第五次作业参考答案

第一种：

Homework5

50

于颖 统计1602 1303160223

2019年5月7日

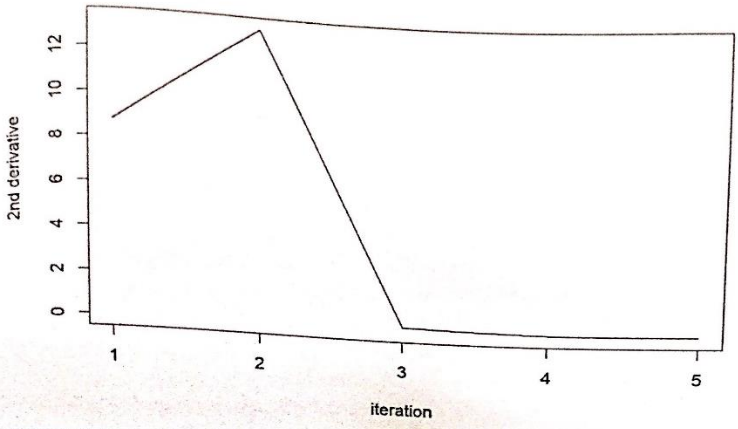
我们将把一个逻辑回归模型与人脸识别算法的测试相关。实验使用识别算法将每个人的第一张图像（称为探针）与剩余的2143张图像之一进行匹配。理想情况下，匹配同一个人的另一个图像（称为目标）。一个成功的匹配产生了yi=1的响应，而与任何其他人的匹配产生了yi=0的响应。预测变量Eyediff测量探针图像与其对应目标之间的眼睛区域像素强度的绝对差异。

第一问、编写您自己的牛顿-拉斐逊算法以适应逻辑回归。初步猜测β (0) = (β (0) 0 , β(0) 1) > = (0.96, 0)T，并设置收敛公差ε=10^5.绘制一个图，显示每次迭代时导数的值

```
load("C:/Users/Dell/Desktop/facerecognition.RData")
y=as.matrix(data$match)
table(y)
```

```
## y
##   0   1
## 282 760
```

```
x=matrix(c(rep(1,1042),data$eyediff),nr=1042,nc=2)
Beta=matrix(c(0.96,0),nr=2,nc=1)
w=matrix(0,nr=1042,nc=1042)
pi=matrix(0,nr=1042,nc=1)
a=NULL
b1=NULL
b2=NULL
j=0
i=1
while(i<=1042){
pi=1/(1+exp(-x%*%Beta))
w[i,i]=(pi[i]*(1-pi[i]))
i=i+1
}
loga=t(x)%*%(y-pi)
logb=t(x)%*%w%*%x
Beta=Beta+solve((logb))%*%loga
h=sum(loga^2)
epsilo = 1e-5
while (sqrt(h)>epsilo){
a=c(a,sqrt(h))
b1=c(b1,Beta[1])
b2=c(b2,Beta[2])
pi=1/(1+exp(-x%*%Beta))
loga=t(x)%*%(y-pi)
logb=t(x)%*%w%*%x
Beta=Beta+solve((logb))%*%loga
i=1
while(i<=1042){
w[i,i]=(pi[i]*(1-pi[i]))
i=i+1
}
j=j+1
h=sum(loga^2)
}
plot(1:j,a,xlab = "iteration",ylab = "2nd derivative",'l')
```

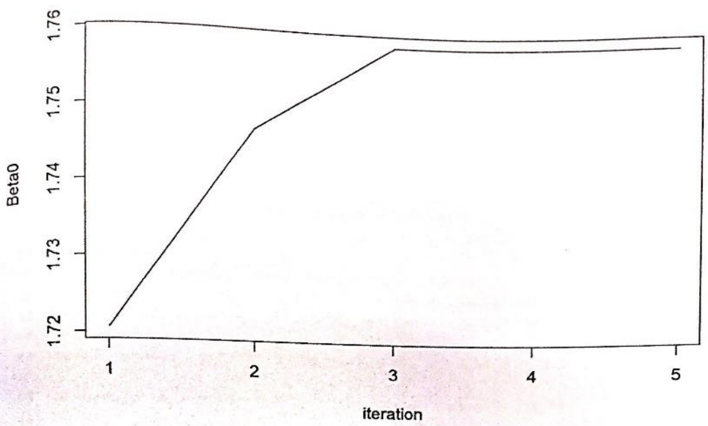



```
Beta
```

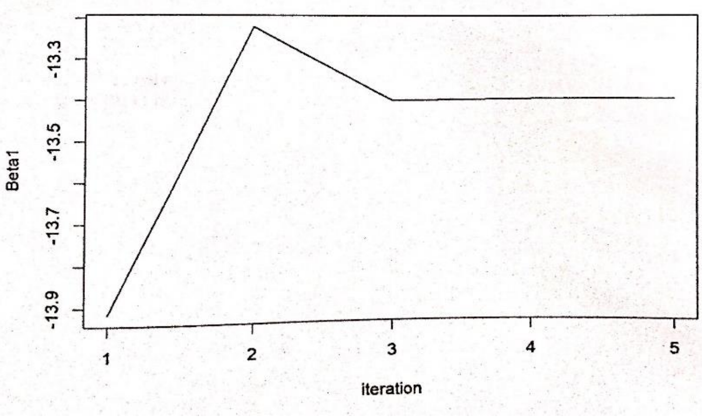
```
##              [,1]
## [1,]    1.758701
## [2,] -13.400040
```

第二问、绘制一个图，显示β（t）如何移动直到收敛。使用r中的estimate from glm（ ）函数检查结果（setting family="binomial"）。

```
plot(1:j,b1,xlab = "iteration",ylab = 'Beta0','l')
```




```
plot(1:j,b2,xlab = "iteration",ylab = 'Beta1','l')
```



```
#library(AER)

fit.full=glm(formula=data$match~data$eyediff,family=binomial())
summary(fit.full)
```

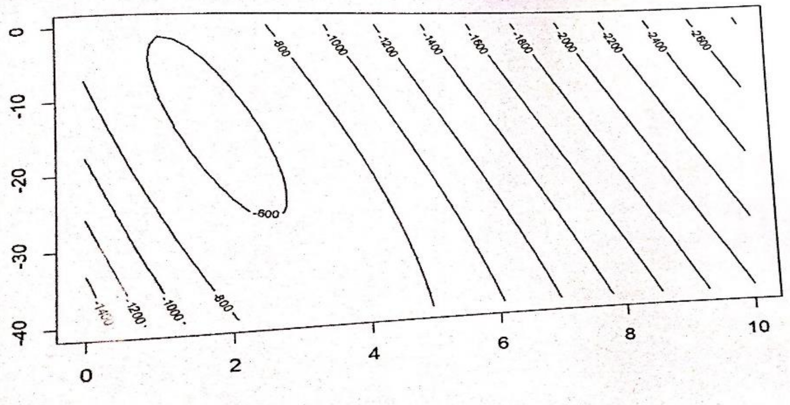
```
##
## Call:
## glm(formula = data$match ~ data$eyediff, family = binomial())
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9562  -0.9227   0.6372   0.7630   1.7173
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)     1.7587     0.1183  14.863   <2e-16 ***
## data$eyediff  -13.4000     1.5502  -8.644   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1216.8  on 1041  degrees of freedom
## Residual deviance: 1134.7  on 1040  degrees of freedom
## AIC: 1138.7
##
## Number of Fisher Scoring iterations: 4
```

```
Beta
```

```
##             [,1]
## [1,]    1.758701
## [2,]  -13.400040
```

第三问、等高线图

```
X=seq(0,10,length=50)
Y=seq(-40,0,length=50)
Z=matrix(nr=50,nc=50)
 f <- function(p,q){
  beta1=c(p,q)
  Pi=1/(1+exp(-x%*%beta1))
  Log=y*(x%*%beta1)+log(1-Pi)
  sum(Log)
}
for (i in 1:50) {
  for (j in 1:50) {
    Z[i,j]=f(X[i],Y[j])
  }
}
contour(X,Y,Z)
```



第二种：

2. Homework 5

50

加载数据集并准备解释与被解释变量：

```
load("D:/Sta_test.RData")
facedata<-data
nrow<-dim(facedata)[1]
# extract data for regression
y<-as.matrix(facedata$match)
x<-as.matrix(facedata$eyediff)
x0<-matrix(1,nrow,1)
x<-cbind(x,x0)
```

模型初始化和参数初值：

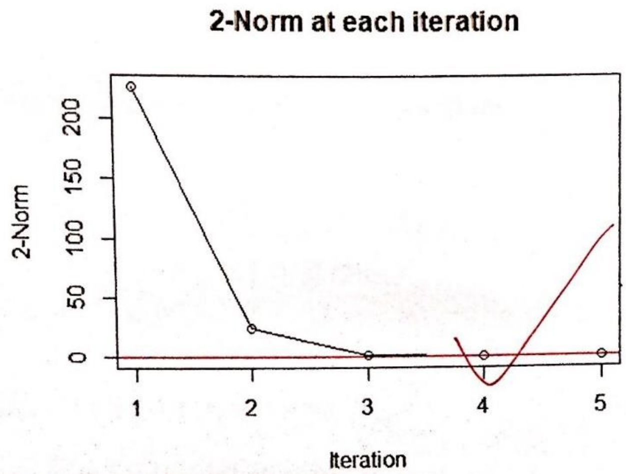
```
# model input parameters
niter<-100
tol<-1e-5
h2<-matrix(0,niter,1)
beta<-matrix(0,2,niter+1)

# initial guess for beta
beta[,1]<-matrix(c(0.96,0))
```

迭代过程：

```
# Newton Raphson
for (i in 1:niter){
    pi<-1/(1+exp(-x%*%beta[,i]))
    z<-y-pi
      h<-t(x)%*%z
      h2[i]<-(h[1,]^2+h[2,]^2)^0.5
    if(h2[i]>tol){
        w<-pi%*%t(1-pi)
            w<-diag(diag(w))
        dh<-(-solve(t(x)%*%w%*%x))
        beta[,i+1]<-beta[,i]-dh%*%h
    }
    else{
        break
    }
}
```

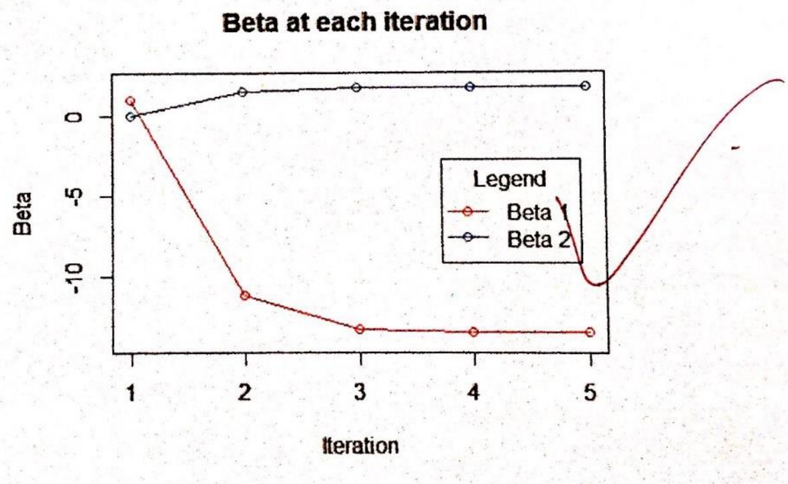
每一步长的二范数：

```
# plot 2 Norm at each iteration
plot(h2[1:i],type="o",main="2-Norm at each iteration",xlab="Iteration",
ylab="2-Norm")
abline(h=tol,lwd=1,col="red")
```



每一步长的参数估计值：

```
# plot beta value at each iteration
plot(beta[1,1:i],type="o",main="Beta at each iteration",xlab="Iteration
",ylab="Beta",ylim=c(-14,2),col="red")
par(new=TRUE)
plot(beta[2,1:i],type="o",main="Beta at each iteration",xlab="Iteration
",ylab="Beta",ylim=c(-14,2),col="blue")
legend("right", inset=.05, title="Legend",c("Beta 1","Beta 2"),lty=c(1,
 1), pch=c(1, 1), col=c("red", "blue"))
```



参数移动绘图（与 contour plot 一起显示）：

```
# plot beta position change during optimization
plot(beta[1,1:i],beta[2,1:i],type="o",main="Beta locations",xlab="Beta 1",ylab="Beta 2",col="red",xlim=c(-13,1),ylim=c(0,2))
```

与广义线性回归模型比较回归参数:

```
# use glm function to check results
glm<-glm(match~eyediff,family = binomial(link = "logit"),data = data)
glm
```

```
##
## Call:  glm(formula = match ~ eyediff, family = binomial(link = "logi
t"),
##     data = data)
##
## Coefficients:
## (Intercept)      eyediff
##       1.759      -13.400
##
## Degrees of Freedom: 1041 Total (i.e. Null);  1040 Residual
## Null Deviance:       1217
## Residual Deviance: 1135  AIC: 1139
```

```
summary(glm)
```

```
##
## Call:
## glm(formula = match ~ eyediff, family = binomial(link = "logit"),
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9562  -0.9227   0.6372   0.7630   1.7173
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.7587     0.1183  14.863   <2e-16 ***
## eyediff     -13.4000     1.5502  -8.644   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1216.8  on 1041  degrees of freedom
## Residual deviance: 1134.7  on 1040  degrees of freedom
## AIC: 1138.7
##
## Number of Fisher Scoring iterations: 4
```

从上述结果可以看出，广义线性回归模型的参数估计为（1.76，-13.40），与迭代得出的结果十分相近。

绘制 log⁡ (L(β))的密度图作为底图：

```
# underlying contour plot for Log(L)
beta1<-matrix(seq(-13,1,0.1))
beta2<-matrix(seq(0,2,0.1))
L<-matrix(0,dim(beta1)[1],dim(beta2)[1])
for (k in 1:dim(beta1)[1]){
for (l in 1:dim(beta2)[1]){
beta0<-rbind(beta1[k],beta2[l])
pi<-1/(1+exp(-x%*%beta0))
L[k,l]<-sum(y*x%*%beta0+log(1-pi))
}
}
par(new=TRUE)
contour(beta1,beta2,L,nlevels=20,xlim=c(-13,1),ylim=c(0,2))
```