Lista 1: Ajustando uma RNA ‘no braço’

César A. Galvão - 190011572

Para essa lista, é considerada a seguinte arquitetura de uma rede neural *feed-forward*:

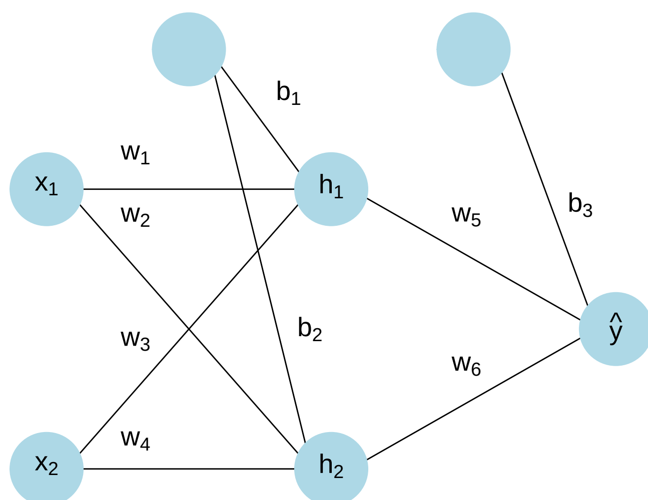

Figura 1: Arquitetura da rede neural artificial. Adotamos função de ativação sigmoide e linear nas camadas escondidas e de saída, respectivamente.

Questão 1

Item a

Crie uma função computacional para calcular o valor previso da variável resposta $\hat{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ em função de $\boldsymbol{x}$ e $\boldsymbol{\theta}$. Use a função para calcular $\hat{y}$ para $\boldsymbol{\theta} = (0.1, \ldots, 0.1)$ e $\boldsymbol{x} = (1, 1)$.

Implementa-se de forma matricial. A função não é adaptativa ao tamanho da rede e exige que o usuário forneça uma lista $\boldsymbol{\theta}$ com os seguintes elementos:

1. $W^{(1)}$ - matriz 2×2 de pesos da camada de entrada $\begin{pmatrix} w_1 & w_2 \\ w_3 & w_4 \end{pmatrix}$. Cada linha deve representar os pesos de cada neurônio para o neurônio subsequente, i.e. w_{ij} representa o peso do neurônio de entrada i para o próximo neurônio j.
2. $\boldsymbol{b}^{(1)}$ - vetor de viés da camada de entrada $\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$.
3. $W^{(2)}$ - matriz 2×1 de pesos da camada de saída $\begin{pmatrix} w_5 \\ w_6 \end{pmatrix}$.
4. $\boldsymbol{b}^{(2)}$ - vetor elemento de viés da camada de saída $\begin{pmatrix} b_3 \end{pmatrix}$.

Como função de ativação foi escolhida a função sigmóide denotada por $\phi = \frac{1}{1+e^{-x}}$. A função de previsão é dada por:

$$\hat{y} = W^{(2)\top}\phi(W^{(1)\top}\boldsymbol{x} + \boldsymbol{b}^{(1)}) + \boldsymbol{b}^{(2)}$$

ou

$$\begin{aligned} \boldsymbol{a} =& W^{(1)\top}\boldsymbol{x} + \boldsymbol{b}^{(\mathbf{1})} \\ \boldsymbol{h} =& \phi(\boldsymbol{a}) \\ \hat{y} =& W^{(2)\top}\boldsymbol{h} + b^{(2)} \end{aligned}$$

```
# Função de ativação
phi <- function(x) {1/(1 + exp(-x))}

# Função de previsão
ffwd <- function(x, theta) {

  if(any(dim(theta[[1]]) != c(2, 2))){
    stop("O primeiro elemento de theta deve ser uma matriz de
    ↪  pesos 2x2 para a aplicação nos dados")
  } else if(length(theta[[2]]) != 2){
    stop("O segundo elemento de theta deve ser um vetor viés de
    ↪  tamanho 2 para somar aos dados")
  } else if(any(dim(theta[[3]]) != c(2, 1))){
    stop("O terceiro elemento de theta deve ser uma matriz de
    ↪  pesos 2x1 para aplicação na única camada h")
  } else if(length(theta[[4]]) != 1){
    stop("O quarto elemento de theta deve ser um vetor viés de
    ↪  tamanho 1 para soma na única camada h")
  } else if(!is.vector(x) & length(x) != 2){
    stop("O x deve ser um vetor de tamanho 2")
  }
```

```r
  x <- as.vector(x)

  W1 <- theta[[1]]
  b1 <- theta[[2]]
  W2 <- theta[[3]]
  b2 <- theta[[4]]

  a1 <- (t(W1)%*%x)+b1
  h <- phi(a1)
  yhat <- (t(W2)%*%h)+b2

  return( # separacao dos elementos de saída para usar no
  ↪  backpropagation
    list(yhat = as.double(yhat),
         hidden = h,
         pre_activation = a1)
  )
}
```

Agora vamos calcular $\hat{y}$ para $\boldsymbol{\theta} = (0.1, \dots, 0.1)$ e $\boldsymbol{x} = (1, 1)$.

```r
x <- c(1,1)

theta <- list(
  matrix(c(0.1), nrow = 2, ncol = 2), #w1
  c(0.1, 0.1), #b1
  matrix(c(0.1), nrow = 2, ncol = 1), #w2
  c(0.1) #b2
)

ffwd(x, theta)
```

Obtemos $\hat{y} = 0.2148885$.

Item b

Crie uma rotina computacional para calcular a função de custo $J(\theta)$. Em seguida, divida o conjunto de dados observados de modo que as **primeiras** 80.000 amostras componham o conjunto de **treinamento**, as próximas 10.000 o de **validação**, e as **últimas** 10.000 o de **teste**. Qual é o custo da rede no **conjunto de teste** quando $\boldsymbol{\theta} = (0.1, ..., 0.1)$?

Primeiro são gerados os dados conforme as instruções da lista:

```
### Gerando dados "observados"
set.seed(1.2024)
m.obs <- 100000
dados <- tibble(x1.obs=runif(m.obs, -3, 3),
                x2.obs=runif(m.obs, -3, 3)) %>%
         mutate(mu=abs(x1.obs^3 - 30*sin(x2.obs) + 10),
                y=rnorm(m.obs, mean=mu, sd=1))
```

Depois, implementamos a função de custo $J(\theta)$, que é dada por

$$
\begin{aligned}
J(\boldsymbol{\theta}) =& \frac{1}{m} \sum_{i=1}^{m} L\left(f(x_{1i}, x_{2i}; \boldsymbol{\theta}), y_i\right) \\
=& \frac{1}{m} \sum_{i=1}^{m} \left(f(x_{1i}, x_{2i}; \boldsymbol{\theta}) - y_i\right)^2 \\
=& \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}_i - y_i\right)^2,
\end{aligned}
$$

onde m é o número de observações.

```
J.Loss <- function(dados, theta, target){

  if(!is.data.frame(dados) | !is_tibble(dados)){
    stop("Dados deve ser uma dataframe ou tibble.")
  } else if (dim(dados)[2] != 2){
    stop("Matriz de dados deve ter 2 colunas.")
  } else if (!is.list(theta)){
    stop("Theta deve ser uma lista com pesos e viéses.")
  } else if (!is.numeric(target)){
    stop("Target deve ser um vetor numérico.")
  }

# aloca memoria para o tamanho dos dados
  yhat <- double(nrow(dados))
# transpoe os dados para termos acesso aos vetores de x que serao
↪  passados para a primeira camada da rede
  tdados <- t(as.matrix(dados))

  for(i in 1:ncol(tdados)){
    yhat[i] <- ffwd(tdados[,i], theta)$yhat
  }
```

```
  return(mean((target - yhat)^2)) #média ja entrega soma/m
}
```

Em seguida, separamos o nosso conjunto de dados em treinamento, validação e teste.

```
dados_train <- dados[1:80000,]
dados_valid <- dados[80001:90000,]
dados_test <- dados[90001:nrow(dados),]
```

Finalmente, executamos a função de *feed-forward* nos dados gerados para calcularmos $\hat{\boldsymbol{y}}$ e em seguida calculamos o custo da rede com $\boldsymbol{\theta} = (0.1, \ldots, 0.1)$.

```
# transformamos os dados em matriz
x_test <- dados_test %>%
  select(x1.obs, x2.obs)

# separamos o target
y_test <- dados_test %>%
  pull(y)

# theta já foi gerado e será reaproveitado
J.Loss(x_test, theta, y_test)
```

Obtemos um custo de 663.1286383.

Item c

Use a regra da cadeia para encontrar expressões algébricas para o vetor gradiente

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \left(\frac{\partial J}{\partial w_1}, \ldots, \frac{\partial J}{\partial b_3}\right).$$

Desejamos $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ tal que

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \left[\frac{1}{m} \sum_{i=1}^{m} (y - \hat{y}_i)^2 \right] \\
&= \frac{2}{m} \sum_{i=1}^{m} (y - \hat{y}_i) \nabla_{\boldsymbol{\theta}} (y - \hat{y}_i), \quad \text{pois } \hat{y}_i = f(x_{1i}, x_{2i}; \boldsymbol{\theta}) \\
&= \frac{2}{m} \sum_{i=1}^{m} (y - \hat{y}_i)(-1) \nabla_{\boldsymbol{\theta}} \hat{y}_i \qquad (1)
\end{aligned}
$$

Resolvemos o gradiente, considerando $\phi(x) = \sigma(x) = \frac{1}{1+e^{-x}}$,

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \hat{y}_i &= \nabla_{\boldsymbol{\theta}} \left[h_{1i} w_5 + h_{2i} w_6 + b_3 \right] \\
&= \nabla_{\boldsymbol{\theta}} \left[\phi(a_{1i}) w_5 + \phi(a_{2i}) w_6 + b_3 \right] \\
&= \nabla_{\boldsymbol{\theta}} \left[\phi(x_{1i} w_1 + x_{2i} w_3 + b_1) w_5 + \phi(x_{1i} w_2 + x_{2i} w_4 + b_2) w_6 + b_3 \right].
\end{aligned}
$$

É imediato que

$$
\frac{\partial \hat{y}_i}{\partial b_3} = 1, \quad \frac{\partial \hat{y}_i}{\partial w_5} = h_{1i}, \quad \frac{\partial \hat{y}_i}{\partial w_6} = h_{2i}.
$$

Para $b_j, j \in \{1, 2\}$ resolvemos de forma análoga:

$$
\begin{aligned}
\frac{\partial \hat{y}_i}{\partial b_1} &= w_5 \frac{\partial h_1}{\partial b_1} = w_5 \frac{\partial}{\partial b_1} \phi(x_{1i} w_1 + x_{2i} w_3 + b_1) \\
&= w_5 \frac{\partial}{\partial b_1} \left(\frac{1}{1 + e^{-(x_{1i} w_1 + x_{2i} w_3 + b_1)}} \right) \\
&= w_5 \frac{(-1) \cdot \frac{\partial}{\partial b_1} \left(1 + e^{-(x_{1i} w_1 + x_{2i} w_3 + b_1)} \right)}{\left(1 + e^{-(x_{1i} w_1 + x_{2i} w_3 + b_1)} \right)^2}
\end{aligned}
$$

e

$$
\begin{aligned}
\frac{\partial}{\partial b_1} \left(1 + e^{x_{1i} w_1 + x_{2i} w_3 + b_1} \right) &= 1 \cdot e^{x_{1i} w_1 + x_{2i} w_3 + b_1} \\
&= e^{x_{1i} w_1 + x_{2i} w_3 + b_1} = e^{a_1}.
\end{aligned}
$$

Portanto,

$$\frac{\partial \hat{y}_i}{\partial b_1} = w_5 \frac{-e^{a_1}}{(1+e^{a_1})^2}, \quad \text{e} \quad \frac{\partial \hat{y}_i}{\partial b_2} = w_6 \frac{-e^{a_2}}{(1+e^{a_2})^2}.$$

Para $w_j, j \in \{1,2,3,4\}$,

$$\begin{aligned}
\frac{\partial \hat{y}_i}{\partial w_1} &= w_5 \frac{\partial h_1}{\partial w_1} = w_5 \frac{-x_{1i}\, e^{a_1}}{(1+e^{a_1})^2} \\
\frac{\partial \hat{y}_i}{\partial w_2} &= w_6 \frac{-x_{1i}\, e^{a_2}}{(1+e^{a_2})^2} \\
\frac{\partial \hat{y}_i}{\partial w_3} &= w_5 \frac{-x_{2i}\, e^{a_1}}{(1+e^{a_1})^2} \\
\frac{\partial \hat{y}_i}{\partial w_4} &= w_6 \frac{-x_{2i}\, e^{a_2}}{(1+e^{a_2})^2}.
\end{aligned}$$

Finalmente, substituimos os as componentes na equação (1) e explicitamos cada componente do gradiente[1]:

[1] Essa notação poderia ser escrita de forma mais elegante e sucinta. No entanto, dessa forma facilita a comparação entre a expressão analítica e a implementação computacional pelo leitor.

$$
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial J}{\partial w_1} \\ \frac{\partial J}{\partial w_2} \\ \frac{\partial J}{\partial w_3} \\ \frac{\partial J}{\partial w_4} \\ \frac{\partial J}{\partial w_5} \\ \frac{\partial J}{\partial w_6} \\ \frac{\partial J}{\partial b_1} \\ \frac{\partial J}{\partial b_2} \\ \frac{\partial J}{\partial b_3} \end{pmatrix} = \begin{pmatrix} \frac{2}{m} \sum_{i=1}^{m} (y - \hat{y}_i)\, w_5\, x_{1i} \frac{e^{a_{1i}}}{(1+e^{a_{1i}})^2} \\ \frac{2}{m} \sum_{i=1}^{m} (y - \hat{y}_i)\, w_6\, x_{1i} \frac{e^{a_{2i}}}{(1+e^{a_{2i}})^2} \\ \frac{2}{m} \sum_{i=1}^{m} (y - \hat{y}_i)\, w_5\, x_{2i} \frac{e^{a_{1i}}}{(1+e^{a_{1i}})^2} \\ \frac{2}{m} \sum_{i=1}^{m} (y - \hat{y}_i)\, w_6\, x_{2i} \frac{e^{a_{2i}}}{(1+e^{a_{2i}})^2} \\ \frac{2}{m} \sum_{i=1}^{m} (y - \hat{y}_i)\, (-h_{1i}) \\ \frac{2}{m} \sum_{i=1}^{m} (y - \hat{y}_i)\, (-h_{2i}) \\ \frac{2}{m} \sum_{i=1}^{m} (y - \hat{y}_i)\, w_5 \frac{e^{a_{1i}}}{(1+e^{a_{1i}})^2} \\ \frac{2}{m} \sum_{i=1}^{m} (y - \hat{y}_i)\, w_6 \frac{e^{a_{2i}}}{(1+e^{a_{2i}})^2} \\ \frac{2}{m} \sum_{i=1}^{m} (y - \hat{y}_i)(-1) \end{pmatrix} \tag{2}
$$

Item d

Crie uma função computacional que receba como entrada o vetor θ, uma matriz design (x) e as respectivas observações (y) e forneça, como saída, o gradiente definido no item c). Apresente o resultado da função aplicada sobre o banco de treinamento, quando $\theta = (0.1, \ldots, 0.1)$. Atenção: implemente o algoritmo *back-propagation* (Algoritmo 6.4 do livro Deep Learning) para evitar realizar a mesma operação múltiplas vezes.

---

na função, para responder de D a F:

function(theta, dados, target, optimize = TRUE, learning_rate = NULL, epochs = NULL, plot = FALSE)

- optimize: se true, otimiza. Se false, retorna o gradiente. padrao é true.
- learning rate

- iteracoes: numero de iteracoes
- plot: se true, gera um elemento da lista de saída com o custo em cada iteracao. padrao é false.

```
backpropagation <- function(theta, dados, target, optimize =
↪  TRUE, learning_rate = NULL, epochs = NULL, plot = FALSE){

  if(!is.list(theta)){
    stop("Theta deve ser uma lista com pesos e viéses.")
  } else if(!is.data.frame(dados) | !is.tibble(dados)){
    stop("Dados deve ser uma dataframe ou tibble.")
  } else if(!is.numeric(target)){
    stop("Target deve ser um vetor numérico.")
  } else if(!is.logical(optimize)){
    stop("Optimize deve ser um valor lógico.")
  } else if(!is.numeric(learning_rate) &
↪  !is.null(learning_rate)){
    stop("Learning rate deve ser um número real.")
  } else if(!is.numeric(epochs) & !is.null(epochs)){
    stop("Epochs deve ser um número inteiro.")
  } else if(!is.logical(plot)){
    stop("Plot deve ser um valor lógico.")
  }

# y é o target
# yhat é o resultado da ffwd$yhat
# todos ws vem da theta
# x vem dos dados
# as vem da ffwd$pre_activation
# h vem da ffwd$hidden

}
```