Lab 1

Welcome to the first lab. The goal of this lab is to perform the forward pass on shallow neural networks with a single hidden layer. Remember that a layer uses a weight matrix $\mathbf{W}$, a bias vector $\mathbf{b}$ and an activation function $\sigma(\cdot)$ to transform its input $\mathbf{x}$ to the output $\mathbf{y}$ with an affine transformation followed by a point-wise non-linearity $\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$.

Exercise 1

Recall that a neural network with a single output neuron and no hidden layer is equivalent to linear or logistic regression, and that the optimal parameters for linear regression are given by $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. Imagine you have four points $x_1 = 2$, $x_2 = -1$, $x_3 = 4$ and $x_4 = -3$ and the corresponding regression targets $y_1 = 3$, $y_2 = -3$, $y_3 = 7$ and $y_4 = -7$. Write code to:

1. Compute the optimal regression parameters with vectorized operations.
2. Use these parameters as weights for a neural network with a single output neuron with linear activation and no hidden layer. Compute the output of the network, again with vectorized operations, and verify that it is very close to the regression targets.

```
xs = matrix(c(
  2, 1,
  -1, 1,
  4, 1,
  -3, 1
), nrow = 4, ncol = 2, byrow=TRUE)
ys = matrix(
  c(3, -3, 7, -7),
  nrow = 4, ncol = 1
)

ws = (
  solve(crossprod(xs, xs)) %*% t(xs) %*% ys
)

ws
```

```
##      [,1]
## [1,]    2
## [2,]   -1
```

Check the weights. Do they make sense?

```
neuron_output = (
  xs %*% ws
)

neuron_output
```

```
##      [,1]
## [1,]    3
## [2,]   -3
## [3,]    7
## [4,]   -7
```

Check that the neuron output matches the targets.

Exercise 2

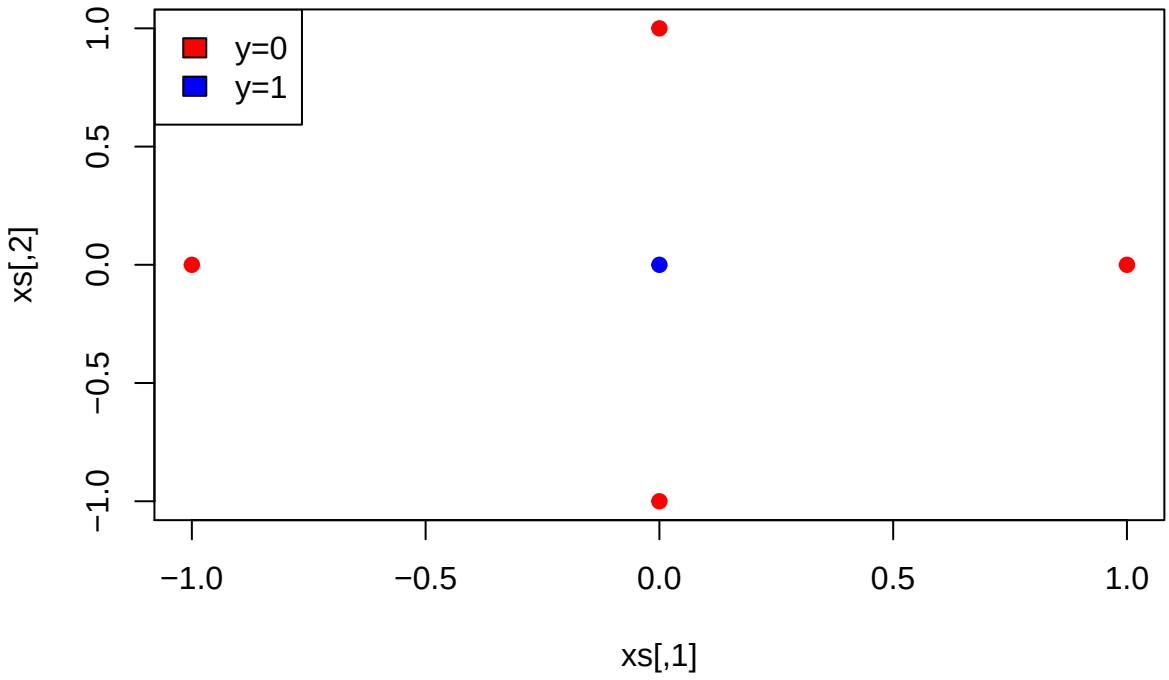
Suppose you have five input points, $\mathbf{x}_1 = |0, 0|^T$, $\mathbf{x}_2 = |1, 0|^T$, $\mathbf{x}_3 = |0, -1|^T$, $\mathbf{x}_4 = |-1, 0|^T$ and $\mathbf{x}_5 = |0, 1|^T$, and the corresponding classes are $y_1 = 1$ and $y_2 = y_3 = y_4 = y_5 = 0$ (see the figure below). The "ReLU" (Rectified Linear Unit) activation function is defined as $\sigma(x) = \max(0, x)$. Design a neural network with one hidden layer that performs this classification:

1. Find the values of $\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2$ to perform the classification correctly.
   - The hidden layer uses the ReLU activation function, the output layer uses the sigmoid.
   - Write down the equation for the forward pass, and find the shapes for the weight matrices and biases $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2$
   - How many neurons do the input and output layers have? How many neurons do you think the hidden layer should have?
   - Compute the loss between the labels and the classification of the neural network.
   - Pen and paper!
2. Write the code to perform point (1).
3. Explain graphically what your neural network is doing.
   - Does your network generalize well?
   - Would it classify correctly the four points $x_6, \ldots, x_9 = |\pm 1, \pm 1|^T$ with class 0?
   - If not, how would you change it?

```
# Plot the dataset
xs = matrix(c(
  0, 0,
  1, 0,
  0, -1,
  -1, 0,
  0, 1
), ncol = 2, byrow = TRUE)

ys = matrix(c(1, 0, 0, 0, 0), nrow = 5, ncol = 1)

plot(xs, col=c("blue", "red", "red", "red", "red"), pch=19)
legend("topleft", c("y=0", "y=1"), fill=c("red", "blue"))
```




```
ws1 = matrix(
  c(
    -20, 20,
    -20, 20
  ), nrow = 2, ncol = 2, byrow = TRUE
)

bs1 = (
  matrix(rep(c(-10, -10), nrow(xs)), ncol = 2)
)

ws2 = matrix(
  c(-1, -1), nrow = 2, ncol = 1, byrow = TRUE
)

bs2 = (
  matrix(rep(5, nrow(xs)), ncol = 1)
)
```

```
relu = function(x) {
  ifelse(x > 0, x, 0)
}

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

```
predictions = (
  sigmoid(relu(xs %*% ws1 + bs1) %*% ws2 + bs2)
)

predictions
```

```
##              [,1]
## [1,] 0.993307149
## [2,] 0.006692851
## [3,] 0.006692851
## [4,] 0.006692851
## [5,] 0.006692851
```

Compare the predicted classes with `ys`, they should be very similar.

```
binary_crossentropy = function(y_true, y_predicted) {
  -mean(ys * log(predictions) + (1 - ys) * log(1 - predictions))
}

loss = binary_crossentropy(ys, predictions)

loss
```

```
## [1] 0.006715348
```

The loss should be very low, as the predictions are all correct.

Solution

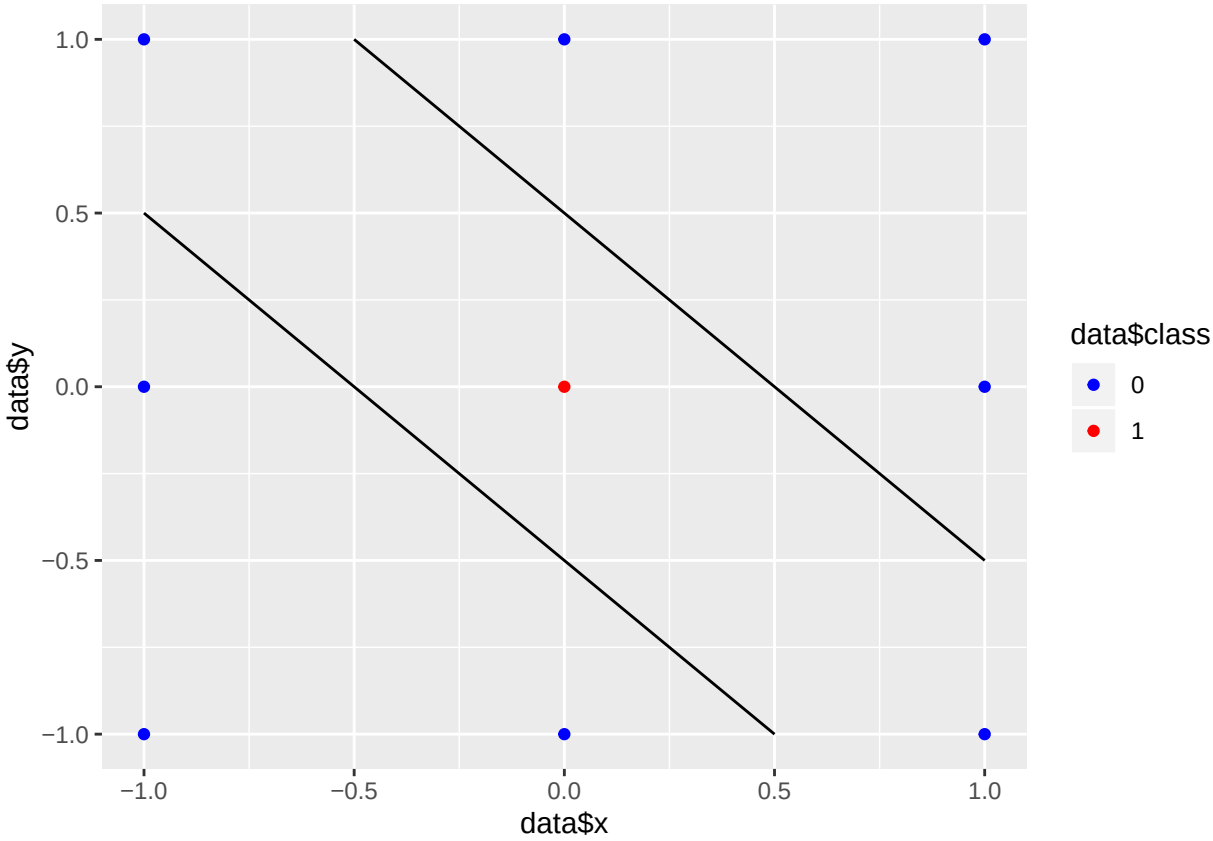
For the third question, with so few points, it is hard to say whether the network above generalizes well, but, as the points seem to form a circle, it probably does not. It would classify correctly $|+1,+1|^T$ and $|-1,-1|^T$, and would be mistaken for $|+1,-1|^T$ and $|-1,+1|^T$. Graphically, the network is using two diagoanl hyperplanes, defined in the first weight matrix, to separate positive and negative class. Specifically, the hyperplanes are the solutions of $-20x - 20y - 10 = 0$ and $20x + 20y - 10 = 0$, i.e. $y = -x - 1/2$ and $y = -x + 1/2$ respectively. Both hyperplanes are oriented "away" from the origin, in the sense that the origin is on the negative side. The network then predicts class 0 for points that are on the positive side of either plane by a sufficient distance (this distance being determined by the bias).

In order to obtain correct predictions for $|+1,-1|^T$ and $|-1,+1|^T$, one would need to add two more neurons to the hidden layer defining suitable hyperplanes, and the output layer would need to be modified accordingly.

```
library(ggplot2)

data = data.frame(
  x=c(0, 1, 0, -1, 0, -1, 1, -1, 1),
  y=c(0, 0, -1, 0, 1, 1, -1, -1, 1),
  class=c("1", rep("0", 8))
)

ggplot() +
  geom_point(aes(data$x, data$y, col=data$class)) +
  scale_color_manual(values=c("blue", rep("red", 8))) +
  geom_line(aes(x=c(-1, 0.5), y=c(0.5, -1))) +
  geom_line(aes(x=c(-0.5, 1), y=c(1, -0.5)))
```


1.0
0.5
0.0
−0.5
−1.0
data$y
−1.0
−0.5
0.0
0.5
1.0
data$x
data$class
0
1