Q25

Rahul Atre

2023-11-03

Q25 [Resampling Methods]

We perform cross-validation on a simulated data set.

1. Generate a simulated data set as follows:

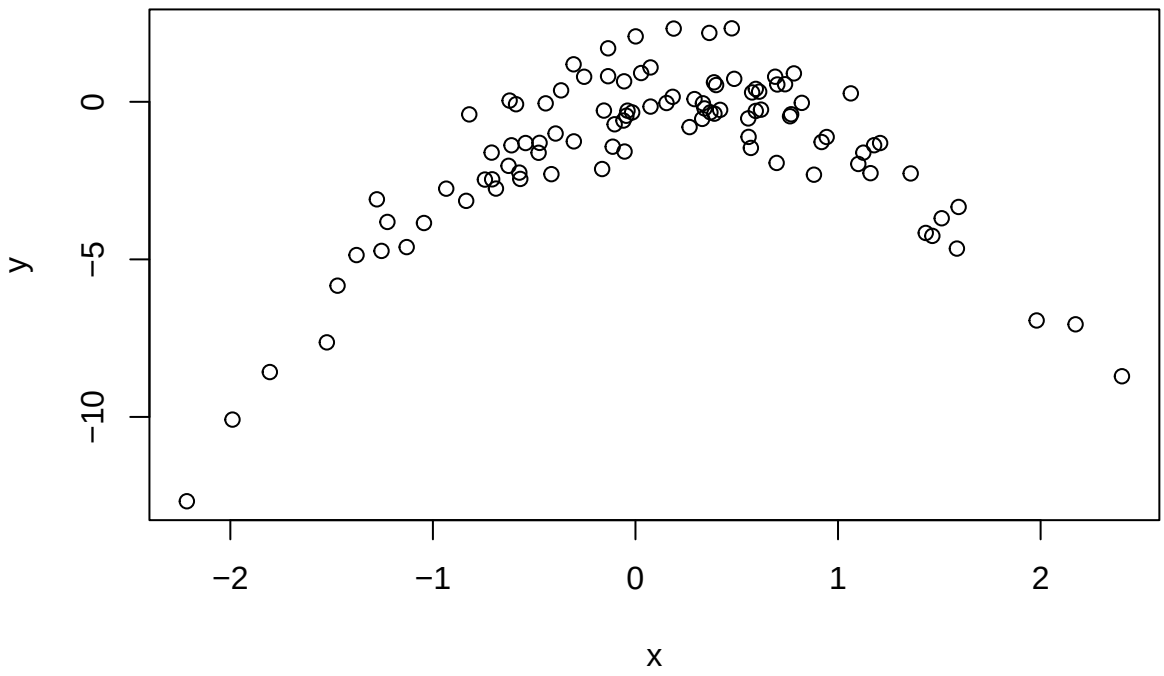
```
set.seed(1)
x=rnorm(100)
y=x-2*x^2+rnorm(100)
```

In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

Ans: From the above equation, it is evident that the equation form of the model is $y = x - 2x^2 + \epsilon$. Since the rnorm() function has a parameter of length 100 for x and y, the value of **n is 100**. Similarly, since the model has two predictors, x and x^2, the value of **p is 2**.

2. Create a scatterplot of X against Y. Comment on what you find.

```
plot(x,y)
```



The above scatterplot shows a non-linear relationship between x and y. The data appears to be a downward parabola (i.e. quadratic).

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

- i. $Y = \beta_0 + \beta_1 X + \epsilon$
- ii. $Y = \beta_0 + \beta_1 X^1 + \beta_2 X^2 + \epsilon$
- iii. $Y = \beta_0 + \beta_1 X^1 + \beta_2 X^2 + \beta_3 X^3 + \epsilon$
- iv. $Y = \beta_0 + \beta_1 X^1 + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon$

Note you may find it helpful to use the data.frame() function to create a single data set containing both X and Y.

```
library(boot) #Load bootstrap and cross-validation functions into R
set.seed(1) #Set seed for reproducibility

poly_df <- data.frame(x,y) #Creating a single data frame for x and y
cv.errors  <- rep(0, 4)
glm.fit <- list()

for (i in 1:4) {
  glm.fit[[i]] <- glm(y ~ poly(x, i, raw = T), data = poly_df) #GLM fit for polynomial deg 1-4
  cv.errors[i] <- cv.glm(poly_df, glm.fit[[i]])$delta[1] #Obtaining cross-validation errors
```

```
}

crossValidation_df <- data.frame(degree=1:4, cv.errors) #Creating dataframe for crossValid. errors
crossValidation_df
```

```
##   degree cv.errors
## 1      1 7.2881616
## 2      2 0.9374236
## 3      3 0.9566218
## 4      4 0.9539049
```

4. Repeat 3. using another random seed, and report your results. Are your results the same as what you got in 3.? Why?

```
set.seed(50)

poly_df <- data.frame(x,y) #Creating a single data frame for x and y
cv.errors  <- rep(0, 4)
glm.fit <- list()

for (i in 1:4) {
  glm.fit[[i]] <- glm(y ~ poly(x, i, raw = T), data = poly_df) #GLM fit for polynomial deg 1-4
  cv.errors[i] <- cv.glm(poly_df, glm.fit[[i]])$delta[1] #Obtaining cross-validation errors
}

crossValidation_df <- data.frame(degree=1:4, cv.errors) #Creating dataframe for crossValid. errors
crossValidation_df
```

```
##   degree cv.errors
## 1      1 7.2881616
## 2      2 0.9374236
## 3      3 0.9566218
## 4      4 0.9539049
```

From the above function call, it appears that despite changing the seed, the results are the exact same. The reason for this is because in LOOCV, there is no randomness since each data point is utilized for both the training and test set. In simpler terms, each data point is partitioned in its own subset, thus allowing consistent and replicable results. In fact, this method's specialty is reducing bias and randomness (i.e. the mean-squared error rate) to avoid overfitting the data.

5. Which of the models in 3. had the smallest LOOCV error? Is this what you expected? Explain your answer.

From the cross-validation error results in 3, we can see that **model 2** had the smallest LOOCV error of 0.9374236. This is indeed what we had expected since the initial equation does not contain an x^3 or x^4, while having a second degree polynomial. The x^2 term contributes the most to the response variable y.

6. Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in 3. using OLS. Do these results agree with the conclusions drawn based on the cross-validation results?

```r
full_model <- lm(y ~ poly(x, 4))
summary(full_model)
```

```
##
## Call:
## lm(formula = y ~ poly(x, 4))
##
## 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
##
## Residual standard error: 0.9591 on 95 degrees of freedom
## Multiple R-squared:  0.8753, Adjusted R-squared:  0.8701
## F-statistic: 166.7 on 4 and 95 DF,  p-value: < 2.2e-16
```

When we include all 4 of the predictors in the polynomial equation, it is apparent that the first two coefficients are sufficient and statistically significant enough to create a good fitted model. The third and fourth coefficient however, have high p-values, and are therefore not significant and can be omitted to minimize MSE or training error.