Chapter 11: Backpropagation

Brandon Morgan

1/20/2021

Contents

E11.1

```r
library(ggplot2) # used for plots
```

For each pattern, we will first create the decision boundaries, form the weight matrix, and then create the network that will AND and OR the appropiate boundaries. Unfortunately, I am unable to recreate the Network diagrams from the book. For all the problems below, we will use a similar network structure as Figure P11.6. It will contain an Initial decisions layer to create boundaries for the categories, and AND layer to solidify those layers, and an OR layer to combine the categories. Parts 2 and 4 were not completed in this example.

1

In part one of Figure E11.1 from the book, there are two triangles, t_1 with the endpoints $(-1,0),(0,0),(0,1)$ and t_2 with the endpoints $(-1,0),(-1,-1),(0,-1)$.

Triangle t_1

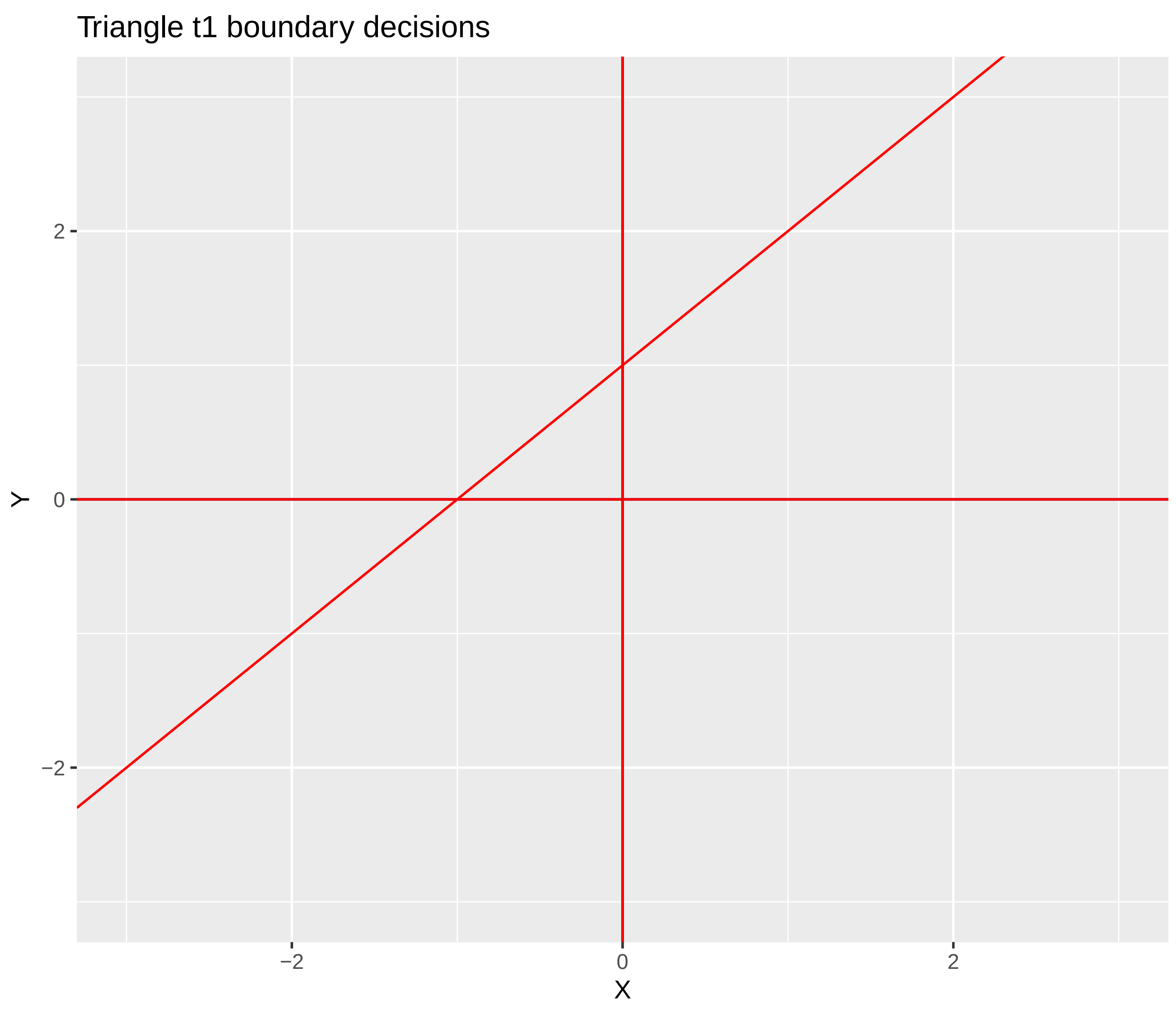
This triangle can be formed by three lines:

$$y_1 = x_1 + 1,\ y_2 = 0,\ x_3 = 0$$

Here we have plotted our decision boundary to show that it replicates our first triangle

```r
ggplot()+xlab("X")+xlim(-3,3)+ylim(-3,3)+
  ylab("Y")+ggtitle("Triangle t1 boundary decisions")+
  geom_hline(yintercept=0)+geom_vline(xintercept=0)+

  geom_abline(intercept=1, colour="red")+
  geom_hline(yintercept=0, colour="red")+
  geom_vline(xintercept=0, colour="red")
```



These three line equations equate to the following weights:

For line $y_1 = x_1$, we want any value below the line to be 1, because we want the area of the triangle to 1 for classification, therefore we would get $p_1 - p_2 + 1$.

For line $x_2 = 0$, we want any value to left of the vertical line to be 1, because we want the area of the triangle to be 1 for classification, therefore we would get $-p_1$.

For line $y_2 = 0$, we want any value above the horizontal line to be 1, because we want the area of the triangle to be 1 for classification, therefore we would get p_2.

Thus, together we get the following weight and bias matrices classifying this traingle, where each column is one of the lines described earlier:

$$w^T = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, b^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

Triangle t_2

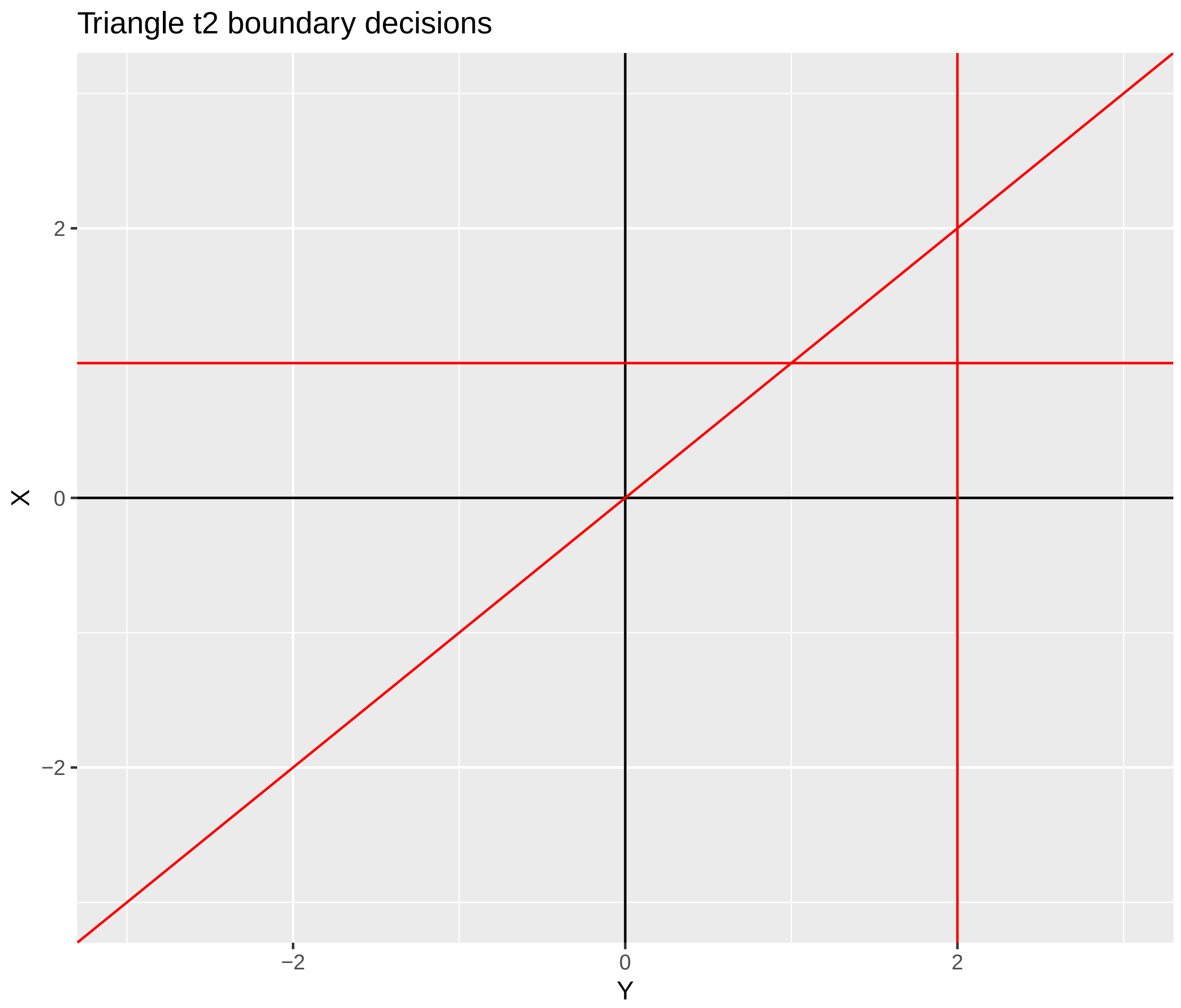
This triangle can be formed by three lines:

$$y_1 = x_1,\ y_2 = 1,\ x_3 = 2$$

Here we have plotted our decision boundary to show that it replicates our first triangle

```r
ggplot()+xlab("Y")+xlim(-3,3)+ylim(-3,3)+
  ylab("X")+ggtitle("Triangle t2 boundary decisions")+
  geom_hline(yintercept=0)+geom_vline(xintercept=0)+

  geom_abline(intercept=0, colour="red")+
  geom_hline(yintercept=1, colour="red")+
  geom_vline(xintercept=2, colour="red")
```



These three line equations equate to the following weights:

For line $y_1 = x_1$, we want any value below the line to be 1, because we want the area of the triangle to 1 for classification, therefore we would get $p_1 - p_2$.

For line $x_3 = 2$, we want any value to left of the vertical line to be 1, because we want the area of the triangle to be 1 for classification, therefore we would get $-p_1 + 2$.

For line $y_2 = 1$, we want any value above the horizontal line to be 1, because we want the area of the triangle to be 1 for classification, therefore we would get $p_2 - 1$.

Thus, together we get the following weight and bias matrices classifying this traingle, where each column is one of the lines described earlier:

$$w^T = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, b^T = \begin{bmatrix} 0 & 2 & -1 \end{bmatrix}$$

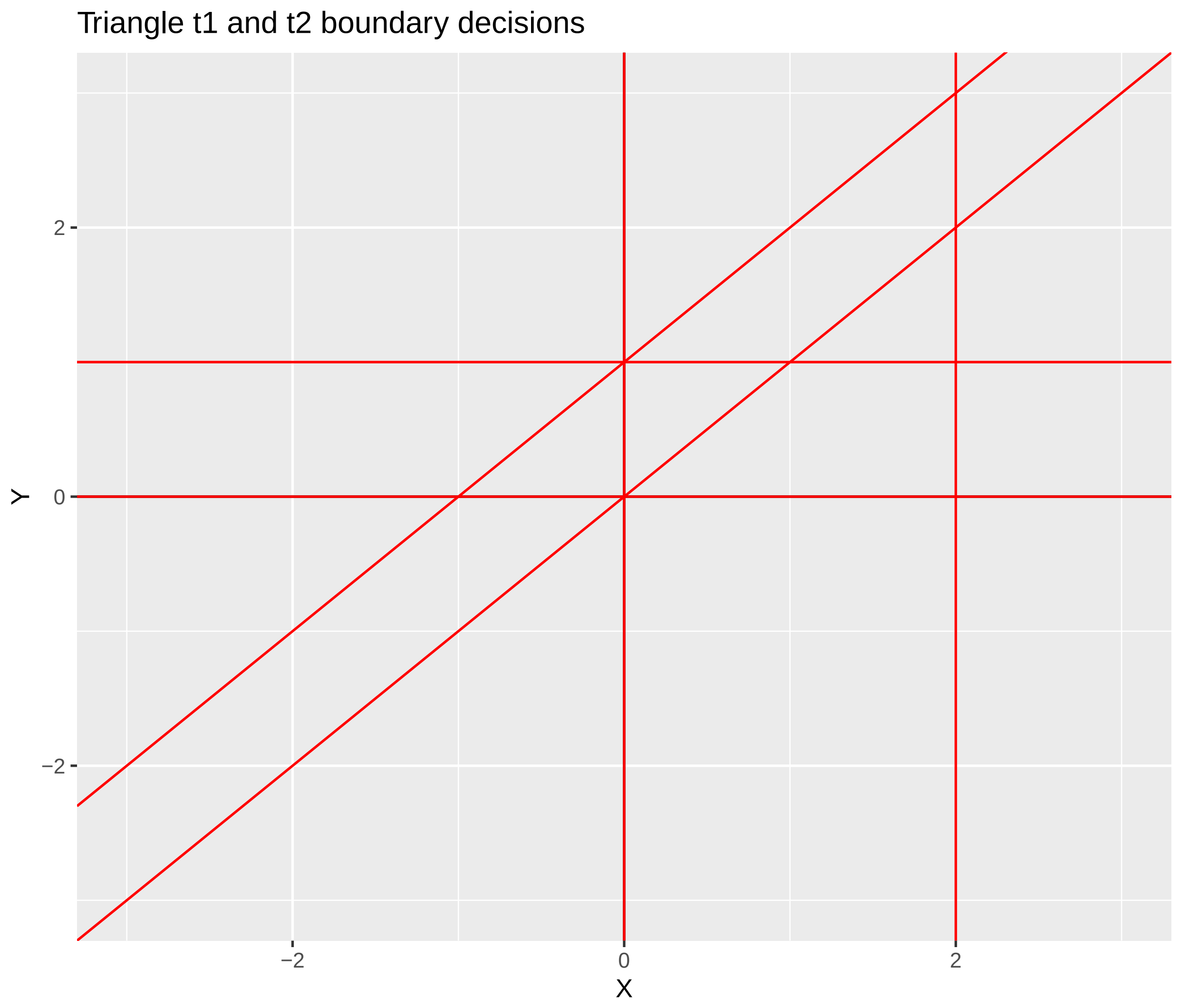
Together

Now we can combine our decision boundaries into one weight and bias matrices:

$$w^T = \begin{bmatrix} 1 & -1 & 0 & 1 & -1 & 0 \\ -1 & 0 & 1 & -1 & 0 & 1 \end{bmatrix}, b^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 & -1 \end{bmatrix}$$

```r
ggplot()+xlab("X")+xlim(-3,3)+ylim(-3,3)+
  ylab("Y")+ggtitle("Triangle t1 and t2 boundary decisions")+
  geom_hline(yintercept=0)+geom_vline(xintercept=0)+

  geom_abline(intercept=0, colour="red")+
  geom_hline(yintercept=1, colour="red")+
  geom_vline(xintercept=2, colour="red")+
  geom_abline(intercept=1, colour="red")+
  geom_hline(yintercept=0, colour="red")+
  geom_vline(xintercept=0, colour="red")
```



AND

Now we need a weight matrix to combine the decision boundaries into their triangular form, instead of infinite lines. This is done by AND-ing together the boundaries that make up the traingle. For t_1, the first three columns of the weight matrix w_1^T make up the boundary, therefore we need to combine these boundaries while discarding the rest. This can be done by the following weight matrix row $[1, 1, 1, 0, 0, 0]$. The same process, but inverse numbers, can be done for t_2, where we get the following weight matrix row $[0, 0, 0, 1, 1, 1]$. In total, we get the following second layer weight matrix:

$$w_2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

For bias of an AND, we need our result to equal one, but in each row of w_2^T we have three one value weights, therefore we need a bias of -2, so that the only way each row would equal 1 is if all inputs were one:

$$b_2 = \begin{bmatrix} -2 & -2 \end{bmatrix}$$

OR

Now that we have solidified our boundaries, we need an OR mechanism that produces the correct category if the input falls into either traingle 1 or 2. We need this layer to return 1 if either input from the AND perceptron returns 1, but returns -1 if both inputs from the AND perceptron return -1. To do this, we need OR to return 1 for the following inputs $[1, -1]$, $[-1, 1]$, and $[1, 1]$; but return -1 if $[-1, -1]$. To do this, we can combine both input values and if they are greater than -2, from $[-1, -1] = -2$, then output -1. This can be done by summing the input values and adding 1, as the *hardlims* transition function returns 1 for $n \geq 0$ and -1 for $n < 0$:

$$w_3 = \begin{bmatrix} 1 & 1 \end{bmatrix}, b_3 = \begin{bmatrix} 1 \end{bmatrix}$$

Conclusion

In concluion, we get the following weight and bias matrices for the first layer of the network, called the Initial Decisions in Figure p11.6

$$w^T = \begin{bmatrix} 1 & -1 & 0 & 1 & -1 & 0 \\ -1 & 0 & 1 & -1 & 0 & 1 \end{bmatrix}, b^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 & -1 \end{bmatrix}$$

Next, we have our hidden layer, AND Operations

$$w_2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \; b_2 = \begin{bmatrix} -2 & -2 \end{bmatrix}$$

Lastly, we have our final layer, OR Operation

$$w_3 = \begin{bmatrix} 1 & 1 \end{bmatrix}, b_3 = \begin{bmatrix} 1 \end{bmatrix}$$

Test inputs

From the graph, the point $(-0.5, 0.25)$ should lie in the first traingle, thus category 1; and point $(0.5, 1)$ should lie outside both traingles, thus category 2.

Now we will test our first point:

$$w^T = \begin{bmatrix} 1 & -1 & 0 & 1 & -1 & 0 \\ -1 & 0 & 1 & -1 & 0 & 1 \end{bmatrix}, b^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 & -1 \end{bmatrix}$$

```r
# hardlims transfer function
hardlims = function(x) {
  result = matrix(0, ncol=ncol(x), nrow=nrow(x))
  for(i in 1:nrow(x)) {
    for(j in 1:ncol(x)) {
      if(x[i,j] < 0) {
        result[i,j] = -1
      }
      else {
        result[i,j] = 1
      }
    }
  }
  result
}

# weight and bias matrices
w1 = matrix(c(1, -1, -1, 0, 0, 1, 1, -1, -1, 0, 0, 1), ncol=2, byrow=TRUE)
b1 = matrix(c(1, 0, 0, 0, 2, -1), ncol=1)
w2 = matrix(c(1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1), nrow=2, byrow=TRUE)
b2 = matrix(c(-2,-2), ncol=1)
w3 = matrix(c(1, 1), nrow=1)
b3 = 1

# test points
p1 = matrix(c(-0.5, 0.25), nrow=2)
p2 = matrix(c(0.5, 1))
```

Layer 1 Initial

Point 1:

```
a11 = hardlims(w1%*%p1+b1)
a11
```

```
##      [,1]
## [1,]    1
## [2,]    1
## [3,]    1
## [4,]   -1
## [5,]    1
## [6,]   -1
```

Point 2:

```
a21 = hardlims(w1%*%p2+b1)
a21
```

```
##      [,1]
## [1,]    1
## [2,]   -1
## [3,]    1
## [4,]   -1
## [5,]    1
## [6,]    1
```

Layer 2 AND

Point 1:

```
w2
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    1    1    0    0    0
## [2,]    0    0    0    1    1    1
```

```
a12 = hardlims(w2%*%a11+b2)
a12
```

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

Point 2:

```
a22 = hardlims(w2%*%a21+b2)
a22
```

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

Layer 3 OR

Point 1:

```
a13 = hardlims(w3%*%a12+b3)
a13
```

```
##      [,1]
## [1,]    1
```

As we can see, our network correctly predicted the following point in cateogry 1, as it lied in the area of the first triangle.

Point 2:

```
a23 = hardlims(w3%*%a22+b3)
a23
```

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

As we can see, our network correctly predicted the following point in cateogyr 2, as it lied in the area outside the triangles.

3

In part one of Figure E11.1 from the book, there are thre figures, t_1 with the endpoints $(-1,0),(-1,2),(0,1)$, t_2 with the endpoints $(0,-1),(-1,-2),(1,-2)$, and t_3 with the endpoints $(0,0),(1,1),(-1,-1),(0,2)$.

Triangle t_1

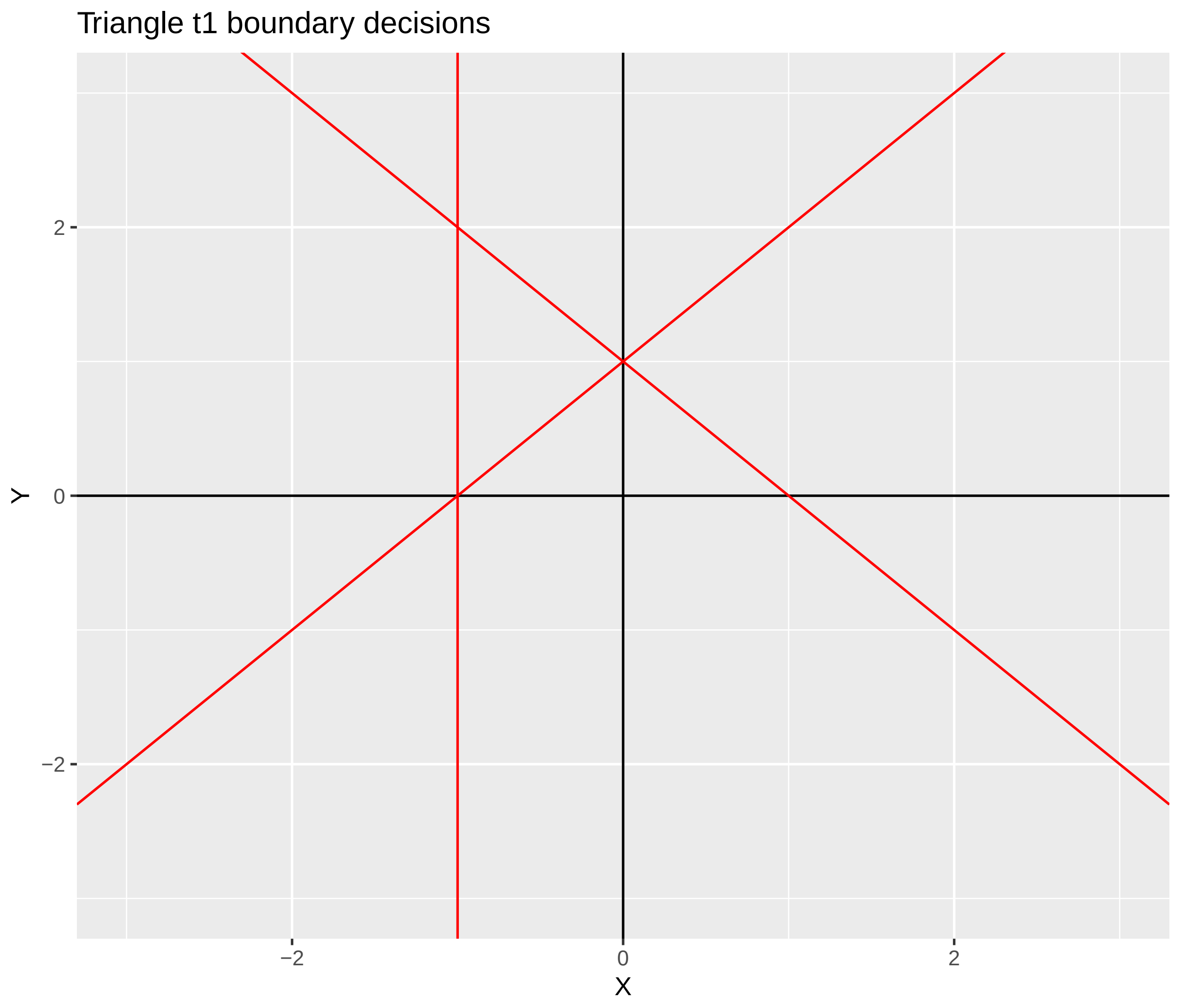
This triangle can be formed by three lines:

$$y_1 = x_1 + 1, \; y_2 = -x_2 + 1, \; x_3 = -1$$

Here we have plotted our decision boundary to show that it replicates our first triangle

```
ggplot()+xlab("X")+xlim(-3,3)+ylim(-3,3)+
  ylab("Y")+ggtitle("Triangle t1 boundary decisions")+
  geom_hline(yintercept=0)+geom_vline(xintercept=0)+

  geom_abline(intercept=1, colour="red")+
  geom_abline(intercept=1, slope=-1, colour="red")+
  geom_vline(xintercept=-1, colour="red")
```



These three line equations equate to the following weights:

For line $y_1 = x_1 + 1$, we want any value above the line to be 1, because we want the area of the triangle to 1 for classification, therefore we would get $-p_1 + p_2 - 1$.

For line $y_2 = -x_2 + 1$, we want any below to line to be 1, because we want the area of the triangle to be 1 for classification, therefore we would get $-p_1 - p_2 + 1$.

For line $x_3 = -1$, we want any value to the right of the vertical line to be 1, because we want the area of the triangle to be 1 for classification, therefore we would get $p_1 + 1$.

Thus, together we get the following weight and bias matrices classifying this traingle, where each column is one of the lines described earlier:

$$w^T = \begin{bmatrix} -1 & -1 & 1 \\ 1 & -1 & 0 \end{bmatrix}, b^T = \begin{bmatrix} -1 & 1 & 1 \end{bmatrix}$$

Triangle t_2

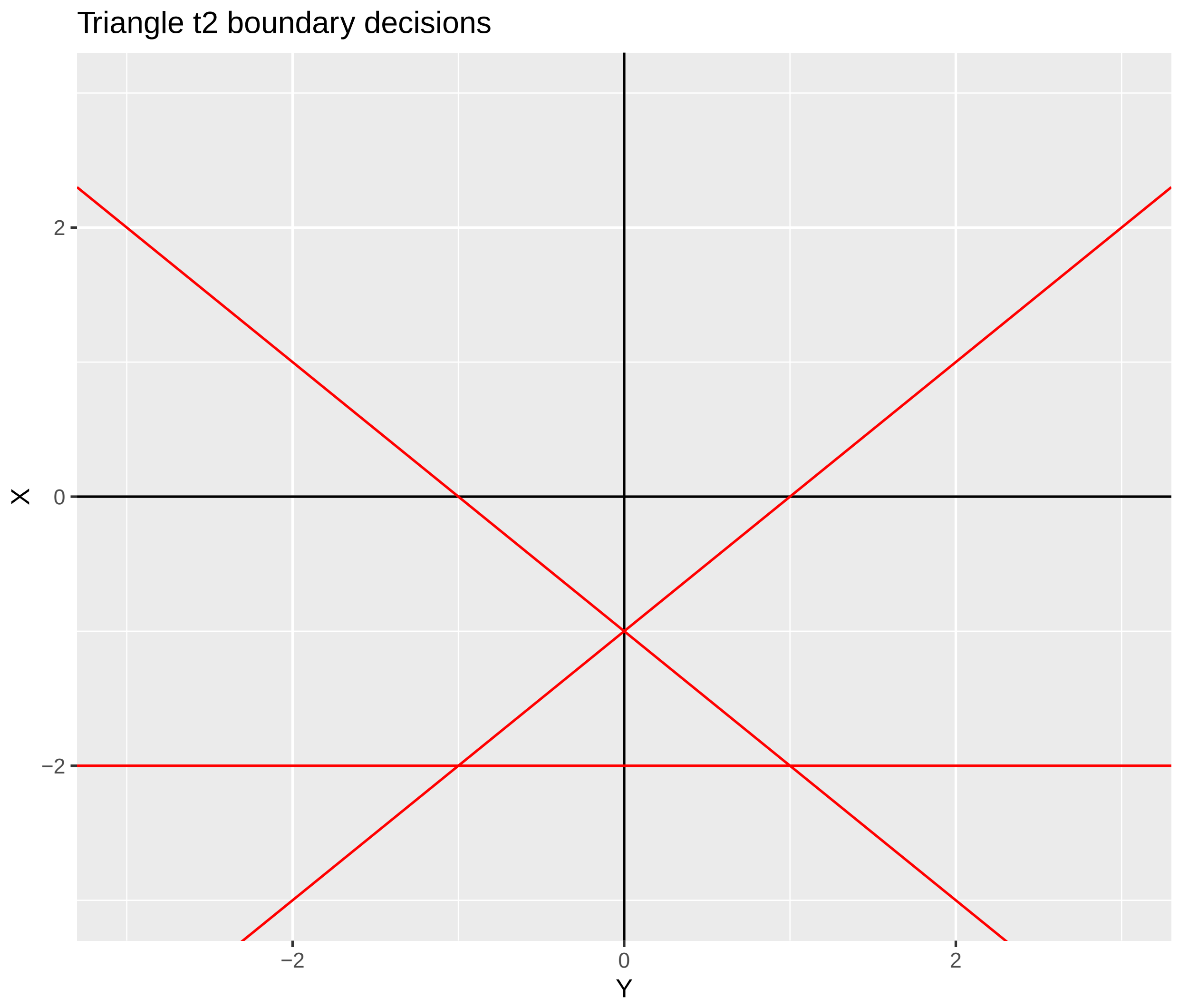
This triangle can be formed by three lines:

$$y_1 = x_1 - 1,\ y_2 = -x_2 - 1,\ y_3 = -2$$

Here we have plotted our decision boundary to show that it replicates our first triangle

```
ggplot()+xlab("Y")+xlim(-3,3)+ylim(-3,3)+
  ylab("X")+ggtitle("Triangle t2 boundary decisions")+
  geom_hline(yintercept=0)+geom_vline(xintercept=0)+

  geom_abline(intercept=-1, slope=-1, colour="red")+
  geom_hline(yintercept=-2, colour="red")+
  geom_abline(intercept=-1, slope=1, colour="red")
```



These three line equations equate to the following weights:

For line $y_1 = x_1 - 1$, we want any value below, (right), the line to be 1, because we want the area of the triangle to 1 for classification, therefore we would get $p_1 - p_2 - 1$.

For line $y_2 = -x_2 - 1$, we want any value below, (left) line to be 1, because we want the area of the triangle to be 1 for classification, therefore we would get $-p_1 - p_2 - 1$.

For line $y_3 = -2$, we want any value above the horizontal line to be 1, because we want the area of the triangle to be 1 for classification, therefore we would get $p_2 + 2$.

Thus, together we get the following weight and bias matrices classifying this traingle, where each column is one of the lines described earlier:

$$w^T = \begin{bmatrix} 1 & -1 & 0 \\ -1 & -1 & 1 \end{bmatrix}, b^T = \begin{bmatrix} -1 & -1 & 2 \end{bmatrix}$$

Square t_3

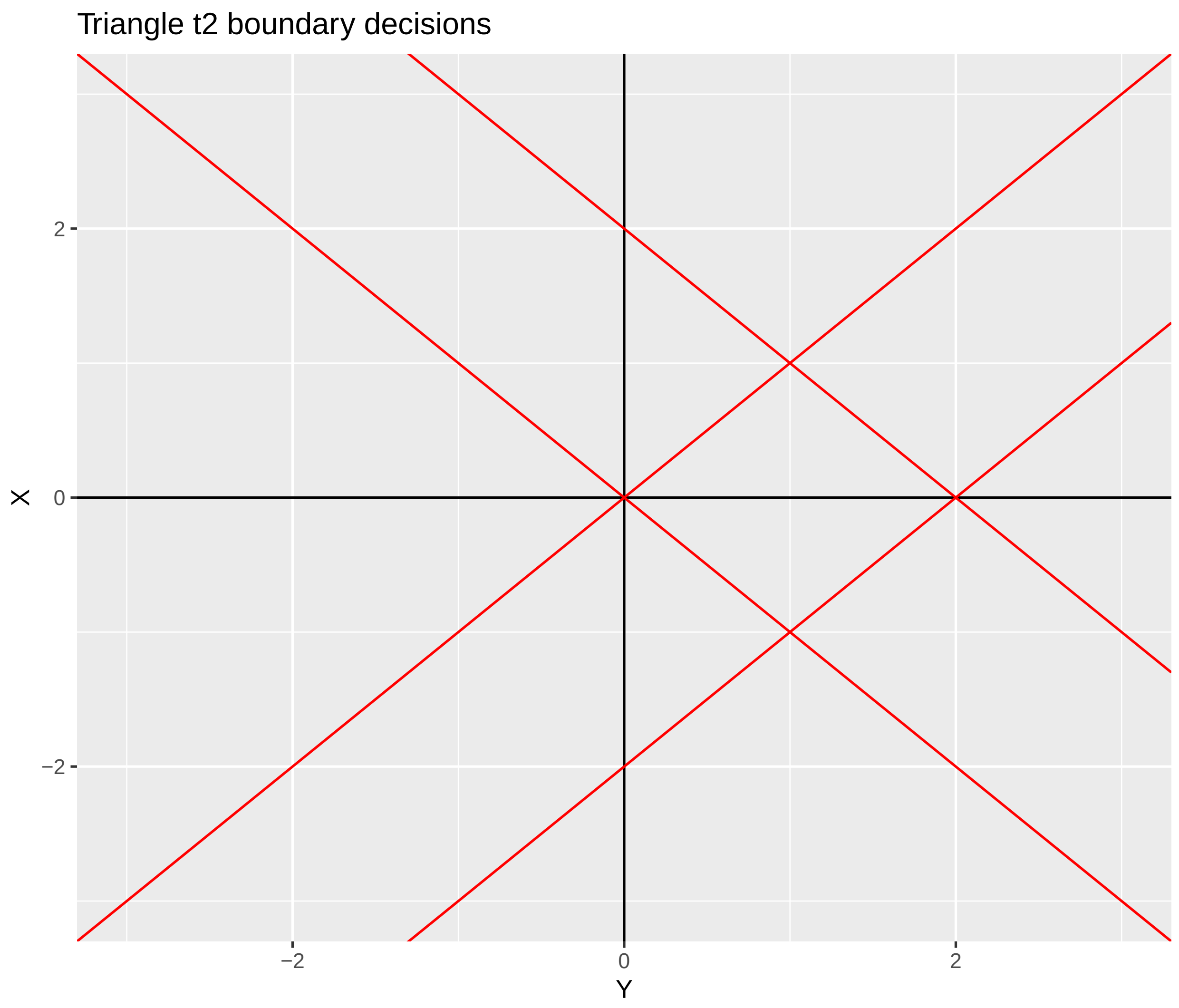
This square can be formed by four lines:

$$y_1 = x_1,\ y_2 = -x_2\ y_3 = -x_3 + 2,\ y_4 = x_4 - 2$$

Here we have plotted our decision boundary to show that it replicates our first triangle

```r
ggplot()+xlab("Y")+xlim(-3,3)+ylim(-3,3)+
  ylab("X")+ggtitle("Triangle t2 boundary decisions")+
  geom_hline(yintercept=0)+geom_vline(xintercept=0)+

  geom_abline(intercept=0, slope=-1, colour="red")+
  geom_abline(intercept=0, slope=1, colour="red")+
  geom_abline(intercept=2, slope=-1, colour="red")+
  geom_abline(intercept=-2, slope=1, colour="red")
```



These three line equations equate to the following weights:

For line $y_1 = x_1$, we want any value below, (right), the line to be 1, because we want the area of the square to 1 for classification, therefore we would get $p_1 - p_2$.

For line $y_2 = -x_2$, we want any value above, (right), the line to be 1, because we want the area of the square to 1 for classification, therefore we would get $p_1 + p_2$.

For line $y_3 = -x_3 + 2$, we want any value below, (left) line to be 1, because we want the area of the square to be 1 for classification, therefore we would get $-p_1 - p_2 + 2$.

For line $y_3 = x_4 - 2$, we want any value above, (left), the line to be 1, because we want the area of the square to be 1 for classification, therefore we would get $-p_1 + p_2 + 2$.

Thus, together we get the following weight and bias matrices classifying this traingle, where each column is one of the lines described earlier:

$$w^T = \begin{bmatrix} 1 & 1 & -1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix}, b^T = \begin{bmatrix} 0 & 0 & 2 & 2 \end{bmatrix}$$

Together

Now we can combine our decision boundaries into one weight and bias matrices:

$$w^T = \begin{bmatrix} -1 & -1 & 1 & 1 & -1 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & -1 & -1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix}, b^T = \begin{bmatrix} -1 & 1 & 1 & -1 & -1 & 2 & 0 & 0 & 2 & 2 \end{bmatrix}$$

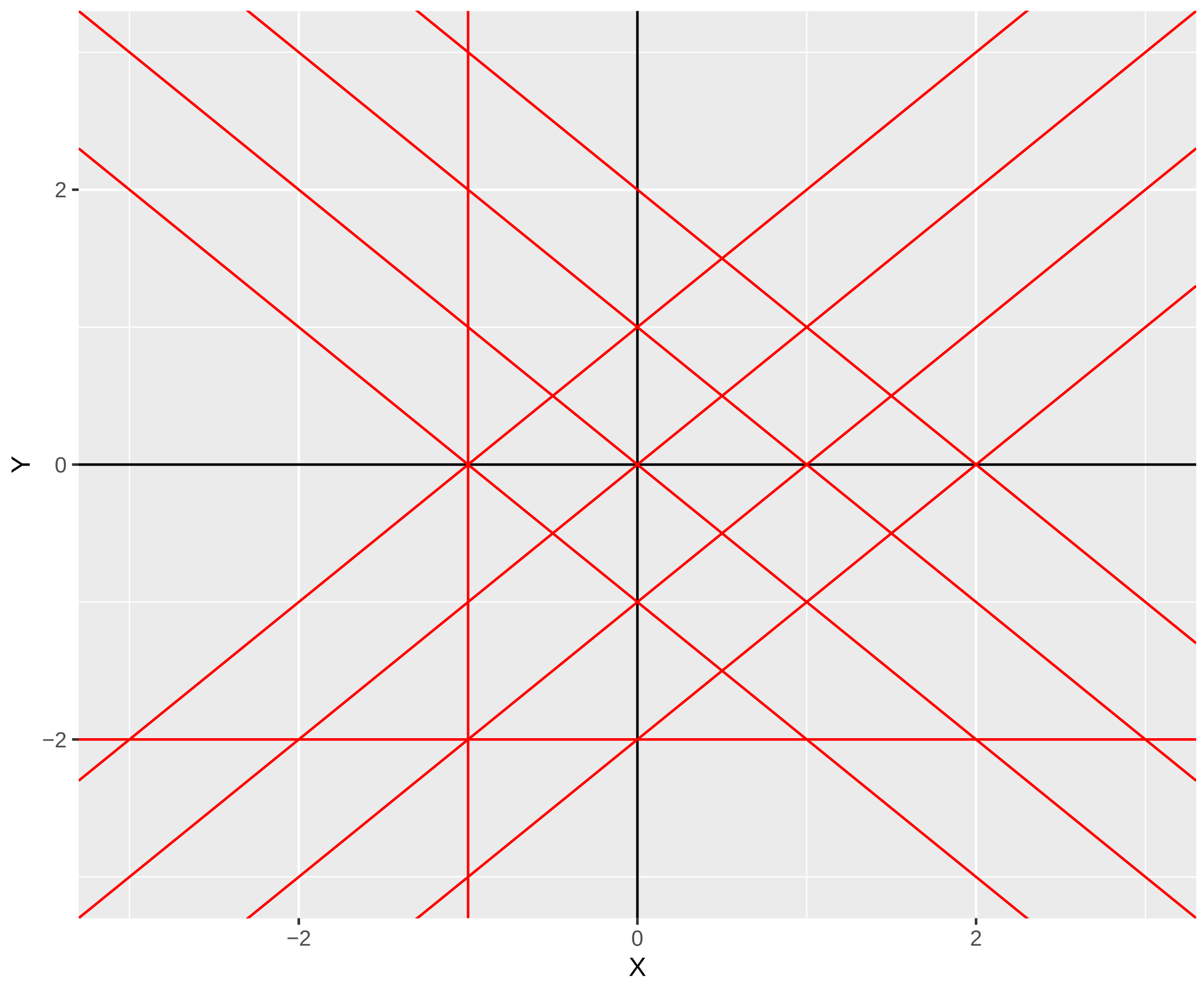
```r
ggplot()+xlab("X")+xlim(-3,3)+ylim(-3,3)+
  ylab("Y")+ggtitle("Figure t1,t2,t3 boundary decisions")+
  geom_hline(yintercept=0)+geom_vline(xintercept=0)+

  geom_abline(intercept=0, slope=-1, colour="red")+
  geom_abline(intercept=0, slope=1, colour="red")+
  geom_abline(intercept=2, slope=-1, colour="red")+
  geom_abline(intercept=-2, slope=1, colour="red")+

  geom_abline(intercept=-1, slope=-1, colour="red")+
  geom_hline(yintercept=-2, colour="red")+
  geom_abline(intercept=-1, slope=1, colour="red")+

  geom_abline(intercept=1, colour="red")+
  geom_abline(intercept=1, slope=-1, colour="red")+
  geom_vline(xintercept=-1, colour="red")
```

Figure t1,t2,t3 boundary decisions



AND

Now we need a weight matrix to combine the decision boundaries into their forms, instead of infinite intersecting lines. This is done by AND-ing together the boundaries that make up the figures. For t_1, the first three columns of the weight matrix w_1^T make up the boundary of the triangle, therefore we need to combine these boundaries while discarding the rest. This can be done by the following weight matrix row $[1, 1, 1, 0, 0, 0, 0, 0, 0, 0]$. The same process can be done for t_2, where we get the following weight matrix row $[0, 0, 0, 1, 1, 1, 0, 0, 0, 0]$. Lastly, the square only uses the last four boundaries, so we get the following weight matrix row: $[0, 0, 0, 0, 0, 0, 1, 1, 1, 1]$. In total, we get the following second layer weight matrix:

$$w_2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

For bias of an AND, we need our result to equal one. For the triangles, each row of the weight matrix has three one value weights, therefore we need a bias of -2, so that the only way each row would equal 1 is if all inputs were one. For the square, we need a bias of -3 because there are four boundaries:

$$b_2 = \begin{bmatrix} -2 & -2 & -3 \end{bmatrix}$$

OR

Now that we have solidified our boundaries, we need an OR mechanism that produces the correct category if the input falls into either traingle 1 or 2, or the square. We need this layer to return 1 if either input from the AND perceptron returns 1, but returns -1 if both inputs from the AND perceptron return -1. To do this, we need OR to return 1 for the following inputs $[1, 1, -1]$, $[1, -1, -1]$, $[-1, 1, 1]$, $[-1, 1, -1]$, $[-1, 1, -1]$, and $[1, 1, 1]$; but return -1 if $[-1, -1, -1]$. To do this, we can combine both input values and if they are greater than -3, from $[-1, -1, -1] = -2$, then output -1. This can be done by summing the input values and adding 1, as the *hardlims* transition function returns 1 for $n \geq 0$ and -1 for $n < 0$:

$$w_3 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, b_3 = \begin{bmatrix} 2 \end{bmatrix}$$

Conclusion

In concluion, we get the following weight and bias matrices for the first layer of the network, called the Initial Decisions in Figure p11.6

$$w^T = \begin{bmatrix} -1 & -1 & 1 & 1 & -1 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & -1 & -1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix}, b^T = \begin{bmatrix} -1 & 1 & 1 & -1 & -1 & 2 & 0 & 0 & 2 & 2 \end{bmatrix}$$

Next, we have our hidden layer, AND Operations

$$w_2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}, b_2 = \begin{bmatrix} -2 & -2 & -3 \end{bmatrix}$$

Lastly, we have our final layer, OR Operation

$$w_3 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, b_3 = \begin{bmatrix} 2 \end{bmatrix}$$

Test inputs

From the graph, the point $(-0.5, 0.5)$ should lie in the first traingle, thus category 1; point $(1, 0)$ should lie in the square, thus category1; and point $(-2, -1)$ should lie outside all three figures, thus category 2.

Now we will test our first point:

```r
# weight and bias matrices
w1 = matrix(c(-1, 1, -1, -1, 1, 0, 1, -1, -1, -1, 0, 1, 1, -1, 1, 1, -1, -1, -1, 1), ncol=2, byrow=TRUE
b1 = matrix(c(-1, 1, 1, -1, -1, 2, 0, 0, 2, 2), ncol=1)
w2 = matrix(c(1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
              0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
              0, 0, 0, 0, 0, 0, 1, 1, 1, 1), nrow=3, byrow=TRUE)
b2 = matrix(c(-2,-2, -3), ncol=1)
w3 = matrix(c(1, 1, 1), nrow=1)
b3 = 2
```

```
# test points
p1 = matrix(c(-0.5, 0.5), nrow=2)
p2 = matrix(c(1, 0), nrow=2)
p3 = matrix(c(-2, -1), nrow=2)
```

Layer 1 Initial

Point 1:

```
a11 = hardlims(w1%*%p1+b1)
a11
```

```
##       [,1]
##  [1,]    1
##  [2,]    1
##  [3,]    1
##  [4,]   -1
##  [5,]   -1
##  [6,]    1
##  [7,]   -1
##  [8,]    1
##  [9,]    1
## [10,]    1
```

Point 2:

```
a21 = hardlims(w1%*%p2+b1)
a21
```

```
##       [,1]
##  [1,]   -1
##  [2,]    1
##  [3,]    1
##  [4,]    1
##  [5,]   -1
##  [6,]    1
##  [7,]    1
##  [8,]    1
##  [9,]    1
## [10,]    1
```

Point 3:

```
a31 = hardlims(w1%*%p3+b1)
a31
```

```
##       [,1]
##  [1,]    1
##  [2,]    1
##  [3,]   -1
##  [4,]   -1
##  [5,]    1
##  [6,]    1
##  [7,]   -1
##  [8,]   -1
##  [9,]    1
## [10,]    1
```

Layer 2 AND

Point 1:

```
a12 = hardlims(w2%*%a11+b2)
a12
```

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

Point 2:

```
a22 = hardlims(w2%*%a21+b2)
a22
```

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

Point 3:

```
a32 = hardlims(w2%*%a31+b2)
a32
```

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

Layer 3 OR

Point 1:

```
a13 = hardlims(w3%*%a12+b3)
a13
```

```
##      [,1]
## [1,]    1
```

As we can see, our network correctly predicted the following point in cateogry 1, as it lied in the area of the first triangle.

Point 2:

```r
a23 = hardlims(w3%*%a22+b3)
a23
```

```
##      [,1]
## [1,]    1
```

As we can see, our network correctly predicted the following point in category 1, as it lied in the area of the square.

Point 2:

```r
a33 = hardlims(w3%*%a32+b3)
a33
```

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

As we can see, our network correctly predicted the following point in category 2, as it lied in the area outside the figures.