PHS 597 – Homework 1 – Regression by Successive Orthogonalization – Fall 2021

Havell Markus, hmm5304@psu.edu

September 6, 2021

Assginment description

- The goal of the code below is to conduct a small simulation to verify that orthogonalization works for bivariate case.
- For a univariate model with no intercept $\hat{\beta} = \langle x, y \rangle / \langle x, x \rangle$ and the $r = y - x\hat{\beta}$
- You can apply principles from simple univariate regression to solve for coeficients for multiple linear regression
- If $x_1, x_2, ..., x_p$ are columns of the data matrix X and they are orthogonal ($\langle x_j, x_k \rangle = 0$ for all $j \neq k$), then $\hat{\beta_j}$ are the univariate estimates $\hat{\beta_j} = \langle x, y \rangle) / (\langle x, x \rangle$
- Thus, in order to solve any coefficient $\hat{\beta_j}$, we can orthogonalize or regress y on the residuals of x_j on remaining columns of the data $x_1, x_2, ..., x_{j-1}, x_{j+1}, ..., x_p$

Bi-variate case

The following code generates the data where $y = x_1 + x_2 + \epsilon$

```r
set.seed(1839) # setting seed
x1 <- rnorm(200, 0, 1) # generating x1
x2 <- x1 + rnorm(200, 1, 3) # generating a correlated x2

eps <- rnorm(200, 0, 6) # generating error
y <- x1 + x2 + eps # making y
```

Conduct multiple regression to check for $\hat{\beta_1}$ and $\hat{\beta_2}$

```r
fit <- lm(y ~ x1 + x2) # fitting overall model
fit
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Coefficients:
## (Intercept)           x1           x2
##     -0.5802       0.1487       1.1272
```

Using Successive Orthogonalization to solve for $\hat{\beta_1}$ and $\hat{\beta_2}$:

1. Regress $y \sim x_1$ and obtain residuals $\epsilon_{y \sim x_1}$

- The residuals are component of y after considering the variation due to x_1

2. Regress $x_2 \sim x_1$ and obtain residuals $\epsilon_{x_2 \sim x_1}$

- The residuals are component of x_2 after considering the variation due to x_1

3. Regress $\epsilon_{y \sim x_1} \sim \epsilon_{x_2 \sim x_1}$ and obtain the coefficient$

- The residuals are component of y after considering the variation due to x_2 after considering variation due to x_1 on both y and x_2
- Thus, $\hat{\beta}_2$ signifies the relationship of y and x_2 independent of x_1

4. Without, loss of generality we can also solve for $\hat{\beta}_1$

```
e_yx1 <- residuals(lm(y ~ x1))
e_x2x1 <- residuals(lm(x2 ~ x1))
beta2 <- coef(lm(e_yx1 ~ e_x2x1))[2]
beta2
```

```
##   e_x2x1
## 1.127219
```

```
e_yx2 <- residuals(lm(y ~ x2))
e_x1x2 <- residuals(lm(x1 ~ x2))
beta1 <- coef(lm(e_yx2 ~ e_x1x2))[2]
beta1
```

```
##    e_x1x2
## 0.1487445
```

- In the book, we can also apply Algorithm 3.1
- In order to do so, we can regress $x_1 \sim x_2$ and regress y on the residuals to solve for $\hat{\beta}_1$
- Similarly, we can regress $x_2 \sim x_1$ and regress y on the residuals to solve for $\hat{\beta}_2$

```
coef(lm(y ~ residuals(lm(x1 ~ x2))))[2]
```

```
## residuals(lm(x1 ~ x2))
##              0.1487445
```

```
coef(lm(y ~ residuals(lm(x2 ~ x1))))[2]
```

```
## residuals(lm(x2 ~ x1))
##               1.127219
```

Implementation of Algorithm 3.1 Regression by Successive Orthogonalization.

- The follwoing is the implementation of algorithm 3.1

```
successive_orthogonalization <- function(input_X, input_Y){
  Z <- cbind(input_X[,1], 0, 0, 0)

  for(j in 2:ncol(Z)){
    l <- seq(1, j-1)
```

```
    lamda_vec <- c()
    res_vector <- input_X[,j]

    for(i in l){
      lamda_vec[i] <- (Z[,i] %*% input_X[,j])/(Z[,i] %*% Z[,i])
      res_vector <- res_vector - (lamda_vec[i] * Z[,i])
    }

    Z[,j] <- res_vector

  }

  (Z[,j] %*% input_Y) / (Z[,j] %*% Z[,j])

}
```

- Apply to trivariate case

```
set.seed(1839) # setting seed
x1 <- rnorm(200, 0, 1) # generating x1
x2 <- x1 + rnorm(200, 1, 3) # generating a correlated x2
x3 <- x1 + x2 + rnorm(200, 3, 3) # generating a correlated x3

eps <- rnorm(200, 0, 6) # generating error
y <- x1 + x2 + x3 + eps # making y
fit <- lm(y ~ x1 + x2 + x3) # fitting overall model
fit
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3)
##
## Coefficients:
## (Intercept)           x1           x2           x3
##     -0.4517       1.2092       0.9679       1.0696
```

```
X <- cbind(x1, x2, x3)
X <- as.matrix(cbind(X, 1))
```

- Solve for β_0
- Solve for β_1
- Solve for β_2
- Solve for β_3

```
successive_orthogonalization(input_X = X[,c(1,2,3,4)], input_Y = y)
```

```
##           [,1]
## [1,] -0.451681
```

```r
successive_orthogonalization(input_X = X[,c(2,3,4,1)], input_Y = y)
```

```
##          [,1]
## [1,] 1.209189
```

```r
successive_orthogonalization(input_X = X[,c(3,4,1,2)], input_Y = y)
```

```
##           [,1]
## [1,] 0.9679488
```

```r
successive_orthogonalization(input_X = X[,c(4,1,2,3)], input_Y = y)
```

```
##        [,1]
## [1,] 1.0696
```