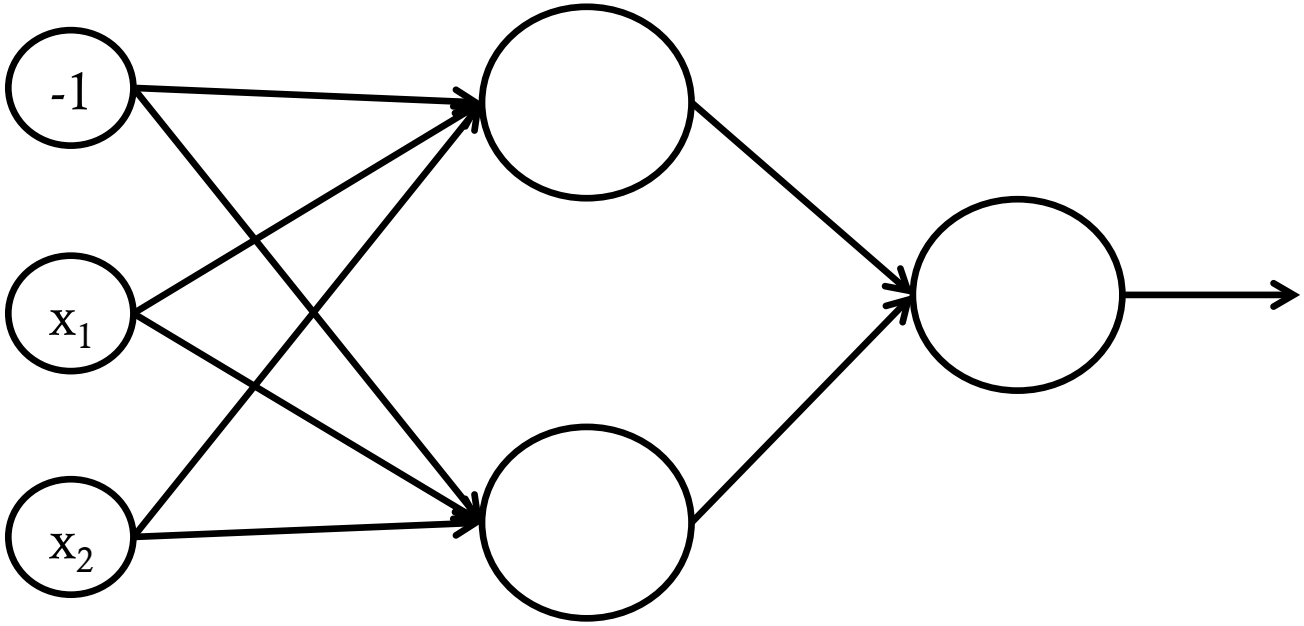
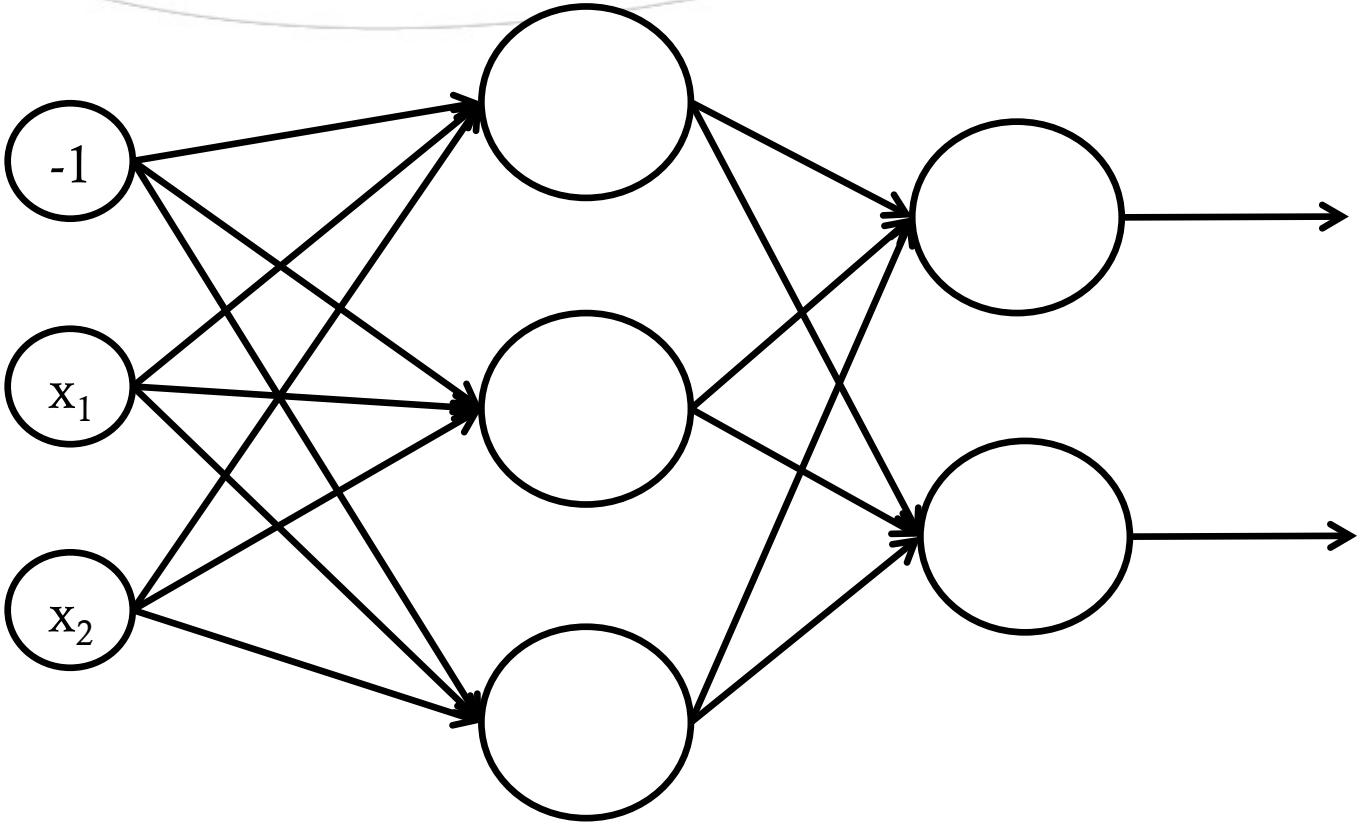
Multi-layer Perceptron

CS 450 – Machine Learning and Data Mining

Multi-Layer Perceptron



Multi-Layer Perceptron



Feed Forward – Hidden Nodes

$$a_j = g(h_j)$$

$$h_j = \sum_i x_i w_{ij}$$

$$g(h_j) = \frac{1}{1 + e^{-h_j}}$$

i – Inputs (actual input or hidden node value from the layer on the left)

j – Current Node

w_{ij} – Weight from input *i* to node *j*

h_j – Weighted sum value for node *j*

a_j – Activation value (output) for node *j*

Feed Forward – Output Nodes

$$a_j = g(h_j)$$

$$h_j = \sum_i x_i w_{ij}$$

$$g(h_j) = \frac{1}{1 + e^{-h_j}}$$

i – Inputs (hidden node value from the layer on the left)

j – Current Node

w_{ij} – Weight from input *i* to node *j*

h_j – Weighted sum value for node *j*

a_j – Activation value (output) for node *j*

Feed Forward

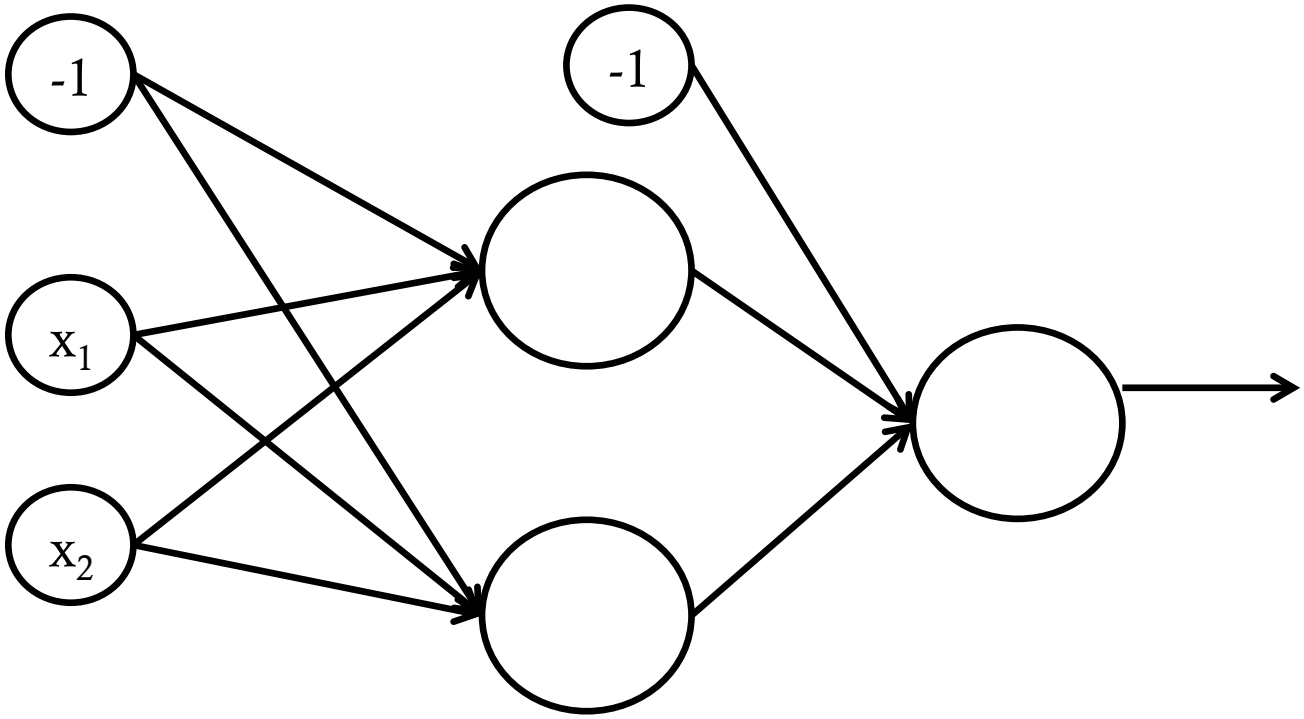
Hidden Node *j* Output Node *j*

$$a_j = g(h_j)$$

$$h_j = \sum_i x_i w_{ij}$$

$$g(h_j) = \frac{1}{1 + e^{-h_j}}$$

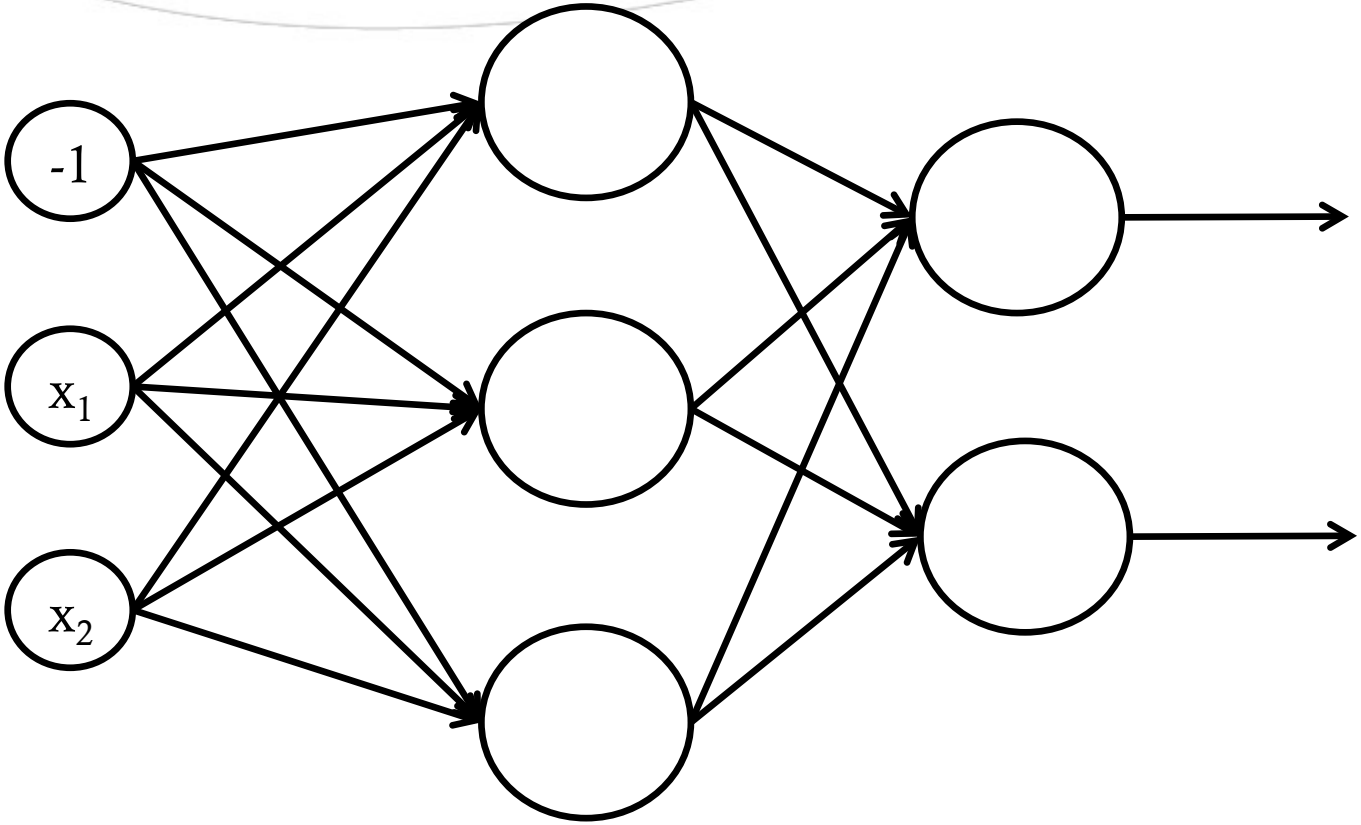
Feed Forward Example



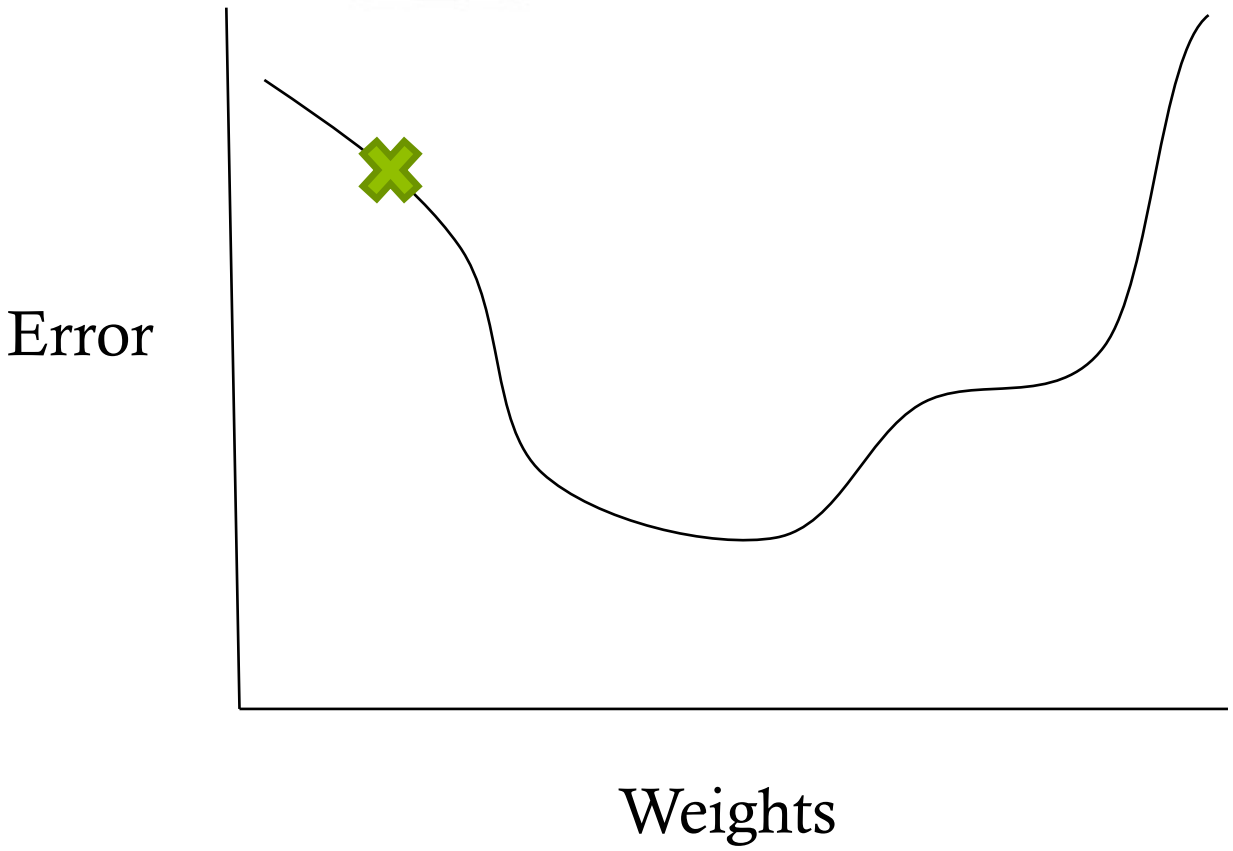
Feed Forward Example

- Input: (0.8, -0.2)
- First Layer Weights:
  - $w_{01} = 0.1$, $w_{11} = -0.1$, $w_{21} = 0.3$
  - $w_{02} = -0.2$, $w_{12} = 0.15$, $w_{22} = 0.25$
- Second Layer Weights:
  - $w_{01} = 0.4$, $w_{11} = 0.2$, $w_{21} = -0.3$

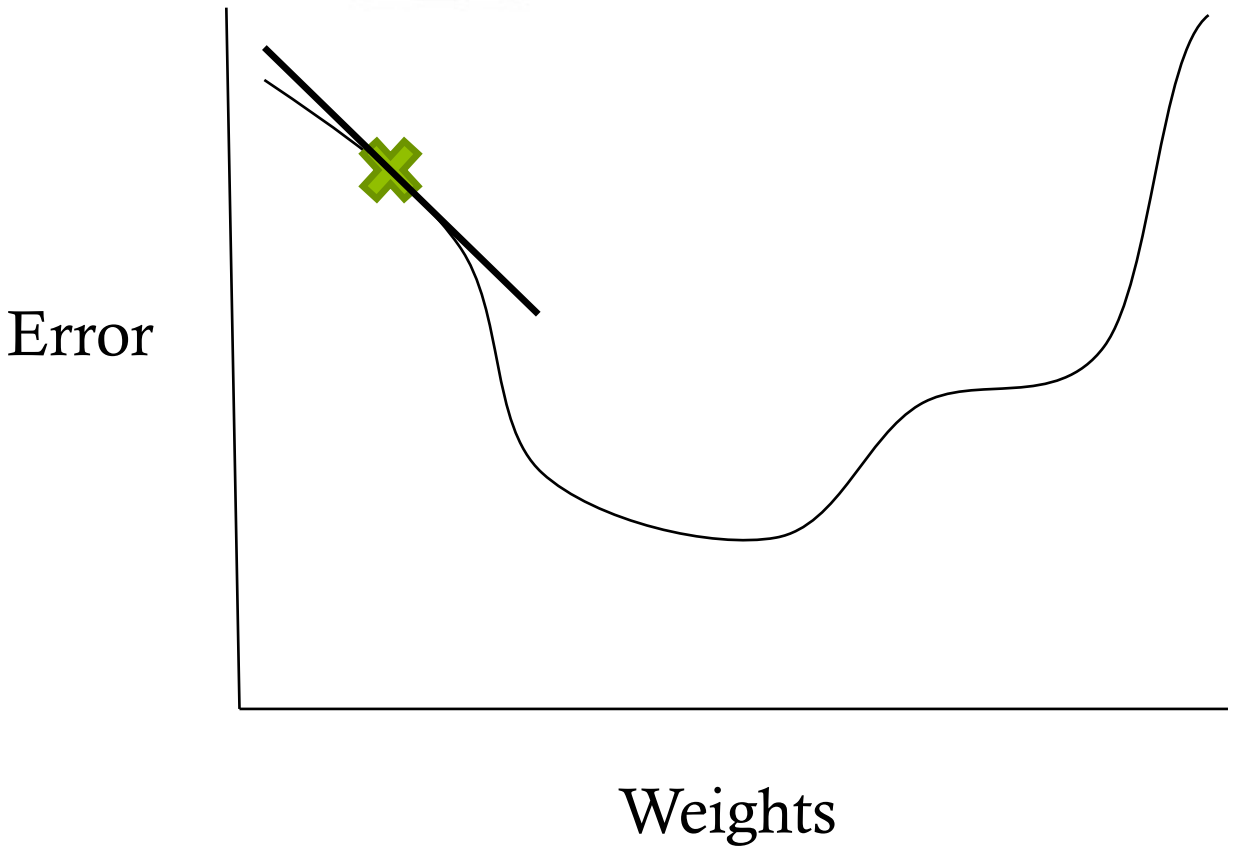
How do we update weights?



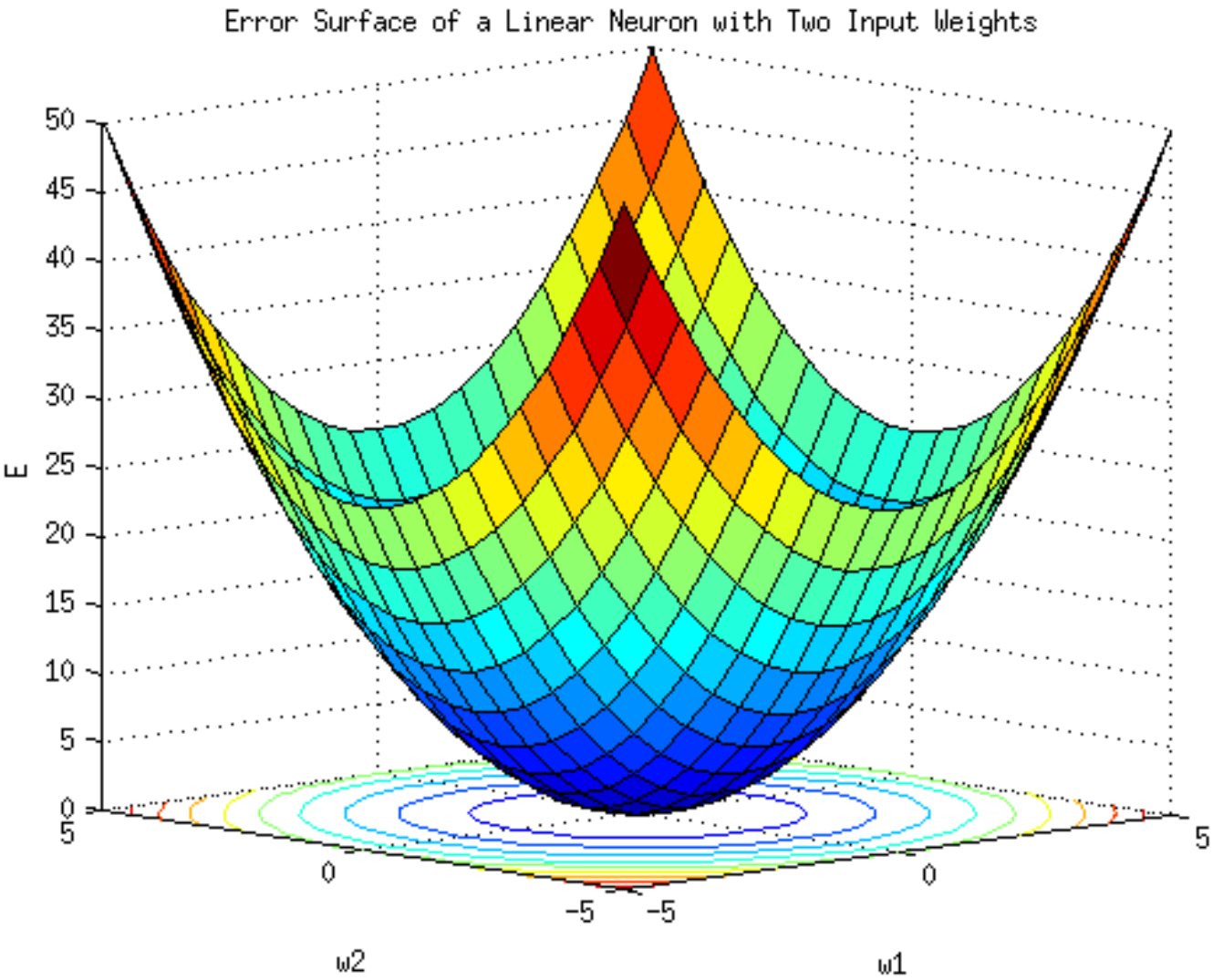
Minimize Error



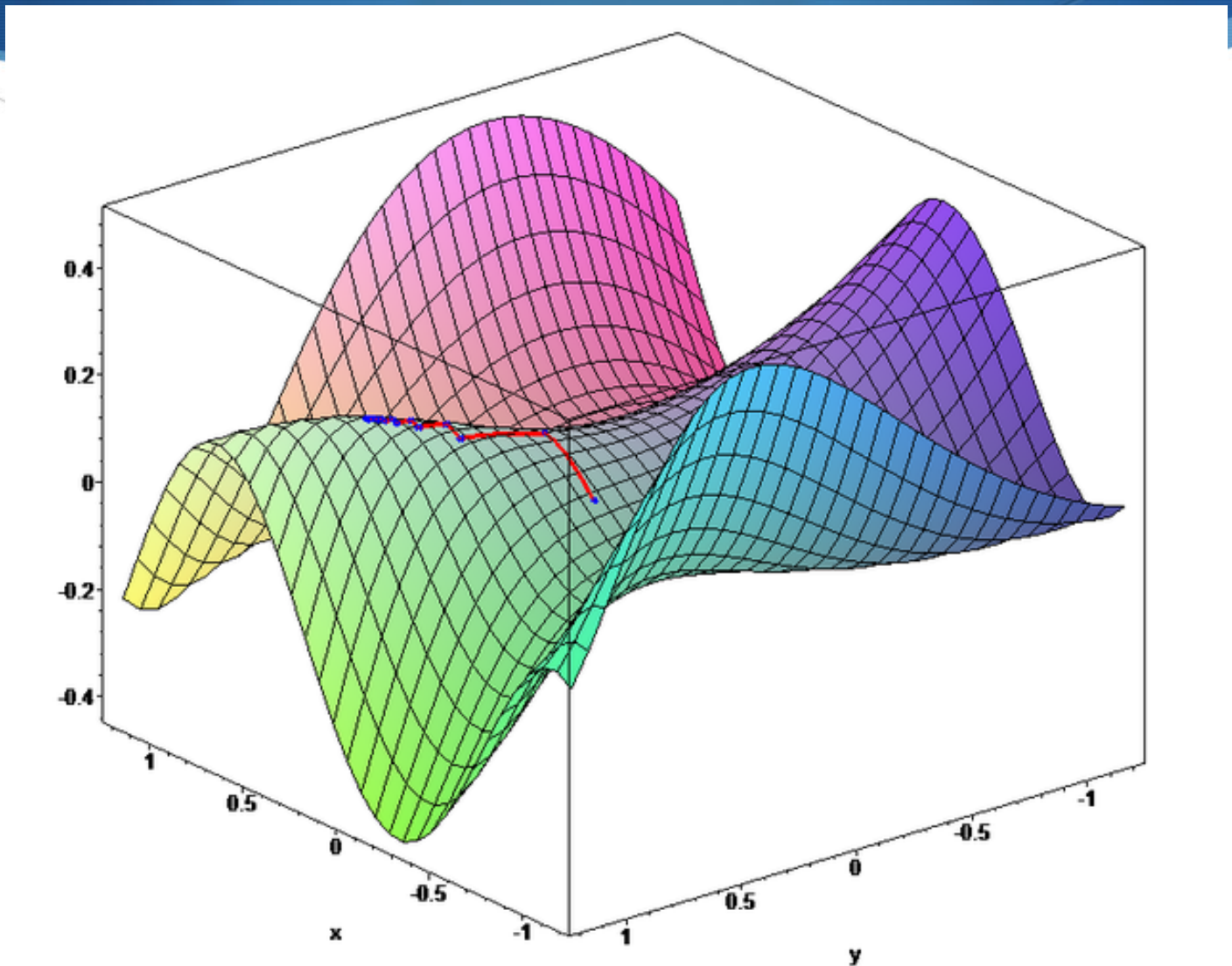
Follow the Derivative



Multiple Dimensions

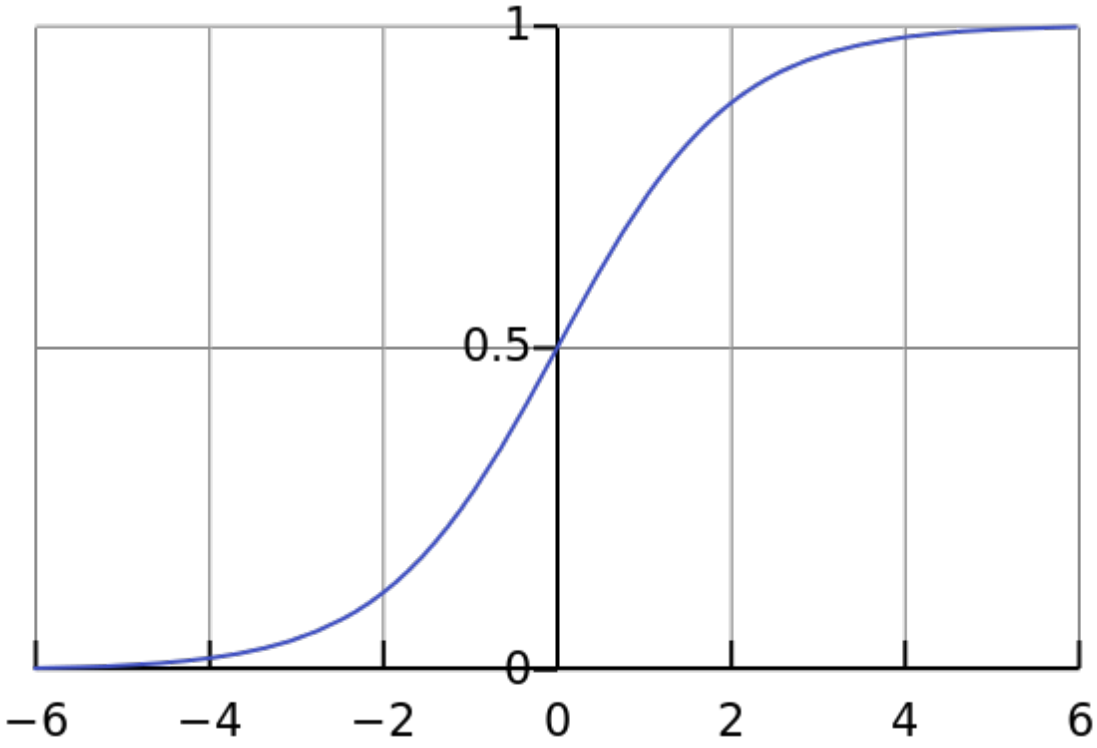


Gradient Descent



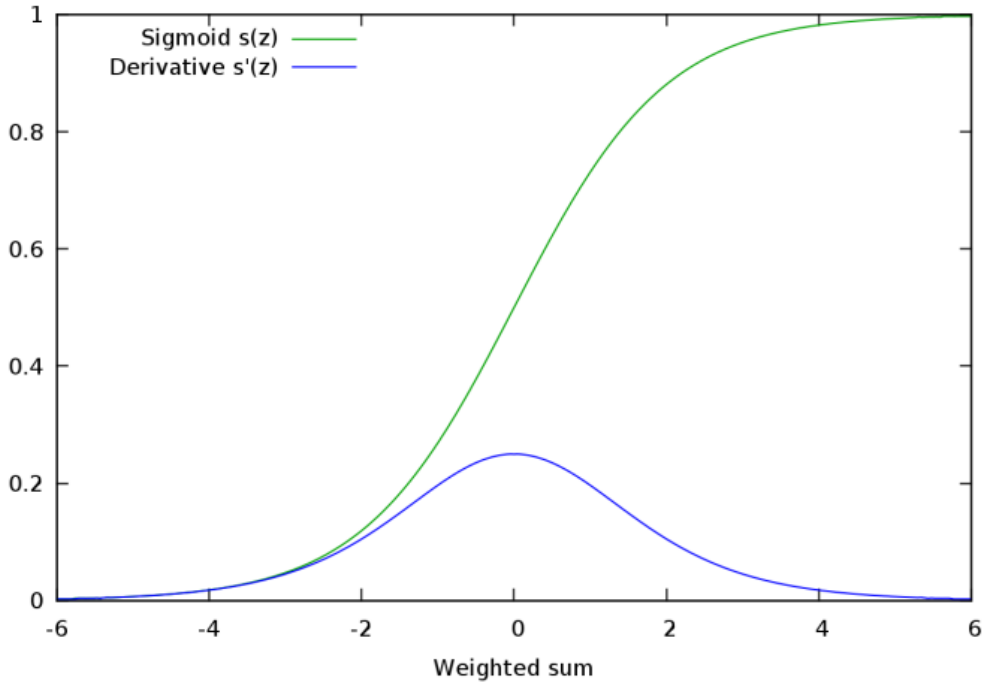
Sigmoid Function

$$f(x) = \frac{1}{1 + e^{-x}}$$



First Derivative

$$f(x) = \frac{1}{1 + e^{-x}} \qquad f'(x) = f(x)(1 - f(x))$$



Error/Update – Output Nodes

$$\delta_j = a_j(1 - a_j)(a_j - t_j)$$

$$w_{ij} \leftarrow w_{ij} - \eta \delta_j a_i$$

i – Inputs (actual input or hidden node value from the layer on the left)

j – Current Node

t_j – Target value of node j

w_{ij} – Weight from input *i* to node *j*

h_j – Weighted sum value for node *j*

a_j – Activation value (output) for node *j*

δ_j – Error of Node *j*

Error – Hidden Nodes

$$\delta_j = a_j(1 - a_j) \sum_{k=1}^{N} w_{jk}\delta_k$$

$$w_{ij} \leftarrow w_{ij} - \eta\delta_j a_i$$

i – Inputs (actual input or hidden node value from the layer on the left)

j – Current Node

k – Node from layer on the right

w_{ij} – Weight from input i to node j

h_j – Weighted sum value for node j

a_j – Activation value (output) for node j

δ_j – Error of Node j

Back-propagation

Hidden Node *j*

$$\delta_j = a_j(1 - a_j) \sum_{k=1}^{N} w_{jk}\delta_k$$

Output Node *j*

$$\delta_j = a_j(1 - a_j)(a_j - t_j)$$

$$w_{ij} \leftarrow w_{ij} - \eta\delta_j a_i$$

Local Minima

Momentum

Batch vs. Sequential Update

Regression