Solution 1: Decision Functions

We would like to implement a piecewise linear target (ground-truth) function f which classifies regions in $\mathbb{R}^2$ into two classes, positive (+) and negative (−), using three independent hypothesis functions: h_1, h_2, h_3.

a) From the figure, it can be seen that there are three regions with the positive label, namely $\mathcal{R}_1 = \overline{h_1}h_2h_3$, $\mathcal{R}_2 = h_1h_2\overline{h_3}$, and $\mathcal{R}_3 = h_1\overline{h_2}h_3$. Therefore, for any given input vector $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top$, $f(\mathbf{x}) = +$ if and only if $\mathbf{x} \in \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$, or equivalently, we can write the resulting expression for f by using the sum of products (SOP) criterion as follows: $f = \overline{h_1}h_2h_3 + h_1h_2\overline{h_3} + h_1\overline{h_2}h_3$. □

b) Yes, similar to above, we can decompose any target decision function using the SOP criterion over all + regions. Furthermore, the Universal Approximation Theorem (for Neural Networks) states that any smooth decision boundary can be approximated by piecewise linear functions which can always be implemented using three layers (including input and output layers) and sufficiently many nodes in the hidden layer.

Solution 2: Decision Function Implementation

a) To extend the logical `OR` function for vectors $\mathbf{x} \in \{-1, 1\}^M$, we observe the following: the output is -1 if and only if all components of the input are -1, i.e. $\sum_{i=1}^M x_i = -M$; otherwise the output remains $+1$. Therefore, the decision rule can be written as $h(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^M x_i + M - \frac{1}{2}\right)$. In all graphical representations below, the activation function ψ is the sign function.

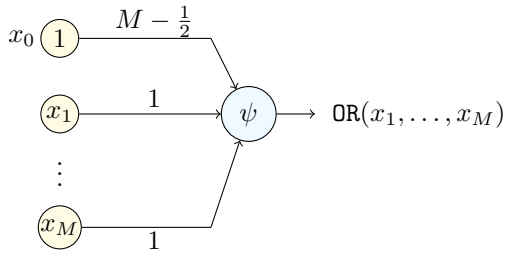

Figure 1: Neural network implementation of logical `OR`

Likewise, the decision rule for `AND` operation can be written as $h(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^M x_i - M + \frac{1}{2}\right)$. Its graphical representation is shown below.

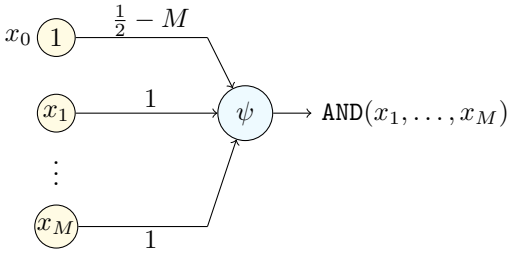

Figure 2: Neural network implementation of logical `AND`

b) Following the above, we can implement the inner product of a vector $\mathbf{x}$ and weights $\mathbf{w}$ by the network given below.

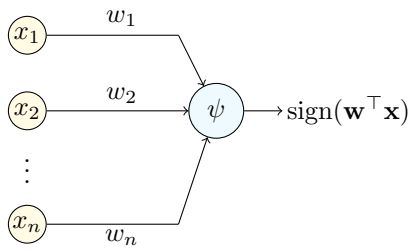

Figure 3: Neural network implementation of $h(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x})$ for $\mathbf{x}, \mathbf{w} \in \mathbb{R}^n$

c) Again, following part a), we can implement `OR` with negated arguments as below (note $M = 3$).

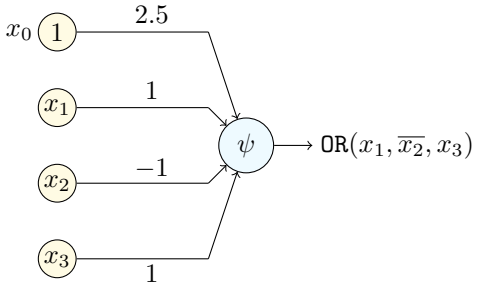

Figure 4: Neural network implementation of $\texttt{OR}(x_1, \overline{x_2}, x_3)$

Solution 3: Decision Function Implementation (Midterm 2019, Problem 3)

a) We get the following truth table after evaluating all combinations of inputs $(x_1, x_2) \in \{0, 1\}^2$.

x_1	x_2	ν	$\psi(\nu)$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

By observation, it is evident that the function is the `AND` operation.

b) Again, by observing the given truth table, it is evident that the new function is the `OR` operation which can be realized by the same neural network in part a), but by modifying of the activation function to be

$$\psi(\nu) = \begin{cases} 1 & \nu \geq 1 \\ 0 & \text{else} \end{cases}.$$

c) The given truth table is shown in graphical form in Figure 5, from which it is evident that the target function (`XOR`) is not separable by a single linear function, or plane, and hence can not be implemented using a single layer. But, the target labels can be computed by one layer if we can allow bivariate non-linear activation functions, such as

$$\psi(x_1, x_2) = \begin{cases} 1 & \frac{1}{2} \leq x_1 + x_2 \leq \frac{3}{2} \\ 0 & \text{else} \end{cases}.$$

Pranshu Malik
1004138916

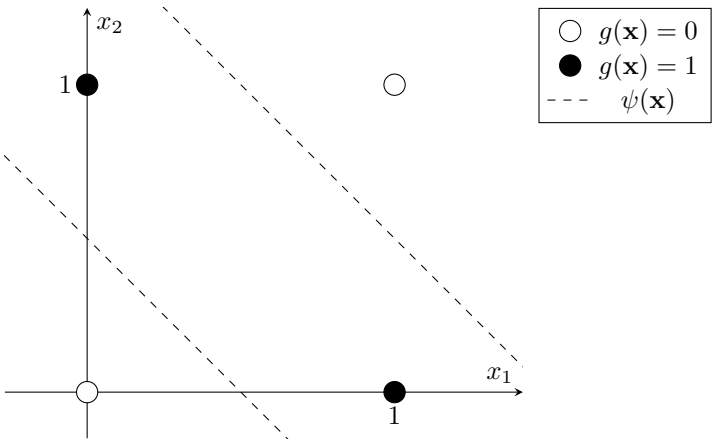

Figure 5: $\texttt{XOR}(\mathbf{x})$ for $\mathbf{x} \in \{0, 1\}^2$