R Tutorial for Statistical Learning

Qidi Peng

12: Subset selection methods

Code	Comments	Results
#Delete variables having missing data from a data set.		
install.packages("ISLR");	The functions detecting and deleting missing data can be found in the package "ISLR".	
library (ISLR);	Load the package "ISLR".	
fix(Hitters);	Check the data frame "Hitters", which is baseball players' personal information. We observe that there are missing data in the variable "Salary".	
is.na(Hitters$Salary);	To check which data is missing. True is existing; False means missing. R regards True as 1 and False as 0.	[1] 59 [1] 263
sum(is.na(Hitters$Salary));	Number of existing data in "Salary".	
sum(1-is.na(Hitters$Salary));	Number of missing data in "Salary".	[1] 0
Hitters =na.omit(Hitters);	Extract the players who have not any missing data.	
sum(is.na(Hitters));	Check the extracted data set. There is no longer missing data.	
#Best subset methods.		
install.packages("leaps");	The best subset methods function can be found in the package "leaps".	
library(leaps);	Load the package.	
help(regsubsets.formula);	The function regsubsets.formula() performs best subset method.	
regfit.full=regsubsets(Salary~.,Hitters);	Let Y=Salary, we perform subset selection among all other variables in the data set Hitters.	
summary(regfit.full);	Print the results. String "*" means the corresponding variable is selected by the method. For instance, this output indicates that the best one-variable model contains only "CRBI" and the best two-variable model contains only "Hits" and "CRBI". In the output, Forced in means whether the variable is contained by all models; Forced out means whether the variable is excluded by all models. By default, regsubsets() only reports results up to the	

regfit.full=regsubsets(Salary~.,data=Hitters,nvmax =19);	best eight-variable model. But the nvmax option can be used in order to return as many variables as are desired. Output the results for up to best 19-variable model.	
reg.summary=summary(regfit.full); Reg.summary;		
#model selection.		
names(reg.summary);	Show all the statistics (variables) contained in reg.summary.	
reg.summary$cp;	Show the Cp statistic for all 19 models, the one which minimizes the test MSE is the one containing 10 variables.	
reg.summary$bic;	Show the BIC, the one containing 6 variables is the minimizer of BIC.	
reg.summary$rsq;	The R2 statistic is increasing. It is not surprising since R2 is a measure for fitting training data. More variables, more accurate.	
reg.summary$adjr2;	The adjusted R2 chooses the 11-variable model. Unlike R2, It does the job to measure the test error.	
par(mfrow =c(2,2));	Open 4 windows of size 2*2. We are going to illustrate the 4 statistics of all the 19 models: RSS, adjusted R2, Cp and BIC.	
plot(reg.summary$rss ,xlab=" Number of Variables ",ylab=" RSS", type="l");	In Window (1,1), present the plot of RSS. Since RSS is to measure training error, it is decreasing as number of variables increases.	
plot(reg.summary$adjr2 ,xlab =" Number of Variables ", ylab=" Adjusted RSq",type="l");	In Window (1,2), present the adjusted R2, it has a maximum.	
which.max(reg.summary$adjr2);	Find the maximizer of the adjusted R2.	
points (11, reg.summary$adjr2[11], col ="red",cex =2, pch =20);	Paint this maximum in red.	[1] 11
plot(reg.summary$cp ,xlab =" Number of Variables ",ylab="Cp", type="l");	In Window (2,1), present Cp. The best model minimizes Cp.	
which.min(reg.summary$cp);	Find the minimizer of Cp.	
points (10, reg.summary$cp [10], col ="red",cex =2, pch =20);	Paint the minimum value point of Cp in red.	[1] 10
plot(reg.summary$bic ,xlab=" Number of Variables ",ylab=" BIC", type="l");	In Window (2,2), plot BIC VS number of variables.	
which.min(reg.summary$bic);	Find the minimizer of BIC.	
points(6, reg.summary$bic[6], col ="red",cex=2,pch =20);	Add this minimum point in red.	[1] 6
	We see that all these statistics propose different models. It is reasonable because each statistic has a different statistical point of view. Coefficient estimates of the best 6-variable model.	

coef(regfit.full,6);		

#Forward and backward stepwise selection methods.
regfit.fwd=regsubsets (Salary~.,data=Hitters,nvmax =19,method="forward");
summary(regfit.fwd);
regfit.bwd=regsubsets (Salary~.,data=Hitters,nvmax =19,method="backward");
summary(regfit.bwd);
coef(regfit.full,7);
coef(regfit.fwd ,7);
coef(regfit.bwd,7);
	In regsubsets(), the argument "method="forward"" can perform forward stepwise selection.
	Output the results.
	The argument "method="backward"" can perform backward stepwise selection
	Output the results..
	We compare the best subsets, forward and backward stepwise methods. We observe that they provide the same one to six-variables models, but the seven-variable models are different. This shows the 3 methods are quite accurate when dealing with small number of predictors.	

# Cross validation. 	
fix(predict.regsubsets);
#In the separate pad write:
function(object,newdata,id){
  form=as.formula(object$call[[2]]);
  mat=model.matrix (form,newdata);
  coefi=coef(object,id=id);
  xvars=names(coefi);
  mat[,xvars]%*%coefi}
 	
k=10;
set.seed (1);
folds=sample (1:k,nrow(Hitters),replace =TRUE);
cv.errors =matrix (NA ,k,19, dimnames =list(NULL , paste (1:19)));
 	
for(j in 1:k){best.fit=regsubsets (Salary~.,data=Hitters[folds !=j,], nvmax =19);for(i in 1:19){pred=predict.regsubsets(best.fit,Hitters[folds==j,], id=i);
cv.errors [j,i]=mean((Hitters$Salary[folds ==j]-pred)^2)}};
 	
mean.cv.errors =apply(cv.errors ,2, mean);
mean.cv.errors;
	There is no predict() function for the subsets methods, we create one here.
	
	10-fold cross validation.
	
	Compute the test errors for each subset, using best subsets method.
	
	We use the apply() function to average over the columns of this matrix in order to obtain a vector for which the jth element is the cross validation error for the j-variable model.	

which.min(mean.cv.errors);	This function tells you which model does the cross-validation selects. Attention, the result is random.	

13: Ridge regression and the lasso

Codes	Comments	Results
# Ridge regression. We perform a regression y~x.		
x=model.matrix (Salary~.,Hitters)[,-1];	Define predictor set (including all predictors, the salaries of players are declined and the qualitative variables are converted to dummy variables).	
y=Hitters$Salary;	Define the response y.	
install.packages("glmnet"); library (glmnet);	Ridge and the lasso are contained in the package "glmnet". We install it.	
grid =10^ seq (10,-2, length =100);	We choose a grid for the tuning parameter lambda, from 10^10 to 10^-2. The mesh is not equal size.	
ridge.mod =glmnet(x,y,alpha =0,lambda=grid);	Perform a ridge regression for y~x. When the argument alpha=0, we fit ridge regression; when alpha=1, we fit the lasso regression; when 0<alpha<1, we fit a ridge-the lasso mixture model.	
help(glmnet);	To check more information on glmnet().	
coef(ridge.mod)[,50];	The estimates of the coefficients of the 50th lambda value (lambda=11498 in this case.)	[1] 11497.57
ridge.mod$lambda [50];	The corresponding lambda.	
predict(ridge.mod,s=50,type ="coefficients")[1:20,];	Predict the first 20 estimates of the coefficients for lambda=50.	
#subset validation.		

set.seed (1);
train=sample (1: nrow(x), nrow(x)/2);
test=(- train);
y.test=y[test];
 	
ridge.mod =glmnet (x[train ,],y[train],alpha =0, lambda =grid , thresh =1e-12);
ridge.pred=predict (ridge.mod ,s=4, newx=x[test,]);
 	
mean((ridge.pred -y.test)^2);
	Set a seed for subset validation.
	Select a training data set index.
	Define test data set index.
	Define test data set.
	
	Perform ridge regression using training data set.
	
	Predict the regression, using lambda=4 (must be the same length as the test set).
	Calculate the test MSE. The result is random.
		[1] 101036.8

# The lasso. 	
lasso.mod =glmnet (x[train ,],y[train],alpha =1, lambda =grid);
plot(lasso.mod);
 	
coef(lasso.mod)[,50];
 	
#Cross validation for choosing lambda; subset validation for
#calculating the test MSE.
set.seed (1);
help(cv.glmnet);
cv.out =cv.glmnet (x[train ,],y[train],alpha =1);
 	
plot(cv.out);
 	
bestlam =cv.out$lambda.min;
bestlam;
lasso.pred=predict (lasso.mod ,s=bestlam ,newx=x[test ,]);
mean((lasso.pred -y.test)^2);
	Perform the lasso on the training data (take alpha=1).
	
	Illustrate the results. We see most of the coefficients are around 0.
	Check the coefficients for lambda=11498. Most are zero.
	
	Check the cross validation function for glmnet.
	Run a 10-fold (by default) cross validation for the lasso regression on the training data set, in order to choose lambda.
	Illustrate the results. We see how the test MSE behaves as log(lambda) increases.
	Return the best lambda among the grid.
	
	Make the lasso prediction using the best lambda.
	Calculate the test MSE by using subset validation.
	The result is random.
		[1] 32.18284
		
		[1] 101918.2

14: Principal component regression (PCR)

Codes	Comments	Results

#Principal component regression.
install.packages("pls");
library(pls);
	Install the package which contains principal component regression function pcr().	

pcr.fit=pcr(Salary~., data=Hitters ,scale=TRUE ,validation="CV");	One directly performs the PCR.	
validationplot(pcr.fit ,val.type="MSEP");	Plot the cross validation MSE VS number of components (predictors). Attention, this cross validation method is used to select the principal components.	
summary(pcr.fit);	Output the results. From the summary we see that the best model contains 7 predictors.	
#Calculate the test MSE using subset validation.		
pcr.pred=predict (pcr.fit ,x[test ,], ncomp =7);	Use the 7 principle components, we make prediction using the training data set.	
mean((pcr.pred -y.test)^2);	Calculate the test MSE.	[1] 85199.48
pcr.fit=pcr(y~x,scale =TRUE ,ncomp =7);	Perform the PCR only with 7 components.	
summary(pcr.fit);	Print the results.	

15: Tree-Based Methods

Codes	Comments	Results
#Classification tree..		
install.packages("tree");	Install the package which contains tree-based functions for classification and regression.	
library(tree);		
library(ISLR);		
attach(Carseats) ;	Consider dataset" Carseat"s for classification.	
High=ifelse(Carseats$Sales <=8,"No","Yes ");	ifelse() function is used to create a variable, which takes on a value of Yes if the Sales variable exceeds 8, and takes on a value of No otherwise. Here we artificially create response labels (2~classes) , called High, for each predictor.	
Carseats =data.frame(Carseats ,High);	Create a data matrix of (X,Y), predictors and responses.	
set.seed(2);		
train=sample (1:nrow(Carseats), 200);	Select 200 training data.	
Carseats .test=Carseats[-train ,] ;	The rest data are for testing. It is validation set.	
High.test=High[-train];	Run classification tree.	
tree.carseats	Output the results.	
=tree(High~.-Sales ,Carseats ,subset=train) ;	Plot the results.	
summary(tree.carseats);	Predict the rest data.	
plot(tree.carseats);	Test the prediction by using validation set. Display	

tree.pred=predict(tree.carseats ,Carseats .test ,type="class") ; table(tree.pred ,High.test)	the results by confusion matrix.	High.test tree.pred No Yes No 86 27 Yes 30 57

#Regression tree..
library(MASS) ;
set.seed(1) ;
train = sample (1:nrow(Boston), nrow(Boston)/2) ;
tree.boston=tree(medv~.,Boston , subset=train) ;
summary(tree.boston);
plot(tree.boston);
 	
yhat=predict (tree.boston ,newdata=Boston[-train ,]) ;
boston.test=Boston[-train ,"medv"] ;
plot(yhat ,boston.test);
abline (0,1) ;
mean((yhat -boston.test)^2)
	
	Use data "Boston".
	
	Use half data for fitting, half for validation.
	Fit regression tree.
	Output the results.
	Plot the results.
	
	Perform subset validation.
	
	Use plot to compare true values and predicted values.
	Calculate the test MSE.
		
		[1] 25.05