Exercie 1: Linear Decision Boundaries

Part A

Lets load and important the iris data set

```
iris_data <- read.csv("./irisdata.csv")
summary(iris_data)
```

```
##   sepal_length    sepal_width     petal_length    petal_width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.054   Mean   :3.759   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
# Lets look at the head of the iris data set
head(iris_data)
```

```
##   sepal_length sepal_width petal_length petal_width species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
# Number of rows of the iris dataset (number of individual observations)
nrow(iris_data)
```

```
## [1] 150
```

```
# All of the columns are numeric except for the species column
# They are continuous because they correspond to length and width (except for species) # The species ar
sapply(iris_data, class)
```

```
## sepal_length  sepal_width petal_length  petal_width      species
##    "numeric"    "numeric"    "numeric"    "numeric"     "factor"
```

```
# The only categorical variable is species in this iris dataset
sapply(iris_data, class)
```

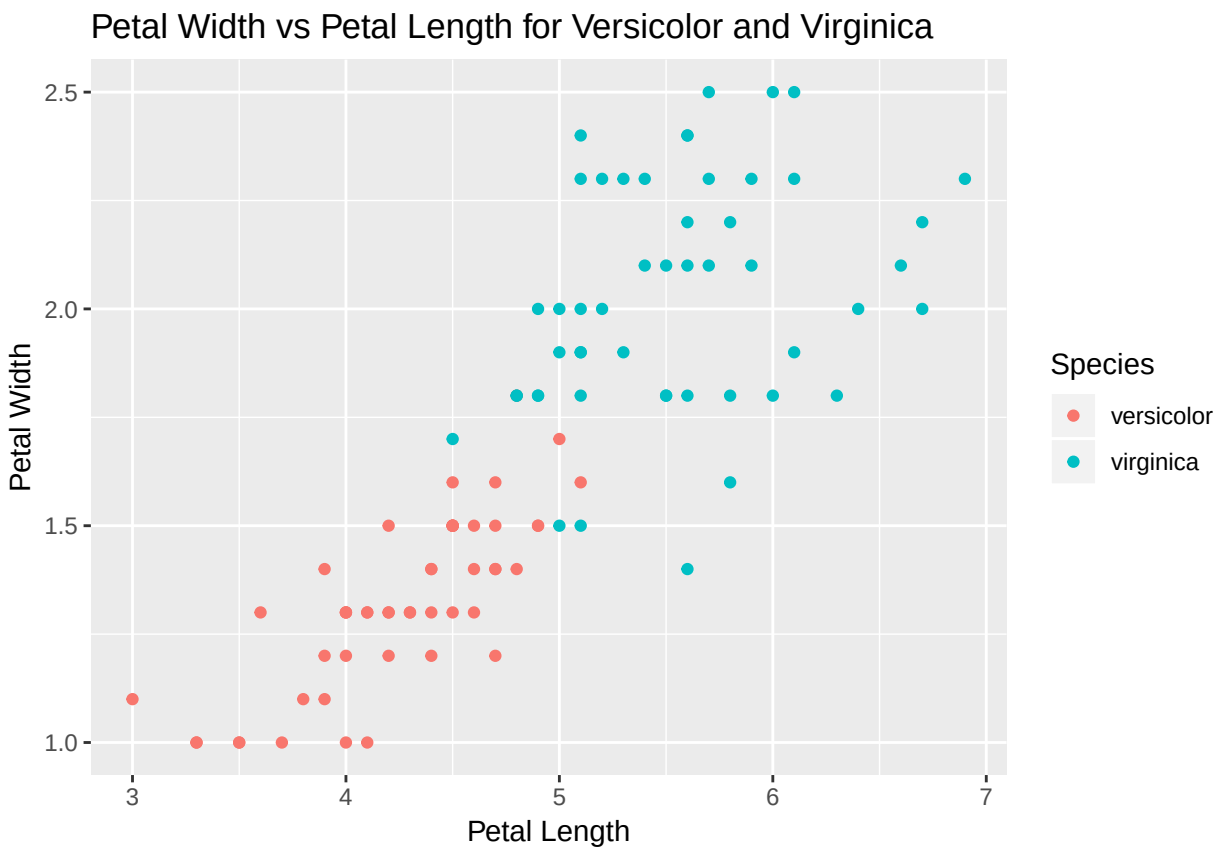
```
## sepal_length  sepal_width petal_length  petal_width      species
##    "numeric"    "numeric"    "numeric"    "numeric"     "factor"
```

From our observations it looks like there are 50 samples of three species: setosa, versicolor and virginica. Each of the samples has measurements of sepal lenght, sepal width and petal length.

Let's plot the second and third iris classses:

```r
# Lets pick the 2nd and 3rd classes: versicolor and virginica and let's plot the subset
plot1 <- iris_data %>% filter(species == 'versicolor' | species == 'virginica') %>%
ggplot(aes(x = petal_length, y = petal_width, color = species)) + geom_point() +
labs(color = "Species") + xlab("Petal Length") + ylab("Petal Width") +
ggtitle("Petal Width vs Petal Length for Versicolor and Virginica")

plot1
```



Part B

The following function will take in five inputs (c0, c1, c2) which are constants that will be estimated later in this exercise and (x1, x2) which correspond to petal width and petal length. The output will return a value between a probability value between 0 and 1 where all values above 0.5 will correspond to the 3rd iris class and values below 0.5 to the 2nd iris class.

```r
# inputs: c0, c1, c2, petal_width, petal_length
# outputs: value between 0 and 1 representing the probability

one_layer_neural_network <- function(c0, c1, c2, petal_width, petal_length) {

  # Linear function
  z <- c0 + c1*petal_length + c2*petal_width

  #result
  result <- 1/(1 + exp(-z))
```

```
  return (result)
}
```

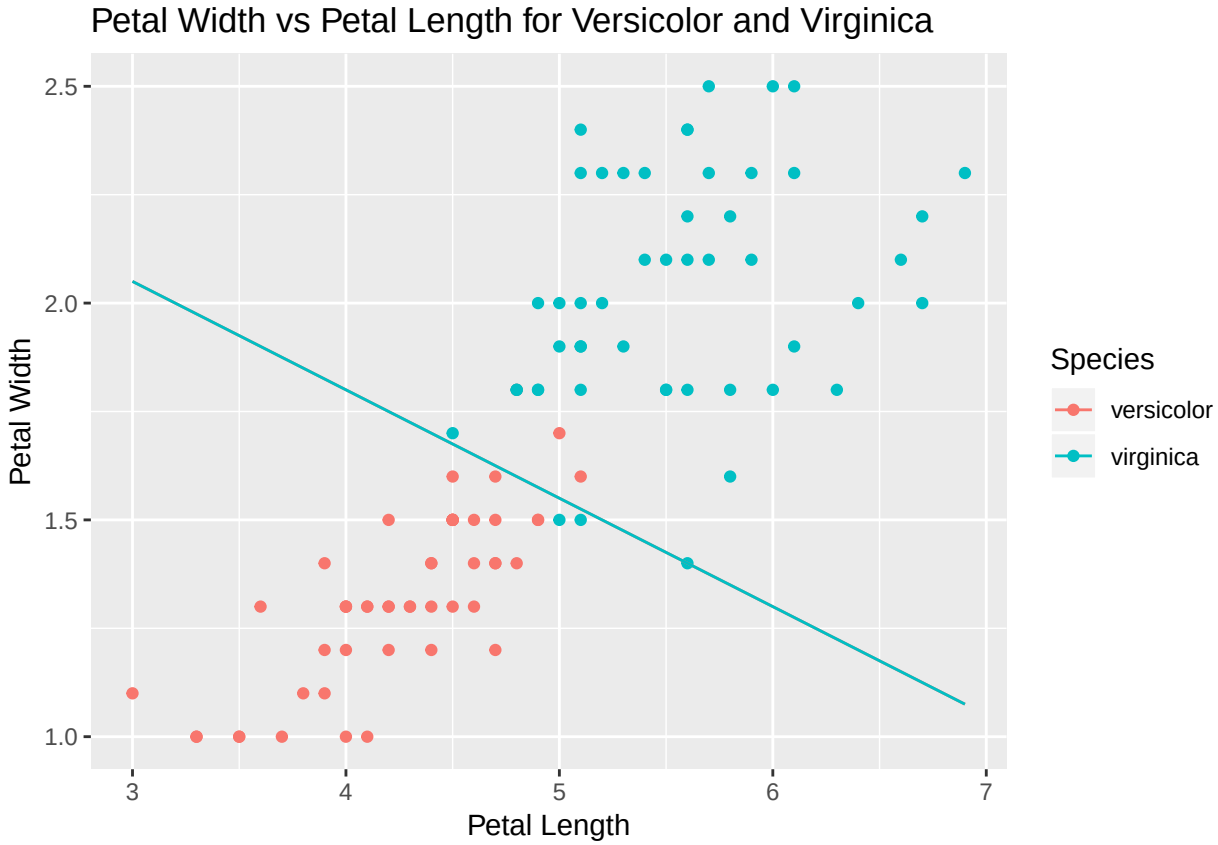
Part C

We are going to estimate the constant values such as c0 = 2.8, c1 = -1/4 and c2 = 1.

```
z_function <- function(x) (2.8 + (-1/4)*x)

# Lets pick the 2nd and 3rd classes: versicolor and virginica and let's plot the subset
plot2 <- iris_data %>% filter(species == 'versicolor' | species == 'virginica') %>%
ggplot(aes(x = petal_length, y = petal_width, color = species)) + geom_point() +
labs(color = "Species") + xlab("Petal Length") + ylab("Petal Width") +
ggtitle("Petal Width vs Petal Length for Versicolor and Virginica") + stat_function(fun = z_function)

plot2
```



Part D

UGH 3D???

Part E

Now lets see how are function from question b performs:

```r
# Versicolor
data1 <- iris_data %>% filter (species == 'versicolor')

for (i in 1:50){

  if (one_layer_neural_network(2.8, -1/4, -1, data1$petal_width[i], data1$petal_length[i]) < 0.5){

     resulting_flower <- "virginica"
  } else {
    resulting_flower <- "versicolor"
  }
  print(paste("actual: versicolor, predicted:", resulting_flower))
}
```

```
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: virginica"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: virginica"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: virginica"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
```

```
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
## [1] "actual: versicolor, predicted: versicolor"
```

```
# Virginica
data2 <- iris_data %>% filter (species == 'virginica')

for (j in 1:50){
  if (one_layer_neural_network(2.8, -1/4, -1, data2$petal_width[j], data2$petal_length[j]) < 0.5){

     resulting_flower <- "virginica"
  } else {
    resulting_flower <- "versicolor"
  }
  print(paste("actual: virginica, predicted: ", resulting_flower))
}
```

```
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  versicolor"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
```

```
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  versicolor"
## [1] "actual: virginica, predicted:  versicolor"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
## [1] "actual: virginica, predicted:  virginica"
```

Results match to what we have seen in the graph above.

Exercie 2: Neural networks

Part A

Mean squared error calculations. The inputs are in order: the data vectors (as the iris dataset that the petal length and petal width information is taken from), the parameters defining the neural network (w0, w1, w2). The pattern classes are computed inside the algorithm.

```
mean_squared_error <- function(data, w0, w1, w2) {

  # Lets setup a vector for the pattern classes
  pattern_classes <- rep(NA, 100)

  for (i in 1:100){
    pattern_classes[i] = one_layer_neural_network(w0, w1, w2, data$petal_width[i+50], data$petal_length
  }

  # what is returned by the program
  resulting_sum <- 0

  # Now lets compute the sum of all the errors
  for (j in 1:100){
    if (j <= 50){
      resulting_sum = resulting_sum + (pattern_classes[j] - 1)^2
    } else {
      esulting_sum = resulting_sum + (pattern_classes[j] - 0)^2
    }
  }
  return(resulting_sum)
}
```

Part B

I have decided to look at three different values. First, I computed the mean square error of the original

```
# Mean squared error for the one we graphed before
mean_squared_error(iris_data, 2.8, -1/4, -1)
```

```
## [1] 8.293887
```

```
# High error

# Low error
```