Assignment 2

Exercise 1

Arthur Junges Schmidt

08 July 2020

Exercise 1a

```
Input1 <- cbind(5, 0.5, 2)
Desired_Output <- 1
learn_rate <- 5
Hidden_Nodes <- 2
Number_Iterations <- 2
Weights_Input <- matrix(0.1, nrow = ncol(Input1), ncol = Hidden_Nodes)
Weights_Output <- matrix(0.5, Hidden_Nodes, 1)



train <- function(x, y, hidden, learn_rate, iterations, Weights1, Weights2) {
  d <- ncol(x)
  w1 <- matrix(Weights1, d, hidden)
  w2 <- matrix(Weights2, nrow = hidden , 1)
  for (i in 1:iterations) {
    ff <- feed_forward(x, w1, w2)
    bp <- feed_backward(x, y,
                        y_hat = ff$output,
                        w1, w2,
                        h = ff$h,
                        learn_rate = learn_rate)
    w1 <- bp$w1; w2 <- bp$w2
  }
  list(output = ff$output, w1 = w1, w2 = w2)

}


feed_forward <- function(x = Input1, w1, w2){
  z1 <- cbind(x) %*% w1
  h <- sigmoid(z1)
  z2 <- cbind(h) %*% w2
  list(output = sigmoid(z2), h = h)
}

sigmoid <- function(x) {
  1 / (1 + exp(-x))
}

feed_backward <- function(x, y = Desired_Output, y_hat, w1, w2, h, learn_rate) {

  dw2 <- (y_hat - y) * y_hat * (1 - y_hat) * as.vector(h)

  dh <- (as.double(y_hat) - y) * as.double(y_hat) * (1 - as.double(y_hat)) %*% w2[1]
```

```
  dw1 <- t(x) %*% (h *  (1 - h) * c(dh))



  w1 <- w1 - learn_rate * dw1
  w2 <- w2 - learn_rate * dw2

  list(w1 = w1, w2 = w2)
}

nnet <- train(x = Input1, y = Desired_Output,
              hidden = Hidden_Nodes,
              learn_rate = learn_rate,
              iterations = Number_Iterations,
              Weights1 = Weights_Input,
              Weights2 = Weights_Output)
```

Exercise 1b

```
# Clear all ---------------------------------------------------------------
rm(list=ls())
gc()
cat("\014")

# Inputs ------------------------------------------------------------------
library(profvis)
#Input1 <- cbind(5, 0.5, 2)
#Desired_Output <- 1
learn_rate <- 5
Hidden_Nodes <- 4
#Number_Iterations <- 2
#Weights_Input <- matrix(0.1, nrow = ncol(Data_set_Y), ncol = Hidden_Nodes)
#Weights_Output <- matrix(0.5, Hidden_Nodes, 1)

# Sigmoid Function --------------------------------------------------------


sigmoid <- function(x) {
  1 / (1 + exp(-x))
}


# Generate sample data ----------------------------------------------------
a1 <- cbind(c(3, 5), c(2, 7), c(3, 8));
a2 <- cbind(c(3, 5), c(-2, -7), c(3, 8));

# Initialize Y variable with 2 rows and 1000 columns
Data_set_Y <- data.frame(replicate(1000, numeric(2))); # Y has 1000 columns and 2 values

# Initialize X variable with 3 rows and 1000 columns
Input_Values_X <- matrix(data = 0, nrow = 3, ncol = 1000);

# Assign the variable with the provided equation
for (n in 1:length(Data_set_Y)) {
  Input_Values_X[,n] <- rnorm(3);
  Data_set_Y[n] <- sigmoid(a1 %*% Input_Values_X[,n]) +
    ((a2 %*% Input_Values_X[,n])^2) + 0.30 * rnorm(2);
}

### ok!
# Dividing Sample data ----------------------------------------------------

#Data will be divided in 70% training data and 30% test data
```

```r
## Sample size
Sample_size <- floor(0.75 * NCOL(Data_set_Y));

## Set the seed to make the partition reproducible
set.seed(91374)

Training_Columns <- sample(seq_len(ncol(Data_set_Y)), Sample_size);
# Choose randomly which collumns will compose the training set

Output_Train_set <- Data_set_Y[, Training_Columns];
Output_Test_set <- Data_set_Y[, -Training_Columns];
Input_Train_set <- Input_Values_X[, Training_Columns];
Input_Test_set <- Input_Values_X[, -Training_Columns];


# Training ----------------------------------------------------------------

# The weights now are N(0,1)

train <- function(Input, Output, hidden, learn_rate, iterations) {
  Number_Inputs <- nrow(Input) # Now inputs are organized by columnns, so the size is 'nrow'
  Weight_Input_Hidden <- matrix(rnorm(Number_Inputs * hidden), hidden, Number_Inputs)

  Weight_Hidden_Output <- matrix(rnorm(hidden * 2), nrow = 2, ncol = hidden)
  # Size of output is 2

  ## Initialize Train Error Matrix --

  Error_Test_Matrix <- matrix(0, nrow = ncol(Input), ncol = 2);

  ## For loops --

  for (j in 1:iterations) {


    for (i in 1:ncol(Input)) {
    ff <- feed_forward(Input, w1 = Weight_Input_Hidden, w2 = Weight_Hidden_Output,
                       Sample_Number = i)

    bp <- feed_backward(Input, Output,
                        y_hat = ff$Output_Activated,
                        Weight_Input_Hidden, Weight_Hidden_Output,
                        Hidden_Activated = ff$Hidden_Activated,
                        learn_rate = learn_rate,
                        Sample_Number = i)
    Weight_Input_Hidden <- bp$w1;
    Weight_Hidden_Output <- bp$w2

    Error_Test_Matrix[i,] <- c(i, bp$error);

    }
    # print(Weight_Input_Hidden)
    # print(Weight_Hidden_Output)
    # print(j)
    # print(Error_Test_Matrix)
```

```
  }
  list(output = ff$output, w1 = Weight_Input_Hidden,
       w2 = Weight_Hidden_Output,
       Error_Test_Matrix = Error_Test_Matrix)

}


feed_forward <- function(x, w1, w2, Sample_Number){

  Hidden_Activated <- sigmoid(w1 %*% x[,Sample_Number]);
  Output_Activated <- sigmoid(w2 %*% Hidden_Activated); ### OK!
  # z1 <- cbind(x) %*% w1
  # h <- sigmoid(z1)
  # z2 <- cbind(h) %*% w2
  # list(output = sigmoid(z2), h = h)

  result <- list(Hidden_Activated = Hidden_Activated, Output_Activated = Output_Activated);
  return(result)
}


feed_backward <- function(Input, Output, y_hat, w1, w2, Hidden_Activated,
                          learn_rate,
                          Sample_Number) {

  Error <- (sum((y_hat - Output[, Sample_Number])^2))*0.5

  dw2 <- ((y_hat - Output[,Sample_Number]) * y_hat * (1 - y_hat)) %*% t(Hidden_Activated)
  ## Hidden_activated is [4x1] so it needs to be transposed to obtain a result dw2[2x4]
  ## > Dot Multiplication!

  dw1 <- ((t(w2) %*% (y_hat - Output[, Sample_Number])) *
            (Hidden_Activated * (1 - Hidden_Activated))) %*%
    t(Input[, Sample_Number]);


  w1 <- w1 + learn_rate * dw1;
  w2 <- w2 + learn_rate * dw2;

  list(w1 = w1, w2 = w2, error = Error)
}
profvis({
Results_0.5 <- train(Input = Input_Train_set, Output = Output_Train_set, hidden = Hidden_Nodes,
                 learn_rate = 0.5,
                 iterations = 1000)
})
Results_5 <- train(Input = Input_Train_set, Output = Output_Train_set, hidden = Hidden_Nodes,
                   learn_rate = 5,
                   iterations = 1000)
```