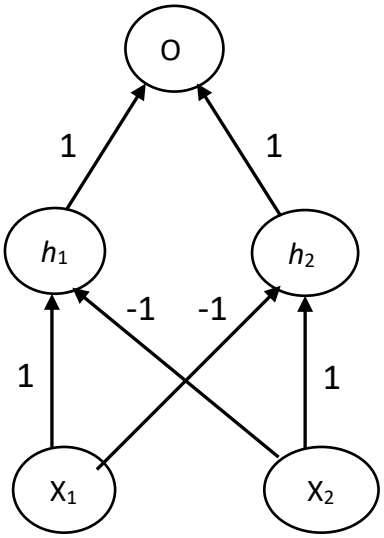
Output layer:

Hidden layer:

Input layer:



Sigmoid activation function at output layer
RELU activation at hidden layer
Learning rate 0.7
Trace the algorithm.

```
repeat
  for each e in examples
    // run forward pass
    for each node i in the input layer do a_i := x_i[e]
    for all hidden layers do:
      for each node j in a hidden layer do:
         for each node i in the prior layer do:
            input_j += w_ij * a_i
        activation_j := relu(input_j)
    for each node j in output layer do:
       for each node i in the prior layer do:
          input_j += w_ij * a_i
      activation_j := sigmoid(input_j)
    // run backprop
    // determine errors
    for each node j in output layer do:
      error_j = activation_j - desiredOutput_j[e]
    for all hidden layers do:
      for each node i in a hidden layer do:
         for each node j in the next layer do:
            e += w_ij * error_j
         error_i = e * relu'(input_i)
    // adjust weights
    for all neuron layers:
     for all nodes at layer j
        for all nodes at prior layer i:
           w_ij -= learning_rate * activation_i * error_j
until some stopping criterion is satisfied
```

XOR FFNet Relu

X1	X2	y	h_{1pre}	h_{2pre}	h_1	h_2	O	e_o	e_{h1}	e_{h2}
0	**0**	0								
0	**1**	1								
1	**0**	1								
1	**1**	0								

X1	X2	w_{h1-O}	w_{h2-O}	w_{x1-h1}	w_{x1-h2}	w_{x2-h1}	w_{x2-h2}
0	**0**						
0	**1**						
1	**0**						
1	**1**						