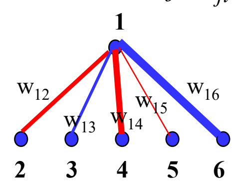
Artificial Neuron Model (Linear Threshold Unit)

- Model network as a graph with cells as nodes and synaptic connections as weighted edges from node *i* to node *j*, w_{ji}



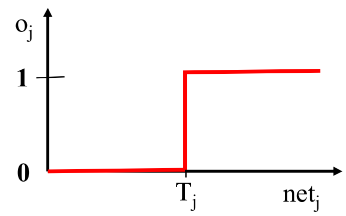
- Model net input to cell as

$$net_j = \sum_i w_{ji} o_i$$

- Cell output is:

$$o_j = \begin{cases} 0 \text{ if } net_j < T_j \\ 1 \text{ if } net_i \geq T_j \end{cases}$$

(T_j is threshold for unit j)



Perceptron Learning Rule

- Update weights by:

$$w_{ji} = w_{ji} + \eta(t_j - o_j)o_i$$

where η is the "learning rate"

t_j is the teacher specified output for unit *j*.

- Equivalent to rules:
  - If output is correct do nothing.
  - If output is high, lower weights on active inputs
  - If output is low, increase weights on active inputs
- Also adjust threshold to compensate:

$$T_j = T_j - \eta(t_j - o_j)$$

Perceptron Learning Algorithm (Rosenblatt, 1957)

- Iteratively update weights until convergence.

Initialize weights to random values
Until outputs of all training examples are correct
    For each training pair, E, do:
        Compute current output o_j for E given its inputs
        Compare current output to target value, t_j, for E
        Update synaptic weights and threshold using learning rule

- Each execution of the outer loop is typically called an *epoch*.

3

3