(1) Perceptron learning algorithm:

1. Categorization Decision
2. \( \text{decision}(\vec{x}, \vec{w}, \theta) \) if \( \vec{w} \cdot \vec{x} > \theta \) then
3. \( \text{return yes} \)
4. else
5. \( \text{return no} \)
6. end

7. Initialization
8. \( \vec{w} = 0 \)
9. \( \theta = 0 \)

10. Perceptron Learning Algorithm
11. while not converged yet do
12. for all elements \( \vec{x}_j \) in the training set do
13. \( d = \text{decision}(\vec{x}, \vec{w}, \theta) \)
14. if class(\( \vec{x}_j \)) = d then
15. continue
16. else if class(\( \vec{x}_j \)) = yes and \( d = no \) then
17. \( \theta = \theta - 1 \)
18. \( \vec{w} = \vec{w} + \vec{x}_j \)
19. else if class(\( \vec{x}_j \)) = no and \( d = yes \) then
20. \( \theta = \theta + 1 \)
21. \( \vec{w} = \vec{w} - \vec{x}_j \)
22. end
23. end
24. end

(2) Exclusive or (XOR):

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(3) Hidden layer can handle XOR:
(4) Output of a node:

\[ A\left(\sum_{i=1}^{N} w_i I_i + b\right) \]

(5) Error of a network:

\[ SSE = \sum (Y_i - Z_i)^2 \]

(6) Neural net training:

1. Calculate output: \( Z_i \)
2. Calculate blame:
   - Blame for output nodes is \( y_i - z_i \)
   - Blame for input and hidden nodes: \( \sum w_k e_k \), where we first reverse the net. Then \( e_k \) is the blame of an out-node and \( w_k \) is the weight of the edge that connects to that out-node.
3. Adjust weights:

\[ w_{ij} = w_{ij} + r \cdot e_j \cdot A'_j(I_j) \cdot O_i \]

where \( r \) is the learning rate, \( e_j \) is the blame of node \( j \), \( A'_j \) is the derivative of node \( j \)’s activation function, \( I_j \) is the input fed to node \( j \) during calculation of the output in the first step above and \( O_i \) is the output of node \( i \) in the first step.
4. Adjust biases:

\[ b_i = b_i + re_i \]