Question 3

99222 - Frederico Silva, 99326 - Sebastião Carvalho

December 13, 2023

Question 3

In this exercise, you will design a multilayer perceptron to compute a Boolean function of D variables, $f : \{-1, +1\}^D \to \{-1, +1\}$, defined as:

(a) Perceptron's Limitations (5 points)

Show that the function above cannot generally be computed with a single perceptron. *Hint: think of a simple counter-example.*

Answer To demonstrate that the specified Boolean function cannot be computed by a single perceptron, we simply need to show that there exists a counter-example where the data is not linearly separable.

Let's consider a simple case where $D = 2$, $A = -1$, and $B = 1$. The function f is defined as:

$$f(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{D} x_i \in [-1, 1], \\ -1 & \text{otherwise,} \end{cases}$$

Now, let's consider the following inputs: $x_1 = (+1, +1)$, $x_2 = (-1, -1)$, $x_3 = (-1, +1)$, and $x_4 = (+1, -1)$.

In this setup:

- For x_1, the sum $\sum x_i = 2$. Since 2 is not in the range [-1, 1], $f(x) = -1$.
- For x_2, the sum $\sum x_i = -2$. Since -2 is also not in the range [-1, 1], $f(x) = -1$.
- For x_3 and x_4, the sum $\sum x_i = 0$. This falls within the range [-1, 1], so $f(x) = 1$ for these inputs.

The visual representation of the points can be seen in Figure 1. The red points represent the inputs that should be classified as $+1$ and the blue points represent the inputs that should be classified as -1.

The critical point here is that a single perceptron is fundamentally a linear classifier, which means it can only separate data points using a straight line in the feature space. However, in this example, there is no straight line that can separate these points accordingly in a 2D space to satisfy the function f.

This example thus serves as a counter-example proving that the given function cannot generally be computed with a single perceptron, as it requires a non-linear decision boundary which a single perceptron cannot provide.

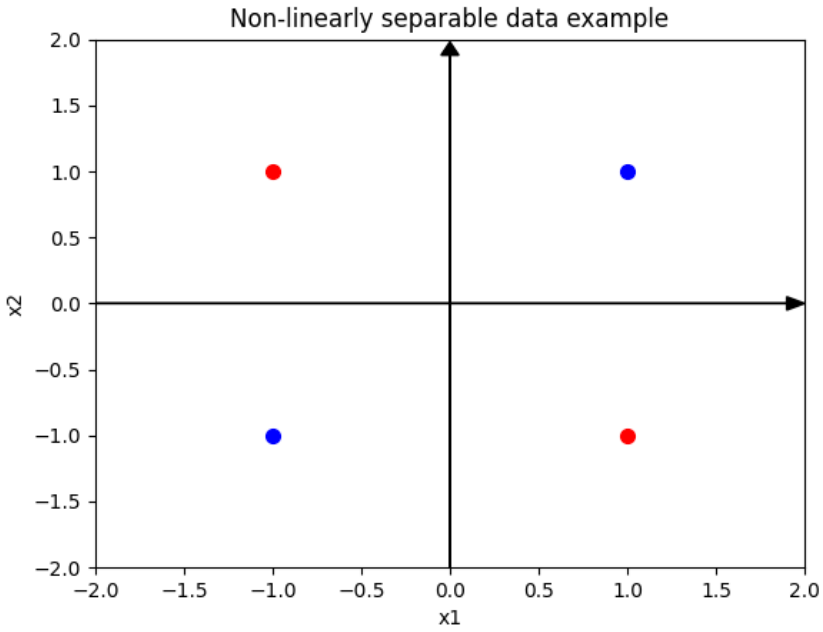

Figure 1: Classification of points using the function f

(b) Neural Network With Hard Threshold Activation (15 points)

Show that the function above can be computed with a multilayer perceptron with a single hidden layer with two hidden units and hard threshold activations $g : \mathbb{R} \to \{-1, +1\}$ with

$$g(z) = \text{sign}(z) = \begin{cases} +1 & \text{if } z \geq 0, \\ -1 & \text{otherwise,} \end{cases}$$

and where all the weights and biases are integers (positive, negative, or zero). Provide all the weights and biases of such network, and ensure that the resulting network is robust to infinitesimal perturbation of the inputs, i.e., the resulting function $h : \mathbb{R}^D \to \mathbb{R}$ should be such that $\lim_{t\to 0} h(\boldsymbol{x} + t\boldsymbol{v}) = h(\boldsymbol{x}) = f(\boldsymbol{x})$ for any $\boldsymbol{x} \in \{-1, +1\}^D$ and $\boldsymbol{v} \in \mathbb{R}^D$.

Answer Firstly, we will start by defining the weights and biases of the network. We will use the notation $W^{(l)}$ and $b^{(l)}$ to represent the weights and the biases, respectively, of the l-th layer. Consider now:

$W^{(1)} = \begin{bmatrix} 1 & \dots & 1 \\ -1 & \dots & -1 \end{bmatrix}$, where $W^{(1)}$ is a matrix of size 2 x D, and D is the size of the input vector;

$b^{(1)} = \begin{bmatrix} -A \\ B \end{bmatrix}$, where A is the lower bound of the sum of the input vector, and B is the upper bound of the sum of the input vector.

The idea behind the weights and biases of the first layer is that we want to verify if the sum of the input vector is within the range [A, B]. However, we have to do this individually, computing the lower bound condition for the first hidden unit, and the upper bound condition for the second hidden unit.

That is why we have the weights of the first layer as all 1's for the first row and all -1's for the second row, and the biases as -A and B.

If $Z^{(1)}$ is the pre-activation of the first layer, we have that $Z^{(1)} = W^{(1)}X + b^{(1)}$. The first hidden

unit's output will be $g((\sum_{i=1}^{D} x_i) - A)$, and it will be 1 if the sum of the input vector is greater than or equal to A, and -1 otherwise. The second hidden unit's output will be $g(B - (\sum_{i=1}^{D} x_i))$, and it will be 1 if the sum of the input vector is less than or equal to B, and -1 otherwise.
This means the first hidden unit's output is 1 if the sum of the input vector respects the lower bound, and the second hidden unit's output is 1 if the sum of the input vector respects the upper bound.

$W^{(2)} = \begin{bmatrix} 1 & 1 \end{bmatrix}$.
$b^{(2)} = \begin{bmatrix} -1 \end{bmatrix}$.
$W^{(2)}$ and $b^{(2)}$ are used to compute an AND function of the two hidden units. We use this since the output of the hidden units is either -1 or 1, and computing the AND we can verify if they respect the lower bound and upper bound, and thus belong to the range $[A, B]$.

With this, if $h(x)$ is the resulting function of the network, we have that $h(x) = 1$ if the sum of the input vector is within the range $[A, B]$, and -1 otherwise, and, thus, $h(x) = f(x)$, and we prove this neural network computes $f(x)$.

Now we need to prove that the network is robust to infinitesimal perturbation of the inputs. The biggest problem here is that the hard threshold activation function is not continuous, and thus we cannot use the continuity of the activation function to prove the continuity of the network.

As it is right now, the network is not robust to infinitesimal perturbation of the inputs. If we have an input vector $X = \begin{bmatrix} 1 \end{bmatrix}$, and we perturb it to make it $X' = \begin{bmatrix} 1 + \epsilon \end{bmatrix}$, with $\epsilon \in \mathbb{R} and \epsilon > 0$, we have that $h(X) = 1$ and $h(X') = -1$, and thus the network is not robust to infinitesimal perturbation of the inputs.

To prevent this from happening, we need to change the interval of values we want to accept from $[A, B]$ to $[A - \epsilon, B + \epsilon]$. This means we have to create new inequations for the lower bound and upper bound conditions, and thus we have to change the weights and biases of the first layer.

The first hidden unit needs to represent the following inequation:
$(\sum_{i=1}^{D} x_i) \geq A - \epsilon \iff (\sum_{i=1}^{D} x_i) - A + \epsilon \geq 0$
The second hidden unit needs to represent the following inequation:
$(\sum_{i=1}^{D} x_i) \leq B + \epsilon \iff B - (\sum_{i=1}^{D} x_i) - \epsilon \geq 0$
Since $\epsilon \in \mathbb{R}$ and $\epsilon > 0$, we can say that $\epsilon = k1/k2$, where $k1, k2 \in \mathbb{Z}$, and $k1 > 0, k2 > 0$ and $k1 < k2 (\epsilon < 1)$. This way we can rewrite the inequations as:
$(\sum_{i=1}^{D} x_i) - A + k1/k2 \geq 0 \iff k2(\sum_{i=1}^{D} x_i) - k2A + k1 \geq 0$
$B - (\sum_{i=1}^{D} x_i) - k1/k2 \geq 0 \iff k2B - k2(\sum_{i=1}^{D} x_i) - k1 \geq 0$
This way, we can rewrite the weights and biases of the first layer as:
$W^{(1)} = \begin{bmatrix} k2 & ... & k2 \\ -k2 & ... & -k2 \end{bmatrix}$
$b^{(1)} = \begin{bmatrix} -k2A + k1 \\ k2B - k1 \end{bmatrix}$

With this, we have that the first hidden unit's output will be $g(k2(\sum_{i=1}^{D} x_i) - k2A + k1)$, and it will be 1 if the sum of the input vector is greater than or equal to A - ϵ, and -1 otherwise. The second hidden unit's output will be $g(k2B - k2(\sum_{i=1}^{D} x_i) - k1)$, and it will be 1 if the sum of the input vector is less than or equal to B + ϵ, and -1 otherwise.
This means the first hidden unit's output is 1 if the sum of the input vector respects the lower bound, and the second hidden unit's output is 1 if the sum of the input vector respects the upper bound.
With this, we have that $h(x) = f(x)$, and the network is robust to infinitesimal perturbation of the inputs, since it now accepts inputs that are within the range $[A - \epsilon, B + \epsilon]$.

(c) MLP with ReLU activation (10 points)

Repeat the previous exercise if the hidden units use rectified linear unit activations instead.

Answer The rectified linear unit (ReLU) activation function is defined as:

$$ReLU(x) = \begin{cases} x & x \geq 0, \\ 0 & \text{otherwise} \end{cases}$$

First we will start by defining the weights and biases of the network. We will use the same notation as the previous exercise for the weights and biases.

The idea behind the weights and biases of the first layer is somewhat the opposite of the first exercise. We want the hidden units to output 0 if the sum of the input vector respects one of the bounds, and a positive value otherwise.
But like in the previous exercise, we must do this individually, computing the lower bound condition for the first hidden unit, and the upper bound condition for the second hidden unit, same as the previous exercise.

For this we have to use the weights and biases to represent the equations that represent the bound but with less or equal to 0, in order for the ReLU output to be 0:

$\sum_{i=1}^{D} x_i \geq A \Rightarrow -A + \sum_{i=1}^{D} x_i \geq 0 \Rightarrow -\sum_{i=1}^{D} x_i + A \leq 0.$
$\sum_{i=1}^{D} x_i \leq B \Rightarrow -B + \sum_{i=1}^{D} x_i \leq 0.$

To represent these equations with weights and biases, we have:

$W^{(1)} = \begin{bmatrix} -1 & ... & -1 \\ 1 & ... & 1 \end{bmatrix}$, where $W^{(1)}$ is a matrix of size 2 x D, and D is the size of the input vector.

$b^{(1)} = \begin{bmatrix} A \\ -B \end{bmatrix}$, where A is the lower bound of the sum of the input vector, and B is the upper bound of the sum of the input vector.

Let's consider $Z^{(1)}$ as the pre-activated output of the first layer. This means $Z^{(1)} = W^{(1)}X + b^{(1)}$, and $Z^{(1)} = \begin{bmatrix} A - \sum_{i=1}^{D} x_i \\ -B + \sum_{i=1}^{D} x_i \end{bmatrix}$.

The first hidden unit's pre-activated output will be $g(Z_1^{(1)})$, and the hidden unit's activated output will be 0 if the sum of the input vector is greater than or equal to A, and a positive value $(g(Z_1^{(1)}))$ otherwise.
The second hidden unit's output will be $g(Z_2^{(1)})$, and it will be 0 if the sum of the input vector is less than or equal to B, and a positive value $(g(Z_2^{(1)}))$ otherwise.
This means the first hidden unit's output is 0 if the sum of the input vector respects the lower bound, and the second hidden unit's output is 0 if the sum of the input vector respects the upper bound.

For the second layer, we want only to accept values that respect both bounds, so we use our weights and biases to verify if any of the values of the hidden layer are positive, and if they are, we want to output -1, because it means one of the bounds was not respected.

For this we have:

$W^{(2)} = \begin{bmatrix} -1 & -1 \end{bmatrix}$.
$b^{(2)} = \begin{bmatrix} -1 \end{bmatrix}$.
The idea behind $W^{(2)}$ and $b^{(2)}$ is to compute a function similar to the logic AND, but since this time our inputs are not in -1, 1, we cannot use the weights used in the previous exercise. Since each hidden unit outputs 0 if the sum of the input vector respects one of the bounds, and a positive

value otherwise, we can set the weights of the second layer as -1's, and the bias as 0, so that the pre-activated output of the network is 0 if the sum of the input vector respects both bounds, and a negative value otherwise.

Since the output this time is not in -1, 1, we need to apply an activation function to the pre-activated output of the network. For this we can use the sign function, which will output 1 if the pre-activated output is positive or zero, and -1 otherwise.
Since when an input vector respects both bounds, the pre-activated output of the network is 0, the sign function will output 1. If the input doesn't respect both bounds, the pre-activated output of the network will be negative, and the sign function will output -1.

With this, if h(X) is the resulting function of the network, we have that h(X) = 1 if the sum of the input vector is within the range [A, B], and -1 otherwise, and, thus, h(X) = f(X), and we prove this neural network computes f(X).

To ensure that our network is robust to infinitesimal perturbation of the inputs, we need to do similar to what we did in the previous exercise.

We need to ensure that the weights and biases of the first layer are such that the hidden units output 0 if the sum of the input vector is within the range [A - ϵ, B + ϵ], and a positive value otherwise.

This means we have to create new inequations for the lower bound and upper bound conditions, and thus we have to change the weights and biases of the first layer.
The first hidden unit needs to represent the following inequation:
$\sum_{i=1}^{D} x_i \geq A - \epsilon \iff -A + \epsilon + \sum_{i=1}^{D} x_i \geq 0 \iff -(\sum_{i=1}^{D} x_i) + A - \epsilon \leq 0.$

The second hidden unit needs to represent the following inequation:
$\sum_{i=1}^{D} x_i \leq B + \epsilon \Rightarrow -B - \epsilon + \sum_{i=1}^{D} x_i \leq 0.$

Since $\epsilon \in \mathbb{R}$ and $\epsilon > 0$, we can say that $\epsilon = k1/k2$, where $k1, k2 \in \mathbb{Z}$, and $k1 > 0, k2 > 0$ and $k1 < k2 (\epsilon < 1)$. This way we can rewrite the inequations as:
$-(\sum_{i=1}^{D} x_i) + A - k1/k2 \leq 0 \iff -k2(\sum_{i=1}^{D} x_i) + k2A - k1 \leq 0$
$-B - k1/k2 + \sum_{i=1}^{D} x_i \leq 0 \iff -k2B + k2(\sum_{i=1}^{D} x_i) + k1 \leq 0$
This way, we can rewrite the weights and biases of the first layer as:
$W^{(1)} = \begin{bmatrix} -k2 & ... & -k2 \\ k2 & ... & k2 \end{bmatrix}$
$b^{(1)} = \begin{bmatrix} k2A - k1 \\ -k2B + k1 \end{bmatrix}$
With this, we have that the first hidden unit's output will be $g(-k2(\sum_{i=1}^{D} x_i) + k2A - k1)$, and it will be 0 if the sum of the input vector is greater than or equal to A - ϵ, and a positive value $(-k2(\sum_{i=1}^{D} x_i) + k2A - k1)$ otherwise. The second hidden unit's output will be $g(-k2B + k2(\sum_{i=1}^{D} x_i) + k1)$, and it will be 0 if the sum of the input vector is less than or equal to B + ϵ, and a positive value $(-k2B + k2(\sum_{i=1}^{D} x_i) + k1)$ otherwise.
This means the first hidden unit's output is 0 if the sum of the input vector respects the lower bound, and the second hidden unit's output is 0 if the sum of the input vector respects the upper bound.
With this, we have that $h(x) = f(x)$, and the network is robust to infinitesimal perturbation of the inputs, since it now accepts inputs that are within the range $[A - \epsilon, B + \epsilon]$.