Resampling Methods

by

AINDRILA GARAI

MSC STATISTICS, IIT KANPUR

aindrilag22@iitk.ac.in

Resampling methods involve repeatedly drawing samples from a training set and refitting a model of interest on each sample in order to obtain additional information about the fitted model. Eg. cross-validation and the bootstrap.

- The process of evaluating a model's performance is known as **model assessment**.
- The process of selecting the proper level of flexibility for a model is known as **model selection**.

cross-validation:

Cross-validation can be used to estimate the test error or to select the appropriate level of flexibility.

1. **The Validation Set Approach:**

The Validation Set Approach involves randomly dividing the available set of observations into two parts(comparable size) a training set and a validation set or hold-out set. The model is fit on the training set and the fitted model is used to predict the responses for the observations in the validation set.

- The resulting validation set error rate provides an estimate of the test error rate. We can get different test error rate for different validation set and that would help us conclude with any confidence is that the linear fit is not adequate for this data.
- The test error rate can be highly variable.
- The validation set error rate may tend to overestimate the test error rate because all the observations are not used to train a model.

```
library (ISLR2)
set.seed (1) # use of different values of set.seed() will provide different v
alidation set
train <- sample (392, 196) # to split the set of observations into two halves
lm.fit <- lm(mpg ~ horsepower , data = Auto , subset = train)
lm.fit

##
## Call:
```

```
## lm(formula = mpg ~ horsepower, data = Auto, subset = train)
##
## Coefficients:
## (Intercept)   horsepower
##     41.2835      -0.1697

attach (Auto)
mean ((mpg - predict (lm.fit , Auto))[-train ]^2) # to calculate MSE(test) in
the validation set

## [1] 23.26601

lm.fit2 <- lm(mpg ~ poly (horsepower , 2), data = Auto , subset = train) # qu
adratic regressions
mean ((mpg - predict (lm.fit2 , Auto))[-train]^2)

## [1] 18.71646\
```

2. **Leave-One-Out Cross-Validation:**

LOOCV involves splitting the set of observations into two parts: a single observation (x_1, y_1) is used for the validation set and the remaining observations $(x_2, y_2), \ldots, (x_n, y_n)$ make up the training set. We can repeat the procedure by selecting (x_i, y_i) for the validation data. The LOOCV estimate for the test MSE is-

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{MSE}_i$$

- It has far less bias.

- If n is large, we use-

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2$$

where $\hat{y}_i$ is the ith fitted value from the original least squares fit, and h_i is the leverage.

```
library (boot)
glm.fit <- glm (mpg ~ horsepower , data = Auto) # to compute LOOCV
coef (glm.fit)

## (Intercept)  horsepower
##  39.9358610  -0.1578447

cv.err <- cv.glm (Auto , glm.fit)
cv.err$delta

## [1] 24.23151 24.23114

a = 0
LOOCV <- function(x)
```

```
{
  for(i in 1:nrow(x))
  {
  train <- (1:nrow(x))[-i]
  lm.fit <- lm(mpg ~ horsepower , data = x , subset = train) # change this co
lumn names for another data
  a <- ((mpg - predict (lm.fit , x))[-train ]^2) + a
  }
  return(a/nrow(x))
}
LOOCV(Auto)

##      397
## 24.23151
```

3. **k-Fold Cross-Validation:**

This approach 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. The k-fold CV estimate is-

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \mathrm{MSE}_i$$

- It is less computational.
- It often gives more accurate estimates of the test error rate than LOOCV.
- It leads to an intermediate level of bias.
- It turns out that LOOCV has higher variance than does k-fold CV with $k < n$ because we are averaging the outputs of k fitted models that are less correlated with each other(than LOOCV) since the overlap between the training sets in each model is smaller.
- So in one note, bias reduces and variance increases with the increase of the value of k.

```
glm.fit3 <- glm (mpg ~ horsepower , data = Auto)
cv.error.10 <- cv.glm (Auto , glm.fit3 , K = 5)$delta[1] # 5-fold CV
cv.error.10

## [1] 24.40949
```

4. **Cross-Validation on Classification Problems:**

In the classification setting, the LOOCV error rate takes the form

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Err}_i$$

where $\mathrm{Err}_i = I(y_i \neq \hat{y}_i)$.

Bootstrap:

Bootstrap provides a measure of accuracy of a parameter estimate or of a given statistical learning method. Each bootstrap data set contains n observations sampled with replacement from the original data set.

- We will invest a fraction α of our money in X and will invest the remaining $1-\alpha$ in Y. Then, α will be-

  $$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$$

  We repeated the process to estimate the standard deviation of $\hat{\alpha}$ for each bootstarp dataset. We would expect $\hat{\alpha}$ to differ from α by approximately the amount of sd of α on average.

```
alpha.fn <- function (data , index)
{
X <- data$X[index]
Y <- data$Y[index]
out <- ( var (Y) - cov (X, Y)) / ( var (X) + var (Y) - 2 * cov (X, Y))
return(out)
}
alpha.fn(Portfolio , 1:100)
```

```
## [1] 0.5758321
```

```
library(boot)
boot (Portfolio , alpha.fn, R = 1000) # R times bootstraps
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Portfolio, statistic = alpha.fn, R = 1000)
##
##
## Bootstrap Statistics :
##      original       bias    std. error
## t1* 0.5758321 -0.001722657  0.09351011
```

```
boot.fn <- function (data , index)
coef (lm(mpg ~ horsepower , data = data , subset = index))

boot (Auto , boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Auto, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##       original        bias    std. error
## t1* 39.9358610  0.0410299775 0.833042888
## t2* -0.1578447 -0.0005189657 0.007216344
```