Math642_HW3_FyonaSun

Fyona Sun

1/29/2020

Problem 5.3

We now review k-fold cross-validation. (a) Explain how k-fold cross-validation is implemented. K-fold cross-validation is implemented by randomly dividing the set of observations into k groups of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining k-1 folds as training set.The test error is then estimated by averaging the k. $CV = 1/k \sum_{i=1}^{k} MSE_i$

(b) What are the advantages and disadvantages of k-fold cross-validation relative to:

i. The validation set approach? There are 2 mian disadvantages of the validation set approach compared to k-fold cross-validation. First of all, the validation estimate of the test error rate can be highly depend on which observations are included in the training set and which observations are included in the validation set. So the test error would be varied time by time. Second, only one subset of the observations 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.

ii. LOOCV? The LOOCV cross-validation approach is a special case of k-fold cross-validation in which k=n. The advantage of the LOOCV cross-validation approach is that it gives approximately unbiased estimates of the test error, since each training set contains n-1 observations. Compared to k-fold cross-validation LOOCV has two disadvantages. First, it is computationally expensive as it requires fitting the model n times compared to k-fold cross-validation which requires the model to be fitted only k times. Second, it has higher variance than k-fold cross-validation. Since we are averaging the outputs of n fitted models trained on an almost identical set of observations, these outputs are highly correlated, and the mean of highly correlated quantities has higher variance than less correlated ones. So, there is a bias-variance trade-off associated with the choice of k in k-fold cross-validation.

Problem 5.8

(a)Generate a simulated data set as follows: In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

```r
set.seed(1)
x=rnorm(100)
y=x-2*x^2+rnorm(100)
```

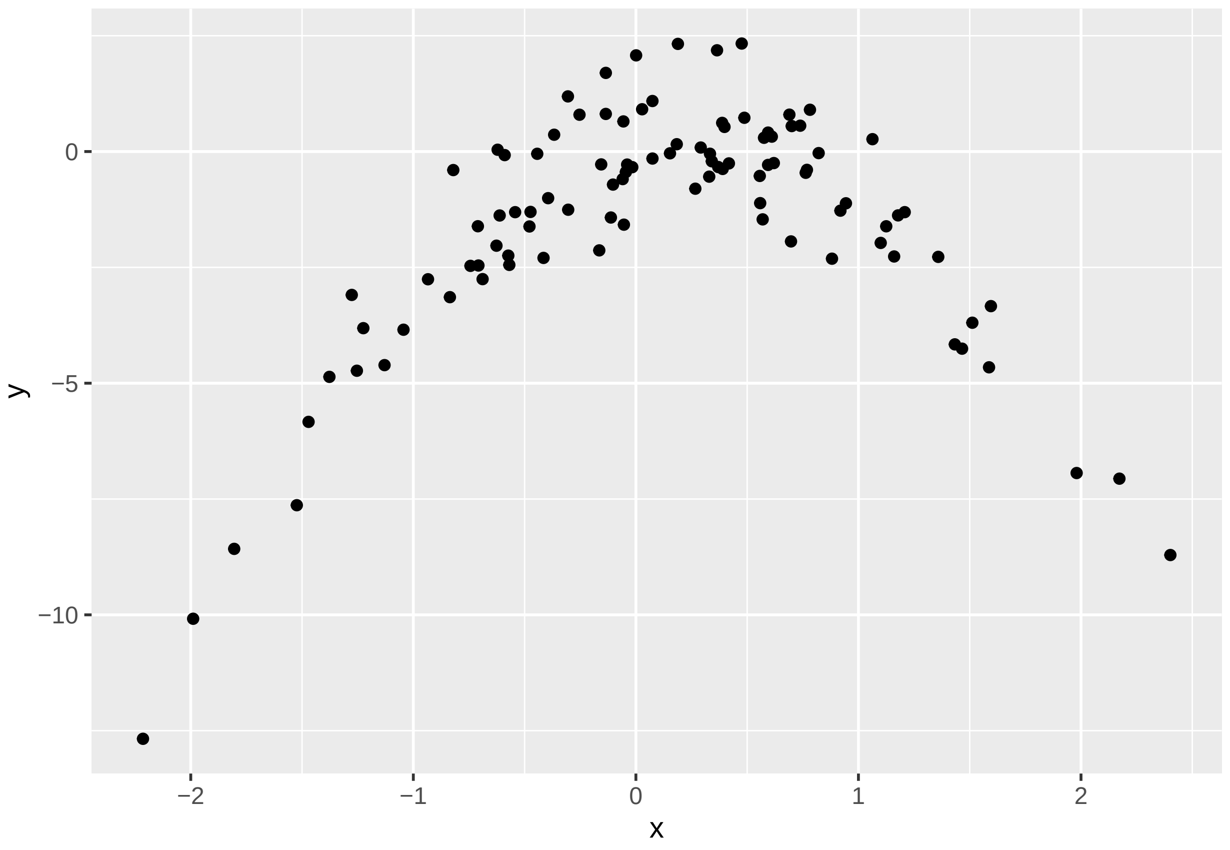
Here we have that n=100 and p=2 and the model is $Y = X - 2X^2 + \epsilon$

(b) Create a scatterplot of X against Y . Comment on what you find.

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```r
ggplot(data.frame(x,y),aes(x=x,y=y)) + geom_point()
```



The plot suggests a down-curved quadratic relationship.

(c) Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares:

```r
# for i
library(boot)
set.seed(1)
Data <- data.frame(x, y)
fit.glm.1 <- glm(y ~ x)
cv.glm(Data, fit.glm.1)$delta[1]
```

```
## [1] 7.288162
```

```r
# for ii
fit.glm.2 <- glm(y ~ poly(x, 2))
cv.glm(Data, fit.glm.2)$delta[1]
```

```
## [1] 0.9374236
```

```r
# for iii
fit.glm.3 <- glm(y ~ poly(x, 3))
cv.glm(Data, fit.glm.3)$delta[1]
```

```
## [1] 0.9566218
```

```r
# for iv
fit.glm.4 <- glm(y ~ poly(x, 4))
cv.glm(Data, fit.glm.4)$delta[1]
```

```
## [1] 0.9539049
```

(d) Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

```
# for i
set.seed(2)
Data <- data.frame(x, y)
fit.glm.1 <- glm(y ~ x)
cv.glm(Data, fit.glm.1)$delta[1]
```

```
## [1] 7.288162
```

```
# for ii
fit.glm.2 <- glm(y ~ poly(x, 2))
cv.glm(Data, fit.glm.2)$delta[1]
```

```
## [1] 0.9374236
```

```
# for iii
fit.glm.3 <- glm(y ~ poly(x, 3))
cv.glm(Data, fit.glm.3)$delta[1]
```

```
## [1] 0.9566218
```

```
# for iv
fit.glm.4 <- glm(y ~ poly(x, 4))
cv.glm(Data, fit.glm.4)$delta[1]
```

```
## [1] 0.9539049
```

The results are the same to the results obtained in (c) since LOOCV evaluates n folds of a single observation.

(e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

fit.glm.2 has the minimal MSE. The result aligned with the plot in (b), which shows a quadratic relationship between X and Y.

(f) Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?

```
summary(fit.glm.2)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 2))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9650  -0.6254  -0.1288   0.5803   2.2700
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.5500     0.0958  -16.18  < 2e-16 ***
## poly(x, 2)1   6.1888     0.9580    6.46 4.18e-09 ***
## poly(x, 2)2 -23.9483     0.9580  -25.00  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9178258)
```

```
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  89.029  on 97  degrees of freedom
## AIC: 280.17
##
## Number of Fisher Scoring iterations: 2
```

```r
summary(fit.glm.4)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0550  -0.6212  -0.1567   0.5952   2.2267
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09591 -16.162  < 2e-16 ***
## poly(x, 4)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, 4)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, 4)3   0.26411    0.95905   0.275    0.784
## poly(x, 4)4   1.25710    0.95905   1.311    0.193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9197797)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  87.379  on 95  degrees of freedom
## AIC: 282.3
##
## Number of Fisher Scoring iterations: 2
```

We can see the model fit.glm.4, a polynomial of degree four that only the quatratic and linear term are significant. If we look at the quatratic fit we can see that both the quatratic term and the linear term are significant.

Problem 5.9

(a) Based on this data set, provide an estimate for the population mean of medv. Call this estimate $\hat{\mu}$.

```r
library(MASS)
data(Boston)
mu.hat <- mean(Boston$medv)
mu.hat
```

```
## [1] 22.53281
```

(b) Provide an estimate of the standard error of $\hat{\mu}$. Interpret this result.

```r
se.hat <- sd(Boston$medv) / sqrt(dim(Boston)[1])
se.hat
```

```
## [1] 0.4088611
```

(c) Now estimate the standard error of $\hat{\mu}$ using the bootstrap. How does this compare to your answer from (b)?

```r
set.seed(1)
mu.function <- function(data, index) {
    mu <- mean(data[index])
    return (mu)
}
boot(Boston$medv, mu.function, 10000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = mu.function, R = 10000)
##
##
## Bootstrap Statistics :
##     original       bias    std. error
## t1* 22.53281 0.0005436561   0.4054831
```

The bootstrap estimated standard error of $\hat{\mu} = 0.4054831$ is very close to the estimate found in (b) of 0.4088611.

(d) Based on your bootstrap estimate from (c), provide a 95 % confidence interval for the mean of medv. Compare it to the results obtained using t.test(Boston$medv).

```r
CI.mu.hat <- c(22.53 - 2 * 0.4119, 22.53 + 2 * 0.4119)
CI.mu.hat
```

```
## [1] 21.7062 23.3538
```

```r
t.test(Boston$medv)
```

```
##
##  One Sample t-test
##
## data:  Boston$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
```

The confidence interval computed by bootstrap estimation is very close to the one provided by the t.test() function.

(e) Based on this dataset,provide an estimate $\hat{\mu}_{med}$ for the median value of medv in the population.

```r
med.hat <- median(Boston$medv)
med.hat
```

```
## [1] 21.2
```

(f)

```
median.function <- function(data, index) {
    median <- median(data[index])
    return (median)
}
boot(Boston$medv, median.function, 10000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = median.function, R = 10000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*     21.2 -0.01445   0.3750377
```

We get an estimated median value of 21.2 which is equal to the value obtained in (e), with a standard error of 0.3750377 which is relatively small compared to median value.

(g)

```
percent.hat <- quantile(Boston$medv, c(0.1))
percent.hat
```

```
##   10%
## 12.75
```

The tenth percentile of medv in Boston suburbs is $\hat{\mu}_{0.1} = 12.75$ (h)

```
tenth.function <- function(data, index) {
    mu_0.1 <- quantile(data[index], c(0.1))
    return (mu_0.1)
}
boot(Boston$medv, tenth.function, 10000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = tenth.function, R = 10000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*    12.75 0.006665   0.5021664
```

We get an estimated tenth percentile value of 12.75 which is again equal to the value obtained in (g), with a standard error of 0.5021664 which is relatively small compared to percentile value.