

Exercie 1: Linear Decision Boundaries

Part A

Lets load and import 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 are
```

```
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)
```

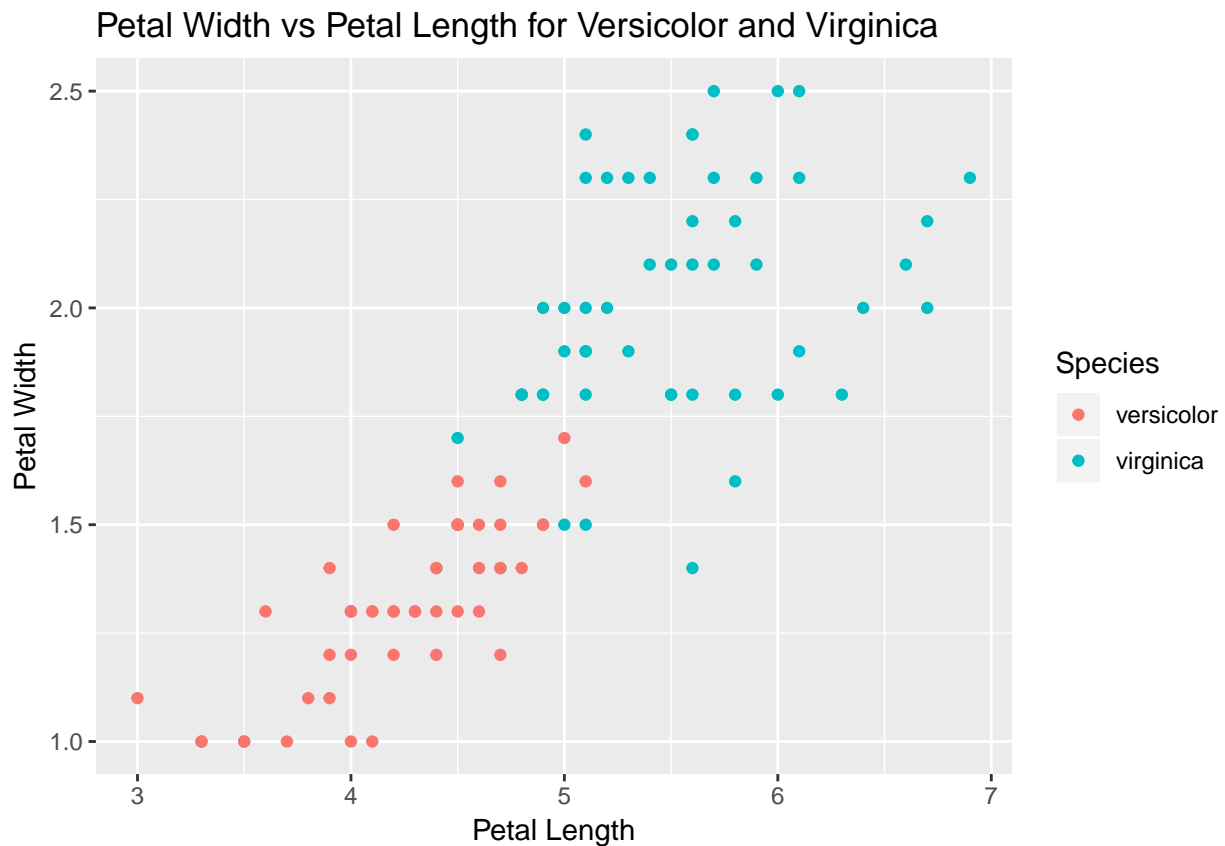
```
## sepal_length  sepal_width petal_length  petal_width    species
##   "numeric"    "numeric"   "numeric"   "numeric"    "factor"
```

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 length, sepal width and petal length.

Let's plot the second (versicolor) and third (virginica) iris classes:

```
# Lets pick the 2nd and 3rd classes: versicolor and virginica and let's plot the subset
plot1 <- iris_data %>% filter(species %in% c('versicolor','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 (w_0 , w_1 , w_2) which are constants that will be estimated later in this exercise and (x_1 , x_2) 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 greater than or equal to 0.5 will correspond to the 3rd iris class (this class will be denoted by 1), and values below 0.5 to the 2nd iris class (this class will be denoted by 0).

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

one_layer_neural_network <- function(w0, w1, w2, petal_length, petal_width) {

  # Linear function
  z <- w0 + w1*petal_length + w2*petal_width

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

```

    return (result)
}

```

Part C

By playing around with the parameters, we select $w_0 = -2.8$, $w_1 = 0.25$ and $w_2 = 1$, which gives a decision boundary that separates the data decently (see below). The boundary that separates the data arises by noting the following:

Starting with the condition that a given iris flower is in class 1 (virginica):

$$\sigma(\mathbf{w}^T \mathbf{x}) \geq 0.5$$

where σ is the logistic function, $\mathbf{w} = (w_0, w_1, w_2)$, and $\mathbf{x} = (1, x_1, x_2)$ (x_1 corresponds to petal length and x_2 corresponds to petal width). After a bit of manipulation, the above inequality implies the following:

$$\mathbf{w}^T \mathbf{x} \geq 0$$

Writing this out with our chosen values for the coefficients above gives us:

$$-2.8 + 0.25x_1 + x_2 \geq 0$$

Consequently, the decision boundary is given by the line $x_2 = 2.8 - 0.25x_1$ (above this line will be class 1 (virginica), and below this line will be class 0 (versicolor)). This result is plotted below:

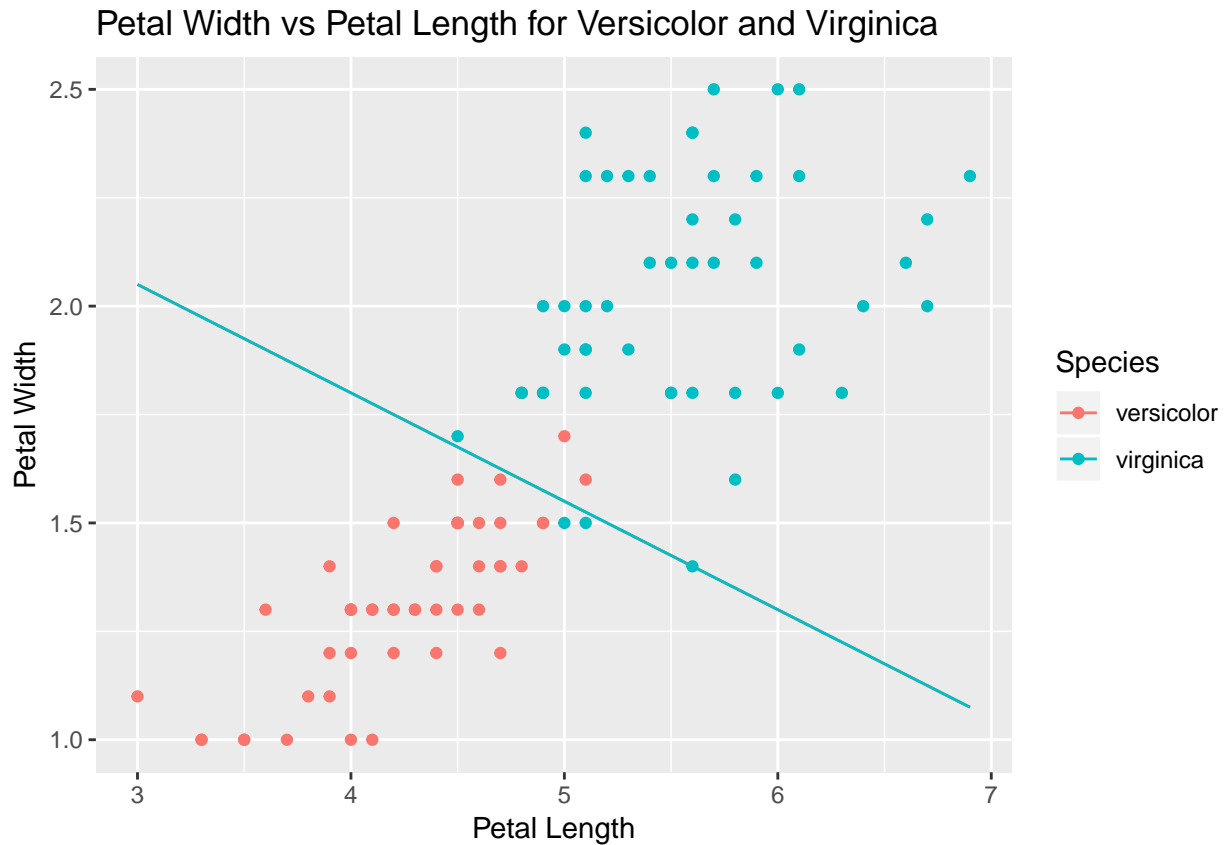
```

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 %in% c('versicolor', '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 let's see how are function from Part B performs:

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

for (i in 1:50){

  if (one_layer_neural_network(-2.8, 0.25, 1, data1$petal_length[i], data1$petal_width[i]) < 0.5){
    resulting_flower <- "versicolor"

  } else {

    resulting_flower <- "virginica"
  }
  print(paste0("actual: versicolor, predicted: ", resulting_flower))
}
```

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



```
} else {  
  resulting_flower <- "virginica"  
}  
print(paste0("actual: virginica, predicted: ", resulting_flower))  
}
```

[illegible]

```
## [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.

Exercise 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 (w_0 , w_1 , w_2).

```
# Input feature vectors
X_data <- iris_data %>% filter(species %in% c("versicolor", "virginica")) %>% select(petal_length, petal_width)
# Pattern classes for each feature vector above
y_data <- iris_data %>%
  filter(species %in% c("versicolor", "virginica")) %>%
  select(species) %>%
  mutate(label = ifelse(species == "versicolor", 0, 1)) %>%
  select(-species)

mean_squared_error <- function(X, y, w0, w1, w2) {

  # Outputs of neural network for each example
  NN_outputs <- rep(0, nrow(X))

  for (i in 1:nrow(X)){
    NN_outputs[i] = one_layer_neural_network(w0, w1, w2, X$petal_length[i], X$petal_width[i])
  }

  # Compute the differences between the NN outputs and the labels
  result = mean((NN_outputs - y$label)^2)

  # Return result
  return(result)
}
```

Part B

I have decided to look at three different values. First, I computed the mean square error of the original and got a mean square error of approximately 0.1489:

```
# Mean squared error for decision boundary we guessed above
paste0("Previous Mean Square Error: ", mean_squared_error(X_data, y_data, -2.8, 0.25, 1))
```

```
## [1] "Previous Mean Square Error: 0.148888121336409"
```

Then I made a completely incorrect estimate of the coefficients such that the boundary is not even touching any of the data points and got a much worse mean square error of 0.4401.

```
# High error
paste0("High Mean Square Error: ", mean_squared_error(X_data, y_data, 1, -0.25, -1))
```

```
## [1] "High Mean Square Error: 0.440092737113904"
```

```

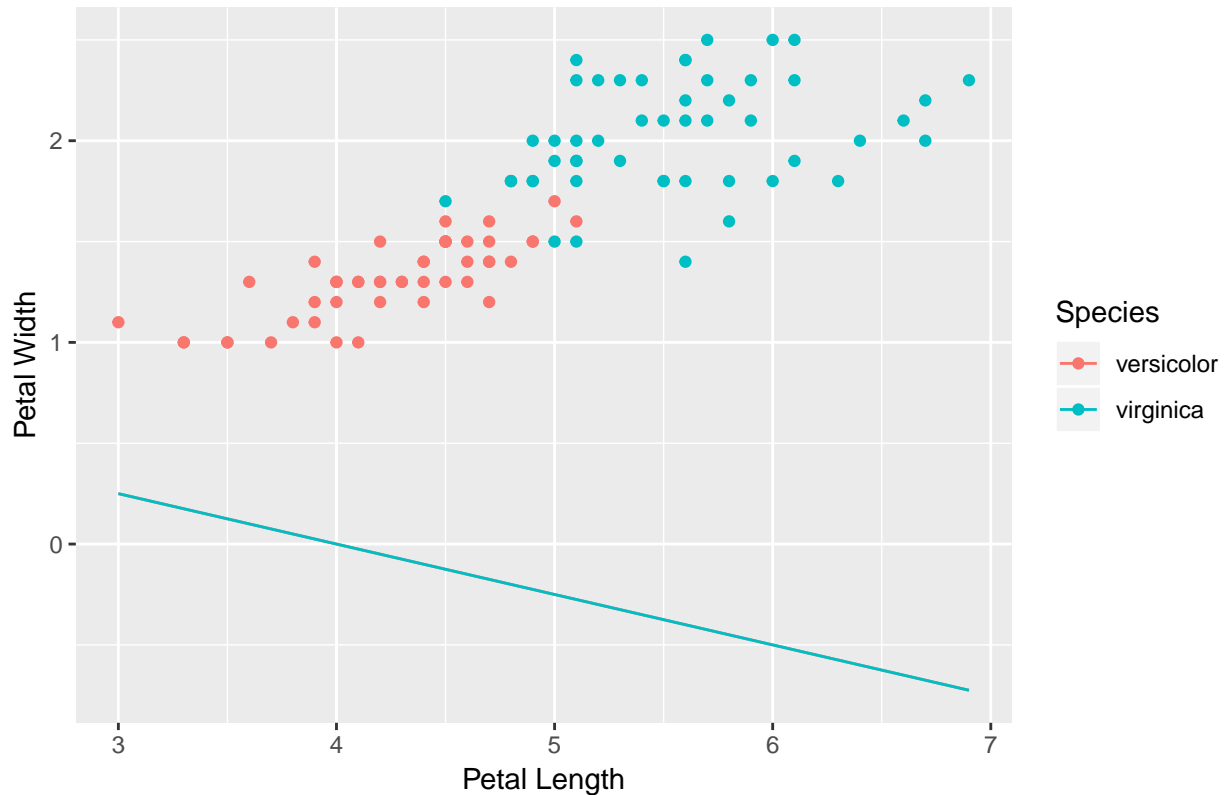
z_function2 <- function(x) (1 - (0.25)*x)

plot3 <- iris_data %>% filter(species %in% c('versicolor','virginica')) %>%
  ggplot(aes(x = petal_length, y = petal_width, color = species)) + geom_point() +
  labs(color = "Species") + xlab("Petal Length") + ylab("Petal Width") +
  ggtitle("High Mean Square Error: Petal Width vs Petal Length for Versicolor and Virginica") +
  stat_function(fun = z_function2)

```

plot3

High Mean Square Error: Petal Width vs Petal Length for Versicolor and Virgi



Finally, I found a slightly better boundary line that made only three points dislocated and decreased the mean square error slightly to 0.1423.

```

# Low error
paste0("Low Mean Square Error: ", mean_squared_error(X_data, y_data, -3.15, 1/3, 1))

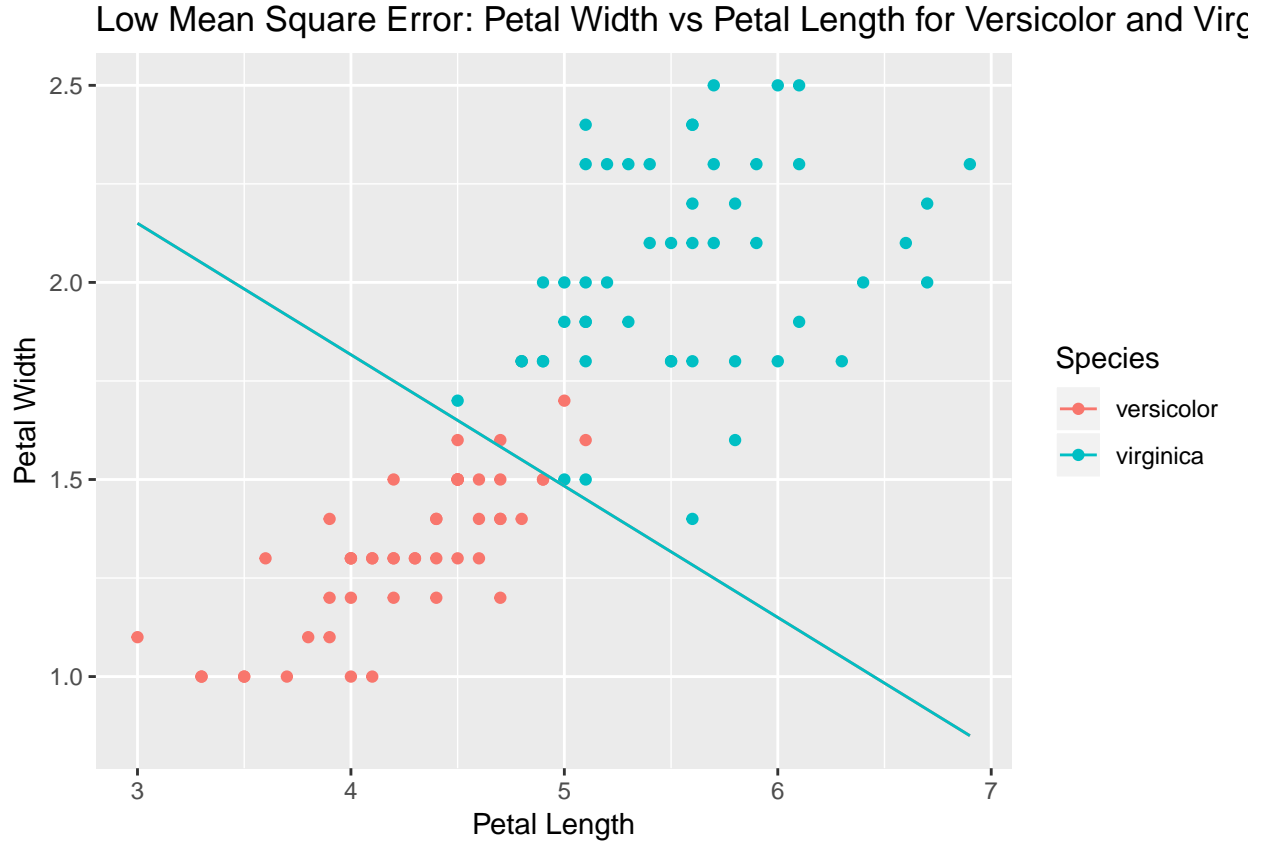
## [1] "Low Mean Square Error: 0.142297154674927"

z_function3 <- function(x) (3.15 + (-1/3)*x)

plot4 <- 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("Low Mean Square Error: Petal Width vs Petal Length for Versicolor and Virginica") +
  stat_function(fun = z_function3)

```

plot4



Part C

Before we begin with the derivation, we will set up some notation for convenience. We will denote the number of examples in our data as m , which is this case is 100. Our data will be collected in the following matrix:

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} \\ \vdots & \vdots & \vdots \\ 1 & x_{m1} & x_{m2} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_m^T \end{bmatrix}$$

where $\mathbf{x}_i^T = [1 \ x_{i1} \ x_{i2}]$ is a single example with petal length x_{i1} and petal width x_{i2} . We will also collect all the corresponding species/classes of these examples in a single vector \mathbf{y} below:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

where y_i is value (0 or 1) corresponding to the respective species of example i . We will also write the coefficients in a single vector $\mathbf{w}^T = [w_0 \ w_1 \ w_2]$. Also, once again, the logistic function is given by $\sigma(z) = (1 + \exp(-z))^{-1}$. With this notation set up, we write the mean squared error (MSE) as a function of the parameter vector as follows:

$$MSE(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m [\sigma(\mathbf{w}^T \mathbf{x}_i) - y_i]^2$$

Noting that the derivative of the logistic function is $\sigma'(z) = \sigma(z)(1 - \sigma(z))$, we can compute the partial derivatives of the MSE with respect to each parameter (w_0, w_1, w_2) as follows:

$$\frac{\partial MSE}{\partial w_0} = \frac{2}{m} \sum_{i=1}^m [\sigma(\mathbf{w}^T \mathbf{x}_i) - y_i] \frac{\partial \sigma(\mathbf{w}^T \mathbf{x}_i)}{\partial w_0} = \frac{2}{m} \sum_{i=1}^m [\sigma(\mathbf{w}^T \mathbf{x}_i) - y_i] [\sigma(\mathbf{w}^T \mathbf{x}_i)] [1 - \sigma(\mathbf{w}^T \mathbf{x}_i)]$$

Similarly, we can write out the partials with respect to the other two parameters:

$$\frac{\partial MSE}{\partial w_1} = \frac{2}{m} \sum_{i=1}^m [\sigma(\mathbf{w}^T \mathbf{x}_i) - y_i] [\sigma(\mathbf{w}^T \mathbf{x}_i)] [1 - \sigma(\mathbf{w}^T \mathbf{x}_i)] x_{i1}$$

$$\frac{\partial MSE}{\partial w_2} = \frac{2}{m} \sum_{i=1}^m [\sigma(\mathbf{w}^T \mathbf{x}_i) - y_i] [\sigma(\mathbf{w}^T \mathbf{x}_i)] [1 - \sigma(\mathbf{w}^T \mathbf{x}_i)] x_{i2}$$

The x_{i1} and x_{i2} arises in the two results above because, for example, $\frac{\partial (\mathbf{w}^T \mathbf{x}_i)}{\partial w_1} = \frac{\partial}{\partial w_1} (w_0 + w_1 x_{i1} + w_2 x_{i2}) = x_{i1}$.

The combination of these three partial derivatives then gives the gradient of the MSE:

$$\nabla_{\mathbf{w}} MSE = \begin{bmatrix} \frac{\partial MSE}{\partial w_0} \\ \frac{\partial MSE}{\partial w_1} \\ \frac{\partial MSE}{\partial w_2} \end{bmatrix}$$

Part D

Now, having derived the scalar form of the result above, we will attempt to vectorize the result in a cleaner, convenient form.

$$\nabla_{\mathbf{w}} MSE = \frac{2}{m} X^T [(\sigma(X\mathbf{w}) - \mathbf{y}) * \sigma(X\mathbf{w}) * (1 - \sigma(X\mathbf{w}))]$$

where $*$ indicates component-wise multiplication of vectors.

Part E