R Tutorial for Statistical Learning

Qidi Peng

7: Resampling by validation methods

Code	Comments	Results

#Validation set approach
install.packages("ISLR");
library(ISLR);
set.seed(1);
	
	Install training database "ISLR".
	Load the package.
	To "freeze" the random variable in the sequel, so that one can obtain the same results when rerunning the simulation following its.	

train=sample(392,196);	In validation set approach, we randomly equally likely select half elements from the sample of size n=392. We use sampling without replacement by default.	
lm.fit=lm(mpg~horsepower,data=Auto,subset=train);	Make a linear regression with Y=mpg, X=horsepower, using half-size of the observed data (training set).	
lm.fit;	Print the least squares estimates.	Call: lm(formula = mpg ~ horsepower, data = Auto, subset = train) Coefficients: (Intercept) horsepower 40.4885 -0.1609
mean((Auto$mpg-predict(lm.fit,Auto))[-train]^2);	Determine the MSE of the 196 training data in the validation set.	[1] 22.50892

set.seed(2);
train=sample(392,196);
lm.fit=lm(mpg~horsepower,data=Auto,subset=train);
mean((Auto$mpg-predict(lm.fit,Auto))[-train]^2);
	If we choose a different training set, we will get a different MSE for validation set.	
		
		[1] 23.29559

# K-fold cross-validation 	
install.packages("boot");
 	
library(boot);
set.seed(1);
 	
fix(cv.error);
function (k,q)
{m=numeric(q);
for(i in
1:q){glm.fit=glm(mpg~poly(horsepower,i),data=Auto);
m[i]=cv.glm(Auto,glm.fit,K=k)$delta[1]};m
}
cv.error(10,10);
	
	Install the generalized linear regression function package "boot".
	Load the package.
	To freeze the following random variable.
	
	Create a K-fold cross-validation test MSE function.
	k=the number of partitions; q denotes the polynomial order of the polynomial regression:
	$Y=beta_0+beta_1X^1+beta_2X^2+...+beta_qX^q$.
	
	Print the results of the 10-fold CV testing MSE corresponding to the polynomial fits of orders 1 to 10.
		
		[1] 93.66531
		97.71557 97.17491
		99.46710 99.87657
		100.68672 99.78368
		[8] 100.02531
		99.91709 101.65488

# Leave-one-out cross validation 	
# One can use the function cv.error(k,q) with k=n. 	
#Another way is to use the delta statistics in the function
#cv.glm().
glm.fit=glm(mpg~horsepower,data=Auto);
cv.glm(Auto,glm.fit)$delta;
	
	Simple linear regression.
	Delta is a vector of length two. They correspond to the testing MSE of LOOCV. The first component is the raw cross-validation estimate of prediction error. The second component is the adjusted cross-validation estimate. The adjustment is designed to compensate for the bias introduced by not using leave-one-out cross-validation. The 2 components are identical up to 2 decimal places.
		
		[1] 24.23151 24.23114

8: Bootstrap

Codes	Comments	Results
# The bootstrap method can be used to estimate the		

accuracy of #the “estimates”.
# In the first example, we do Exercise 2 in HW3. More 	
#precisely, we use the bootstrap method to estimate alpha.
 	
fix(alpha.fn);
function (data,index)
{X=data$X[index];Y=data$Y[index];
(var(Y)-cov(X,Y))/(var(X)+var(Y)-2*cov(X,Y))
}
set.seed(1);
alpha.fn(Portfolio,1:100);
 	
# The bootstrap method can be applied as: 	
 	
alpha.fn(Portfolio,sample(100,100,replace=T));
 	
boot(Portfolio,alpha.fn,R=1000);
	Create a function to calculate the estimate of alpha, which maximizes Var(alpha X+(1-alpha)Y)). X,Y are observed values of 2 portfolios in the data set “data”. Index is the subset we select from the original sample.
	
	Use the data set “Portfolio” in “ISLR”. Take the first 100 training data (in fact we have taken all training data) to estimate alpha.
	
	Produce 1 bootstrap sample to estimate alpha.
	
	The boot() function estimates the returns of the function alpha.fn, using the original sample “Portfolio” and produce R=1000 bootstrap samples. The final result shows the estimate of alpha is 0.5758 and SE(alpha) is 0.0925.
		[1] 0.5758321
		
		[1] 0.5313198
		
		ORDINARY NONPARAMETRIC BOOTSTRAP
		Call:
		boot(data = Portfolio, statistic = alpha.fn, R = 1000)
		Bootstrap Statistics :
		original
		bias std. error
		t1* 0.5758321
		0.001237822
		0.09252988

# Second example: estimating the accuracy of a linear 	
#regression model. We use the bootstrap method to assess
#the variability of the estimates $beta_0$ and $beta_1$ in a simple
#linear regression. The SE of these 2 estimates will be
#estimated by using bootstrap. As alpha.fn, we have to
#create a function which returns the estimates of $beta_0$ and
#$beta_1$.
fix(boot.fn);
function (data,index)
{coef(lm(mpg~horsepower,data=data,subset=index))
}
boot.fn(Auto,1:392);
	Create a function which returns the least squares estimates of $beta_0$ and $beta_1$.
	
	Test this function with all training data.
		(Intercept)
		horsepower

set.seed(1); boot.fn(Auto,sample(392,392,replace=T));	A bootstrap estimate from 1 bootstrap sample.	39.9358610 -0.1578447 (Intercept) horsepower 39.3809602 -0.1531129
boot(Auto,boot.fn,R=1000);	The estimation results by 1000 bootstrap samples. We see SE(beta$_0$)=0.8512 and SE(beta$_1$)=0.0073.	ORDINARY NONPARAMETRIC BOOTSTRAP Call: boot(data = Auto, statistic = boot.fn, R = 1000) Bootstrap Statistics : original bias std. error t1* 39.9358610 0.0326353594 0.851224457 t2* -0.1578447 -0.0002322221 0.007347569
#Now we estimate the standard errors of the estimates using classical methods (introduced in CH1): summary(lm(mpg~horsepower,data=Auto))$coef;	This method suggests SE(beta$_0$)=0.7175 and SE(beta$_1$)=0.0064, which are different from the ones from bootstrap. We claim that the SE by bootstrap are more accurate, since it does not rely on any extra assumptions (no need to be normal, no need to estimate sigma,....).	Estimate Std. Error t value Pr(>|t|) (Intercept) 39.9358610 0.717498656 55.65984 1.220362e-187 horsepower -0.1578447 0.006445501 -24.48914 7.031989e-81

9: Generalized linear regressions

Codes	Comments	Results
#Poisson model.		

<table>
<tr><td>#(https://onlinecourses.science.psu.edu/stat504/node/223)
#The study investigated factors that affect whether the
#female crab had any other males, called satellites,
#residing near her. Explanatory variables that are thought
#to affect this included the female crab’s color (C), spine #condition (S), weight (Wt), and carapace width (W). The #response outcome for each female crab is her number
#of satellites (Sa). There are 173 females in this study.</td><td></td><td></td></tr>
<tr><td>Crab=read.table("C:/Program Files/R/crab.txt");

colnames(Crab)=c("Obs","C","S","W","Wt","Sa");
fix(Crab);
Expectation=glm(Crab$Sa~1, family=poisson(link=log));
summary(Expectation);</td><td>Download the textfile “crab.txt” from Canvas. Save it to local disk. Load it and name this data set “Crab”.
Name the columns.
One can manually edit the data frame using fix().
Poisson regression of Sa on intercept only (no predictors).
This model shows the mean of Sa: E(Sa)=exp(1.0713).</td><td></td></tr>
<tr><td>model.1=glm(Crab$Sa~1+Crab$W,family=poisson(link=log));
summary(model.1);</td><td>Poisson regression of Sa on W.</td><td>Coefficients:
Estimate St
Error z value Pr(>|z|)
(Intercept) -3.30476
0.54224 -6.095 1.1e-09
Crab$W 0.16405
0.01997 8.216 < 2e-1
***</td></tr>
<tr><td>data.frame(Crab,pred=model.1$fitted);
plot(Crab$W,Crab$Sa);</td><td>Prediction formula, which forecasts Sa values.
Illustrate the relationship between Sa and W.</td><td></td></tr>
<tr><td>model.disp=glm(Crab$Sa~Crab$W, family=quasipoisson(link=log), data=Crab);
summary(model.disp);</td><td>Fitting the overdispersed Poisson model. This is a generalized Poisson model Poisson(theta,omega), where omega is the dispersion parameter. The variance of the data is equal to omega times the data mean. When omega=1, this model reduces to Poisson model.</td><td></td></tr>
<tr><td>#Logistic-binomial model
#In the "MASS" library there is a data set called "menarche"
#(Milicer, H. and Szczotka, F., 1966, Age at Menarche in #Warsaw girls in 1965, Human Biology, 38, 199-203), in which #there are three variables: "Age" (average age of age #homogeneous groups of girls), "Total" (number of girls in #each group), and "Menarche" (number of girls in the group #who have reached menarche)</td><td></td><td></td></tr>
<tr><td>library(MASS);
data("menarche");
glm.out=glm(cbind(Menarche, Total-Menarche) ~ Age,family=binomial(logit),data=menarche);</td><td>Load package MASS.
Load data set “menarche”.
Logistic-binomial regression of Menarche on age.</td><td></td></tr>
</table>

plot(Menarche/Total ~ Age, data=menarche);
 	
lines(menarche$Age, glm.out$fitted, type="l", col="red");
title(main="Menarche Data with Fitted Logistic Regression Line");
 	
glm.disp=glm(cbind(Menarche, Total-Menarche) ~ Age,family=quasibinomial(logit),data=menarche);
 	
summary(glm.disp);
	Plot of the ratio Menarche/Total via Age in the data set "menarche".
	Add fitted line.
	Add title.
	
	This is the corresponding logistic-binomial overdispersion model.	

10: Bass model

Code	Comments	Results
#We show a typical Bass curve of the yearly sales of #VCRs in the US home market between 1980 and 1989 using #the non-linear least squares function nls(). The variable T79 is the #year from 1979, and the variable Tdelt is the time from 1979 at #a finer resolution of 0.1 year for plotting the Bass curves. The #cumulative sum function cumsum() is useful for monitoring #changes in the mean level of the process.		
T79=1:10;	Label the 10 years from 1980 to 1989.	
Tdelt =(1:100) / 10;	Set a mesh=1/10 year.	
Sales=c(840,1470,2110,4000, 7590, 10950, 10530, 9470, 7790, 5890);	Enter the sales data per year S(t).	
Bass.nls= nls(Sales ~ M * (((P+Q)^2 / P) * exp(-(P+Q) * T79)) /(1+(Q/P)*exp(-(P+Q)*T79))^2, start = list(M=sum(Sales), P=0.03, Q=0.38));	Use non-linear least squares to solve the parameters m, p, q. Note that the initial values P=0.03, Q=0.38 are values for a typical product (this often comes from media published data for similar products).	Parameters:
summary(Bass.nls);	Print the results.	Estimate Std. Error t value Pr(>|t|)
Cusales=cumsum(Sales);	Calculate the cumulative sum N_t.	M 6.798e+04 3.128e+03 21.74 1.10e-07 *** P 6.594e-03 1.430e-03 4.61 0.00245 ** Q 6.381e-01 4.140e-02 15.41 1.17e-06 ***

Bcoef=coef(Bass.nls);
m=Bcoef[1];
p=Bcoef[2];
q=Bcoef[3];
	Extract the estimates m,p,q.	

ngete = exp(-(p+q) * Tdelt);		
Bpdf=m * ((p+q)^2 / p) * ngete / (1 + (q/p) * ngete)^2;	Calculate the estimated density function S(t);	
plot(Tdelt, Bpdf, xlab = "Year from 1979",ylab = "Sales per year", type='l');	Plot the estimated S(t);	
points(T79, Sales);	Add the observed S(t) to compare with the one from fitted model.	
Bcdf= m * (1 - ngete)/(1 + (q/p)*ngete);	Calculate the estimated cumulative distribution function N_t.	
plot(Tdelt, Bcdf, xlab = "Year from 1979",ylab = "Cumulative sales", type='l');	Plot the estimated pdf.	
points(T79, Cusales);	Join the observed N_t to the previous plot.	

11: Exponential smoothing

Code	Comments	Results
#The number of letters of complaint received each month by a motoring #organization over the four years 1996 to 1999 are available on the website. At #the beginning of the year 2000, the organization wishes to estimate the #current level of complaints and investigate any trend in the level of #complaints. We should first plot the data, and, even though there are only #four years of data, we should check for any marked systematic trend or #seasonal effects.		
Complaints= c(30, 15, 34, 21, 16, 33, 24, 29, 52, 41, 25, 36, 11, 33, 18, 14, 25, 62, 41, 33, 26, 42, 33, 16, 25, 36, 25, 35, 52, 41, 52, 15, 42, 7, 12, 15, 28, 15, 30, 20, 20, 35, 17, 24, 33, 14, 30, 22);	Load complaints data with respect to month (from Jan 1996 to Dec 1999).	
Comp.ts <- ts(complaints, start = c(1996, 1), freq = 12);	Define a time series.	
plot(Comp.ts, xlab = "Time / months", ylab = "Complaints");	Plot it. From the picture, we see there is no seasonal feature detected. Then one can use exponential smoothing method to predict this time series.	
Comp.hw1 = HoltWinters(Comp.ts, beta = 0, gamma = 0); Comp.hw1;	The exponential smoothing estimates. Print the estimates alpha and the coefficients of the time series.	Smoothing parameters: alpha: 0.1464355
Comp.hw2 = HoltWinters(Comp.ts, alpha=0.2,beta = 0, gamma = 0); plot(Comp.hw2);	We compare the above results with the ones with	

Comp.hw1$SSE Comp.hw2$SSE	alpha=0.2. The 2 results are similar. Compare the SSE of the 2 models.	beta : 0 gamma: 0 [1] 10004.98 [1] 10077.95

#Now we consider a Holt-Winters seasonal model.
Comp.hw3= HoltWinters (Comp.ts, seasonal = "mult");
plot(Comp.hw3);
Comp.hw3$SSE;
	Consider the seasonals.
	Plot the estimates and observed data.
	Check its SSE. It is slightly less than the above 2 models. We can claim that this model is slightly better than the other 2. And the complaints data does have minor seasonal feature.
		[1] 9668.28