frame()
258   for(g in levels(data_draw$color)){
259     data_ell = rbind(data_ell,
cbind(as.data.frame(with(data_draw[factor(data_draw$color)==g,],ellipse(cor(x1,
x2),
260
                                  scale=c(sd(x1),sd(x2)),
261
                                  centre=c(mean(x1),mean(x2)))),color=g))
262   }
263   colnames(data_ell)=c("x1","x2","color")
264   p=p+geom_path(data=data_ell, aes(x=x1, y=x2,color=color), size=1, linetype=1)
265   print(p)
266 }
267
    EM=function(number_data,k,data,data_dim,parameter_pi,parameter_mu,parameter_si
gma,standard)
268 {
269
270   mu_cov=rep(0,k)
271   par_cov=0
272   error=0.0001
273   num_ite=0#迭代次数
274   cov1=rep(0,1000)
275   for (l in 1:1000) {
276
277
278     #更新 gamma
279
    gamma_val=gamma_value(number_data,k,data,parameter_pi,parameter_mu,parameter_s
igma)
```

```
280
281       #更新 pi
282       pi_new=rep(0,k)
283       mu_new=matrix(rep(0,data_dim*k),nrow = data_dim)
284
285
286       for (j in 1:k) {
287         sigma_new=list(rep(1,data_dim*data_dim))
288         sigma_matrix=matrix(rep(0,data_dim*data_dim),nrow=data_dim)
289         for (i in 1:num_data) {
290           pi_new[j]=pi_new[j]+gamma_val[i,j]#gamma 按行求和
291           mu_new[,j]=mu_new[,j]+gamma_val[i,j]*t(data[i,])
292           sigma_matrix=sigma_matrix+
293             gamma_val[i,j]*
294             (((as.matrix(data[i,]))-as.matrix(parameter_mu[,,j]))%*%
295                (data[i,]-t(as.matrix(parameter_mu[,,j]))))
296         }
297
298         parameter_pi[,,j]=pi_new[j]/number_data
299         parameter_mu[,,j]=mu_new[,j]/pi_new[j]
300         parameter_sigma[,,j]=sigma_matrix/pi_new[j]
301         sigma_matrix=matrix(rep(0,data_dim*data_dim),nrow=data_dim)
302       }
303       cla=Cla(number_data,gamma_val)
304       cov1[l+1]=Cov(number_data,k,cla,standard)
305       num_ite=num_ite+1
306       draw_po(number_data,k,data,cla,parameter_mu,parameter_sigma)
307       draw_po2(number_data,k,data,cla,parameter_mu,parameter_sigma)
308       if(cov1[l+1]-cov1[l]<error)
309       {
310         print(num_ite)
311         param=list(parameter_pi,parameter_mu,parameter_sigma)
312         return(param)
313       }
314     }
315 }
316
    KM=function(number_data,k,data,data_dim,parameter_pi,parameter_mu,parameter_si
gma,standard)
317 {
318
319   mu_cov=rep(0,k)
320   par_cov=0
321   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,mu3_initial,c(0,0)),dim =
c(2,1,4))

    parameter_pi=array(c(pi1_initial,pi2_initial,pi3_initial-pi4,pi4),dim=c(1,1,4)
)

    parameter_sigma=array(c(Sigma1_initial,Sigma2_initial,Sigma3_initial,Sigma1_in
itial),dim = c(2,2,4))
parameter_sigma1=array(c(Unit_Sigama,Unit_Sigama,Unit_Sigama,Unit_Sigama),dim =
c(2,2,4))
k=4#类数量

    parameter_fin=EM(num_data,k,data,data_dim,parameter_pi,parameter_mu,parameter_
sigma,standard)

    #parameter_fin=KM(num_data,k,data,data_dim,parameter_pi,parameter_mu,parameter
_sigma1,standard)
print(parameter_fin)
#2
#生成数据
library(MASS)
library(ggplot2)
library(mvtnorm)
library(ellipse)
```

```
397 library(FactoMineR)
398 library(factoextra)
399 data=read.table("D:\\学习\\大三下\\生物统计\\第一次作业
\\GSE68849_series_matrix.txt",header=T,
400                 encoding="UTF-8",comment.char = '',quote = "")
401 data=t(data)
402 View(data[1:11,1:100])
403 data_pca <- as.data.frame((data[2:11, 2:100]))
404 colnames(data_pca)=data[1,2:100]
405 matrix1 =as.numeric(as.matrix(data_pca))
406 dim(matrix1) <- dim(data_pca)
407 num_data=10
408 scale(data_pca)
409 res.pca <- PCA(matrix1,scale.unit = TRUE, graph = T)
410 standard=c(0,1,0,1,0,1,0,1,0,1)
411 eig.val <- get_eigenvalue(res.pca)
412 eig.val
413 var<-get_pca_var(res.pca)
414 var$cos2
415 var$contrib
416 md<-res.pca$ind
417 matrix2=md$cos2[1:10,1:2]
418 scale(matrix2)
419 #matrix=md[]
420 #em
421 #计算 gamma 值
422
    gamma_value=function(number_data,k,data,parameter_pi,parameter_mu,parameter_si
gma)
423 {
424   number_data=num_data
425   list1=matrix(rep(0,time=number_data),nrow = number_data)
426   gamma_val=matrix(rep(0,number_data*k),nrow=number_data)
427
428   for (i in 1:number_data) {
429
430     for (j in 1:k) {
431       data_ij=data[i,]
432
list1[i]=list1[i]+parameter_pi[,,j]*dmvnorm(as.vector(data[i,]),as.matrix(paramete
r_mu[,,j]),
433
as.matrix(parameter_sigma[,,j]))
434
```

```
    }

    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*2),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
```

```
519
520   )
521   points$x2[1:400] <- sqrt(((a*b)^2-(b*x)^2)/a^2)
522   points$x2[401:800] <- -sqrt(((a*b)^2-(b*x)^2)/a^2)
523   return(points)
524 }
525 ellipse.general <- function(ell_mu, ell_sig){
526
527   #ell_C=2
528   lambda <- diag(eigen(ell_sig)$values)  # 特征值
529   P <- eigen(ell_sig)$vectors            # 特征向量
530   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
531   X <- t(P%*%t(Y) + rep(ell_mu,800))  # 对坐标旋转位移
532   X <- as.data.frame(X)
533   colnames(X) <- c('x1', 'x2')
534   return(X)
535 }
536 #根据协方差阵画图
537 draw_po=function(number_data,k,data,cla,ell_mu1,ell_sigma)
538 {
539   if(FALSE)
540   {
541     num_class=rep(0,k)
542     data_class=c()
543     for (j in 1:k) {
544       for (i in 1:number_data) {
545
546         if(cla[i]==j)
547         {
548           data_class=append(data_class,data[i,])
549           num_class[j]=num_class[j]+1
550         }
551       }
552
553     }
554     data_class=matrix(data_class,nrow = 2)
555     #data_class=data.frame(data_class)
556     #rownames(data_class)=c("x1","x2")
557     p=ggplot()
558     num_p=1
559     #data_class_draw_list=list()
560     for (i in 1:3) {
561       #data_class_draw=data.frame(data_class[,num_p:(num_p+num_class[i]-1)])
```

```
562         #rownames(data_class_draw)=c("x1","x2")
563         #data_class_draw_list=append(data_class_draw_list,data_class_draw)
564         p=p +
565           geom_point(data =
data.frame(data_class[,num_p:(num_p+num_class[i]-1)],row.names =c("x1","x2")),
566                      mapping = aes(x = x1,  y = x2),
567                      color="blue")
568         num_p=num_p+num_class[i]
569
570       }
571     }
572     data_draw=data.frame(matrix(c(data,cla),ncol=3))
573     colnames(data_draw)=c("x1","x2","color")
574     p=ggplot()
575     p=p +
576       geom_point(data = data_draw,
577                  mapping = aes(x = x1,  y = x2),
578                  color=data_draw$color)
579     data_ell=c()
580     data_ell_color=c()
581
582     for (i in 1:k) {
583
    data_ell=append(data_ell,ellipse.general(as.matrix(ell_mu1[,,i]),as.matrix(ell
_sigma[,,i])))
584       data_ell_color=append(data_ell_color,c(rep(i,800)))
585
586     }
587     data_ell=matrix(unlist(data_ell),ncol = 2)
588     data_ell=data.frame(data_ell,data_ell_color*3)
589     #data_ell_color=factor(data_ell_color*2)
590     colnames(data_ell)=c("y3","y4","color")
591     p=p+
592       geom_path(data = data_ell,  mapping = aes(x = y3,  y = y4,color
=factor(data_ell$color)))
593     print(p)
594     return(p)
595 }
596 draw_po3=function(number_data,k,data,cla,ell_mu1,ell_sigma)
597 {
598     data_draw=data.frame(matrix(c(data,cla),ncol=3))
599     colnames(data_draw)=c("x1","x2","color")
600     p=ggplot()
601     p=p +
```

```
602     geom_point(data = data_draw,
603                mapping = aes(x = x1, y = x2),
604                color=data_draw$color)
605   data_ell=c()
606   data_ell_color=c()
607
608   for (i in 1:k) {
609     data_ell=append(data_ell,ellipse(ell_sigma[1,2,i],
610                                      c(ell_sigma[1,1,i],ell_sigma[2,2,i]),
611                                      ell_mu1[,,i]))
612     data_ell_color=append(data_ell_color,c(rep(i,800)))
613
614   }
615   data_ell=matrix(unlist(data_ell),ncol = 2)
616   data_ell=data.frame(data_ell,data_ell_color*3)
617   #data_ell_color=factor(data_ell_color*2)
618   colnames(data_ell)=c("y3","y4","color")
619   p=p+
620     geom_path(data = data_ell, mapping = aes(x = y3, y = y4,color
=factor(data_ell$color)))
621   print(p)
622   return(p)
623 }
624 #根据分类点画图
625 draw_po2=function(number_data,k,data,cla,ell_mu1,ell_sigma)
626 {
627   data_draw=data.frame(data,factor(cla))
628   colnames(data_draw)=c("x1","x2","color")
629   p=ggplot()
630   p=p +
631     geom_point(data = data_draw,
632                mapping = aes(x = x1, y = x2,color=data_draw$color))
633   data_ell=data.frame()
634   for(g in levels(data_draw$color)){
635     data_ell = rbind(data_ell,
cbind(as.data.frame(with(data_draw[factor(data_draw$color)==g,],ellipse(cor(x1,
x2),
636
                                scale=c(sd(x1),sd(x2)),
637
                                centre=c(mean(x1),mean(x2)))),color=g))
638   }
639   colnames(data_ell)=c("x1","x2","color")
640   p=p+geom_path(data=data_ell, aes(x=x1, y=x2,color=color), size=1, linetype=1)
```

```
641   print(p)
642 }
643
    EM=function(number_data,k,data,data_dim,parameter_pi,parameter_mu,parameter_si
gma,standard)
644 {
645
646   mu_cov=rep(0,k)
647   par_cov=0
648   error=0.0001
649   num_ite=0#迭代次数
650   cov1=rep(0,1000)
651   for (l in 1:1000) {
652
653
654     #更新 gamma
655
    gamma_val=gamma_value(number_data,k,data,parameter_pi,parameter_mu,parameter_s
igma)
656
657     #更新 pi
658     pi_new=rep(0,k)
659     mu_new=matrix(rep(0,data_dim*k),nrow = data_dim)
660
661
662     for (j in 1:k) {
663       sigma_new=list(rep(1,data_dim*data_dim))
664       sigma_matrix=matrix(rep(0,data_dim*data_dim),nrow=data_dim)
665       for (i in 1:num_data) {
666         pi_new[j]=pi_new[j]+gamma_val[i,j]#gamma 按行求和
667         mu_new[,j]=mu_new[,j]+gamma_val[i,j]*t(data[i,])
668         sigma_matrix=sigma_matrix+
669           gamma_val[i,j]*
670           (((as.matrix(data[i,]))-as.matrix(parameter_mu[,,j]))%*%
671              (data[i,]-t(as.matrix(parameter_mu[,,j]))))
672       }
673
674       parameter_pi[,,j]=pi_new[j]/number_data
675       parameter_mu[,,j]=mu_new[,j]/pi_new[j]
676       parameter_sigma[,,j]=sigma_matrix/pi_new[j]
677       sigma_matrix=matrix(rep(0,data_dim*data_dim),nrow=data_dim)
678     }
679     cla=Cla(number_data,gamma_val)
680     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_si
gma,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 1:10) {


      #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
          }
```

```
723
724         }
725
726         parameter_mu[,,j]=mu_new[,j]/mu_num[j]
727       }
728
729
730       cov1[l+1]=Cov(number_data,k,cla,standard)
731       num_ite=num_ite+1
732       #draw_po3(number_data,k,data,cla,parameter_mu,parameter_sigma)
733       #draw_po2(number_data,k,data,cla,parameter_mu,parameter_sigma)
734
735     }
736     print(num_ite)
737     param=list(parameter_pi,parameter_mu,parameter_sigma)
738     return(cla)
739 }
740 data_dim=5#数据维数
741 pi1_initial=0.3
742 pi2_initial=0.3
743 pi3_initial=1-pi1_initial-pi2_initial
744 Sigma1_initial <- matrix(c(5,1.5,1.5,1),2,2)
745 Sigma2_initial <- matrix(c(10,4,4,6),2,2)
746 Sigma3_initial <- matrix(c(2,0,0,2),2,2)
747 Unit_Sigama=matrix(c(1,0,0,1),2,2)
748 mu1_initial=matrix(c(0.02,0.0005,0.00002,0.00004,0.02))
749 mu2_initial=matrix(c(0,0.0002,0.0001,0.2,0.001))
750 mu3_initial=matrix(c(4,9))
751 parameter_mu=array(c(mu1_initial,mu2_initial),dim = c(5,1,2))
752 parameter_pi=array(c(pi1_initial,pi2_initial,pi3_initial),dim=c(1,1,2))
753 parameter_sigma=array(c(Sigma1_initial,Sigma2_initial),dim = c(2,2,2))
754 parameter_sigma1=array(c(Unit_Sigama,Unit_Sigama),dim = c(2,2,2))
755 k=2#类数量
756
    #parameter_fin=EM(num_data,k,data,data_dim,parameter_pi,parameter_mu,parameter
_sigma,standard)
757parameter_fin=KM(num_data,k,md$cos2,data_dim,parameter_pi,parameter_mu,parameter
_sigma1,standard)
758 print(parameter_fin)
```

Python

#autoencoder

（sourcehttps://baijiahao.baidu.com/s?id=1621473711897747154&wfr=spider&for=pc）

```
from keras.models import Model

from keras import optimizers
from keras.layers import Input,Dense
def autoencoder(dim,act):
  input_tensor= Input(shape=(28*28,))
  encoder=Dense(dim,activation=act,name='encode')(input_tensor)
  decoder = Dense(28*28, activation='sigmoid',name='decode')(encoder)
  model = Model(input_tensor,decoder)
  model.compile(optimizer=optimizers.Adam(), loss='binary_crossentropy')
  return model

from keras.datasets import mnist

from sklearn.preprocessing import StandardScaler
(X_train,y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000,28*28)
X_train = X_train.astype('float32') /255
X_test = X_test.reshape(10000, 28*28)
X_test = X_test.astype('float32') /255
scale=StandardScaler().fit(X_train)
X_train=scale.transform(X_train)
X_test =scale.transform(X_test)

AE=autoencoder(32,'relu')

his=AE.fit(X_train,X_train,batch_size=256,verbose=1,epochs=10)

from sklearn import manifold

import matplotlib.pyplot as plt
tsne =manifold.TSNE(n_components=2, init='random', random_state=0)
X_tsne =tsne.fit_transform(model_out)
plt.figure(figsize=(15,10))
for i in range(10):
  plt.scatter(X_tsne[y_test==i,0],X_tsne[y_test==i,1],s=20,label='%d'%i)
  plt.title('simple AE')
  plt.legend()
```

#LeNet

```
from __future__ import print_function
import keras

from keras.datasets import mnist
```

```
from keras.layers import Input, Dense, Dropout, Flatten, add
from keras.layers import Conv2D, Activation, MaxPooling2D, AveragePooling2D
from keras import backend as K
from keras.callbacks import ModelCheckpoint
import tensorflow as tf
from keras.models import Model
from keras.utils import plot_model

batch_size = 128
num_classes = 10
epochs = 12  # 训练次数=12*60000
model_path = 'modeldir.mnist'
# input image dimensions
img_rows, img_cols = 28, 28

flag = "train"  # one of "train" or "eval"


def res_block(x, channels, i):
    if i == 1:  # 第二个block
        strides = (1, 1)
        x_add = x
    else:  # 第一个block
        strides = (2, 2)
        # x_add 是对原输入的bottleneck操作
        x_add = Conv2D(channels,
                       kernel_size=(3, 3),
                       activation='relu',
                       padding='same',
                       strides=strides)(x)

    x = Conv2D(channels,
               kernel_size=(3, 3),
               activation='relu',
               padding='same')(x)
    x = Conv2D(channels,
               kernel_size=(3, 3),
               padding='same',
               strides=strides)(x)
    x = add([x, x_add])
    Activation(K.relu)(x)
    return x


```

```
def build_model(input_shape):
    inpt = Input(shape=input_shape)
    # conv_1  28x28→28x28x16
    x = Conv2D(16,
               kernel_size=(7, 7),
               activation='relu',
               input_shape=input_shape,
               padding='same'
               )(inpt)
    # x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2))(x)
    # conv_2 28x28x16→14x14x16
    for i in range(2):
        x = res_block(x, 16, i)
    # conv_3 14x14x16→7x7x32
    for i in range(2):
        x = res_block(x, 32, i)

    x = AveragePooling2D(pool_size=(7, 7))(x)
    x = Flatten()(x)
    x = Dense(num_classes, activation='softmax')(x)
    # Construct the model.
    model = Model(inputs=inpt, outputs=x)
    plot_model(model, to_file='resnet.png')
    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adadelta(),
                  metrics=['accuracy'])
    return model


def train(model, x_train, y_train, x_test, y_test):
    checkpoint = ModelCheckpoint(model_path,
                                 monitor='val_loss',  # 保存模型的路径。
                                 verbose=1,  # 详细信息模式，0 或者 1 。
                                 save_best_only=True,
                                 save_weights_only=False,
                                 mode='auto',
                                 period=1  # 每个检查点之间的间隔（训练轮数）
                                 )
    history = model.fit(x_train, y_train,
                        batch_size=batch_size,  # 每次梯度更新的样本数
                        epochs=epochs,  # 训练模型迭代轮次。一个轮次是在整个训练集上的一轮迭代
                        verbose=2,  # verbose: 0, 1 或 2。日志显示模式。 0 = 安静模式, 1 = 进度条, 2 = 每轮一行。
```

```
                          validation_data=(x_test, y_test),
                          callbacks=[checkpoint])
    return history


def test(model, x_test, y_test):
    model.load_weights(model_path)
    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    score = model.evaluate(x_test, y_test, verbose=0)
    print('Test loss:', score[0])
    print('Test accuracy:', score[1])

    # def main(_):
    # the data, split between train and test sets
    (x_train, y_train), (x_test, y_test) = mnist.load_data()  # 数据集会默认下
载到C:\Users\xxx\.keras\datasets 中
    if K.image_data_format() == 'channels_first':
        x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
        x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
        input_shape = (1, img_rows, img_cols)
    else:
        x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
        x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
        input_shape = (img_rows, img_cols, 1)

    x_train = x_train.astype('float32')
    x_test = x_test.astype('float32')
    # 将像素范围缩至0到1
    x_train /= 255
    x_test /= 255
    # convert class vectors to binary class matrices
    y_train = keras.utils.to_categorical(y_train, num_classes)  # 60000个
    y_test = keras.utils.to_categorical(y_test, num_classes)  # 10000个

    _model = build_model(input_shape)
    # x_train(60000, 28, 28, 1),  y_train(60000,)

    if flag == "train":
        hisrory = train(_model, x_train, y_train, x_test, y_test)
    elif flag == "eval":
        test(_model, x_test, y_test)
    print(history.history.key)
# main(_)
```

```
133 # if __name__ == '__main__':
```