Question 3

99222 - Frederico Silva, 99326 - Sebastião Carvalho

December 9, 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.

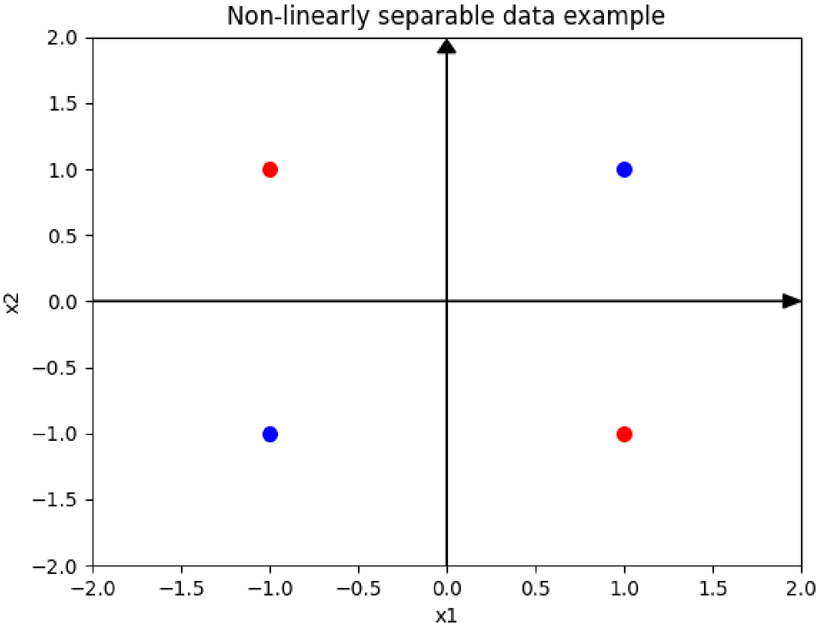

Figure 1: Classification of points using the function f

(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:

$W^{(1)} = \begin{bmatrix} 1 & ... & 1 \\ -1 & ... & -1 \end{bmatrix}$, 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.

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.

So, to prove that the network is robust to infinitesimal perturbation of the inputs, we need to prove that before it reaches the hard threshold, the network is continuous.

So in sum, to prove our network is robust to infinitesimal perturbation of the inputs, we need to prove that $\lim_{t\to 0} W^{(1)}(X + tV) + b^{(1)} = W^{(1)}X + b^{(1)}, V \in \mathbb{R}^D$.
$\lim_{t\to 0} W^{(1)}(X + tV) + b^{(1)} = \lim_{t\to 0} W^{(1)}X + W^{(1)}tV + b^{(1)}$.
Since $W^{(1)}$ is a matrix, and t is a scalar, and we know that a scalar multiplied by a matrix is a matrix, we know that:
$\lim_{t\to 0} W^{(1)}tV = 0$, which means $\lim_{t\to 0} W^{(1)}(X + tV) + b^{(1)} = W^{(1)}X + b^{(1)}$.
If $Z^{(1)} = W^{(1)}X + b^{(1)}$, is continuous, we know that even tho the activation function is not continuous, the first layer of the network is robust to infinitesimal perturbation of the inputs.

Since the second layer uses the same calculation as the first one for the pre-activation, and we don't use an activation function, we know that the second layer is also robust to infinitesimal perturbation of the inputs.