hw5

顾格非 3210103528

2

1.

$$f_m(X) = \Pr(Y = m|X) = \frac{e^{Z_m+c}}{\sum_{\ell=0}^{9} e^{Z_\ell+c}}$$
$$= \frac{e^c * e^{Z_m}}{e^c * \sum_{\ell=0}^{9} e^{Z_\ell}} = \frac{e^{Z_m}}{\sum_{\ell=0}^{9} e^{Z_\ell}}$$

所以前后的值没有变。

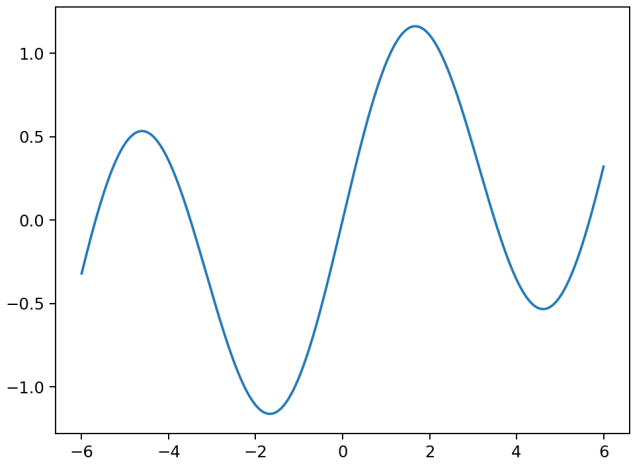
2.

$$\Pr(Y = k|X = x) = \frac{e^{(c+\beta_{k0})+(c+\beta_{k1})x_1+\cdots+(c+\beta_{kp})x_p}}{\sum_{l=1}^{K} e^{(c+\beta_{l0})+(c+\beta_{l1})x_1+\cdots+(c+\beta_{lp})x_p}}$$
$$= \frac{e^c * e^{\beta_{k0}+\beta_{k1}x_1+\cdots+\beta_{kp}x_p}}{e^c * \sum_{l=1}^{K} e^{\beta_{l0}+\beta_{l1}x_1+\cdots+\beta_{lp}x_p}}$$
$$= \frac{e^{\beta_{k0}+\beta_{k1}x_1+\cdots+\beta_{kp}x_p}}{\sum_{l=1}^{K} e^{\beta_{l0}+\beta_{l1}x_1+\cdots+\beta_{lp}x_p}}$$

所以前后的prediction没有变。

6

1. 如图所示：

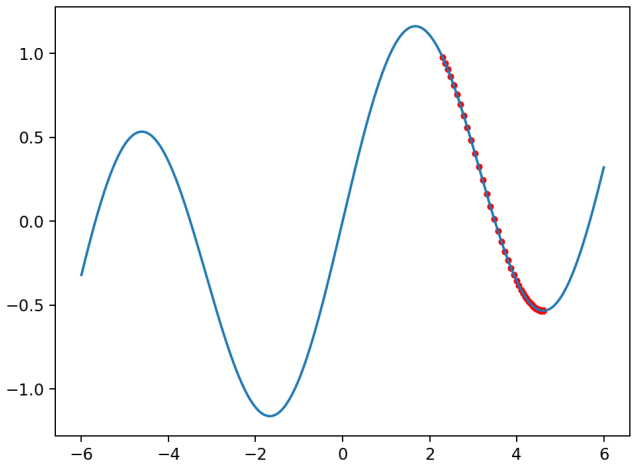


2. 导函数$R'(\beta) = cos(\beta) + 0.1$
3. 编写如下的python代码，用$x_{k+1} = x_k - 0.1 * \nabla f$更新参数，其中当$|\nabla f| < 0.001$时终止算法：

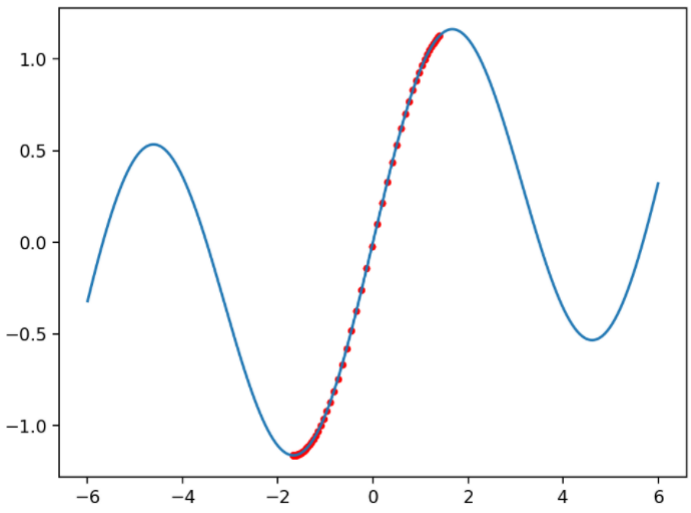
```
import numpy as np
import matplotlib.pyplot as plt

beta = np.linspace(-6,6,1000)
R = np.array([np.sin(x)+x/10 for x in beta])
```

```
def f(x):
    return np.sin(x)+x/10
def derive_f(x):
    return np.cos(x)+0.1
x_0 = 2.3

x = x_0
ans = []
while (np.abs(derive_f(x))>0.001):
    ans.append(x)
    x = x - 0.1*derive_f(x)
print(x)
Y = np.array([f(x) for x in ans])
plt.plot(beta,R)
plt.scatter(ans,Y,c = 'r',s = 10)
plt.show()
```

得到最终的$x* = -4.6112187$，迭代过程如图所示：



4. 用同样的方式迭代，得到最终的$x* = -1.670004$，迭代过程如图所示，可以看到两个初值经过迭代都能收敛到local minimum。



1

1.

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}\left(x_{ij}-x_{i'j}\right)^2=\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}\left[\left(x_{ij}-\bar{x}_{kj}\right)-\left(x_{i'j}-\bar{x}_{kj}\right)\right]^2$$

$$=\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}\left[\left(x_{ij}-\bar{x}_{kj}\right)^2-2\left(x_{ij}-\bar{x}_{kj}\right)\left(x_{i'j}-\bar{x}_{kj}\right)+\left(x_{i'j}-\bar{x}_{kj}\right)^2\right]$$

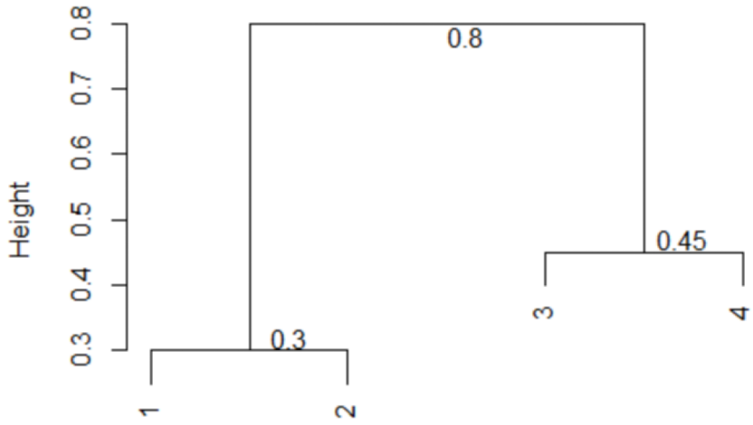
$$=\frac{|C_k|}{|C_k|}\sum_{i\in C_k}\sum_{j=1}^{p}\left(x_{ij}-\bar{x}_{kj}\right)^2+\frac{|C_k|}{|C_k|}\sum_{i'\in C_k}\sum_{j=1}^{p}\left(x_{i'j}-\bar{x}_{kj}\right)^2-\frac{2}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}\left(x_{ij}-\bar{x}_{kj}\right)\left(x_{i'j}-\bar{x}_{kj}\right)$$

$$=2\sum_{i\in C_k}\sum_{j=1}^{p}\left(x_{ij}-\bar{x}_{kj}\right)^2$$

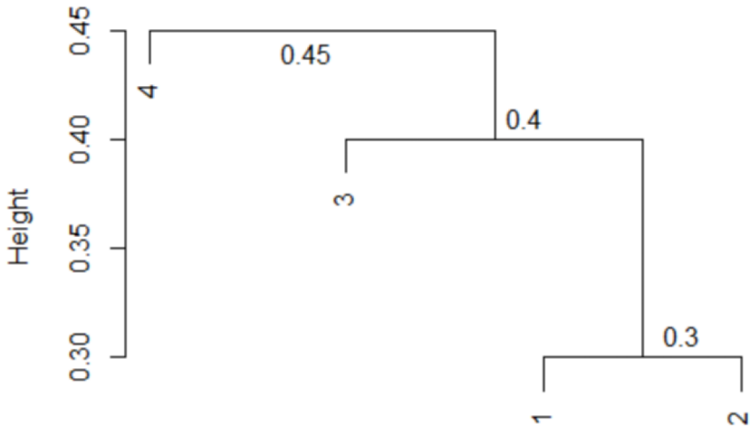
2. 上面的式子告诉我们：减小每个类中每个观测与类中心欧式距离的平方之和，相当于减小每个类中各个观测之间所有成对的欧式距离的平方之和除以类中观测总数。所以说，kmeans的迭代把新的中心选到所有点的均值处，相当于减少了（12.17）的值。

2

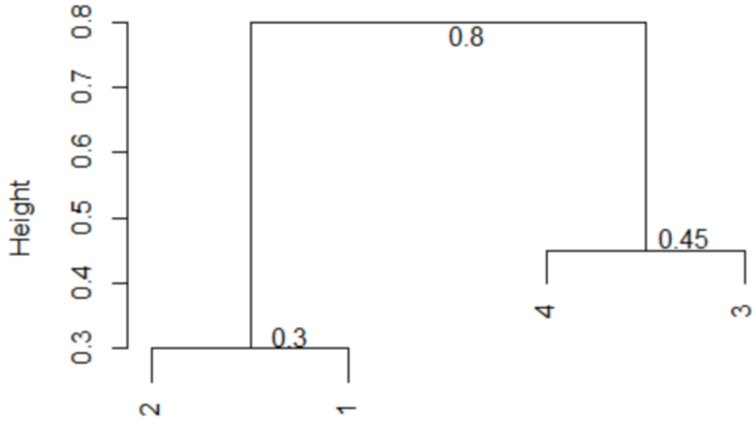
1. 用最长距离法得到的图：



2. 用最短距离法：

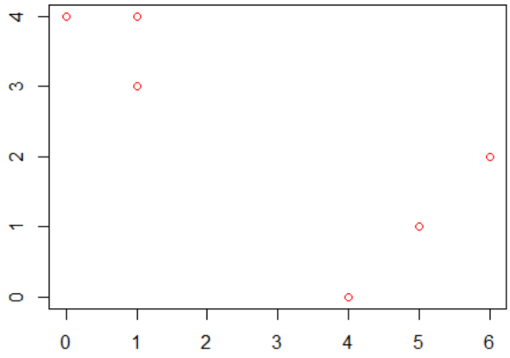


3. 根据1中的图，1,2为一类，3,4 为一类
4. 根据2中的图，1,2,3 为一类，4为一类。
5. 这张图跟1中的图是等价的，只不过交换了一些将要汇合的两个点的位置：



3

```
X = matrix(c(1, 1, 0, 5, 6, 4, 4, 3, 4, 1, 2, 0), ncol = 2)
plot(X[,1],X[,2],xlab = "",ylab = "",col = "red")
```




```
> label = sample(2, nrow(X), replace=T)
> label
[1] 2 1 2 2 2 1
```

计算两个center：

```
> center1 = c(mean(X[label == 1, 1]), mean(X[label == 1, 2]))
> center2 = c(mean(X[label == 2, 1]), mean(X[label == 2, 2]))
> center1
[1] 2.5 1.5
> center2
[1] 3.00 2.75
```

把每个点按欧式距离分到最近的center上，得到各自的label：

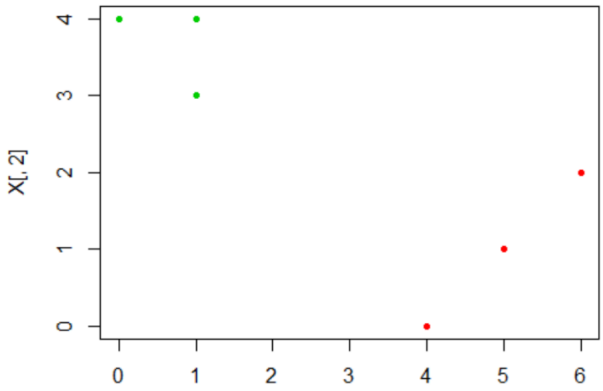
```
> Eculidean = function(x,y){
+   return(sqrt((x[1]-y[1])^2+(x[2]-y[2])^2))
+ }
> k2 = function(X,center1,center2){
+   label = rep(NA, nrow(X))
+   for (i in 1:nrow(X)) {
+     if(Eculidean(X[i,],center1) <= Eculidean(X[i,],center2)){
```

```
+       label[i] = 1
+     }
+     else{
+       label[i] = 2
+     }
+   }
+   return(label)
+ }
> k2(X,center1,center2)
[1] 2 2 2 1 2 1
```

不断重复上述过程，知道结果不再改变：

```
> finallabel = rep(0, 6)
> while (!all(finallabel == label)) {
+   finallabel = label
+   center1 = c(mean(X[label == 1, 1]), mean(X[label == 1, 2]))
+   center2 = c(mean(X[label == 2, 1]), mean(X[label == 2, 2]))
+   label = k2(X, center1, center2)
+ }
>
> label
[1] 2 2 2 1 1 1
```

最后给每个点按照label上色：



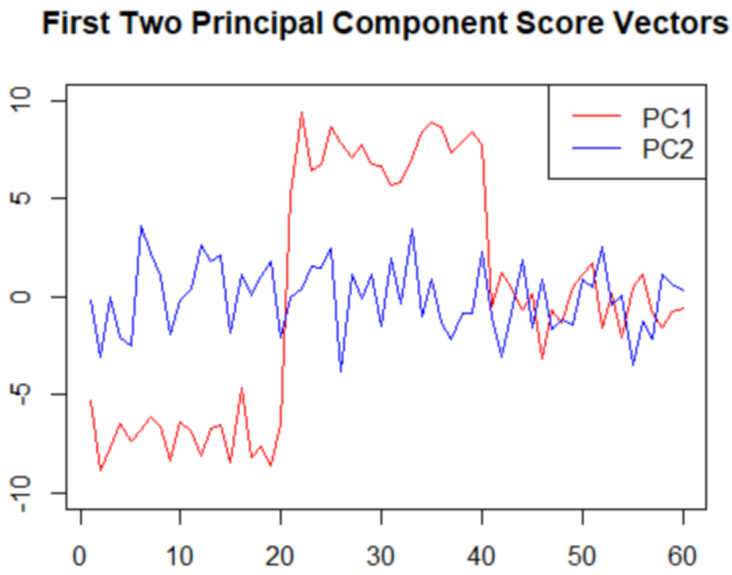
10

1. 用`rnorm`产生数据，接下来，将x的前20行第1列和第2列的元素都加上了5，而x的第21到40行第1列和第2列的元素都减去了5，相当于在这些元素上手动加上偏移量。

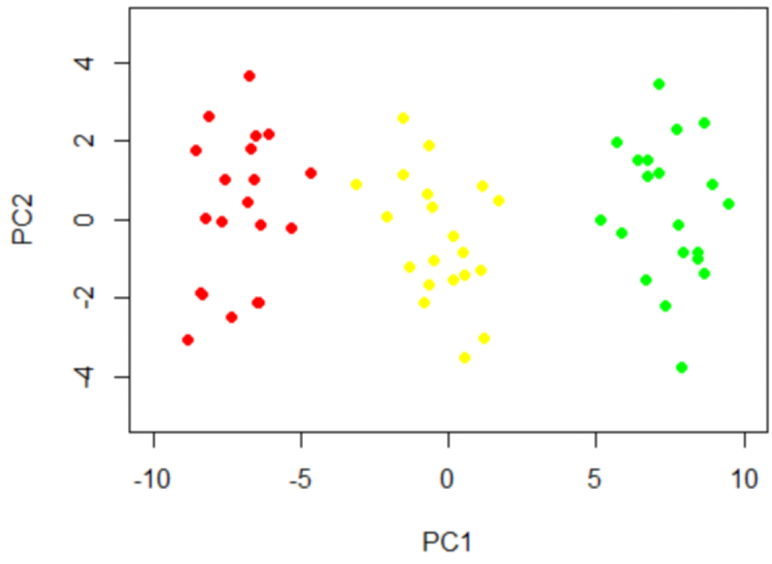
```
> set.seed (2)
> x=matrix (rnorm (60*50) , ncol = 50)
> x[1:20 ,1]= x[1:20 ,1]+5
> x[1:20 ,2]= x[1:20 ,2]+5
> x[21:40 ,1]=x[21:40 ,1] -5
> x[21:40 ,2]=x[21:40 ,2] -5
```

2. 通过`prcomp`函数对矩阵`x`进行主成分析，得到如下的结果：

```
> pr.out = prcomp(x)
> plot(c(1:60),pr.out$x[,1],col = "red",type = "l",xlab = "",ylab = "",main = "First
Two Principal Component Score Vectors",ylim = c(-10,10))
> lines(c(1:60),pr.out$x[,2],col = "blue")
> legend("topright",c("PC1","PC2"),col = c("red","blue"),cex = 1, lty = 1)
```




```
> plot(pr.out$x[1:20,1:2], col="red", xlab="PC1", ylab="PC2", pch=19,xlim =
c(-10,10),ylim = c(-5,5))
> points(pr.out$x[21:40,1:2], col="green",pch=19)
> points(pr.out$x[41:60,1:2], col="yellow",pch=19)
```



3.
```
> km.out$cluster
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[59] 3 3
> table(km.out$cluster, c(rep(1,20), rep(2,20), rep(3,20)))
     1  2  3
  1  0 20  0
  2 20  0  0
  3  0  0 20
```

因为K-means方法会随机为类进行编号，所以K-means得到的第1类，第2类和第3类分别对应原数据的第2类，第1类和第3类，所以说K-means的聚类效果很好，成功将原数据聚成3类没有误差。

4. 两类时：

```
> km.out = kmeans(x, 2, nstart=20)
> km.out$cluster
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
[59] 1 1
> table(km.out$cluster, c(rep(1,20), rep(2,20), rep(3,20)))

     1  2  3
  1 20  0 19
  2  0 20  1
```

第1类和第2类的聚类结果是完全正确的。对于第3类，K-means将19个第3类的数据聚为第1类，将1个第3类数据聚为第2类。

5. K=4:

```
> km.out = kmeans(x, 4, nstart=20)
> km.out$cluster
 [1] 1 1 1 1 1 2 1 2 1 1 2 2 2 2 1 2 1 2 2 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
[59] 4 4
> table(km.out$cluster, c(rep(1,20), rep(2,20), rep(3,20)))

     1  2  3
  1 11  0  0
  2  9  0  0
  3  0 20  0
  4  0  0 20
```

对于第1类数据，K-means将11个第1类的数据聚为第1类，将9个第1类数据聚为第2类。K-means得到的第3类和第4类分别对应原数据的第2类和第3类,分别对应原来的2和3。

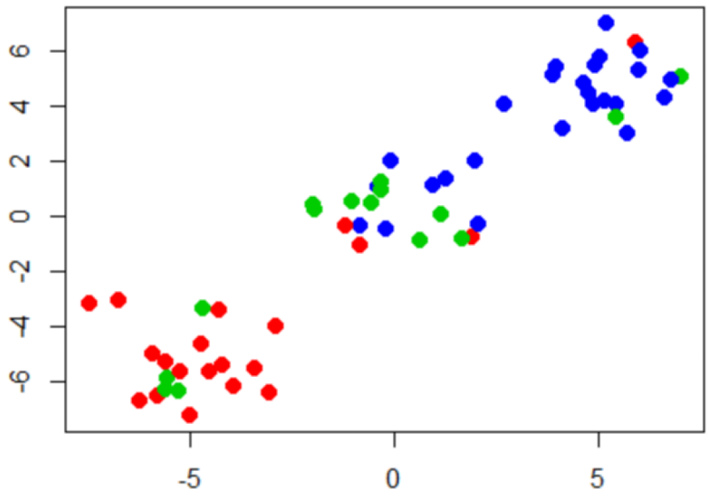
6.

```
> km.out = kmeans(pr.out$x[,1:2], 3, nstart=20)
> km.out$cluster
 [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[59] 1 1
> table(km.out$cluster, c(rep(1,20), rep(2,20), rep(3,20)))

     1  2  3
  1  0  0 20
  2  0 20  0
  3 20  0  0
```

可以看到分类结果完全正确。十分优秀。

7.

```
> km.out = kmeans(scale(x), 3, nstart=20)
> km.out$cluster
 [1] 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 1 3 3 2 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 1 1 2 1
3 2 3 1 2 3 2 3 3 1 1 3 2 3 2 2 2 3
[59] 2 2
> table(km.out$cluster, c(rep(1,20), rep(2,20), rep(3,20)))

     1  2  3
  1  1 16  3
  2  2  4  9
  3 17  0  8
> plot(x, col = (km.out$cluster +1), xlab ="", ylab="", pch =20, cex =2)
```



标准化之后数据的距离变得更加接近，所以Kmeans的聚类效果变差。