Exercise - 2

Ans - 1 Hand- Grafted Network:

- We assign a three- layer network using the provided building blocks and specifications:

1) Logical OR: ($f(z)$)

To design a single neuron that performs a logical OR operation, we can set the weights and bias as follows.

⟹ weights: $w = [1, 1]$
Bias: $b = -0.5$
Activation function: step function (threshold at 0)

⟹ This neuron will output 1 if, any element in the input vector z is equat to 1,
Otherwise it will be an output 0.

2) Masked logical OR:- ($g(z; c)$):-

To design a single neuron that performs a masked logical OR operation with a fixed binary vector c, we can modify the logical OR neuron as follows.

⇒ weights : $W = [c_1, c_2, \ldots\ldots c_D]$
bias : $b = -0.5$
Activation function : step function
(threshold at 0).

Here,
- c is the fixed binary vector of length D.
- This neuron will output 1 if, any element in the input vector z is equal to 1
&
- The corresponding element in c is also 1.
- Otherwise
it will be an output 0.

2) perfect Match : (h(z; c)):

To design a single neuron that performs a perfect match operation with a fixed binary vector c, we can use a simple comparison operation.

⇒ weights : $W = [c_1, c_2, \ldots\ldots c_D]$.
bias : $b = -0.5$
Activation function : step function
(threshold at 0).

⇒ This neuron will output 1 if the vector z matches the fixed binary vector c element-wise, otherwise it will o/p 0.

- Now let's design a three-layer Network for the given dataset X_i, Y in figure 1.

4] First layer:

- Each feature vector x_i will be mapped onto once corner of a hypercube $\{0,1\}^m$

- The dimenssion m of the hypercube need to be adjusted based on the training set and complexity of the decision boundaries and the complexity of the decision boundaries required to classify the classes correctly.
- The hypercube corners will correspond to the classes in the dataset.

- Each corner will contain only points of the same class.
- The normal vector of these hyperplanes will point towards the positive pre-activation side.

5] second-layer!
- This layer consists of three neurons, each representing one class, red minus, blue plus and green circle.
- The inputs to these neurons will come from the first layer, indicating which hypercube corner the input feature belongs to.

- The weights and biases of these neurons will be adjusted to produce the desired output for each class.
- The activation function used in this layer can be the identity function since we want the neurons to directly represent the classes.

6) Third layer : (output layer)

- The third layer is the output layer which produces a one - hot encoding of the class labels.
- It consists of three neurons, each representing the one class.
- The activation function for this layer can be the softmax function, which normalizes the outputs and represents class probabilities.
- The equations for the output layer will be adjusted to produce the desired one-hot encoding of the class labels.

=> The resulting network can be generalized by arbitrary input dimensions by adjusting the size of the hypercube in the first layer and ~~out~~ adjusting the weights and biases accordingly.

- The number of neurons in the second and third layers will match the number of classes in the classification task.
- Additionally, the hand-crafted nature of the network limits its ability to learn complex and non-linear decision boundaries efficiently.

2> Linear Activation function

To prove that if the activation function ϕ_l is the identity function, then any network with depth $L > 1$ is equivalent to a 1-Layer neural network, we need to show that the output of the L-layer network can be expressed as a linear transformation without any non-linear activation.

Let's consider a feed-forward neural network with L layers. The output of each layer is calculated iteratively using the following equations:

$$Z_0 = X$$

$$\tilde{Z}_l = Z_{l-1} \cdot B_l + b_l$$

$$Z_l = \phi(\tilde{Z}_l)$$

where x is the input vector. $\tilde{Z}_l$ represents the pre-activation values of layer l, B_l is the weight matrix of layer l, b_l is the bias vector of layer l, and ϕ_l is the activation function of layer l.

Now, let's consider the case where ϕ_l is the identity function ($\phi_l(x) = x$). In this case, the activation function does not introduce any non-linearity, and the output of each layer is simply a linear transformation of the pre-activation values.

Let's rewrite the equations for the L-layer network using the identity activation function:

$$\tilde{Z}_l = Z_{l-1} \cdot B_l + b_l$$

$$Z_l = \tilde{Z}_l$$

Since the activation function is the identity function Z_l is equal to $\tilde{Z}_l$. We can substitute this back into the equation for $\tilde{Z}_l$:

$$\tilde{Z}_l = Z_{l-1} \cdot B_l + b_l$$

$$Z_l = Z_{l-1} \cdot B_l + b_l$$