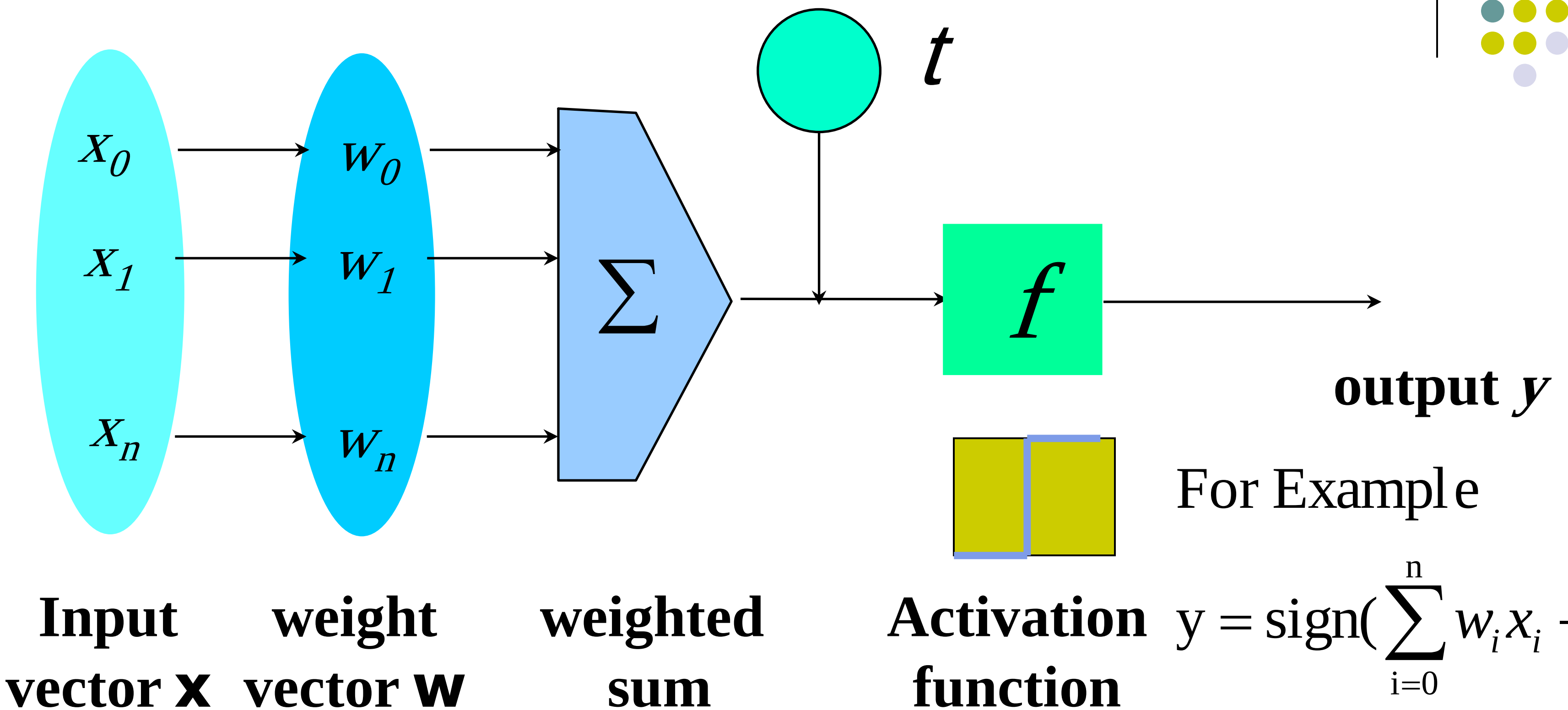
When to consider neural networks

- Input is high-dimensional discrete or raw-valued
- Output is discrete or real-valued
- Output is a vector of values
- Possibly noisy data
- Form of target function is unknown
- Human readability of the result is not important

Examples:

- Speech phoneme recognition
- Image classification
- Financial prediction

A Neuron (= a perceptron)



$$y = \mathrm{sign}\left(\sum_{i=0}^{n} w_i x_i - t\right)$$

- The n-dimensional input vector **x** is mapped into variable y by means of the scalar product and a nonlinear function mapping

Perceptron

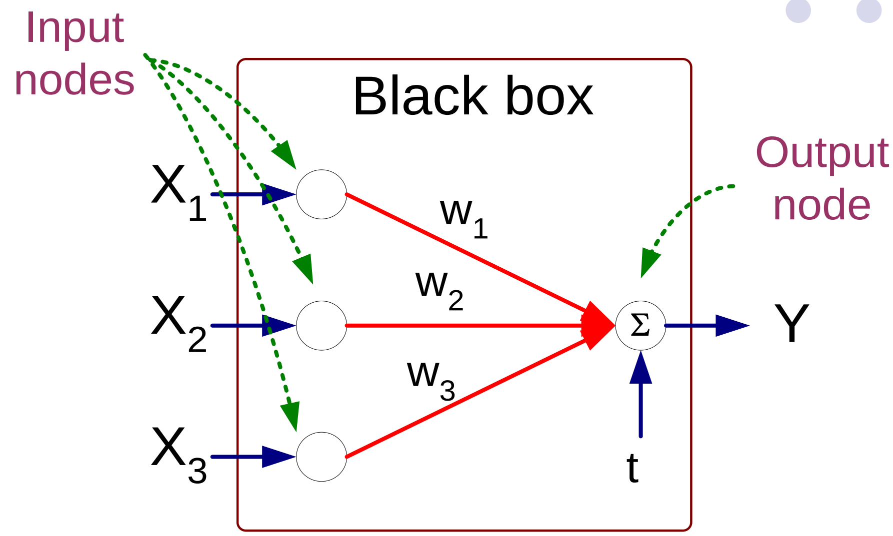
- Basic unit in a neural network
- Linear separator
- Parts
  - N inputs, $x_1 \ldots x_n$
  - Weights for each input, $w_1 \ldots w_n$
  - A bias input x_0 (constant) and associated weight w_0
  - Weighted sum of inputs, $y = w_0x_0 + w_1x_1 + \ldots + w_nx_n$
  - A threshold function or activation function,
    - i.e 1 if $y > t$, -1 if $y \le t$

Artificial Neural Networks (ANN)

- Model is an assembly of inter-connected nodes and weighted links

- Output node sums up each of its input value according to the weights of its links

- Compare output node against some threshold t



Perceptron Model

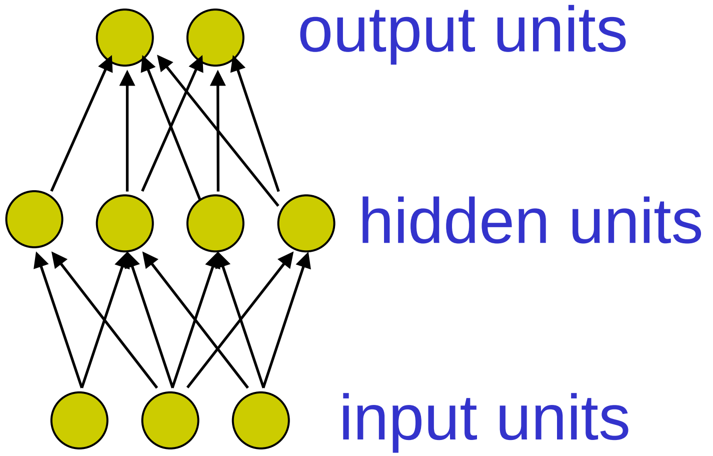
$$Y = I(\sum_i w_i x_i - t)$$

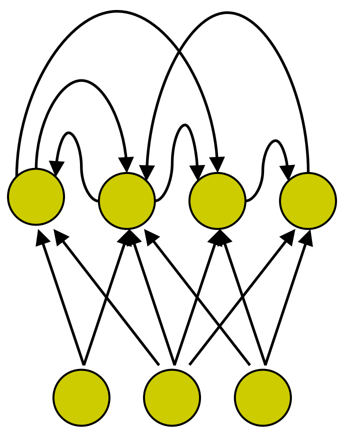
or

$$Y = sign(\sum_i w_i x_i - t)$$

Types of connectivity

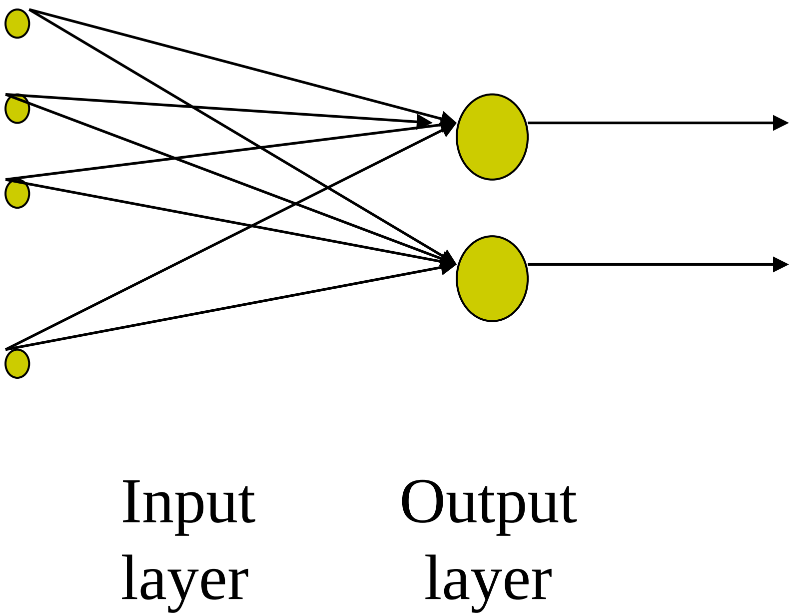
- Feedforward networks
  - These compute a series of transformations
  - Typically, the first layer is the input and the last layer is the output.
- Recurrent networks
  - These have directed cycles in their connection graph. They can have complicated dynamics.
  - More biologically realistic.





Different Network Topologies

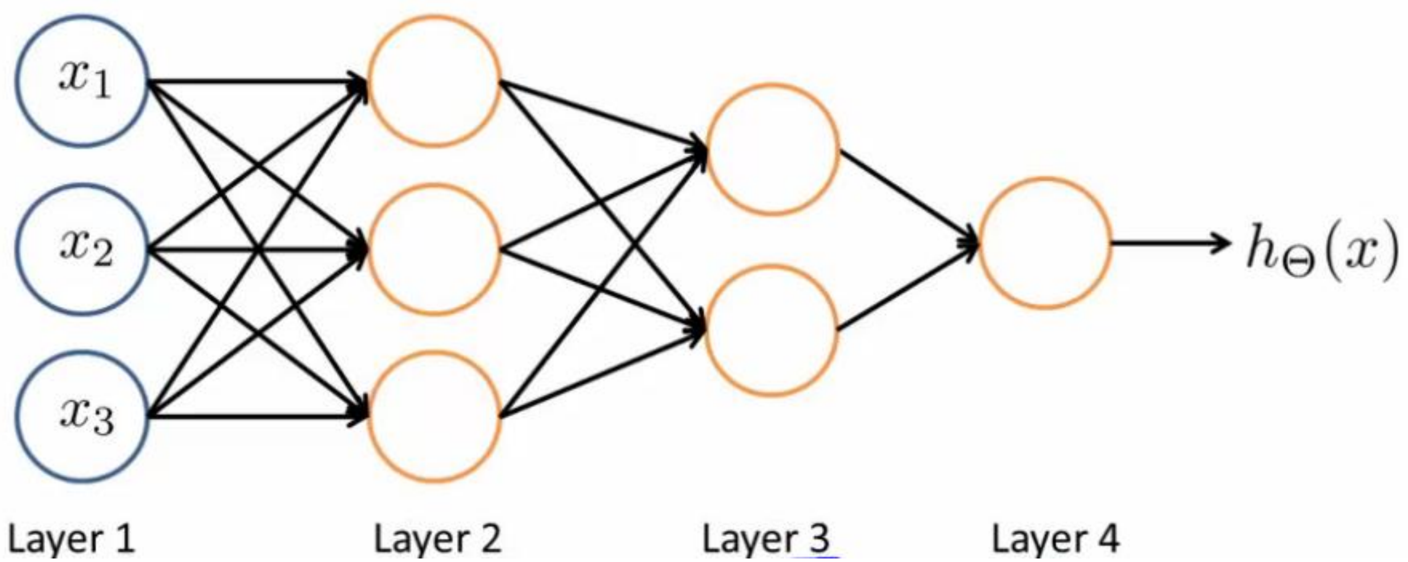
- Single layer feed-forward networks
  - Input layer projecting into the output layer



Single layer
network

Different Network Topologies

- Multi-layer feed-forward networks



Input layer

Hidden layers

Output layer

Algorithm for learning ANN

- Initialize the weights ($w_0, w_1, \ldots, w_k$)

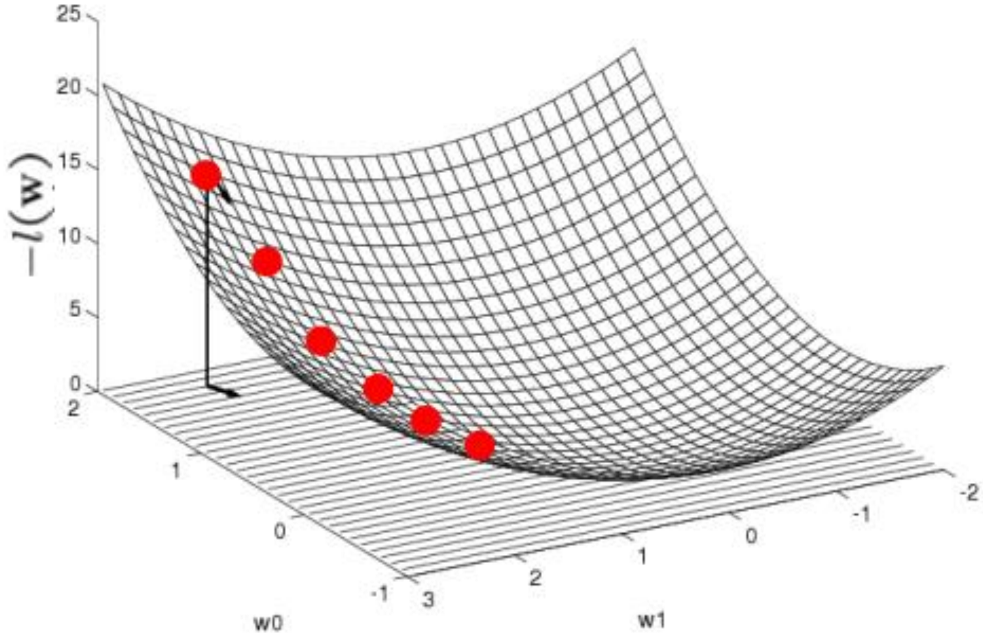
- Adjust the weights in such a way that the output of ANN is consistent with class labels of training examples
  - Error function: $$\text{Loss} = -\frac{1}{\substack{\text{output}\\\text{size}}} \sum_{i=1}^{\substack{\text{output}\\\text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

  - Find the weights w_i's that minimize the above error function
    - e.g., gradient descent, backpropagation algorithm

Optimizing concave/convex function

- Maximum of a concave function = minimum of a convex function

Gradient ascent (concave) / Gradient descent (convex)



Gradient:

$$\nabla_{\mathbf{w}} l(\mathbf{w}) = [\frac{\partial l(\mathbf{w})}{\partial w_0}, \ldots, \frac{\partial l(\mathbf{w})}{\partial w_d}]'$$

Update rule: Learning rate, η>0

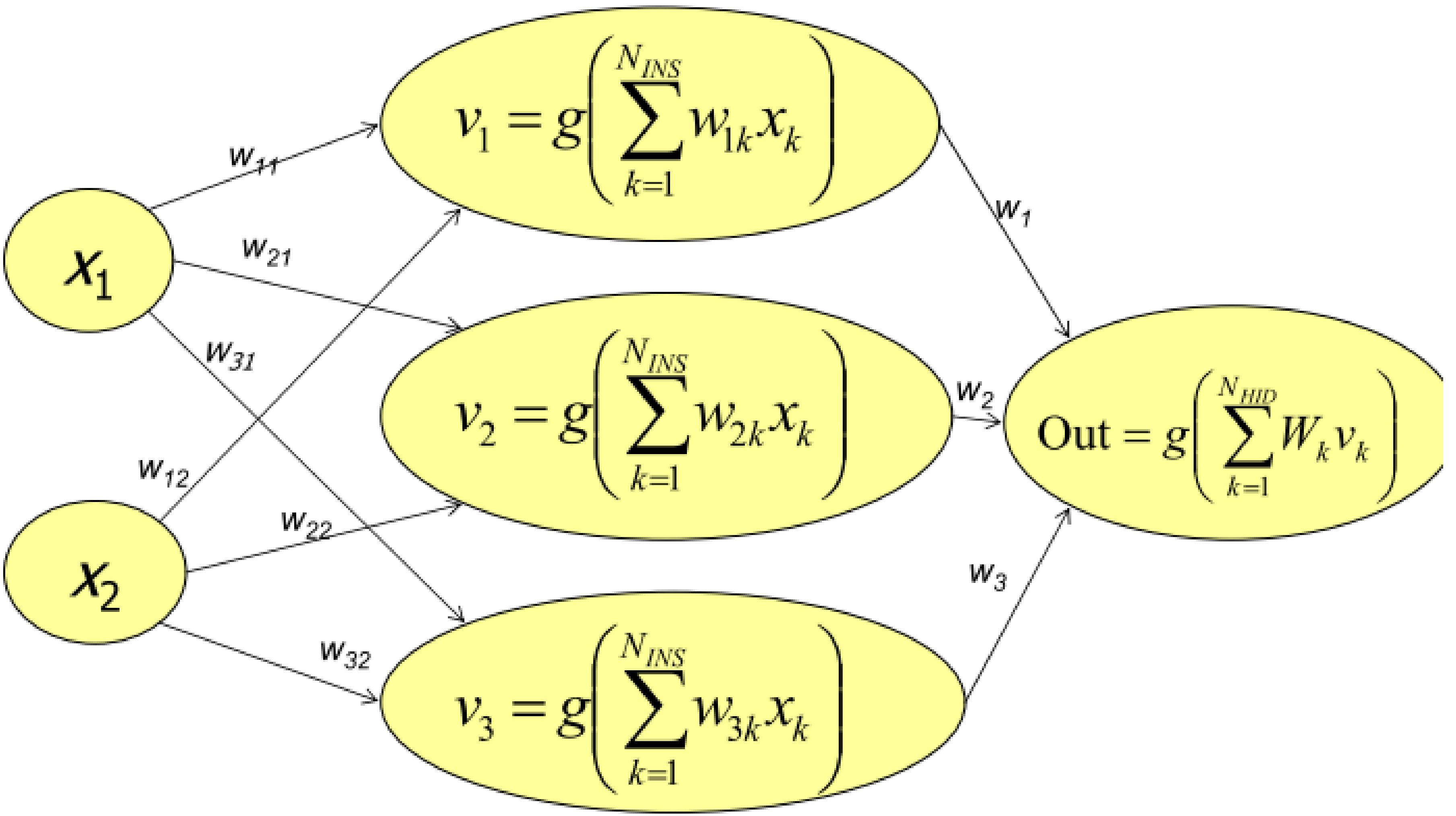
$$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left. \frac{\partial l(\mathbf{w})}{\partial w_i} \right|_t$$

Gradient ascent rule

A 1-HIDDEN LAYER NET

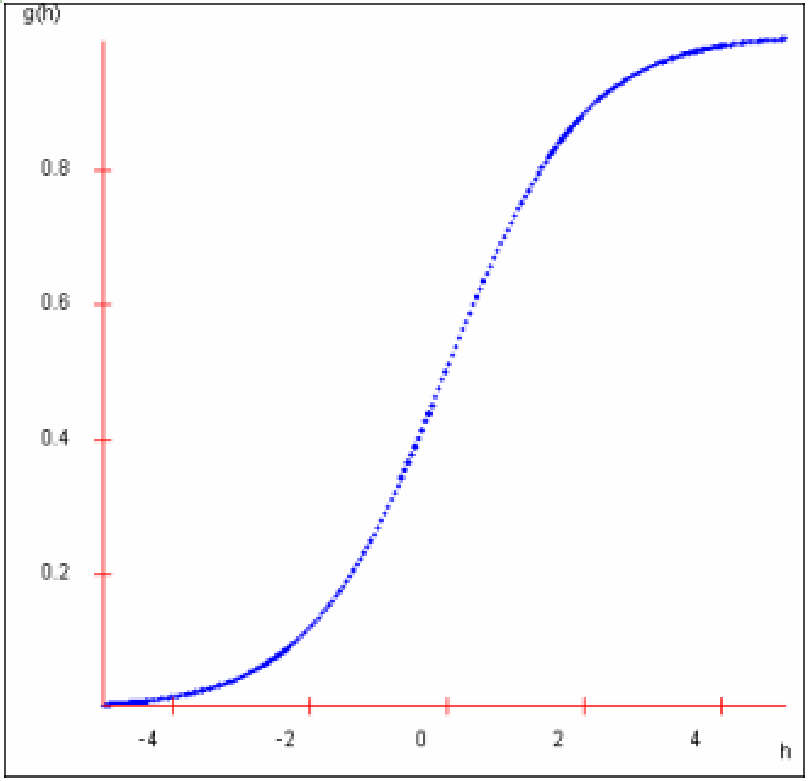
$N_{INPUTS} = 2$ $N_{HIDDEN} = 3$



Copyright © 2001, 2003, Andrew W. Moore

The Sigmoid

$$g(h) = \frac{1}{1+\exp(-h)}$$

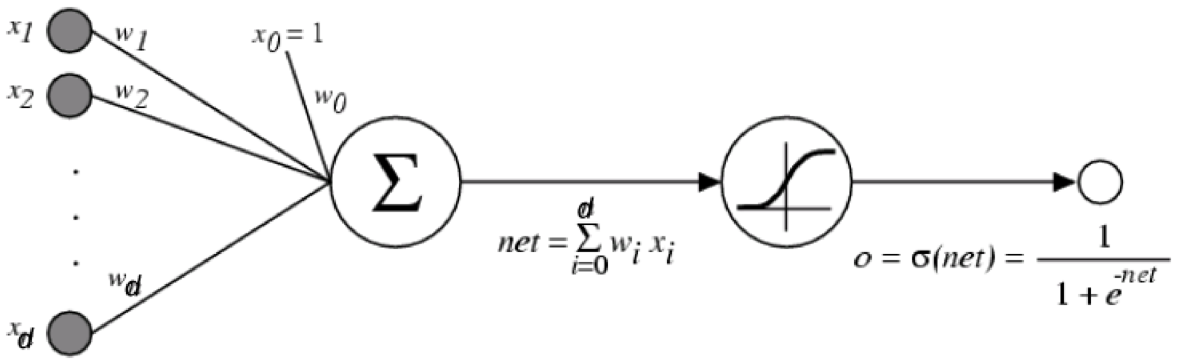


Now we choose ***w*** to minimize

$$\sum_{i=1}^{R} [y_i - \mathrm{Out}(\mathbf{x}_i)]^2 = \sum_{i=1}^{R} [y_i - g(\mathbf{w}^T \mathbf{x}_i)]^2$$

Copyright © 2001, 2003, Andrew W. Moore

Sigmoid Unit



$\sigma(x)$ is the sigmoid function/activation function (also linear, threshold)

$$\frac{1}{1+e^{-x}}$$

Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1-\sigma(x))$ Differentiable

We can derive gradient decent rules to train

- One sigmoid unit
- *Multilayer networks* of sigmoid units $\rightarrow$ Backpropagation

Backpropagation

$$\text{Out}(\mathbf{x}) = g\left(\sum_j W_j \, g\left(\sum_k w_{jk} x_k\right)\right)$$

Find a set of weights $\{W_j\},\{w_{jk}\}$

to minimize

$$\sum_i \left(y_i - \text{Out}(\mathbf{x}_i)\right)^2$$

by gradient descent.

That's it!

That's the backpropagation algorithm.

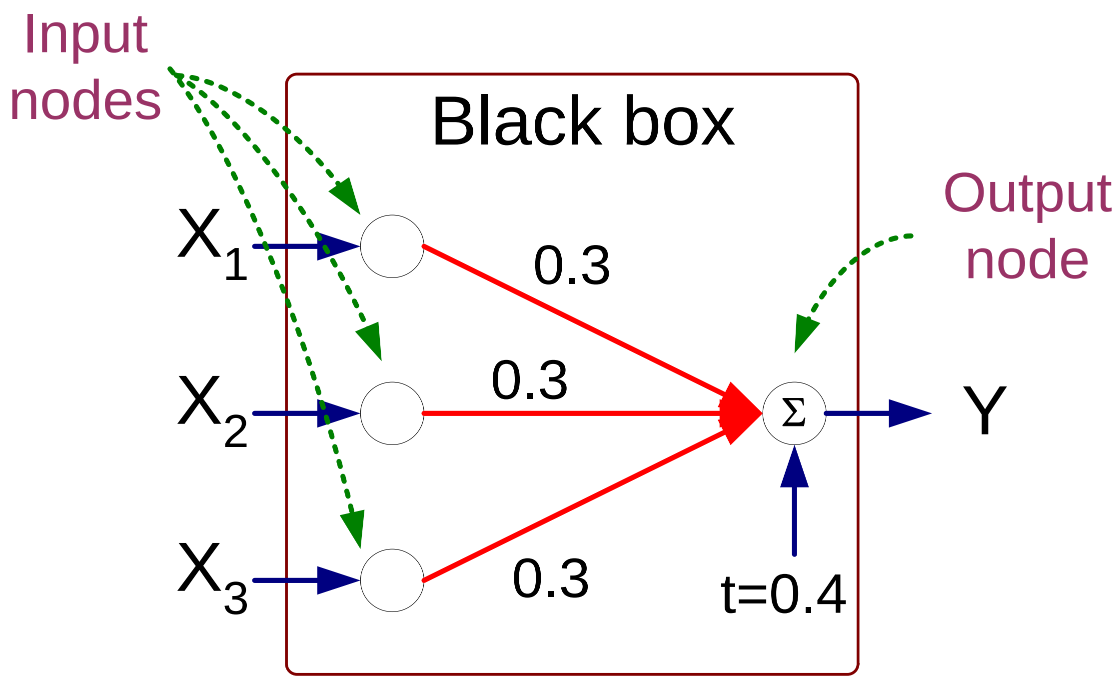
Copyright © 2001, 2003, Andrew W. Moore

Backpropagation

- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value
- Modifications are made in the "**backwards**" direction: from the output layer, through each hidden layer down to the first hidden layer, hence "**backpropagation**"
- Steps
  - Initialize weights (to small random #s) and biases in the network
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

Artificial Neural Networks (ANN)

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0

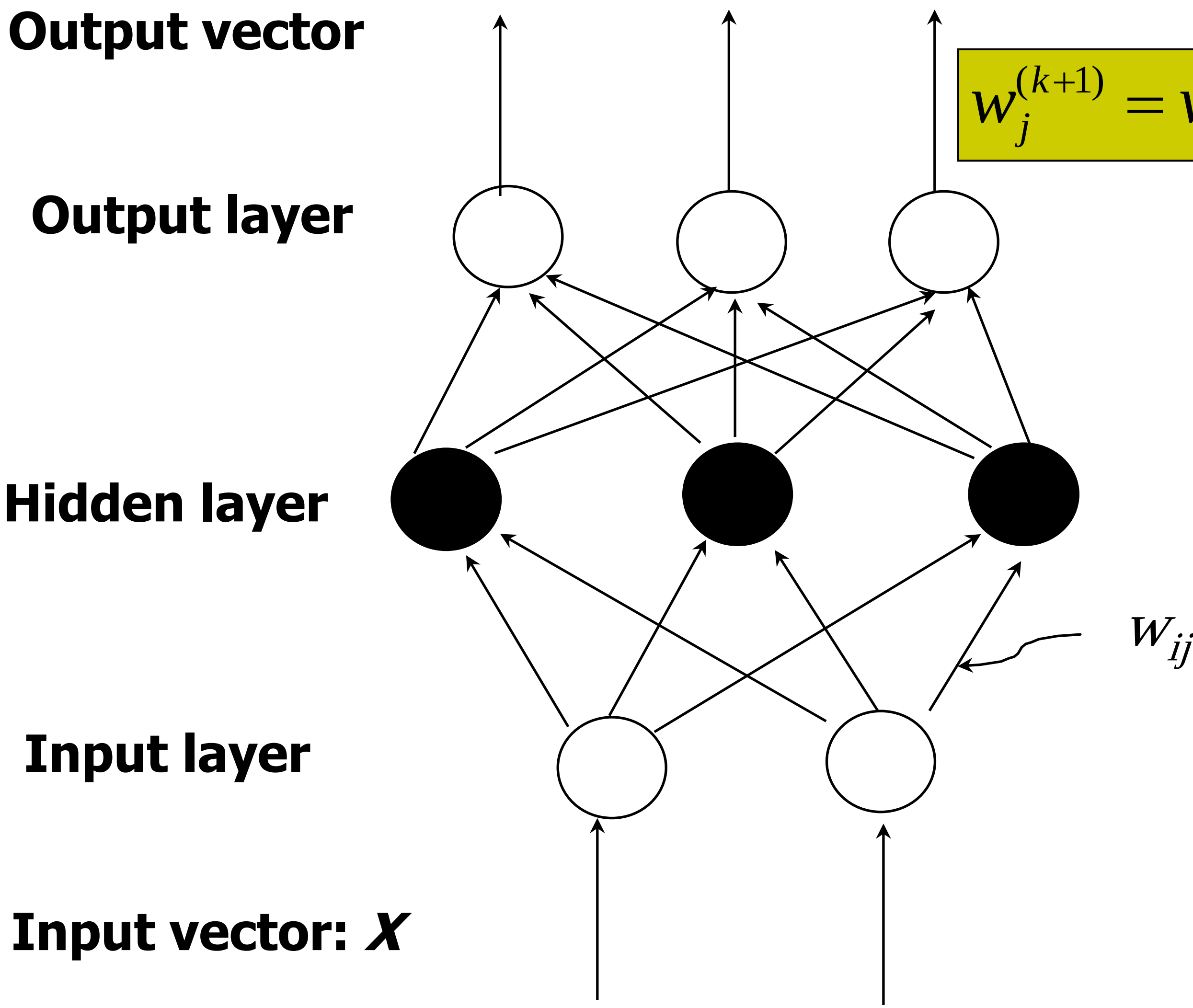


$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

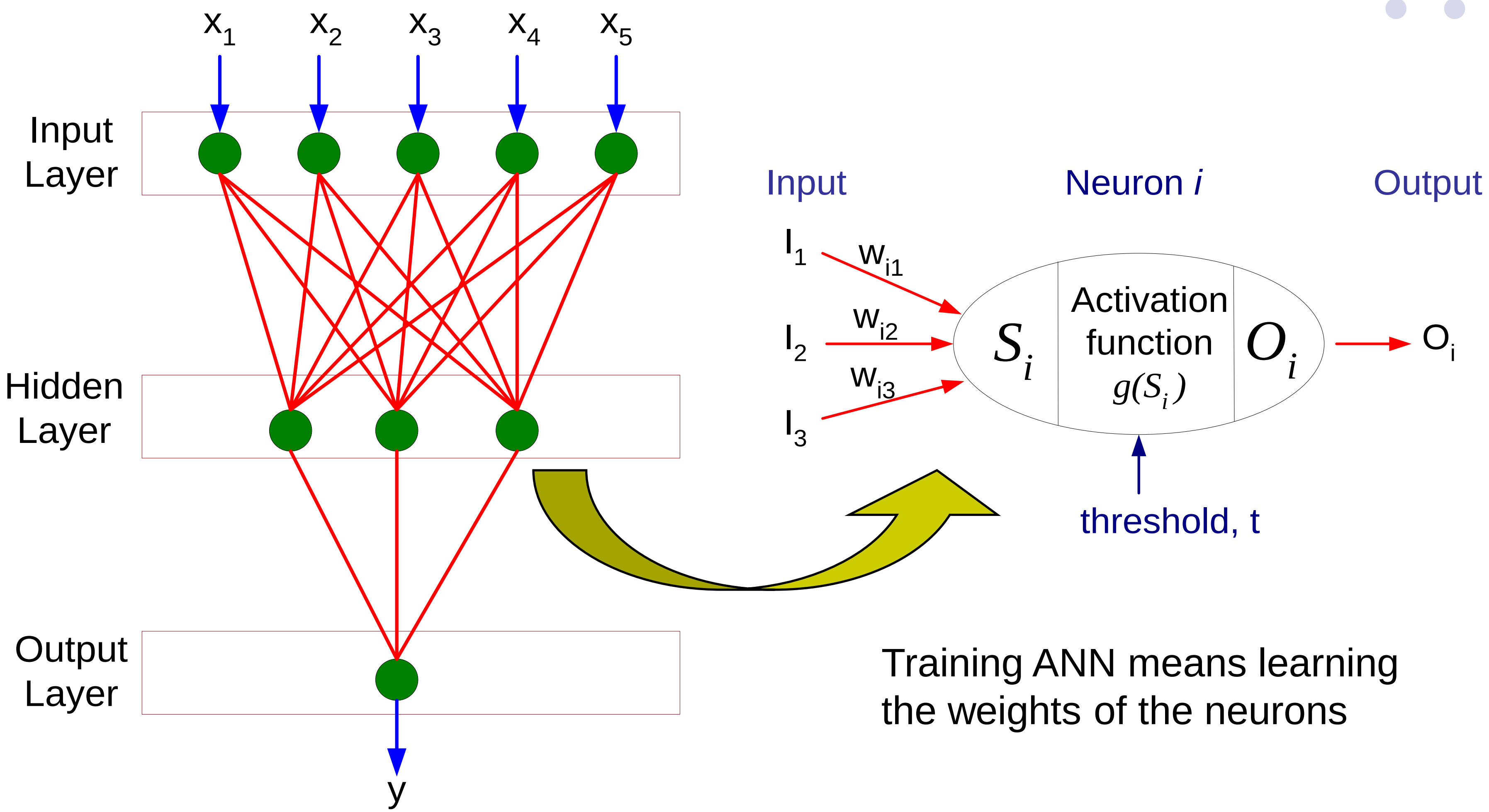
$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

A Multi-Layer Feed-Forward Neural Network

$$w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij}$$



General Structure of ANN



Training ANN means learning the weights of the neurons