Lecture 14
Minimum Distance
&
Neural Network

Minimum Distance Classifier

- For a test sample X, compute $D_j(X)$ for each class j
- Assign class with minimum D(x) value

$$D_j(\mathbf{x}) = \left\| \mathbf{x} - \mathbf{m}_j \right\|$$

$$= \left[(\mathbf{x} - \mathbf{m}_j)^T (\mathbf{x} - \mathbf{m}_j) \right]^{1/2}$$

- Here m_j is mean value of training samples from j^{th} class

Minimum Distance Classifier

Manipulating $D_j(X)$

$$
\begin{aligned}
D_j^2(\mathbf{x}) &= \left\|\mathbf{x}-\mathbf{m}_j\right\|^2 = (\mathbf{x}-\mathbf{m}_j)^T(\mathbf{x}-\mathbf{m}_j) \\
&= \mathbf{x}^T\mathbf{x}-2\mathbf{x}^T\mathbf{m}_j+\mathbf{m}_j^T\mathbf{m}_j \\
&= \mathbf{x}^T\mathbf{x}-\mathbf{2}\left(\mathbf{x}^T\mathbf{m}_j-\frac{1}{2}\mathbf{m}_j^T\mathbf{m}_j\right).
\end{aligned}
$$

Minimum Distance Classifier

- Now instead of $D_j(X)$, we compute discriminent function $d_j(X)$ for each class
- Assign class with maximum $d_j(X)$ value

$$d_j(\mathbf{x}) = \mathbf{x}^T\mathbf{m}_j - \frac{1}{2}\mathbf{m}_j^T\mathbf{m}_j \qquad j = 1, 2, \ldots, W$$

- Equation for decision boundary between two classes *i* and *j*

$$d_{ij}(\mathbf{x}) = \mathbf{x}^T(\mathbf{m}_i - \mathbf{m}_j) - \frac{1}{2}(\mathbf{m}_i^T\mathbf{m}_i - \mathbf{m}_j^T\mathbf{m}_j)$$

Minimum Distance Classifier

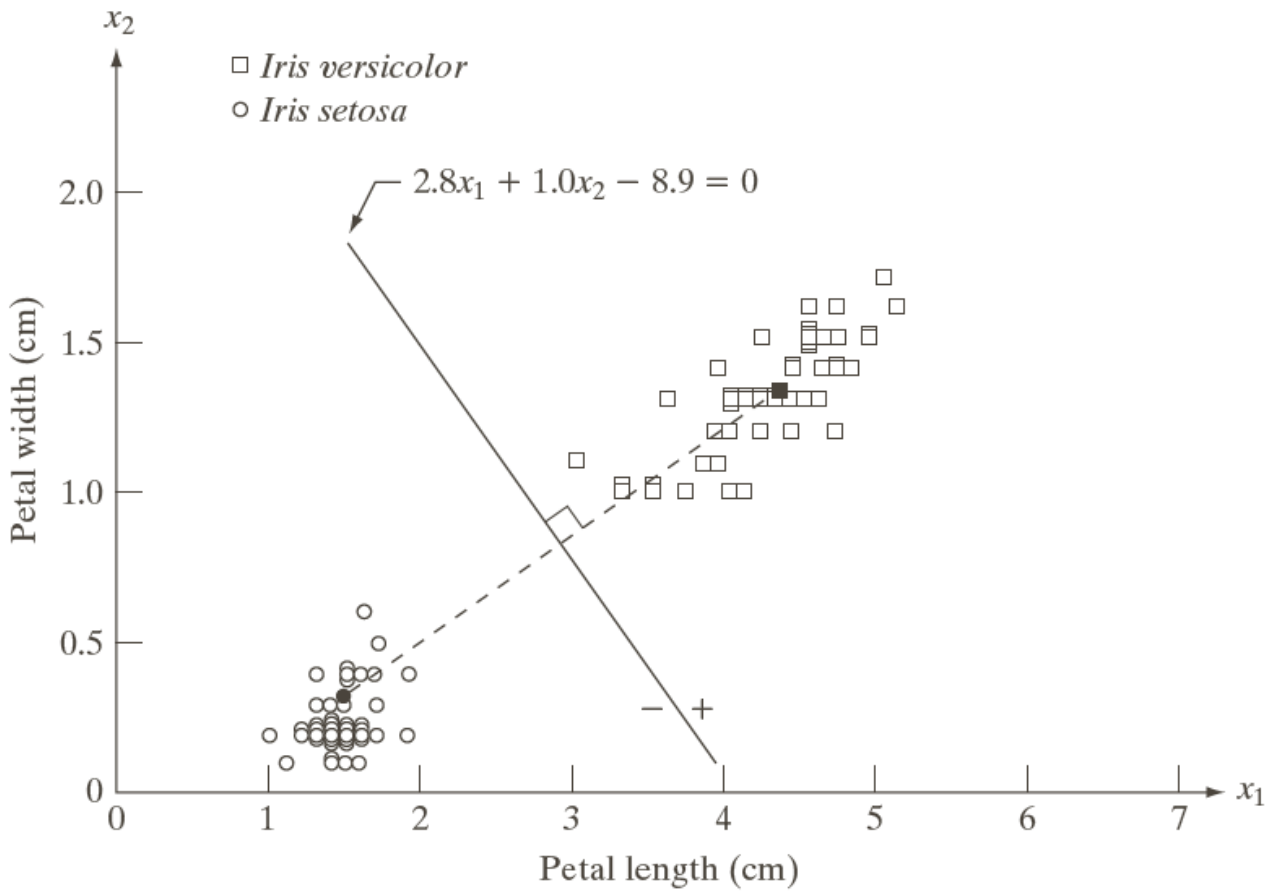
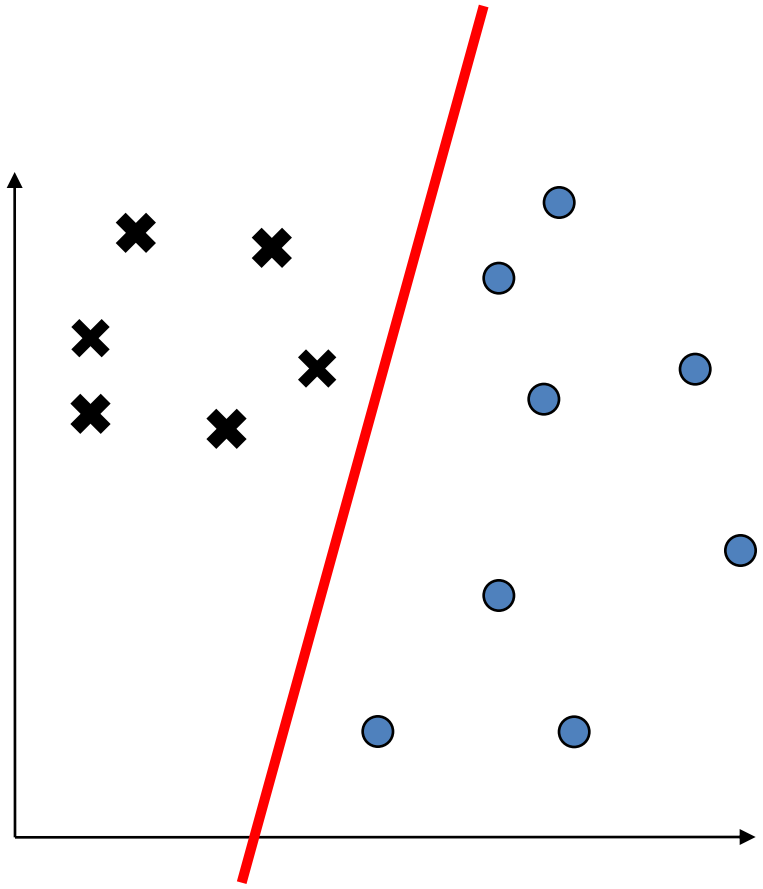


FIGURE 12.6 Decision boundary of minimum distance classifier for the classes of *Iris versicolor* and *Iris setosa*. The dark dot and square are the means.

Artificial Neural Network - Perceptron

A (Linear) Decision Boundary



Represented by:

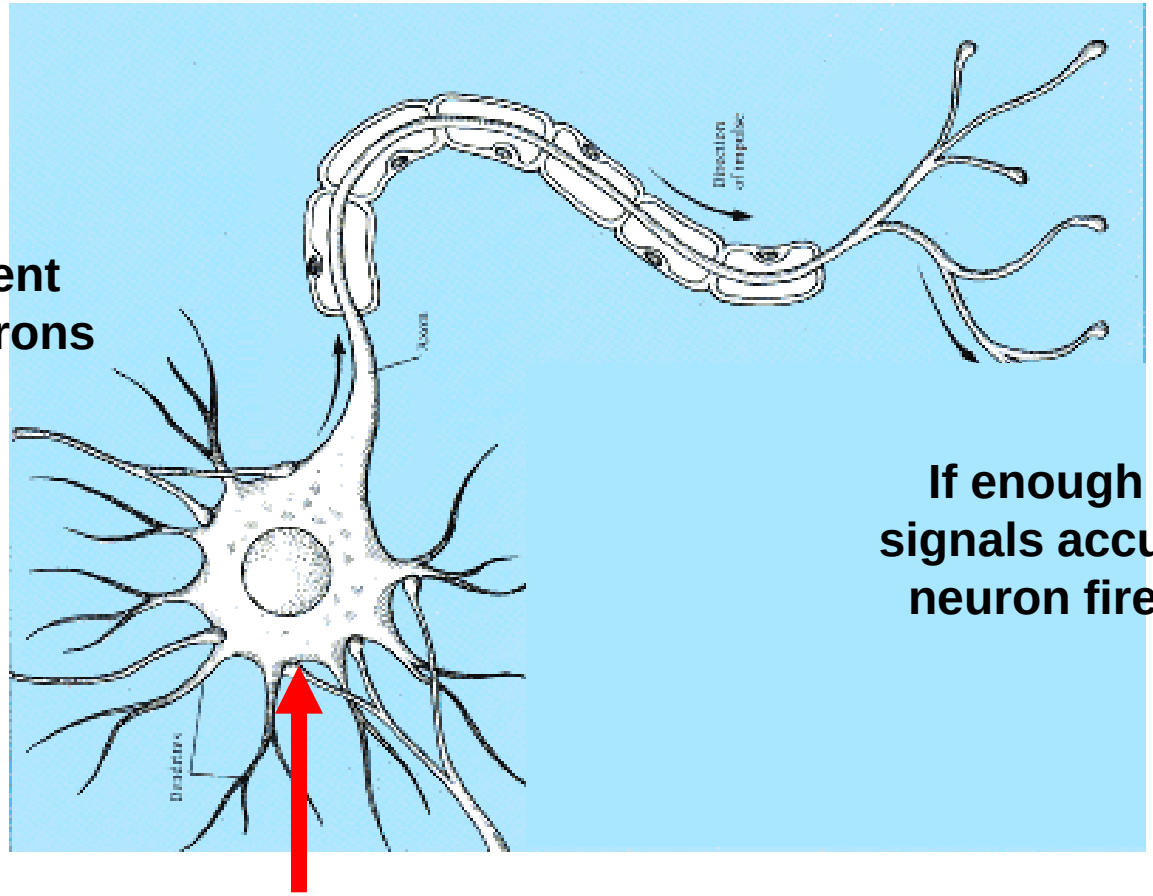
One artificial neuron called a "Perceptron"

Low accuracy (mostly)

Low space complexity

Low time complexity

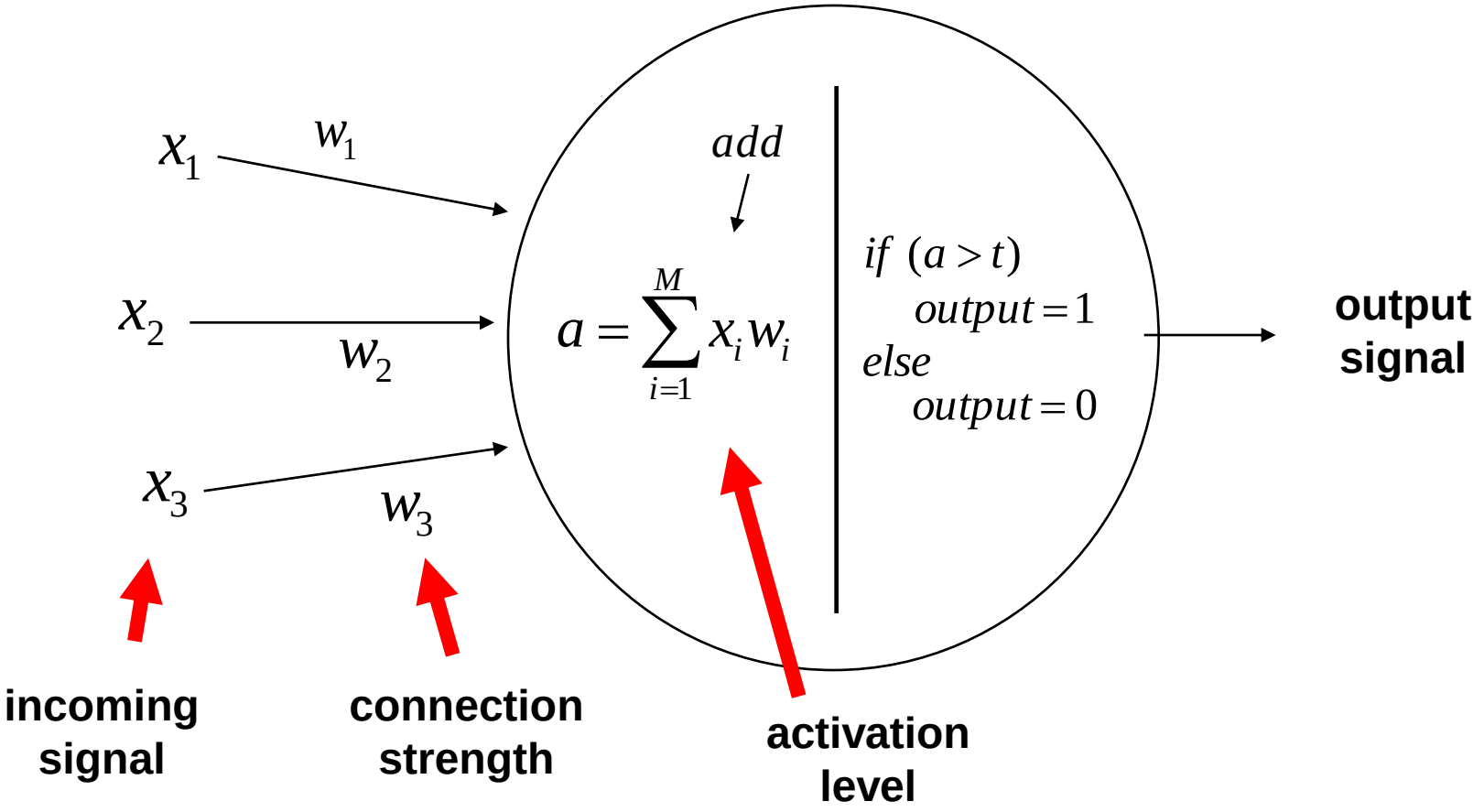
Input signals sent from other neurons



If enough sufficient signals accumulate, the neuron fires a signal.

Connection strengths determine how the signals are accumulated

- input signals ‘x’ and weights ‘w’ are multiplied
- weights correspond to connection strengths
- signals are added up – if they are enough, FIRE!



Calculation…

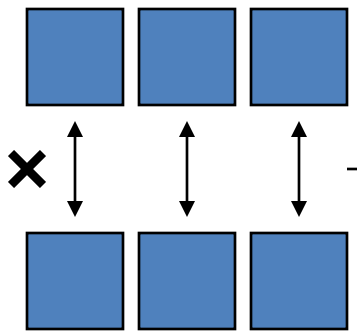
$$a = \sum_{i=1}^{M} x_i w_i$$

Sum notation

(just like a loop from 1 to M)

`double[] x =`

`double[] w =`



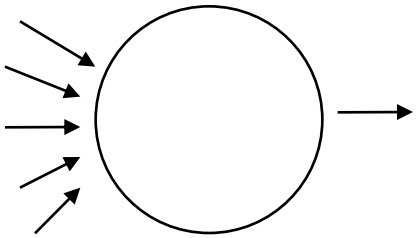
`a` (activation)

Multiple corresponding elements and add them up

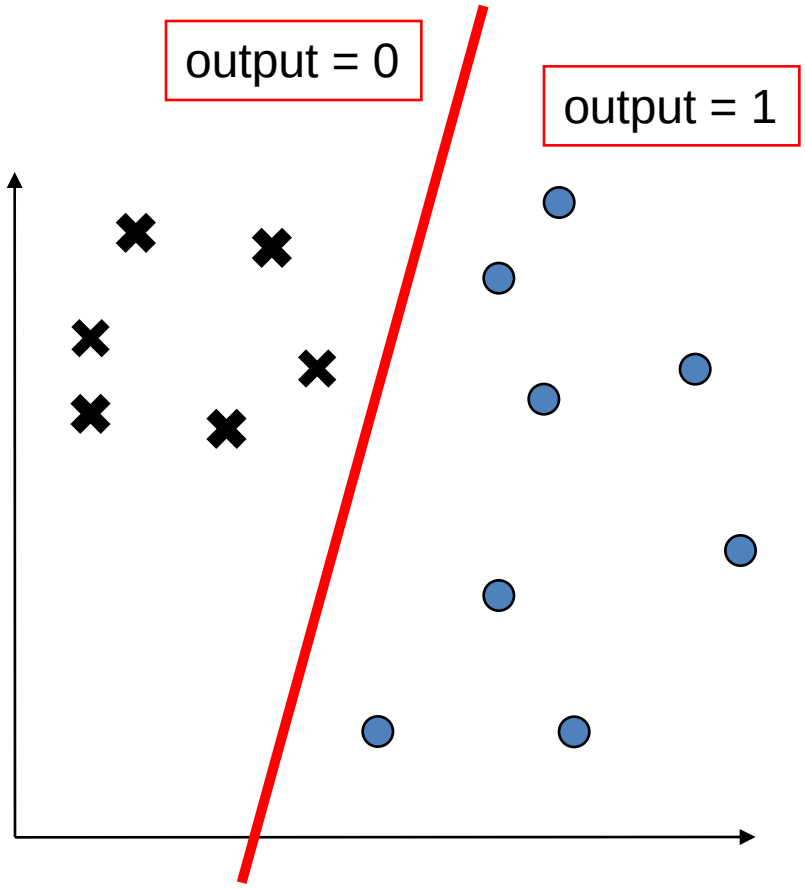
if (activation > threshold) FIRE !

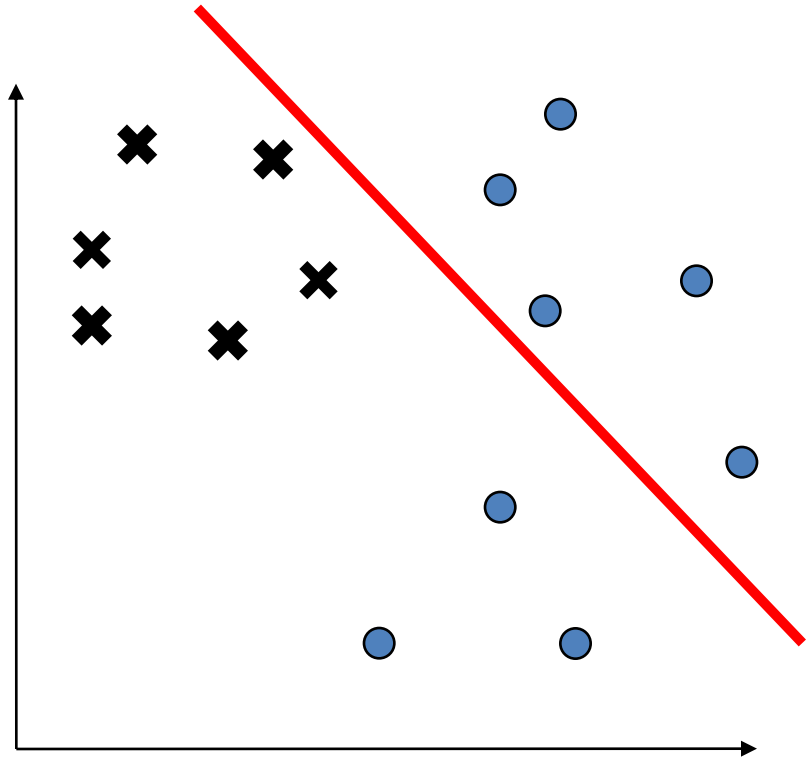
Perceptron Decision Rule

$$\textit{if} \quad \left(\sum_{i=1}^{M} x_i w_i \right) \quad > \quad t \qquad \text{then } output = 1, \text{ else } output = 0$$



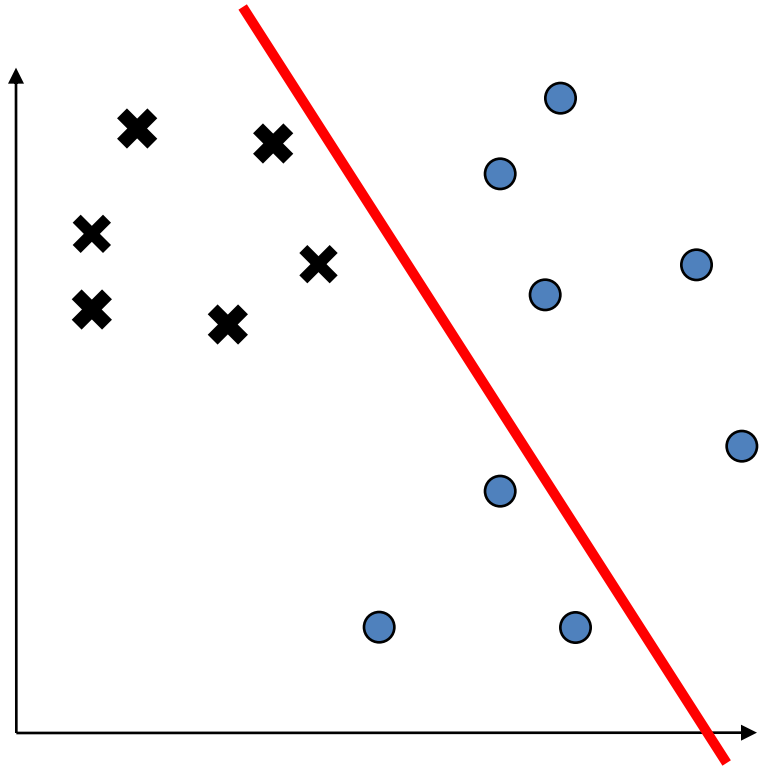
$$\text{if} \quad \left(\sum_{i=1}^{M} x_i w_i \right) \quad > \quad t \qquad \text{then } output = 1, \text{ else } output = 0$$





Is this a good decision boundary?

$$\textit{if} \quad \left(\sum_{i=1}^{M} x_i w_i \right) \quad > \quad t \qquad \text{then } output = 1, \text{ else } output = 0$$

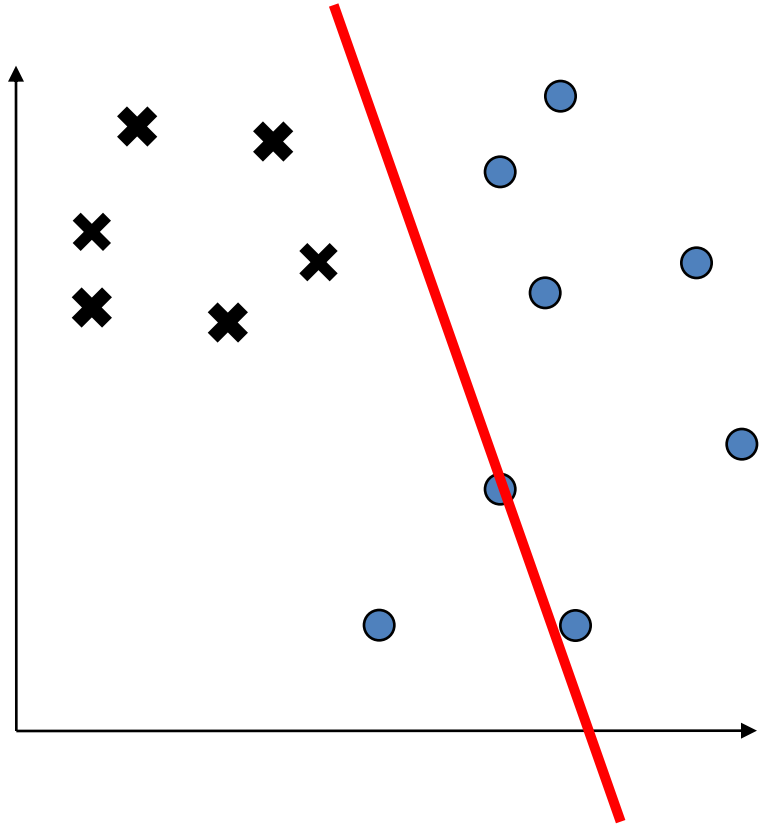


$w_1 = 1.0$

$w_2 = 0.2$

$t = 0.05$

$$\textit{if} \quad \left(\sum_{i=1}^{M} x_i w_i \right) \quad > \quad t \qquad \text{then } output = 1, \text{ else } output = 0$$

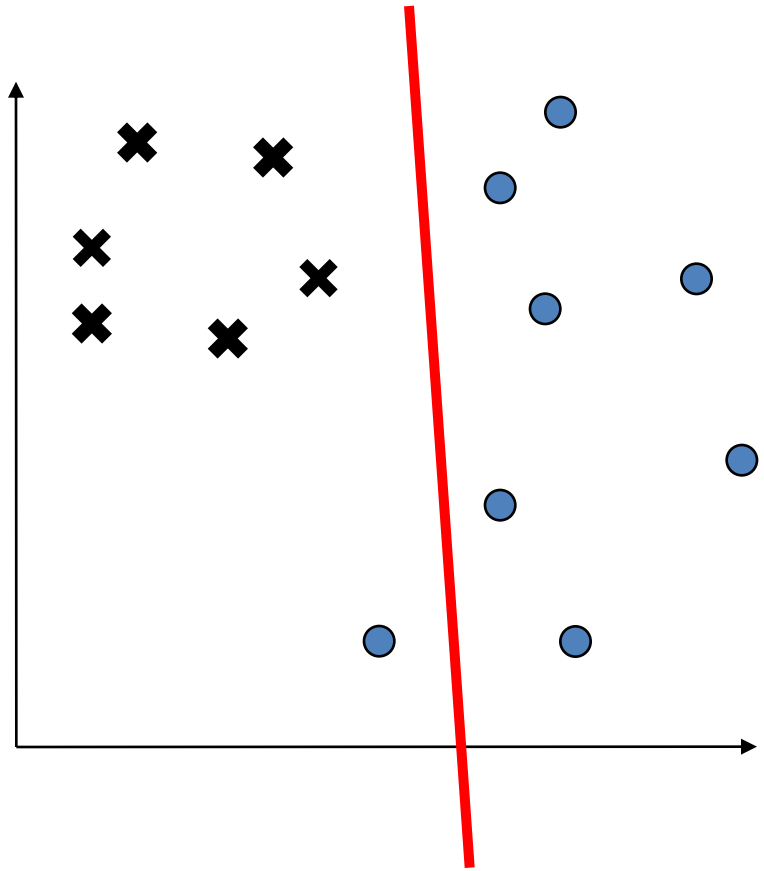


$w_1 = 2.1$

$w_2 = 0.2$

$t = 0.05$

$$\textit{if} \quad \left(\sum_{i=1}^{M} x_i w_i \right) \quad > \quad t \qquad \text{then } output = 1, \text{ else } output = 0$$

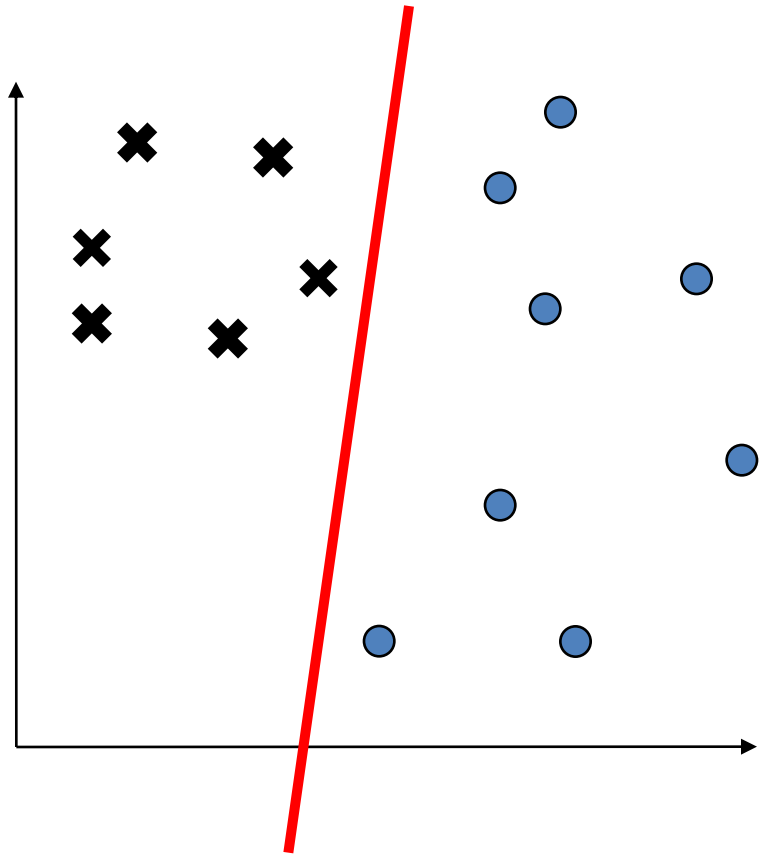


$w_1 = 1.9$

$w_2 = 0.02$

$t = 0.05$

$$\textit{if} \quad \left(\sum_{i=1}^{M} x_i w_i \right) \quad > \quad t \qquad \text{then } output = 1, \text{ else } output = 0$$



$w_1 = -0.8$

$w_2 = 0.03$

$t = 0.05$

Changing the weights/threshold makes the decision boundary move.

Pointless / impossible to do it by hand – only ok for simple 2-D case.

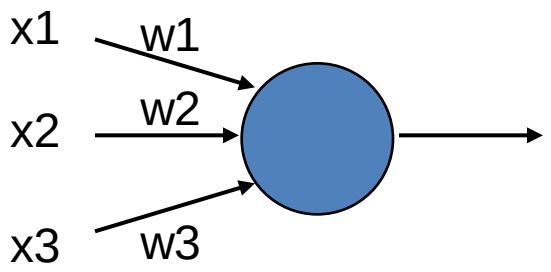
We need an algorithm….

$$x = [\ 1.0,\ 0.5,\ 2.0\ ]$$

$$w = [\ 0.2,\ 0.5,\ 0.5\ ]$$

$$t = 1.0$$

$$a = \sum_{i=1}^{M} x_i w_i$$



Q1. What is the activation, ***a***, of the neuron?

Q2. Does the neuron fire?

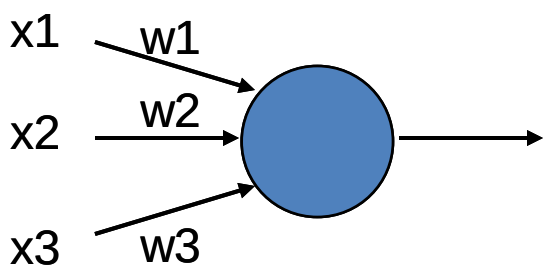
Q3. What if we set threshold at 0.5 and weight #3 to zero?

$$x = [\ 1.0,\ 0.5,\ 2.0\ ]$$

$$w = [\ 0.2,\ 0.5,\ 0.5\ ]$$

$$t = 1.0$$

$$a = \sum_{i=1}^{M} x_i w_i$$



Q1. What is the activation, *a*, of the neuron?

$$a = \sum_{i=1}^{M} x_i w_i = (1.0 \times 0.2) + (0.5 \times 0.5) + (2.0 \times 0.5) = 1.45$$

Q2. Does the neuron fire?

if (activation > threshold) output=1 else output=0

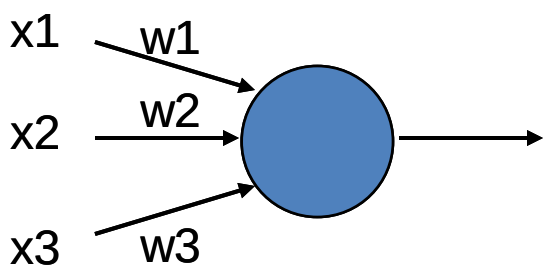
…. So yes, it fires.

$$x = [\ 1.0,\ 0.5,\ 2.0\ ]$$

$$w = [\ 0.2,\ 0.5,\ 0.5\ ]$$

$$t = 1.0$$

$$a = \sum_{i=1}^{M} x_i w_i$$



Q3. What if we set threshold at 0.5 and weight #3 to zero?

$$a = \sum_{i=1}^{M} x_i w_i = (1.0 \times 0.2) + (0.5 \times 0.5) + (2.0 \times 0.0) = 0.45$$

if (activation > threshold) output=1 else output=0

…. So no, it does not fire..

We can rearrange the decision rule….

$$\text{if} \quad \left(\sum_{i=1}^{M} x_i w_i \right) > t \quad \text{then } output = 1, \text{ else } output = 0$$

$$\text{if} \quad \left(\sum_{i=1}^{M} x_i w_i \right) - t > 0 \quad \text{then } output = 1, \text{ else } output = 0$$

$$\text{if} \quad \left(\sum_{i=1}^{M} x_i w_i \right) + (-1 \times t) > 0 \quad \text{then } output = 1, \text{ else } output = 0$$

$$\text{if} \quad \left(\sum_{i=1}^{M} x_i w_i \right) + (x_0 \times w_0) > 0 \quad \text{then } output = 1, \text{ else } output = 0$$

$$\text{if} \quad \left(\sum_{i=0}^{M} x_i w_i \right) > 0 \quad \text{then } output = 1, \text{ else } output = 0$$

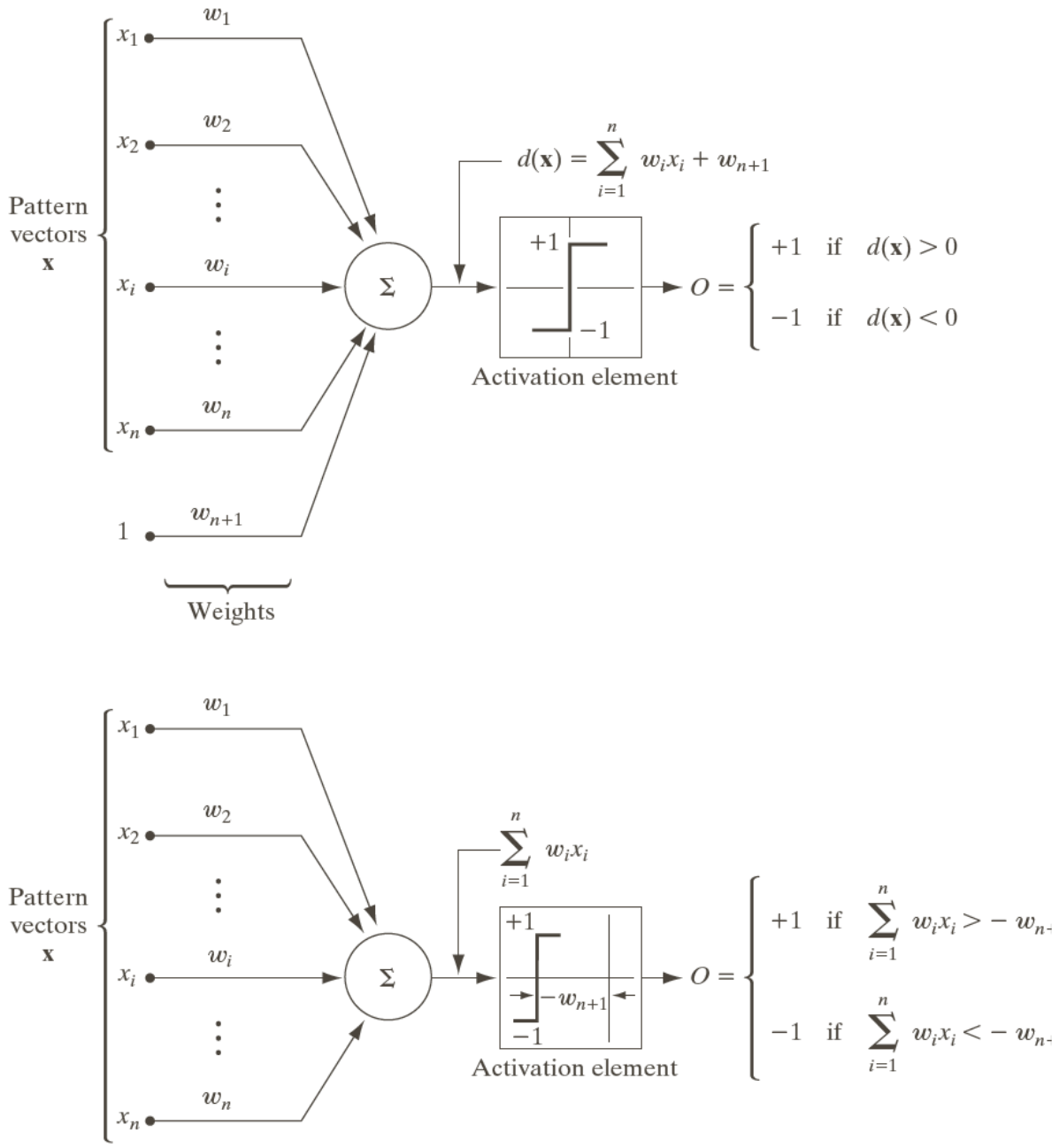
We now treat the threshold like any other weight with a permanent input of -1

Perceptron Learning Algorithm

```
initialise weights (w)
Repeat until all points are correctly classified
        Repeat for each point
                Calculate margin (yiwXi) for point i)
                If margin > 0, point is correclty classified
                Else change the weights to increase margin such
                that Δw = ηyiXi and wnew = wold + Δw
        end
end
```

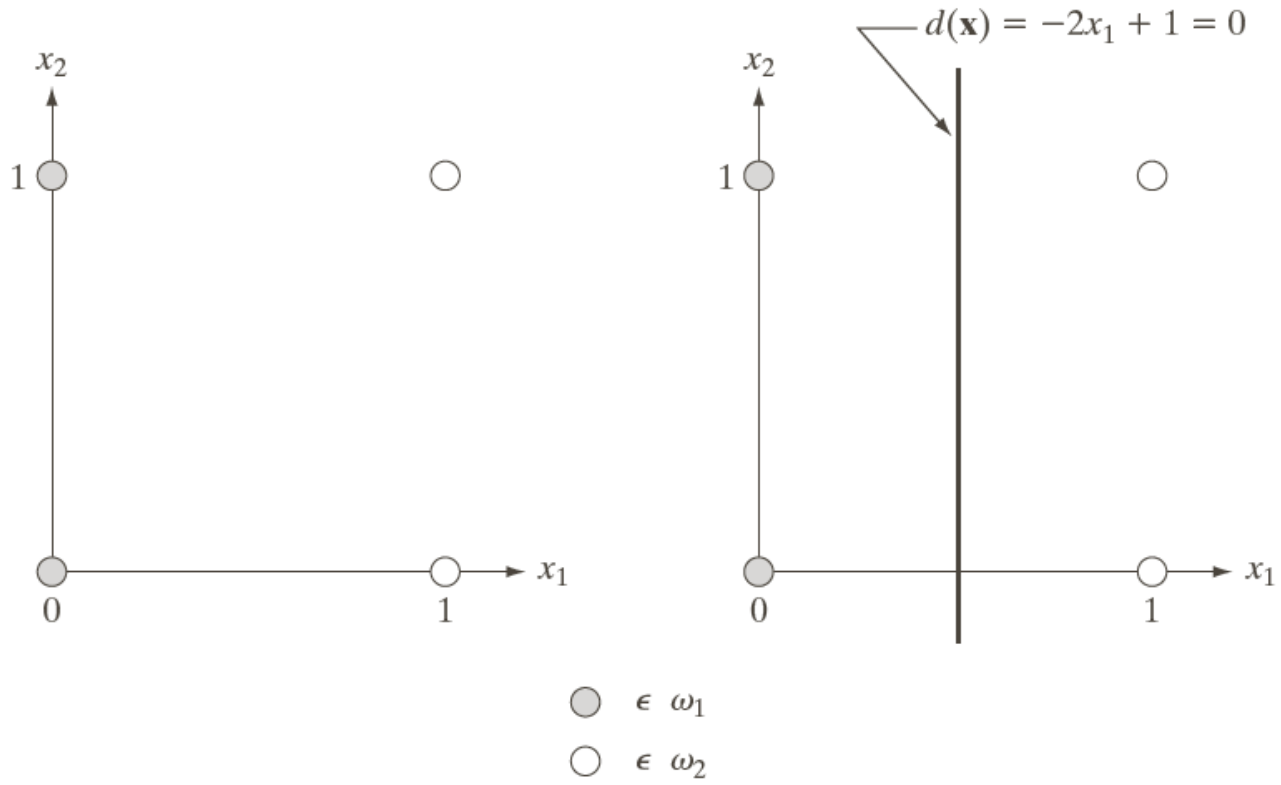
Perceptron convergence theorem:

If the data is linearly separable, then application of the Perceptron learning rule will find a separating decision boundary, within a finite number of iterations



a
b **FIGURE 12.14** Two equivalent representations of the perceptron model for two pattern classes.

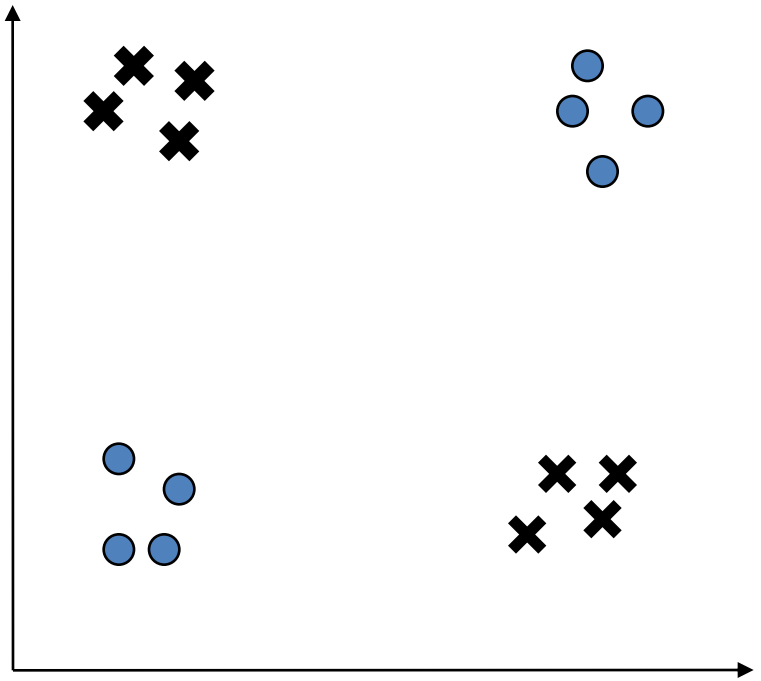
Decision Boundary Using Perceptron



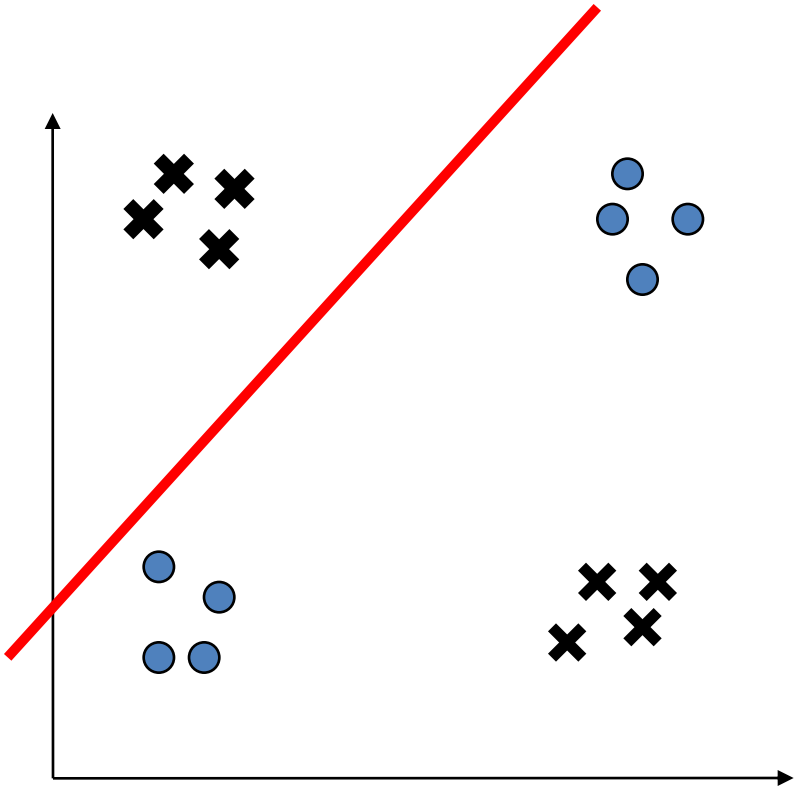
a b

FIGURE 12.15 (a) Patterns belonging to two classes. (b) Decision boundary determined by training.

Can a Perceptron solve this problem?



Can a Perceptron solve this problem? ….. NO.



Perceptrons only solve

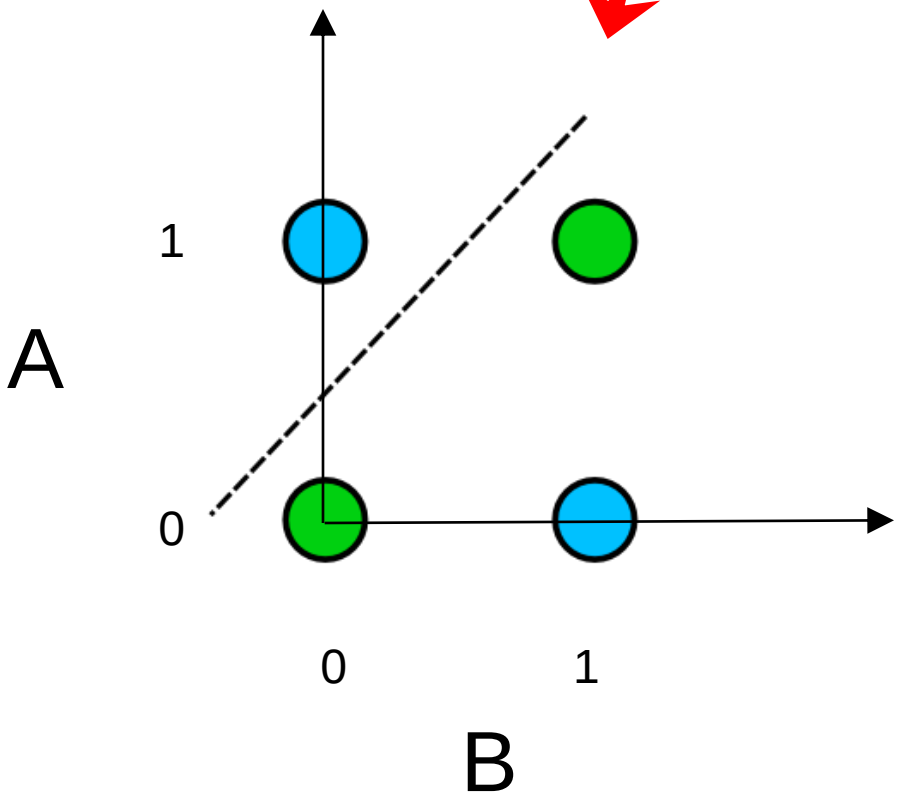
LINEARLY SEPARABLE

problems

With a perceptron…
the decision boundary is
LINEAR

Exclusive-OR gate

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0



Multilayer Neural Network

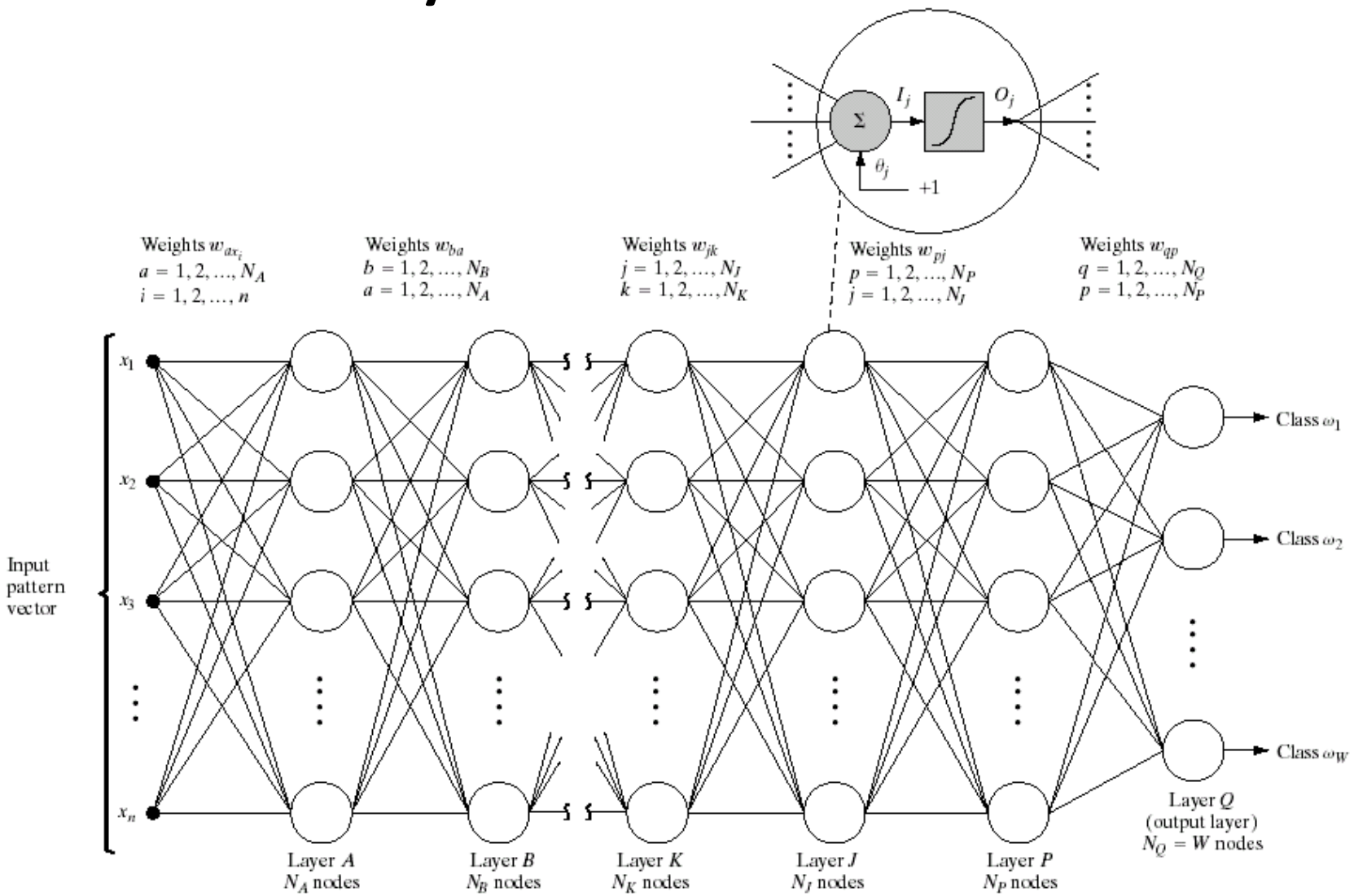


FIGURE 12.16 Multilayer feedforward neural network model. The blowup shows the basic structure of each neuron element throughout the network. The offset, θ_j, is treated as just another weight.

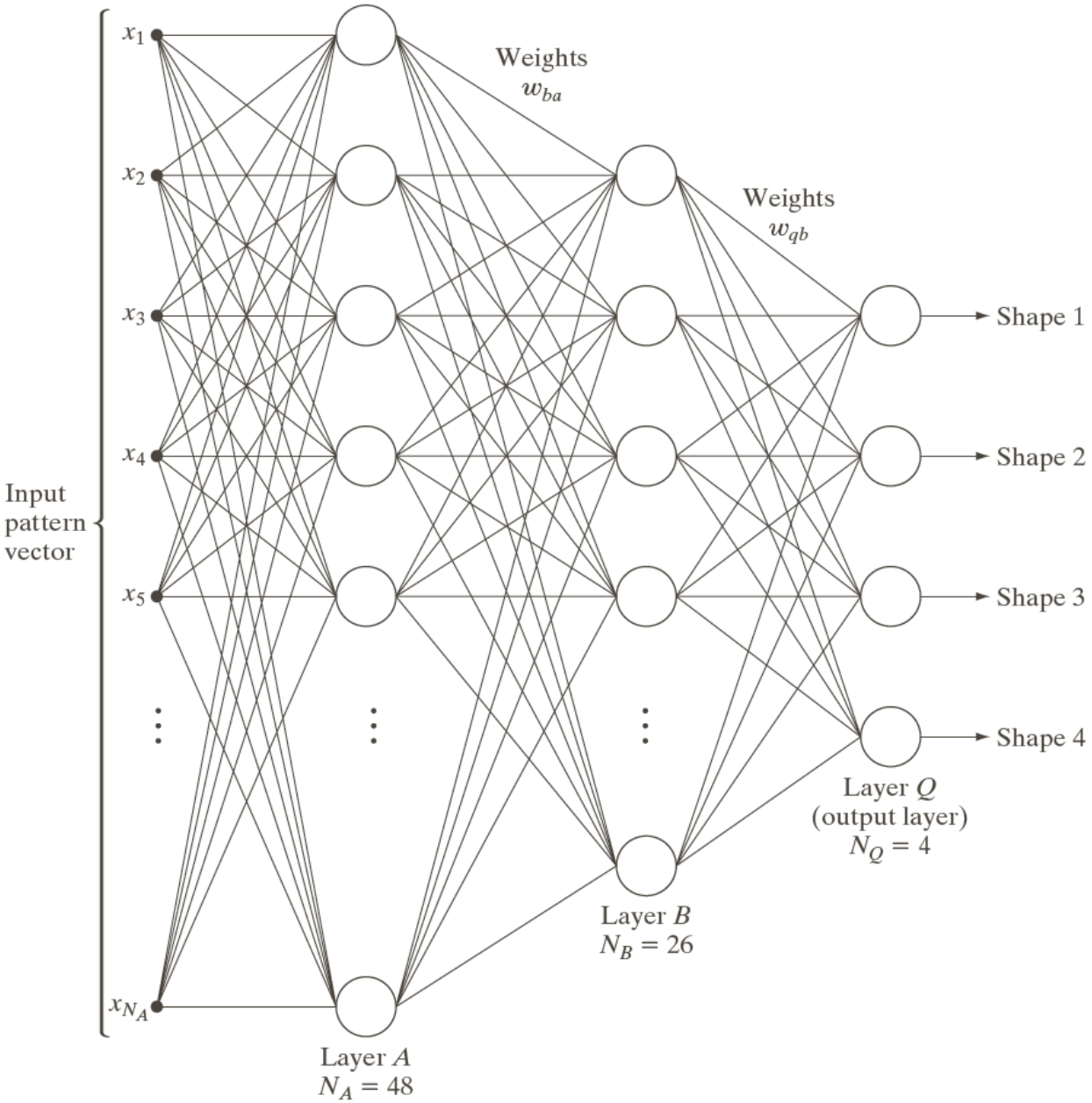
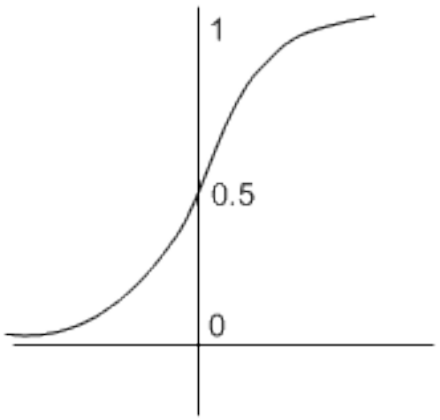


FIGURE 12.19 Three-layer neural network used to recognize the shapes in Fig. 12.18. (Courtesy of Dr. Lalit Gupta, ECE Department, Southern Illinois University.)

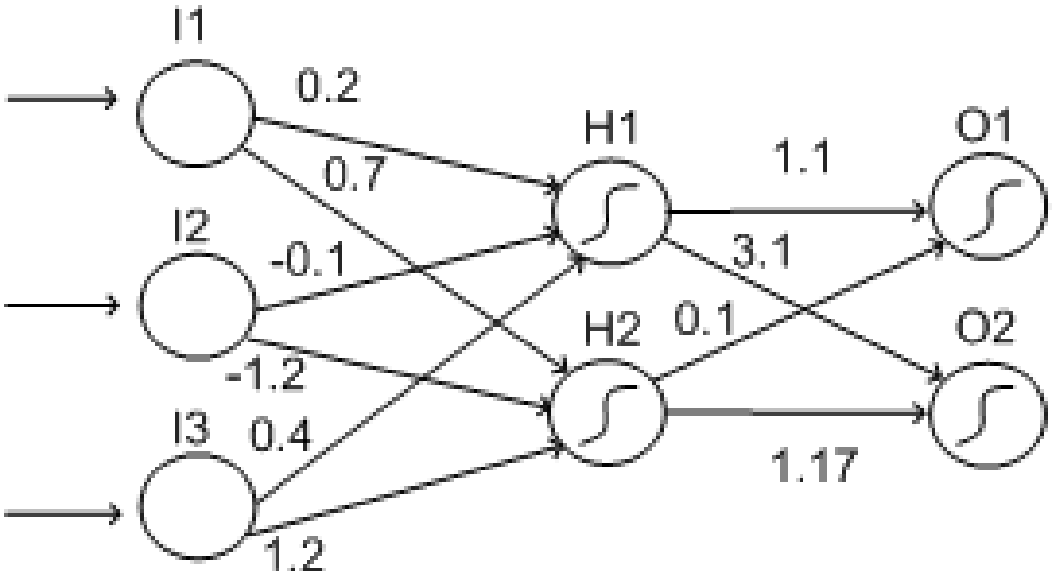
Sigmoid Function

- The function inside units take as input the weighted sum, S, of the values coming from the units connected to it

$$\sigma(S) = \frac{1}{1 + e^{-S}}$$



Example of multilayer Neural Network



- Suppose input values are 10, 30, 20
- The weighted sum coming into H1

$$S_{H1} = (0.2 * 10) + (-0.1 * 30) + (0.4 * 20)$$
$$= 2 -3 + 8 = 7.$$

- The σ function is applied to S_{H1}:

$$\sigma(S_{H1}) = 1/(1+e^{-7}) = 1/(1+0.000912) = 0.999$$

- Similarly, the weighted sum coming into H2:

$$S_{H2} = (0.7 * 10) + (-1.2 * 30) + (1.2 * 20)$$
$$= 7 - 36 + 24 = -5$$

- σ applied to S_{H2}:

$$\sigma(S_{H2}) = 1/(1+e^{5}) = 1/(1+148.4) = 0.0067$$

- Now the weighted sum to output unit O1 :

  $S_{O1} = (1.1 * 0.999) + (0.1*0.0067) = 1.0996$
- The weighted sum to output unit O2:

  $S_{O2} = (3.1 * 0.999) + (1.17*0.0067) = 3.1047$
- The output sigmoid unit in O1:

  $\sigma(S_{O1}) = 1/(1+e^{-1.0996}) = 1/(1+0.333) = 0.750$
- The output from the network for O2:

  $\sigma(S_{O2}) = 1/(1+e^{-3.1047}) = 1/(1+0.045) = 0.957$
- **The input triple (10,30,20) would be categorised with O2, because this has the larger output.**