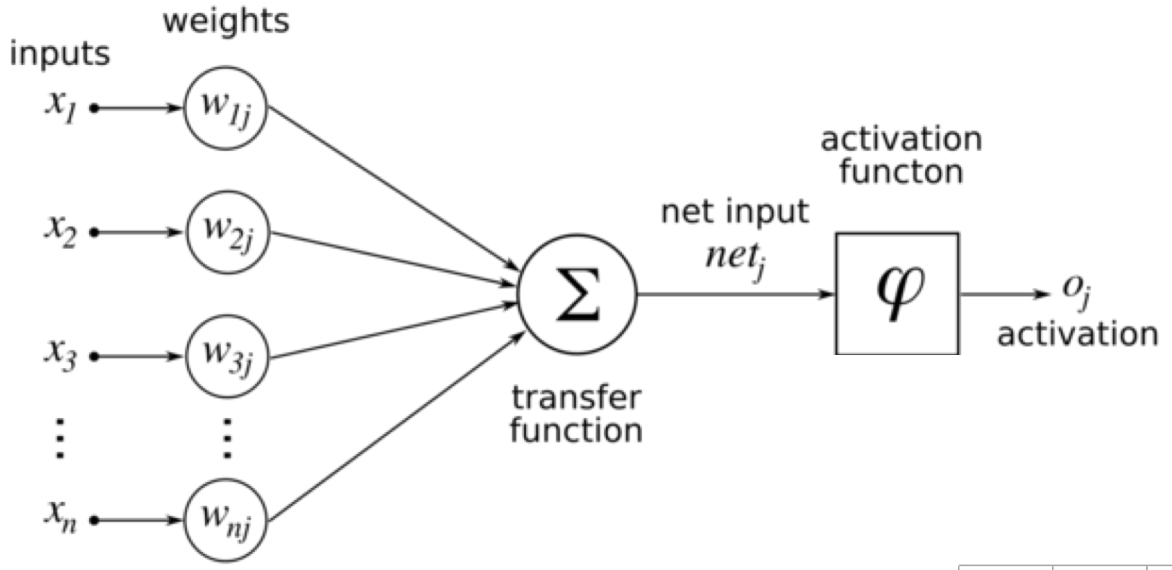
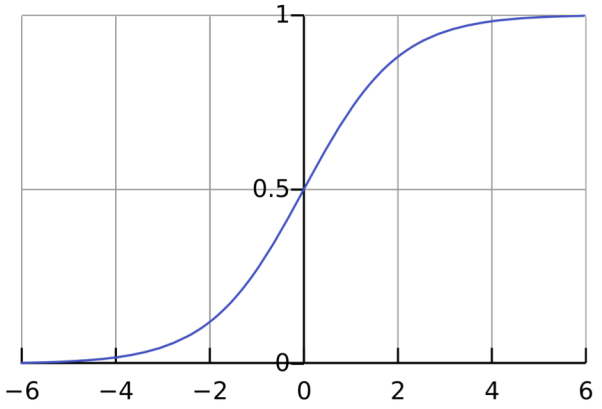
Back-Prop

An artificial Neuron



Logistic activation function: $\varphi(x) = \dfrac{1}{1+e^{-x}}$

Derivative: $\dfrac{d}{dx}\varphi(x) = \dfrac{e^{-x}}{\left(1+e^{-x}\right)^2} = \varphi(x)\left(1-\varphi(x)\right)$



Learning update of a single neuron

A source of data: $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots$ where $\vec{x}_t \in R^n, y_t \in [0,1]$

Goal : Find a weight vector $\vec{w}$ such that $\vec{w} \cdot \vec{x}$ is close to y

Loss function: $\mathrm{L}(\vec{x}, y, \vec{w}) = \left(\varphi\left(\vec{w} \cdot \vec{x}\right) - y\right)^2$

Gradient: $\nabla_{\vec{w}} \mathrm{L}(\vec{x}, y, \vec{w}) = 2\varphi\left(\vec{w} \cdot \vec{x}\right)\left(1 - \varphi\left(\vec{w} \cdot \vec{x}\right)\right)\left(\varphi\left(\vec{w} \cdot \vec{x}\right) - y\right)\vec{x}$

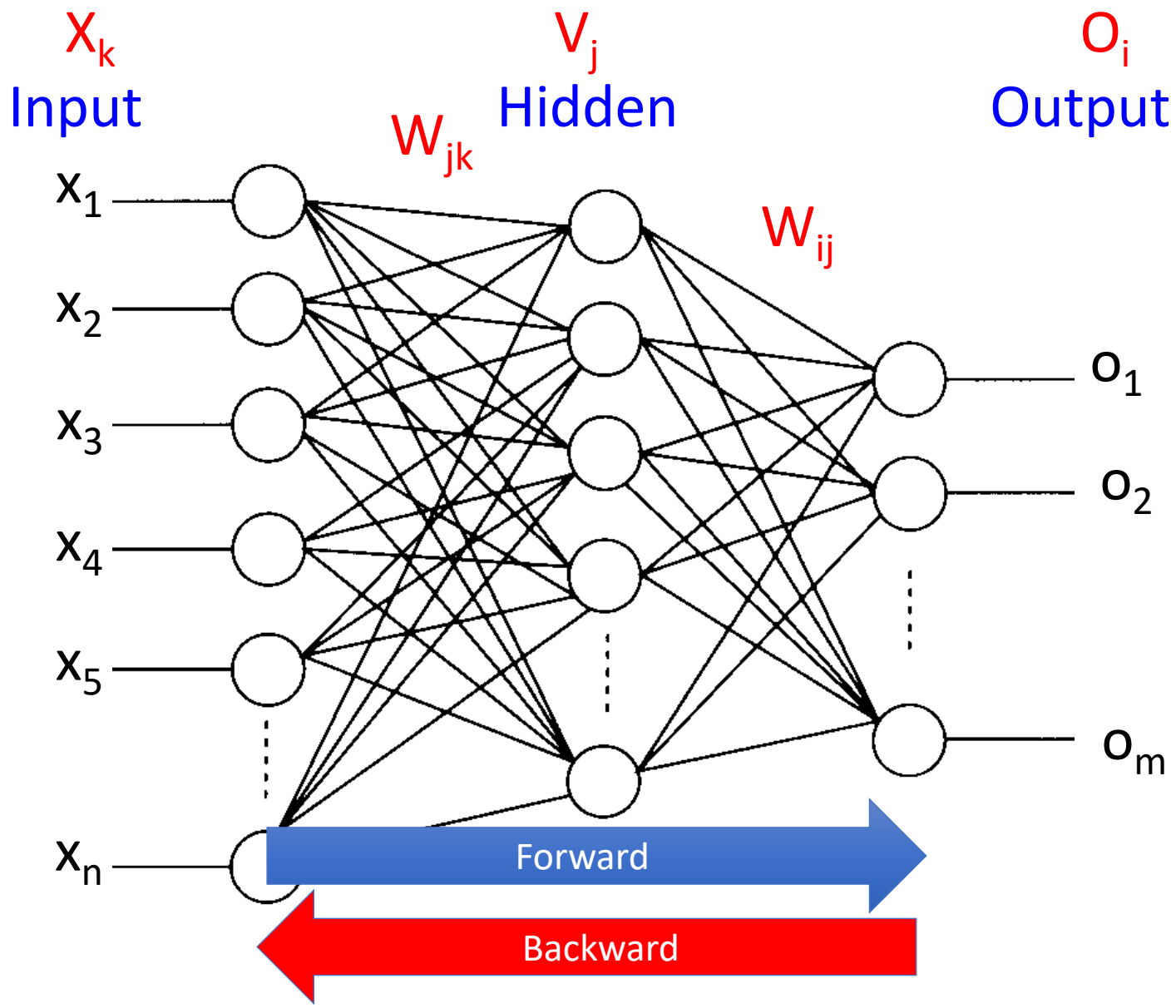
Update rule: $\vec{w}_{t+1} = \vec{w}_t - 2\eta_t \varphi\left(\vec{w}_t \cdot \vec{x}_t\right)\left(1 - \varphi\left(\vec{w}_t \cdot \vec{x}_t\right)\right)\left(\varphi\left(\vec{w}_t \cdot \vec{x}_t\right) - y_t\right)\vec{x}_t$

Define the output at time t to be: $o_t = \varphi\left(\vec{w}_t \cdot \vec{x}_t\right)$

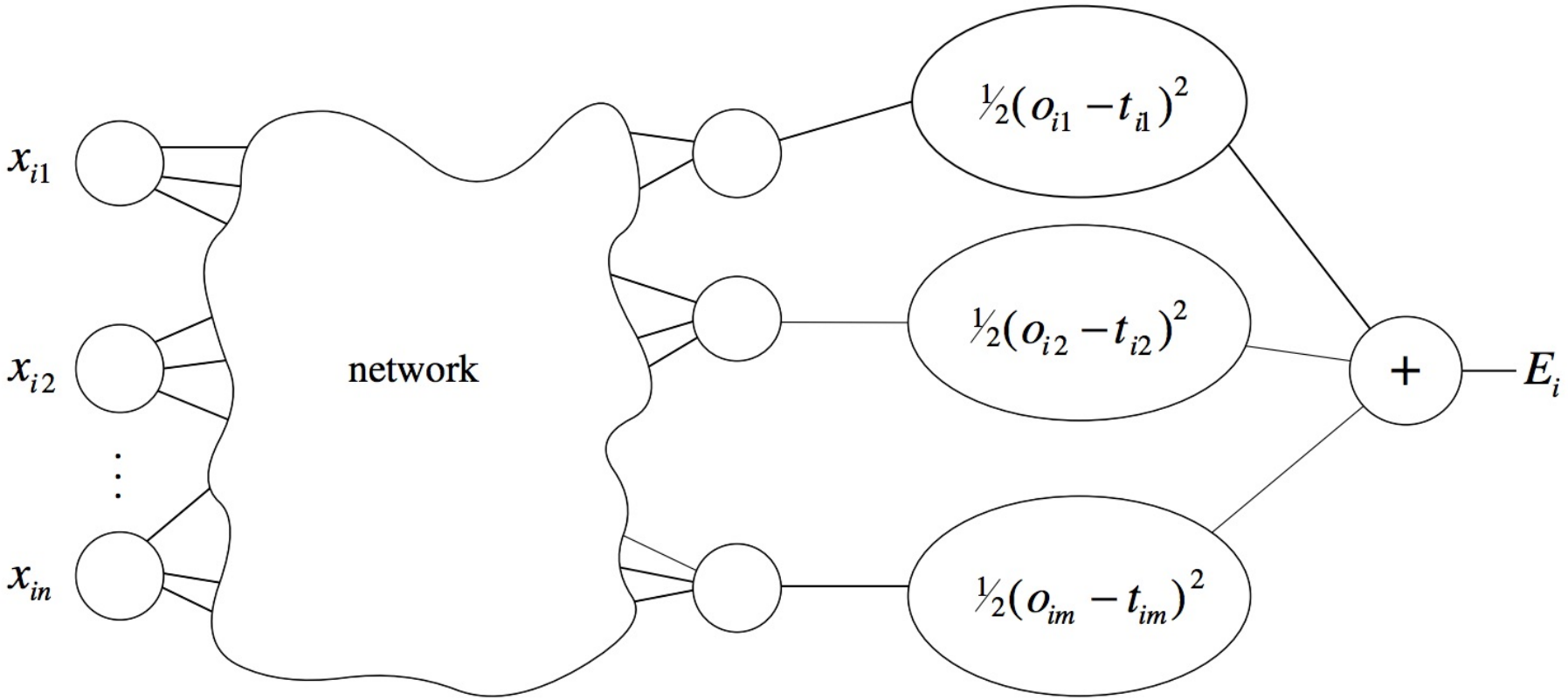
Simplified update rule: $\vec{w}_{t+1} = \vec{w}_t - 2\eta_t o_t \left(1 - o_t\right)\left(o_t - y_t\right)\vec{x}_t$

A feed-forward Artificial Neural Network (ANN)

Based on “Introduction to the theory of Neural computation” Hertz, Krogh and Palmer

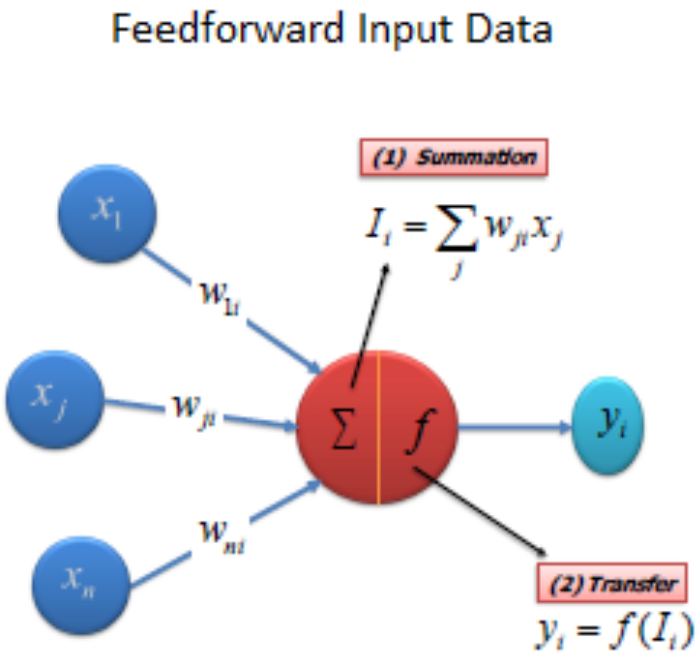


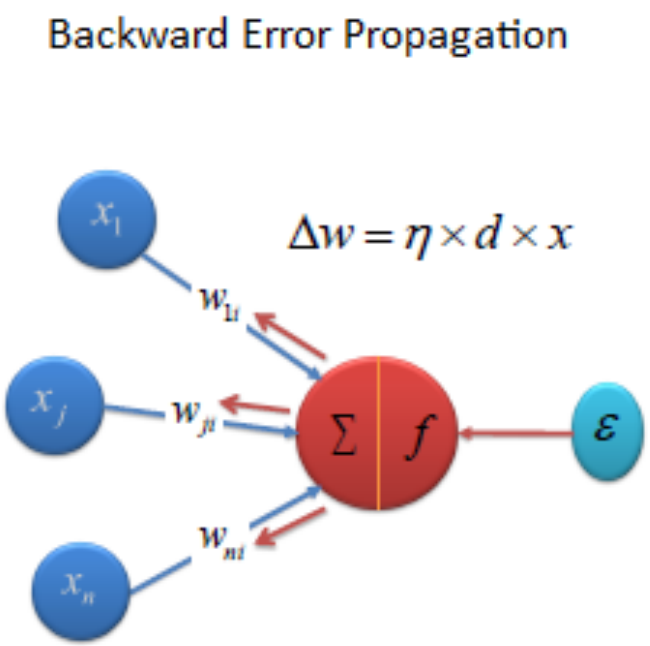
ANN loss function



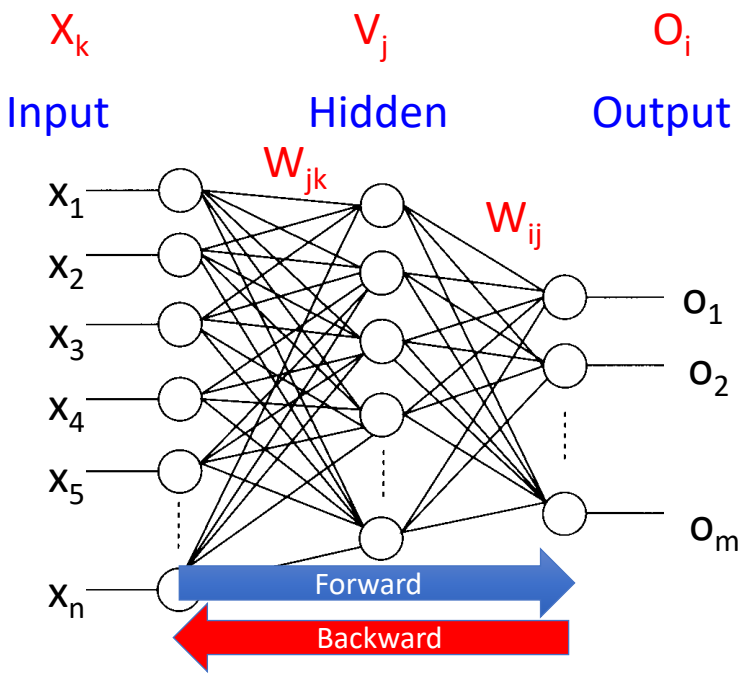
Computing the gradient

- Forward pass: propagate activations.
- Backward pass: propagate errors.





Forward-Backward Update rules



1) Forward Activation Propagation:

$$v_j = \varphi\left(\sum_{k=1}^{n} W^1{}_{jk} x_k\right);\ o_i = \varphi\left(\sum_{j} W^2{}_{ij} v_j\right)$$

2) Backward propagation of errors:

Update of connection from hidden node j to output node i

$$\Delta W_{ij} = \underbrace{o_i(1-o_i)(t_i-o_i)}_{\delta_i} v_j = \delta_i v_j;\ W'_{ij} \leftarrow W_{ij} + \eta \Delta W_{ij}$$

Update of connections from input node k to hidden node j

$$\underbrace{\delta_j = v_j(1-v_j)\sum_{i=1}^{m} W_{ij}\delta_i}_{\text{Back-Prop}};\quad \Delta W_{jk} = \delta_j x_k;\ W'_{jk} \leftarrow W_{jk} + \eta \Delta W_{jk}$$

Summary

- Most NN are feed-forward
  - There are NN with Feedback, we will not talk about them here.
- A Layered FF-NN has connection going from one layer to the next.
- **The forward pass** computes the activation level of all neurons/
- Given the output we compute the **errors** for output nodes.
- **The backwards pass** computes the effective error for each internal neuron.
- In the backward pass the weights are updated.