**Chapter 5: Resampling Methods**

**1**



对求导并令其等于0得到



整理后得到



**2(a)**

易知，。

**2(b)**

每次抽样是独立的，易知。

**2(c)**

等价于对于n次抽样，每次都没有抽到第j个观测的概率。易知。

**2(d)**

。

**2(e)**

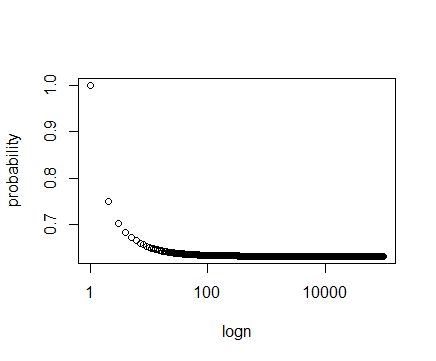
。

**2(f)**

。

**2(g)**

plot(n,p,log = "x",xlab = "logn",ylab = "probability")



随着n变大，概率趋近于。

**2(h)**

> plot(n,p,log = "x",xlab = "logn",ylab = "probability")

> store=rep (NA , 10000)

> for (i in 1:10000) {

+ store[i]=sum(sample (1:100 , rep = TRUE)==4) >0

+ }

> mean(store)

[1] 0.6413

与概率值很接近。

**3(a)**

k-Fold Cross-Validation involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining k − 1 folds. The mean squared error, MSE1, is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error, MSE1, MSE2, . . . , MSEk. The k-fold CV estimate is computed by averaging these values,



**3(b)**

The Validation Set Approach:

Advantages: conceptually simple and is easy to implement.

Disadvantages:

1. The validation estimate of the test error rate can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.

2. In the validation approach, only a subset of the observations—those that are included in the training set rather than in the validation set—are used to fit the model. Since statistical methods tend to perform worse when trained on fewer observations, this suggests that the validation set error rate may tend to overestimate the test error rate for the model fit on the entire data set.

LOOCV:

It is not hard to see that LOOCV is a special case of k-fold CV in which k is set to equal n.

Advantages:

1. it has far less bias. LOOCV repeatedly fit the statistical learning method using training sets that contain n − 1 observations, almost as many as are in the entire data set. This is in contrast to the validation set approach, in which the training set is typically around half the size of the original data set. Consequently, the LOOCV approach tends not to overestimate the test error rate as much as the validation set approach does.

2. in contrast to the validation approach which will yield different results when applied repeatedly due to randomness in the training/validation set splits, performing LOOCV multiple times will always yield the same results: there is no randomness in the training/validation set splits.

Disadvantages:

LOOCV has the potential to be expensive to implement, since the model has to be fit n times. This can be very time consuming if n is large, and if each individual model is slow to fit.

**4**

If we suppose using some statistical learning method to make a prediction for the response Y for a particular value of the predictor X we might estimate the standard deviation of our prediction by using the bootstrap approach. We randomly select n observations from the data set in order to produce a bootstrap data set, Z∗1. The sampling is performed with replacement, which means that the same observation can occur more than once in the bootstrap data set. In this example, Z∗1 contains the third observation twice, the first observation once, and no instances of the second observation. Note that if an observation is contained in Z∗1, then both its X and Y values are included. We can use Z∗1 to produce a new bootstrap estimate for α, which we call . This procedure is repeated B times for some large value of B, in order to produce B different bootstrap data sets, Z∗1, Z∗2, . . . , Z∗B, and B corresponding α estimates, , , . . . , . We can compute the standard error of these bootstrap estimates using the formula

.

This serves as an estimate of the standard error of ˆα estimated from the original data set.

**5(a)**

> set.seed(1)

> glm.fit5=glm(default~income+balance,data = default,family = binomial)

**5(b)**

> f5=function(){

+ train = sample(dim(Default)[1], dim(Default)[1]/2)

+ Default.test=Default[-train,]

+ default.test=default[-train]

+ glm.fit = glm(default ~ income + balance, data = Default, family = binomial, subset = train)

+ glm.probs=predict(glm.fit,Default.test,type = "response")

+ glm.pred=rep("No",dim(Default)[1]/2)

+ glm.pred[glm.probs>0.5]="Yes"

+ return(mean(glm.pred != default.test))

+ }

**5(c)**

> f5()

[1] 0.0252

> f5()

[1] 0.0246

> f5()

[1] 0.0266

**5(d)**

> train = sample(dim(Default)[1], dim(Default)[1]/2)

> Default.test=Default[-train,]

> default.test=default[-train]

> glm.fit5\_ = glm(default ~ income + balance+student, data = Default, family = binomial, subset = train)

> glm.probs=predict(glm.fit5\_,Default.test,type = "response")

> glm.pred=rep("No",dim(Default)[1]/2)

> glm.pred[glm.probs>0.5]="Yes"

> mean(glm.pred != default.test)

[1] 0.028

The dummy variable for student doesn’t lead to a reduction in the test error rate.

**6(a)**

> set.seed(1)

> glm.fit6 = glm(default ~ income + balance, data = Default, family = binomial)

> summary(glm.fit6)

Call:

glm(formula = default ~ income + balance, family = binomial,

data = Default)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.4725 -0.1444 -0.0574 -0.0211 3.7245

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -1.154e+01 4.348e-01 -26.545 < 2e-16 \*\*\*

income 2.081e-05 4.985e-06 4.174 2.99e-05 \*\*\*

balance 5.647e-03 2.274e-04 24.836 < 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: 2920.6 on 9999 degrees of freedom

Residual deviance: 1579.0 on 9997 degrees of freedom

AIC: 1585

Number of Fisher Scoring iterations: 8

**6(b)**

> boot.fn = function(data, index){

+ return(coef(glm(default ~ income + balance,

data = data, family = binomial, subset = index)))

+ }

**6(c)**

> library(boot)

> boot(Default, boot.fn, 1000)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:

boot(data = Default, statistic = boot.fn, R = 1000)

Bootstrap Statistics :

original bias std. error

t1\* -1.154047e+01 -8.008379e-03 4.239273e-01

t2\* 2.080898e-05 5.870933e-08 4.582525e-06

t3\* 5.647103e-03 2.299970e-06 2.267955e-04

**6(d)**

对于斜率及income的系数的标准误差，logistic回归略大于bootstrap方法，balance系数的标准误差，logistic回归略小于bootstrap方法。

**7(a)**

> attach(Weekly)

> set.seed(1)

> glm.fit7 = glm(Direction ~ Lag1 + Lag2, data = Weekly, family = binomial)

**7(b)**

glm.fit7\_ = glm(Direction ~ Lag1 + Lag2, data = Weekly[-1, ], family = binomial)

**7(c)**

> predict(glm.fit7\_,Weekly[1, ],type = "response")

1

0.5713923

预测第一个观测是“Up”，预测错了。

**7(d)**

> for (i in 1:dim(Weekly)[1]) {

+ glm.fit = glm(Direction ~ Lag1 + Lag2, data = Weekly[-i, ], family = binomial)

+ up = predict.glm(glm.fit, Weekly[i, ], type = "response") > 0.5

+ up\_true = Weekly[i, ]$Direction == "Up"

+ if (up != up\_true)

+ count[i] = 1

+ }

> sum(count)

[1] 490

有490个观测，Direction是“Down”被错误地预测为“Up”。

**7(e)**

> mean(count)

[1] 0.4499541

test error rate约为45.0%。

**8(a)**

> set.seed(1)

> y=rnorm(100)

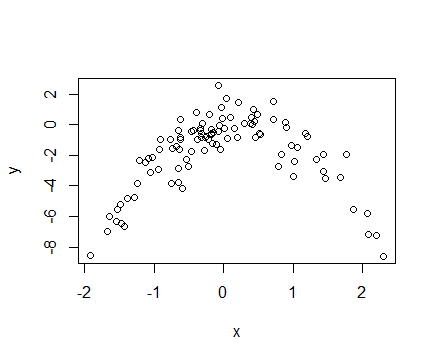
> x=rnorm(100)

> y=x-2\*x^2+rnorm(100)

，。

**8(b)**

> plot(x,y)



符合二次函数的特征，x取值大约在区间[-2,2]，y取值大约在区间[-8,2]。

**8(c)**

> set.seed(100)

> dataxy=data.frame(x,y)

> cv.error=rep(0,4)

> for (i in 1:4) {

+ glm.fit8=glm(y~poly(x,i),data = dataxy)

+ cv.error[i]=cv.glm(dataxy,glm.fit8)$delta[1]

+ }

> cv.error

[1] 5.890979 1.086596 1.102585 1.114772

**8(d)**

> set.seed(1000)

> dataxy=data.frame(x,y)

> cv.error=rep(0,4)

> for (i in 1:4) {

+ glm.fit8=glm(y~poly(x,i),data = dataxy)

+ cv.error[i]=cv.glm(dataxy,glm.fit8)$delta[1]

+ }

> cv.error

[1] 5.890979 1.086596 1.102585 1.114772

结果是一样的。对每个模型，LOOCV都循环了n次，训练集和测试集的分割方式相同。

**8(e)**

第二个模型LOOCV误差最小，因为原本的模型就是二次的。

**8(f)**

> glm.fit8\_1=glm(y~poly(x,1),data = dataxy)

> summary(glm.fit8\_1)

Call:

glm(formula = y ~ poly(x, 1), data = dataxy)

Deviance Residuals:

Min 1Q Median 3Q Max

-7.3469 -0.9275 0.8028 1.5608 4.3974

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.8277 0.2362 -7.737 9.18e-12 \*\*\*

poly(x, 1) 2.3164 2.3622 0.981 0.329

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 5.580018)

Null deviance: 552.21 on 99 degrees of freedom

Residual deviance: 546.84 on 98 degrees of freedom

AIC: 459.69

Number of Fisher Scoring iterations: 2

> glm.fit8\_2=glm(y~poly(x,2),data = dataxy)

> summary(glm.fit8\_2)

Call:

glm(formula = y ~ poly(x, 2), data = dataxy)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.89884 -0.53765 0.04135 0.61490 2.73607

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.8277 0.1032 -17.704 <2e-16 \*\*\*

poly(x, 2)1 2.3164 1.0324 2.244 0.0271 \*

poly(x, 2)2 -21.0586 1.0324 -20.399 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 1.06575)

Null deviance: 552.21 on 99 degrees of freedom

Residual deviance: 103.38 on 97 degrees of freedom

AIC: 295.11

Number of Fisher Scoring iterations: 2

> glm.fit8\_3=glm(y~poly(x,3),data = dataxy)

> summary(glm.fit8\_3)

Call:

glm(formula = y ~ poly(x, 3), data = dataxy)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.87250 -0.53881 0.02862 0.59383 2.74350

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.8277 0.1037 -17.621 <2e-16 \*\*\*

poly(x, 3)1 2.3164 1.0372 2.233 0.0279 \*

poly(x, 3)2 -21.0586 1.0372 -20.302 <2e-16 \*\*\*

poly(x, 3)3 -0.3048 1.0372 -0.294 0.7695

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 1.075883)

Null deviance: 552.21 on 99 degrees of freedom

Residual deviance: 103.28 on 96 degrees of freedom

AIC: 297.02

Number of Fisher Scoring iterations: 2

> glm.fit8\_4=glm(y~poly(x,4),data = dataxy)

> summary(glm.fit8\_4)

Call:

glm(formula = y ~ poly(x, 4), data = dataxy)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.8914 -0.5244 0.0749 0.5932 2.7796

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.8277 0.1041 -17.549 <2e-16 \*\*\*

poly(x, 4)1 2.3164 1.0415 2.224 0.0285 \*

poly(x, 4)2 -21.0586 1.0415 -20.220 <2e-16 \*\*\*

poly(x, 4)3 -0.3048 1.0415 -0.293 0.7704

poly(x, 4)4 -0.4926 1.0415 -0.473 0.6373

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 1.084654)

Null deviance: 552.21 on 99 degrees of freedom

Residual deviance: 103.04 on 95 degrees of freedom

AIC: 298.78

Number of Fisher Scoring iterations: 2

只有一次和二次项的系数统计上是显著的，这和cross-validation得到的结论一致。

**9(a)**

> miu=mean(medv)

> miu

[1] 22.53281

**9(b)**

> miu.se=sd(medv)/sqrt(length(medv))

> miu.se

[1] 0.4088611

**9(c)**

> boot.fn = function(data, index){

+ return(mean(data[index]))

+ }

> library(boot)

> bootstrap\_ = boot(medv, boot.fn, 1000)

> bootstrap\_

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:

boot(data = medv, statistic = boot.fn, R = 1000)

Bootstrap Statistics :

original bias std. error

t1\* 22.53281 -0.01598083 0.4131197

**9(d)**

> t.test(medv)

One Sample t-test

data: medv

t = 55.111, df = 505, p-value < 2.2e-16

alternative hypothesis: true mean is not equal to 0

95 percent confidence interval:

21.72953 23.33608

sample estimates:

mean of x

22.53281

> c(mean(medv) - 2 \* 0.4131197, mean(medv) + 2 \* 0.4131197)

[1] 21.70657 23.35905

**9(e)**

> miu\_med=median(medv)

> miu\_med

[1] 21.2

**9(f)**

> boot.fn\_=function(data, index){

+ return(median(data[index]))

+ }

> boot(medv, boot.fn\_, 1000)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:

boot(data = medv, statistic = boot.fn\_, R = 1000)

Bootstrap Statistics :

original bias std. error

t1\* 21.2 -0.01105 0.3754415

**9(g)**

> miu\_0.1=quantile(medv,probs = c(0.1))

> miu\_0.1

10%

12.75

**9(h)**

> boot.fn\_\_=function(data,index){

+ return(quantile(data[index],probs = c(0.1)))

+ }

> boot(medv, boot.fn\_\_, 1000)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:

boot(data = medv, statistic = boot.fn\_\_, R = 1000)

Bootstrap Statistics :

original bias std. error

t1\* 12.75 -0.0031 0.5280523