SDSC3006 Lab
4-Cross Validation

Langming LIU langmiliu2-c@my.cityu.edu.hk

School of Data Science
City University of Hong Kong

Contents

- Validation Set Approach
- LOOCV
- K-fold CV

Validation Set Approach

Introduction

- Data set: Auto dataset in ISLR.
- Target 1: find how mpg(response) depends on horsepower(predictor).
- Possible models:
  - mpg ~ horsepower (linear)
  - mpg ~ horsepower + horspower^2 (quadratic)
  - mpg ~ horsepower + horspower^2 + horspower^3 (cubic)
- Target 2: find the best model among above.
- Method: validation set approach
  - Randomly split the data set into training set and validation set.
  - Fit each model using the training data set.
  - Estimate test error rate using the validation data set.
  - The model with the lowest validation (testing) MSE is the winner!

Code

```
library(ISLR)
attach(Auto)

#generate same set of random numbers every time this code is
#executed
set.seed(1)
#try set.seed(2) to generate another set after one trial

#pick half of the samples in dataset randomly to be training set
l=length(mpg)
train = sample(l , l/2)    #set of indexes
mpg.test = mpg[-train]    #rest is validation set
```

Code

```
#fit 3 models respectively using the training data
lm.fit1 = lm(mpg~horsepower,data=Auto,subset=train)
lm.fit2 = lm(mpg~poly(horsepower,2),data=Auto,subset=train)
lm.fit3 = lm(mpg~poly(horsepower,3),data=Auto,subset=train)
#make predictions for each model
lm.pred1 = predict(lm.fit1,Auto[-train,])
lm.pred2 = predict(lm.fit2,Auto[-train,])
lm.pred3 = predict(lm.fit3,Auto[-train,])
#calculate MSE for each model
mean((mpg.test-lm.pred1)^2)     #linear
mean((mpg.test-lm.pred2)^2)     #quadratic
mean((mpg.test-lm.pred3)^2)     #cubic
```

LOOCV

Introduction

- Data set: Auto dataset in ISLR.
- LOOCV(Leave-One-Out Cross Validation) involves splitting the set of observations into two parts like the validation set approach but it's not random.
- Steps of LOOCV:
  - Split the data set into training set(whole dateset except (xi,yi)) and validation set(xi,yi).
  - Fit model using the training data set.
  - Estimate test error rate: $MSE_i = (y_i - \hat{y}_i)^2$
  - Repeat above process for n times(i=1,2,...,n), LOCCV estimate for the test MSE is the average:

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

R implemention

- Notice:
  - LOOCV can be automatically computed for generalized linear models using the glm() and cv.glm() functions.
  - Since linear regression belongs to generalized linear models, we can use the glm() function rather than lm() to fit each model.
  - The cv.glm() function is in the boot library.
- Code

```
library(ISLR)
attach(Auto)
library(boot)
glm.fit = glm(mpg~horsepower,data=Auto)
cv.err = cv.glm(Auto,glm.fit)
cv.err$delta    #average MSE and adjusted MSE
```

R implemention

- Code:

```
#Write loop statement to repeat LOOCV process for all models
cv.error = rep(0,5)    #initial value
#for polynomials from order 1 to 5 calculate average MSE
for (i in 1:5){
glm.fit = glm(mpg~poly(horsepower,i),data=Auto)
cv.error[i] = cv.glm(Auto,glm.fit)$delta[1]
}
cv.error
```

K-fold CV

Introduction

- Data set: Auto dataset in ISLR.
- This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size.
- Steps of K-fold CV:
  - Randomly divide the set of observations into k groups, or folds, of approximately equal size.
  - Choose the ith fold to be validation set, remaining k-1 folds form the training set.
  - Fit model using the training data set.
  - Estimate test error rate of ith fold(test set): MSE_i
  - Repeat above process for k times(i=1,2,...,k), take average of MSE values:

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

R implemention

- Notice: the cv.glm() function can also be used to implement k-fold CV, just set k value.
- Code

```
library(ISLR)
attach(Auto)
library(boot)
set.seed(1)
#test polynomials from order 1 to 10 (loop)
cv.error.10 = rep(0,10)
for (i in 1:10) {
glm.fit = glm(mpg~poly(horsepower,i),data=Auto)
cv.error.10[i] = cv.glm(Auto,glm.fit,K=10)$delta[1]
}
cv.error.10
```