11주차 과제

과 목	머신러닝을이용한재난설계
담 당 교 수	이 두 호
학 번	201720970
학 과	소프트웨어·미디어·산업공학부
이 름	권 대 한

머신러닝을이용한재난설계 제출일 2021. 11. 15.

201720970_권대한

1. NOR operation 퍼셉트론을 구현하시오.

<코드>

```
perceptron <- function(x1, x2, w1, w2, b) {
  if(w1*x1 + w2*x2 + b <= 0) {
    return(0)
  } else {
    return(1)
  }
}
NOR <- function(x1, x2) {
  return(perceptron(x1, x2, -0.5, -0.5, 0.2))
}
NOR(0,0); NOR(0,1); NOR(1,0); NOR(1,1)
```

<결과>

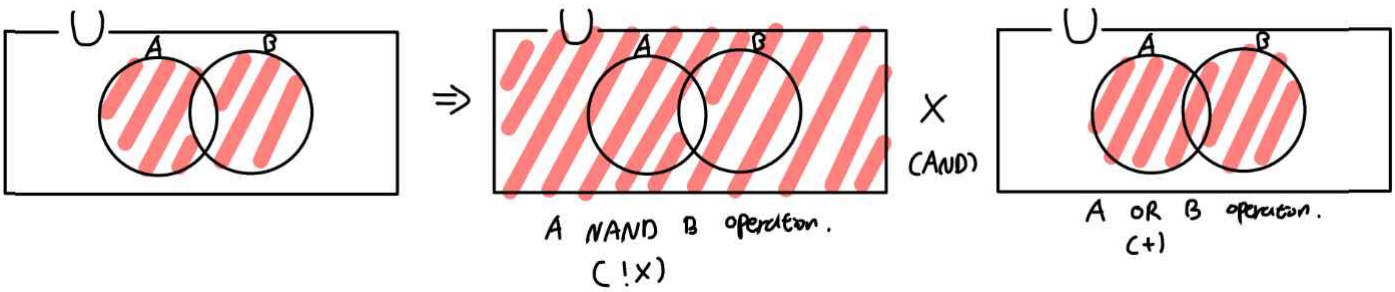
```
> perceptron ← function(x1, x2, w1, w2, b) {
+   if(w1*x1 + w2*x2 + b ≤ 0) {
+     return(0)
+   } else {
+     return(1)
+   }
+ }
> NOR ← function(x1, x2) {
+   return(perceptron(x1, x2, -0.5, -0.5, 0.2))
+ }
> NOR(0,0)
[1] 1
> NOR(0,1)
[1] 0
> NOR(1,0)
[1] 0
> NOR(1,1)
[1] 0
> 
```

2. XOR operation with three inputs 퍼셉트론을 구현하시오

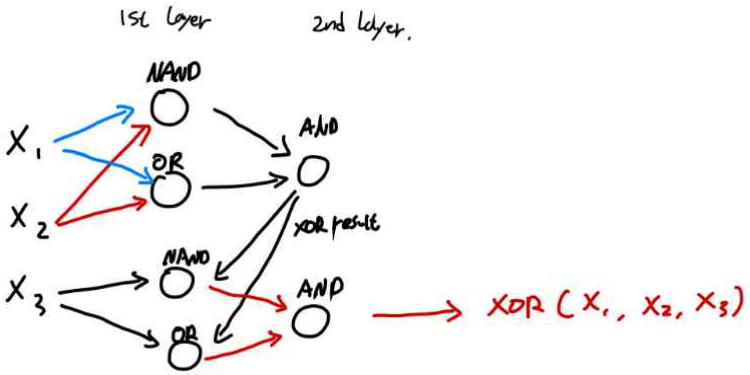
<코드>

```
perceptron <- function(x1, x2, w1, w2, b) {
  if(w1*x1 + w2*x2 + b <= 0) return(0)
  else return(1)
}
AND <- function(x1, x2) return(perceptron(x1, x2, 0.5, 0.5, -0.7))
OR <- function(x1, x2) return(perceptron(x1, x2, 0.5, 0.5, -0.2))
# Single Layer Perceptron으로는 XOR Operation을 계산할 수 없으므로,,,
# NAND(!AND) X OR Operation Perceptron으로 XOR Operation을 구현할 수 있다.
XOR <- function(x1, x2) return(AND(!AND(x1,x2), OR(x1, x2)))
# x1 XOR x2 XOR x3 == (x1 XOR x2) XOR x3 == x1 XOR (x2 XOR x3) 이므로
XOR_3Inputs <- function(x1,x2,x3) return(XOR(x1, XOR(x2, x3)))
XOR_3Inputs(0,0,0); XOR_3Inputs(0,0,1); XOR_3Inputs(0,1,0); XOR_3Inputs(0,1,1); XOR_3Inputs(1,0,0); XOR_3Inputs(1,0,1); XOR_3Inputs(1,1,0); XOR_3Inputs(1,1,1)
```

<XOR Operation>



<Black Box>



<출력 결과>

```
> perceptron ← function(x1, x2, w1, w2, b) {
+   if(w1*x1 + w2*x2 + b ≤ 0) {
+     return(0)
+   } else {
+     return(1)
+   }
+ }
>
> AND ← function(x1, x2) {
+   return(perceptron(x1, x2, 0.5, 0.5, -0.7))
+ }
> AND(0,0); AND(0,1); AND(1,0); AND(1,1)
[1] 0
[1] 0
[1] 0
[1] 1
> OR ← function(x1, x2) {
+   return(perceptron(x1, x2, 0.5, 0.5, -0.2))
+ }
> OR(0,0); OR(0,1); OR(1,0); OR(1,1)
[1] 0
[1] 1
[1] 1
[1] 1
```

```
> XOR.three.inputs ← function(x1,x2,x3){
+   return(XOR(XOR(x1, x2), x3))
+ }
> XOR.three.inputs(0,0,0)
[1] 0
> XOR.three.inputs(0,0,1)
[1] 1
> XOR.three.inputs(0,1,0)
[1] 1
> XOR.three.inputs(0,1,1)
[1] 0
> XOR.three.inputs(1,0,0)
[1] 1
> XOR.three.inputs(1,0,1)
[1] 0
> XOR.three.inputs(1,1,0)
[1] 0
> XOR.three.inputs(1,1,1)
[1] 1
> 
```

3. XNOR operation 퍼셉트론을 구현하시오

<코드>

```
perceptron <- function(x1, x2, w1, w2, b) {
  if(w1*x1 + w2*x2 + b <= 0) return(0)
  else return(1)
}
AND <- function(x1, x2) {
  return(perceptron(x1, x2, 0.5, 0.5, -0.7))
}
OR <- function(x1, x2) {
  return(perceptron(x1, x2, 0.5, 0.5, -0.2))
}
XNOR <- function(x1, x2){
  return(OR(AND(x1,x2), !OR(x1, x2)))
}
XNOR(0,0)
XNOR(0,1)
XNOR(1,0)
XNOR(1,1)
```

<출력 결과>

```
> perceptron ← function(x1, x2, w1, w2, b) {
+   if(w1*x1 + w2*x2 + b ≤ 0) {
+     return(0)
+   } else {
+     return(1)
+   }
+ }
>
> AND ← function(x1, x2) {
+   return(perceptron(x1, x2, 0.5, 0.5, -0.7))
+ }
> OR ← function(x1, x2) {
+   return(perceptron(x1, x2, 0.5, 0.5, -0.2))
+ }
> XNOR ← function(x1, x2){
+   return(OR(AND(x1,x2), !OR(x1, x2)))
+ }
> XNOR(0,0)
[1] 1
> XNOR(0,1)
[1] 0
> XNOR(1,0)
[1] 0
> XNOR(1,1)
[1] 1
> >
```

4. 입력변수(X) 중 1이 5개 이상일 때, 출력변수(Y)=1을 출력하는 퍼셉트론을 구현하시오

```
# rbind를 위한 임의 데이터 선언
df <- data.frame(t(rep(NA, 9)))
# 9개의 반복문을 사용하여, data.frame에 순차적으로 숫자 데이터를 적는다.
for(x1 in 0:1)
```

```
  for(x2 in 0:1)
    for(x3 in 0:1)
      for(x4 in 0:1)
        for(x5 in 0:1)
          for(x6 in 0:1)
            for(x7 in 0:1)
              for(x8 in 0:1)
                for(x9 in 0:1)
                  df <- rbind.data.frame(df, c(x1,x2,x3,x4,x5,x6,x7,x8,x9))
# 초기에 생성한 임의 데이터 배제
df <- df[-1,]
# 문제의 조건에 맞게, x의 개수(sum == 5)일 경우, 1, 반대의 경우 -1을 Y column에 기록
df$Y <- ifelse(rowSums(df) >= 5, 1, -1)
# 경사하강법 기반으로 데이터 학습을 진행하므로, 열 벡터 1을 추가한 X를 정리하였으며, y, w를 선언하였다.
X <- cbind(1, df[,-10]) %>% as.matrix
y <- df$Y %>% as.matrix
w <- rep(0, ncol(X)) %>% as.matrix
# 학습률을 0.3으로 지정하였다.
lambda <- 0.3
# 단지 w는 열 벡터의 형태를 가지므로 기록을 위해 Transpose 과정을 거쳤다.
W <- t(w) %>% as.data.frame
# 편의를 위해 각 열에 이름을 붙혀주었다.
names(W) <- paste0("w", 0:(length(w)-1))


# 무한 반복을 통해, X와 w간 행렬 곱을 하였을 때 예측되는 값이 0보다 작다면, y.hat을 -1, 반대의 경우 1로 기록하도록 하였으며, 이는 매 반복문에서 갱신되는 Weight를 기반하여 갱신된다.
# 끝으로 모든 예측 값과 실제 Y 값이 같다면 종료한다.
repeat{
  for(i in 1:nrow(X)) {
    y.hat <- ifelse(X %*% w < 0, -1, 1)
    w <- w + lambda * (y[i] - y.hat[i]) * t(X)[,i]
    W <- rbind.data.frame(W, as.vector(w))
  }
  if(all(y == y.hat)) break
}
data_index <- sample(1:512, 10)
View(data.frame(df[data_index, ], y.hat[data_index]))
```

<출력 결과>

	X1	X2	X3	X4	X5	X6	X7	X8	X9	Y	y.hat.data_index.
375	1	0	1	1	1	0	1	0	1	1	1
349	1	0	1	0	1	1	0	1	1	1	1
127	0	0	1	1	1	1	1	0	1	1	1
13	0	0	0	0	0	1	0	1	1	-1	-1
305	1	0	0	1	0	1	1	1	1	1	1
139	0	1	0	0	0	1	0	0	1	-1	-1
113	0	0	1	1	0	1	1	1	1	1	1
264	1	0	0	0	0	0	1	1	0	-1	-1
413	1	1	0	0	1	1	0	1	1	1	1
266	1	0	0	0	0	1	0	0	0	-1	-1