14주차 과제

과 목	최적화및기계학습
담 당 교 수	이 두 호
학 번	201720970
학 과	컴퓨터공학과
이 름	권 대 한

201720970_권대한 최적화및기계학습 제출일 2022. 05. 29.

Population, Income, Illiteracy, Life Exp, Frost를 입력변수로 하고, Murder를 출력변수로 하여 경사하강법을 이용해 다중회귀분석을 실시하라. 단, 학습률과 종료조건은 본인이 정할 것!

<코드>

```
state <- as.data.frame(state.x77)

# Weight Cross Check, Linear Model Declaration

model1 <- lm(Murder ~ Population + Income + Illiteracy + `Life Exp` + Frost, data=state)

# state에 저장된 정보만 불러오기 위함. 입력 데이터, 출력 데이터를 각 X, Y에 저장하였음.

x <- cbind(1, state$Population, state$Income, state$Illiteracy, state$`Life Exp`, state$Frost) %>% as.matrix(.)

y <- as.matrix(state$Murder)

# SSE Function, 1/2n * y_hat - y %>% square

cost <- function(x, y, w) {

  sum( (x %*% w - y)^2 ) / (2*length(y))

}

# 시행 착오적으로 Hyper Parameter를 구하였음.

alpha <- 0.00000003

num_iters <- 1000000

cost_history <- double(num_iters)

#w_history <- list(num_iters)

# Weight 행렬 선언. 정규 분포에 따르는 -1:1 범위의 Weight를 선언하였을 때 극소 값을 구할 수 없었다.

w <- matrix(rep(0, 6), nrow=6)

for (i in 1:num_iters) {

  error <- (x %*% w - y)

  delta <- t(x) %*% error # X^T %*% (X %*% w - y)

  w <- w - alpha * delta / length(y)

  cost_history[i] <- cost(x, y, w)

  #w_history[[i]] <- w

  #print(c(cost(x, y, w)))

}

# iteration 중 가장 작은 cost 확인.

which.min(cost_history)
```

높은 확률로 마지막 iteration 의 cost 가 가장 낮을 것.

회귀 문제 풀이지만, 상당히 높은 Cost 값을 가지게 된다. 4.907642

print(c(cost(x, y, w)))

LSE 의 loss

model1$residuals %>% sum()

Gradient Descent 에서의 Weight 출력.

w %>% print

하지만 많은 차이가 존재한다.

model1$coefficients

<실행 결과>

```
> # 벡터, 행렬로 표현
> x ← cbind(1, state$Population,  state$Income, state$Illiteracy, state$`Life Exp`, state$Frost) %>% as.matrix(.)
> y ← as.matrix(state$Murder)
>
>
> cost ← function(x, y, w) {
+   sum( (x %*% w - y)^2 ) / (2*length(y))
+ }
>
> alpha ← 0.00000003
> num_iters ← 1000000
>
> cost_history ← double(num_iters)
> w_history ← list(num_iters)
>
> #w ← matrix(c(1.215e+02, 1.700e-04, 4.749e-04, 1.529e+00, -1.658e+00, -1.142e-02), nrow=6)
> w ← matrix(rep(0, 6), nrow=6)
>
> # gradient descent
> for (i in 1:num_iters) {
+   error ← (x %*% w - y)
+   delta ← t(x) %*% error
+   w ← w - alpha * delta / length(y)
+   cost_history[i] ← cost(x, y, w)
+   #w_history[[i]] ← w
+   #print(c(cost(x, y, w)))
+ }
> which.min(cost_history)
[1] 1000000
> print(c(cost(x, y, w)))
[1] 4.907642
> model1$residuals %>% sum()
[1] -1.561251e-17
>
> w %>% print
              [,1]
[1,]  0.0037782538
[2,]  0.0001992445
[3,] -0.0004657382
[4,]  0.0340923533
[5,]  0.1635512812
[6,] -0.0296879301
> model1$coefficients
  (Intercept)    Population        Income    Illiteracy    `Life Exp`         Frost
 1.214934e+02  1.699728e-04  4.748577e-04  1.529070e+00 -1.658323e+00 -1.142001e-02
>
```

결국 해당 코드는 SSE 인 Loss Function(!= Cost Function)의 수치를 최소화 하도록 작성하였으므로, 연속된 데이터를 예측하기 위한 모델로는 부적합하다는 것을 알 수 있다.