번호 : /

강원대학교 시험 답안지

		감독교수확인	이 두 호	
2022 학년도 1학기 기말고사	시험과목 : 최적화 및 기계학습 2반	전공 : 컴퓨터공학과 학번: 201720970 성명: 권 대 한	출 제 자	이두호
			채점성적	

1. (10점) 아래의 20개의 관측치에 대해서 예측 모델을 구축하려고 한다. 선형계획법을 이용하여 최적 모델 파라미터를 추정하시오.

<실행 코드>

```
library(dplyr)
library(lpSolve)

data.1 <- data.frame("X1" = c(5.22, 9.76, 2.13, 1.38, 7.80, 2.59, 3.80, 3.84,3.63, 6.97, 6.80, 6.09, 6.43, 2.44, 1.07, 5.80, 5.93, 7.85, 5.97, 4.27),
                     "X2" = c(2.74, 19.88, 6.16, 12.83, 11.60, 3.79, 17.25, 16.81, 16.39, 4.89, 6.46, 16.19, 3.78, 15.38, 9.48, 5.51, 3.53, 13.76, 10.95, 4.81),
                     "Y" = c(25.43, 52.99, 26.00, 30.52, 41.76, 21.29, 38.91, 38.06, 38.72, 33.56, 32.88, 42.04, 30.06, 36.19, 26.97, 31.06, 27.66, 44.62, 37.22, 27.19))

#선형회귀 모델 생성
model.1 <- lm(Y ~ X1 + (X2) %>% sqrt(.), data = data.1)
#model.1 %>% summary(.)

# 편과 x1, x2의 입력
X <- cbind(1, data.1$X1, (data.1$X2) %>% sqrt(.))
y <- data.1$Y

n <- nrow(data.1)
p <- ncol(data.1)

direction <- "min"

# 0.5, 1 - 0.5

objective.in  <-  c(rep(0.5, n), rep(1-0.5, n), rep(0, 2*ncol(X)))

# x1 * x2^2 = y
#diag(n) %*% -diag(n) %*% X * -X
# diagonal matrix, 입력 행렬의 제곱이 출력과 같아야 한다는 제약 조건
const.mat  <- cbind(diag(n), -diag(n), X, -X)
const.dir <- "="

model.2  <-  lp(direction, objective.in, const.mat, const.dir, y)

result.1 <- model.2$solution
result.2 <- matrix()
for (i in 1:length(result.1))
  result.2[i] <- (if(result.1[i]<=0)-1 else +1) * max(result.1[i], result.1[i+p])


# 앞 부분 가중치 제거
w.1  <-  result.2[((2*n)+1):((2*n)+p)]

cat("선형 회귀 가중치: ", model.1$coefficients, '\n')
cat("선형계획법 가중치: ", w.1, '\n')

 # y.hat 계산
 calc.1 <- matrix(y - X %*% w.1)

 cat("선형 회귀 절댓값 오차: ", model.1$residuals %>% abs(.) %>% max(.), '\n')
 cat("선형계획법 절댓값 오차: ", calc.1 %>% abs(.) %>% max(.), '\n')
```

<실행 화면>

```
> cat("선형 회귀 가중치: ", model.1$coefficients, '\n')
선형 회귀 가중치:  4.36895 1.905877 6.650111
> cat("선형계획법 가중치: ", w.1, '\n')
선형계획법 가중치:  4.527187 1.940267 6.509185
>
> # y.hat 계산
> calc.1 ← matrix(y - X %*% w.1)
>
> cat("선형 회귀 절댓값 오차: ", model.1$residuals %>% abs(.) %>% max(.), '\n')
선형 회귀 절댓값 오차:  1.351187
> cat("선형계획법 절댓값 오차: ", calc.1 %>% abs(.) %>% max(.), '\n')
선형계획법 절댓값 오차:  1.401266
```

문제를 풀기 위한 목적함수는 min [1, 1, C] [w^+, w^-, di]^T, 제약 식은 [[X, -X, -1] [-X, X, -1]] [w^+, w^-, d]^T <= [y, -y]가 된다. 그리고 w^+, w^-, d 모두 0보다 크거나 같은 조건을 만들 수 있다.
그러나 실제 코드에서 구현했을 때, 왜인지는 모르겠지만 가중치가 모두 0이 나오는 문제가 있어, 제약식의 조건을 “=”으로 수정하였으며, 우변의 경우 y로 수정하였다.
마지막으로 w = w^+ + w^-에 의해 w^+와 w^-의 값 중 큰 값을 가중치로 결정하도록 하였다.
w[1] + w[2] * x1 + w[3] * (x2)^(1/2)의 가중치와 절댓값 오차를 선형회귀 모델과 선형계획법 모델을 만들어 계산하였으며, 실제 residual과 매우 근사한 것을 알 수 있다.

답: 4.527187 1.940267 6.509185

2. (10점) 아래의 선형계획 문제를 푸시오.

<실행 코드>

```r
library(dplyr)
library(primes)
library(lpSolve)

direction <- "max"

objective.in <- c(1:200) %>% matrix(., nrow = 1)

# xi 계수 표현
for (i in 1:200)
  objective.in[i] <- sin(i)^2 + cos(i)


# 1~198에 대한 조건을 저장할 공간 선언
const.1 <- rep(0, 200 * 198) %>% matrix(., ncol = 200)

for(i in 1:198)
{
  # const matrix의 case를 계산한다.
  j <- (i - 1) * 201
  # 계산된 인덱스를 기반으로 조건을 표기한다.
  const.1[j+1] <- const.1[j+2] <- const.1[j+3] <- 1

  # 실질적으로 1~198에 대한 입력의 합이 2.758이므로... 저장하는 공간이 추가적으로 필요할 것.
  #const.1[i] <- const.1[i+1] <- const.1[i+2] <- 1
}

# 2부터 200까지 소수 찾기
prime <- generate_primes(min = 2, max = 200)

# 소수일 경우의 조건 저장 공간 선언
const.2 <- rep(0, 200*length(prime)) %>% matrix(., ncol = 200)

# 소수개 만큼 조건 저장
for(i in 1:length(prime))
{
  j <- (i - 1) * (length(prime)) + prime[i]
  const.2[j] <- 1
}

# 1~200에 대한 조건 저장 공간 선언
const.3 <- rep(0, 200 * 200) %>% matrix(., ncol = 200)
```

```
43
44    # 1~200의 조건 저장
45    for(i in 1:200)
46    {
47      j <- (i - 1) * 201
48      const.3[j+1] <- 1
49    }
50
51    # 모든 조건을 하나의 매트릭스로 합친다.
52    const.mat <- matrix(c(const.1, const.2, const.3), nrow=444, byrow=TRUE)
53
54    # 1번 조건 198개, 2번 조건 46개, 3번 조건 200개
55    const.dir  <- c(rep("<=", 198), rep("<=", length(prime)), rep(">=", 200))
56
57    const.rhs <- c(rep(2.758, 198), rep(1.56, 46), rep(0, 200))
58
59    model <- lp(direction, objective.in, const.mat, const.dir, const.rhs)
60
61    cat("최적해: ", model$objval, "\n")
62    cat("x1 ~ 200: ", model$solution, "\n")
63
64    model$solution
```

<실행 결과>

```
> cat("최적해: ", model$objval, "\n")
최적해:  161.7312
> cat("x1 ~ 200: ", model$solution, "\n")
x1 ~ 200:  1.56 0 0 0 2.758 0 0 2.758 0 0 2.758 0 0 2.758 0 0 2.758 0 0 0 2.758 0 0 2.758 0 0 2.758 0 0 2.758 0 0
2.758 0 0 2.758 0 0 1.56 0 1.198 1.56 0 0 0 1.56 1.198 0 1.56 0 0 2.758 0 0 2.758 0 0 2.758 0 0 0 2.758 0 0 2.758
0 0 2.758 0 0 2.758 0 0 2.758 0 0 2.758 0 0 2.758 0 0 2.758 0 0 0 1.56 1.198 0 1.56 0 0 0 2.758 0 0 0 0 2.758 0 0 2.758 0 0
1.198 1.56 0 1.198 1.56 0 0 2.758 0 0 2.758 0 0 2.758 0 0 2.758 0 0 2.758 0 0 2.758 0 0 0 2.758 0 0 2.758 0 0 1.56 1.198 0
1.56 0 0 0 2.758 0 0 2.758 0 0 2.758 0 0 2.758 0 0 2.758 0 0 2.758 0 0 1.56 1.198 0 1.56 0 0 2.758 0 0 2.758 0 0 0 2.758 0
0 2.758 0 0 2.758 0 0 2.758 0 0 0 0 0 2.758 0 0 0 2.758
>
> model$solution
  [1] 1.560 0.000 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000
 [17] 2.758 0.000 0.000 2.758 0.000 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000
 [33] 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 1.560 0.000 1.198 1.560 0.000 0.000 0.000
 [49] 1.560 1.198 0.000 1.560 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758
 [65] 0.000 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758
 [81] 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 0.000 1.560 1.198 0.000 1.560
 [97] 0.000 0.000 0.000 2.758 0.000 0.000 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 1.198 1.560
[113] 0.000 1.198 1.560 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000
[129] 0.000 2.758 0.000 0.000 2.758 0.000 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 1.560 1.198
[145] 0.000 1.560 0.000 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000
[161] 0.000 2.758 0.000 0.000 2.758 0.000 0.000 1.560 1.198 0.000 1.560 0.000 0.000 2.758 0.000 0.000
[177] 2.758 0.000 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000 2.758 0.000 0.000
[193] 0.000 0.000 0.000 2.758 0.000 0.000 0.000 2.758
```

Xi의 계수가 정해져 있는 문제로 1~200까지 반복문을 돌려 sin(i)^2 + cos(i)의 값을 구했으며, 이후 모든 상태에 대한 제약식을 1개의 매트릭스에 합쳐 선형계획 모델을 구축하였다.
이후 1부터 198까지의 모든 경우의 수를 저장할 공간을 선언하였으며, 그 크기는 200*198이 된다.
198개의 I에 대한 경우의 수가 200개가 존재하기 때문이다.
계산한 인덱스를 기반으로 모든 조건의 제약을 저장한다.
마지막으로 좌변, 조건, 우변을 구성하였으며, 1번째 조건 198개의 좌, 우변, 2번째 조건 소수 46개, 3번째 조건 200개를 모두 포함해, 총 444개의 제약조건이 만들어졌으며, 이를 토대로 선형 계획법을 구성하였다.

답: 최적해: 161.7312, 목적함수: 캡쳐 화면 참조.

3. (10점) 삼척시청을 출발해 강원도 내 모든 도청, 시청, 군청을 방문하고 다시 삼척시청으로 돌아오는 최적경로를 구하라. 단, 이동은 차량을 이용하며, 단위는 km로 한다.

<실행 코드>

```
library(dplyr)
library(TSP)


# csv 파일로부터 거리(km) 데이터 불러오기(data.frame type)
nw.tsp  <- read.csv("Optimization/Final/3.csv", header=TRUE)
nw.tsp <- nw.tsp[,-1]
rownames(nw.tsp) <- colnames(nw.tsp)
name <- colnames(nw.tsp)

#TSP 함수에 사용하기 위해, data.frame에서 행렬로 바꿔줌
nw.tsp <- as.matrix(nw.tsp)
nw.tsp %>% class

name <- c('삼척시청','강원도청','강릉시청','동해시청','속초시청','원주시청','춘천시청','태백시청','고성군청','양구군청',
          '양양군청','영월군청','인제군청','정선군청','철원군청','평창군청','홍천군청','화천군청','횡성군청')

# 거리 데이터의 열 이름을 지정해주었다.
colnames(nw.tsp) <- rownames(nw.tsp) <- as.character(name)

# Triangular Matrix + Transposed Triangular Matrix, 삼각행렬의 데이터를 전치함수와 더해줌으로, 비어있는 삼각행렬에 데이터를 넣어준다.
nw.tsp <- nw.tsp + t(nw.tsp)

#TSP 함수에 데이터 전달
tsp <- TSP(nw.tsp)

#노드 수 출력
n_of_cities(tsp)
#labels(tsp)

# 함수를 통해 최단 거리 출력해준다.
tour <- solve_TSP(tsp)
tour
# 도출된 최단 거리를 따로 저장하였다
tour_route <- names(tour)

# 삼척시청의 인덱스, 위치를 찾아냄.
idx <- which(tour_route == "삼척시청")
# 삼척시청부터 마지막 장소의 인덱스까지... 정렬해서 최단 경로를 알려주었다.
cat(tour_route[idx:length(tour_route)], tour_route[1:idx])
```

<실행 결과>

```
> tour
object of class 'TOUR'
result of method 'arbitrary_insertion+two_opt' for 19 cities
tour length: 768
> # 도출된 최단 거리를 따로 저장하였다
> tour_route ← names(tour)
>
> # 삼척시청의 인덱스, 위치를 찾아냄.
> idx ← which(tour_route == "삼척시청")
> # 삼척시청부터 마지막 장소의 인덱스까지... 정렬해서 최단 경로를 알려주었다.
> cat(tour_route[idx:length(tour_route)], tour_route[1:idx])
삼척시청 태백시청 정선군청 평창군청 영월군청 원주시청 횡성군청 홍천군청 춘천시청 강원도청 철원군청 화천군청 양구군청 인제군청 고성군청 속초시청 양양군청 강릉시청 동해시청 삼척시청
>
```

카카오맵 기반으로 삼척시청과 강원도 내 관공서 간의 거리(km 단위)를 조사하였으며, csv 파일에 거리 정보를 모두 저장하여 코드를 작성하였다. 코드를 실행하면 모든 관공서를 경유하고 돌아오는 최적의 거리가 768km이며, 순서는 삼척시청 태백시청 정선군청 평창군청 영월군청 원주시청 횡성군청 홍천군청 춘천시청 강원도청 철원군청 화천군청 양구군청 인제군청 고성군청 속초시청 양양군청 강릉시청 동해시청 삼척시청이 최적 탐색법이다.

4. (10점) 기존의 경사하강법 알고리즘에서 학습률을 너무 크게 잡으면 비용함수가 발산(overshooting)되는 상황이 발생한다. 학습이 진행될수록 학습률을 감소시키는 Adagrad 알고리즘 이용하여 다중선형회귀모델을 구하라. 단, w, h의 초기치는 0이고, 학습률은 0.7, 반복 횟수는 5,000회이다.

<실행 코드>

```r
library(dplyr)

input <- data.frame("x1" = c(9, 12, 17, 25, 7, 23, 27, 16, 15, 3, 5, 20, 13, 8, 18, 28, 11, 24, 6, 30, 10, 21, 29, 2, 14),
                    "x2" = c(7, 12, 11, 7, 14, 14, 7, 13, 10, 10, 10, 8, 12, 7, 9, 13, 14, 8, 10, 6, 11, 9, 13, 7, 9),
                    "y" = c(26.42, 39.44, 46.57, 52.09, 34.85, 60.54, 56.56, 48.81, 40.36, 23.46, 24.68, 45.53,
                            40.63, 26.02, 45.08, 66.78, 40.16, 52.47, 28.26, 60.02, 34.42, 48.83, 68.50, 14.41, 37.98))

x <- cbind(1, input$x1, input$x2) %>% as.matrix(.)
y <- input$y %>% as.matrix(.)

# 학습률, 반복횟수
alpha <- 0.7
num_iters <- 5000


# adagrad 알고리즘을 위한 h 선언
# 그리고 동적 weight를 적용하기 위함
h <- w <- matrix(x %>% ncol(.) %>% rep(0, .))

# gradient descent
for (i in 1:num_iters) {

  grad <- t(x) %*% (x %*% w - y) / length(y)
  h <- h + grad^2
  w <- w - alpha * (h^(-1/2)) * grad
}

w %>% cat(.)
```

<실행 결과>

```
> x ← cbind(1, input$x1, input$x2) %>% as.matrix(.)
> y ← input$y %>% as.matrix(.)
>
> # 학습률, 반복횟수
> alpha ← 0.7
> num_iters ← 5000
>
>
> # adagrad 알고리즘을 위한 h 선언
> # 그리고 동적 weight를 적용하기 위함
> h ← w ← matrix(x %>% ncol(.) %>% rep(0, .))
>
> # gradient descent
> for (i in 1:num_iters) {
+
+   grad ← t(x) %*% (x %*% w - y) / length(y)
+   h ← h + grad^2
+   w ← w - alpha * (h^(-1/2)) * grad
+ }
>
> w %>% cat(.)
1.416274 1.611763 1.569919
>
```

gradient는 cost function w에 대해 w로 편미분을 하는 것이다.
편미분을 시행할 때, 전치(Transpose)가 지워지므로, 증명에 의해 X^T (Xw-y) / n이 된다.
이를 토대로 Adagrad의 식에 맞게 element-wise operation을 시행하였으며, 이를 weight에 반영하였다.

답: w0: 1.58016, w1: 1.593763, w3: 1.578293

5. (5점) 아래의 정수계획법 문제의 최적해와 목적함수 값을 분지한계법을 이용하여 구하시오.

<실행 코드>

```r
library(dplyr)
library(lpSolve)

direction <- "max"
objective.in <- c(20, 50)
const.mat <- matrix(c(1.2, 4,
                      0.5, 1,
                      0, 1,
                      1, 0,
                      0, 1), ncol = 2, byrow = TRUE)

const.dir <- c("<=", "<=", "<=", ">=", ">=")
const.rhs <- c(240, 81, 40, 0, 0)

cat("최적해: ", lp(direction, objective.in, const.mat, const.dir, const.rhs, all.int = TRUE)$objval, "\n")
cat("목적함수 x1: ", lp(direction, objective.in, const.mat, const.dir, const.rhs, all.int = TRUE)$solution[1], "\n")
cat("목적함수 x2: ", lp(direction, objective.in, const.mat, const.dir, const.rhs, all.int = TRUE)$solution[2], "\n")
```

<실행 결과>

```
> library(dplyr)
> library(lpSolve)
>
> direction ← "max"
> objective.in ← c(20, 50)
> const.mat ← matrix(c(1.2, 4,
+                      0.5, 1,
+                      0, 1,
+                      1, 0,
+                      0, 1), ncol = 2, byrow = TRUE)
>
> const.dir ← c("≤", "≤", "≤", "≥", "≥")
> const.rhs ← c(240, 81, 40, 0, 0)
>
> cat("최적해: ", lp(direction, objective.in, const.mat, const.dir, const.rhs, all.int = TRUE)$objval, "\n")
최적해:  3520
> cat("목적함수 x1: ", lp(direction, objective.in, const.mat, const.dir, const.rhs, all.int = TRUE)$solution[1], "\n")
목적함수 x1:  106
> cat("목적함수 x2: ", lp(direction, objective.in, const.mat, const.dir, const.rhs, all.int = TRUE)$solution[2], "\n")
목적함수 x2:  28
```

lpSolve, Lindo 모두 분지한계법을 사용해 최적해를 찾아내므로, 문제에서 제시한 제약조건을 lpSolve 내장함수 lp에 입력시켜줌으로 lpSolve 모델을 생성하였다.

답: 최적해: 3520, 목적함수 x1: 106, 목적함수 x2: 28

6. (10점) 아래의 인공신경망에 대해서 오차역전파 알고리즘을 적용하여 가중치를 갱신할 경우, 31회 갱신된 후의 출력값 y.hat을 계산하시오. 단, 오차역전파의 학습률은 0.37 이다.

<실행 코드>

```
library(dplyr)

# 입력 변수 선언.
x <- matrix(c(2,9))
y <- 0.82

# 각 로지스틱, 하이퍼볼릭 탄젠트 함수를 활성 함수로 사용한다.
f.1 <- function(z)
{
 1/(1+exp(-z))
}
f.2 <- function(z)
{
 tanh(z)
}

# hidden layer weight initialized
a.0 <- x
b.1 <- matrix(c(0.1, 0.5))
b.2 <- matrix(c(3.9))
w.1 <- matrix(c(3.0, 0.5,
                0.7, 2.0), byrow = T, ncol=2)
w.2 <- matrix(c(1.2, -7.0), byrow = T, ncol=2)

df <- data.frame(t(rep(NA,10)))
names(df) <- c("b1.1", "b2.1",
               "w11.1", "w12.1","w21.1","w22.1",
               "b2.1",
               "w11.2","w12.2",
               "y.hat")

for(i in 1:31) {
 # 초기값 기록.
 a.1 <- f.1(b.1 + w.1 %*% a.0)
 a.2 <- as.vector(f.2(b.2 + w.2 %*% a.1))
 y.hat <- a.2
 df[i,] <- c(b.1[1,],b.1[2,],
             w.1[1,1],w.1[1,2],w.1[2,1],w.1[2,2],
             b.2[1,],
             w.2[1,1],w.2[1,2],
             y.hat)

 # 학습률은 0.37이다.
 alpha <- 0.37

 # 2nd layer output - real value => delta 2 (증명에 의해...)
 delta.2 <- a.2 - y

 # tanh를 미분한다면 1 - tanh(x)^2 이므로...
 gr.2 <- delta.2 * (1 - a.2^2) * rbind(1, a.1)

 b.2 <- b.2 - alpha * gr.2[1]      # b.2 update
 w.2 <- w.2 - alpha * gr.2[-1]     # w.2 update

 # 결국 activation의 통과 값에 대해 편미분을 수행하므로...
 delta.1 <- delta.2 * (1 - a.2^2) * w.2

 # a.1은 로지스틱 함수를 사용하므로, bias를 위해 1에 대한 행렬 곱을 수행한다.
 gr.1 <- (t(delta.1) * (a.1 * (1 - a.1))) %*% cbind(1, t(a.0))

 b.1 <- b.1 - alpha * gr.1[,1]     # b.1 update
 w.1 <- w.1 - alpha * gr.1[,-1]    # w.1 update
}
a.1 <- f.1(b.1 + w.1 %*% a.0)
y.hat <- a.2 <- as.vector(f.2(b.2 + w.2 %*% a.1))
df <- rbind.data.frame(df, c(b.1[1,],b.1[2,],
                             w.1[1,1],w.1[1,2],w.1[2,1],w.1[2,2],
                             b.2[1,],
                             w.2[1,1],w.2[1,2],
                             y.hat))
View(df)
```

1st Layer의 활성 함수가 로지스틱 함수, 그리고 2nd Layer의 활성 함수가 하이퍼볼릭 탄젠트 함수를 사용하므로 각 함수와 입력 변수를 선언하였다.
이후 문제에서 정의한 은닉층의 가중치 계수와 편의 계수를 선언해주었으며, 순 전파 계산을 시행한 값을 df 데이터 프레임에 저장한 후 역 전파 알고리즘에 의해 갱신된 계수를 통해 값을 계산하는 코드다.
본질적으로 Gradient, 변화량을 구하기 위해 Loss 함수를 편미분한 식의 연쇄 법칙을 이용하게 된다.
손실 함수 값과 해당 노드 출력 값의 편미분(델타) * 출력 값과 활성 함수 입력값(노드 계산 값)의 편미분(활성 함수의 미분) * 계산 값과 가중치의 편미분(wx + b와 같은 식에서 가중치에 대해 편미분을 시행한다면, 실제 노드의 입력 값, 이전 노드의 출력 값)으로 구할 수 있다.
출력 노드의 델타 값은, 증명에 따라 예측 값 - 실제 값으로 계산되므로 delta.2를 y.hat - y로 설정하였으며,
가중치 변화량인 gr.2는 delta.2 * 활성 함수 값 편미분 * (1, a.1)이 되는데, 노드 계산값이 편의에 의해 편미분 될 경우 1이 되기 때문이다.
2nd Layer의 bias는 학습률과 1번 행에 저장된 변화량을 기반으로 갱신하며, weight는 2, 3번 행에 저장된 변화량을 기반으로 갱신하게 된다.
delta.1은 은닉 층의 델타 증명에 의해, delta.1+1 * 2nd Layer 활성 함수 편미분 * 2nd Layer 가중치로 계산이 된다.
그러므로 delta.1은 delta.2 * (1 - tanh^2) * w.2가 된다.
1st Layer의 변화량은 delta.1 * (a.1 * (1 - a.1)) %*% (1, a.0)이 될 것이다.
delta.1는 행 벡터, a.1은 열 벡터이므로, 전치 과정을 수행하였다.
1st Layer bias는 학습률과 1번 열에 저장된 변화량을 기반으로 갱신하며, weight는 2, 3번 열에 저장된 변화량을 기반으로 갱신하고, 순 전파 계산을 수행하며 코드가 종료된다.

<실행 결과>

	b1.1	b2.1	w11.1	w12.1	w21.1	w22.1	b2.1	w11.2	w12.2	y.hat
1	0.1000458	0.5	3.000092	0.5004119	0.7	2	4.916742	2.216717	-5.983258	0.8178023

답: 0.8178023

7. (45점) 여름 방학 계획에 대해서 공백 포함 290~300자로 서술하시오.
간략하게 취업 준비에 많은 시간을 쏟을 생각을 하고 있다.
방학 기간에 학부 과정에서 이룬 것과 활동들을 정리하며, 학원에 다니면서 토익 점수를 올릴 계획이다.
주관적으로 본인은 다른 학생에 비해 뭐든 열심히 하는 편이라고 생각하지만, 그 정도에 그쳤다.
특히 3년간 연구생 활동을 하면서, 기업에 내세울 게 없다는 것이 가장 큰 문제점이라는 생각이 들었다.
조금 늦은 감이 있지만, 이것저것 따지면서 여러 부분에 시간 낭비하는 것보다
당장 할 수 있는 것부터 조금씩 행동하는 것이 현 위치에서의 바른 방향이라 생각하므로 토익 공부를 계획했다.
(300자)