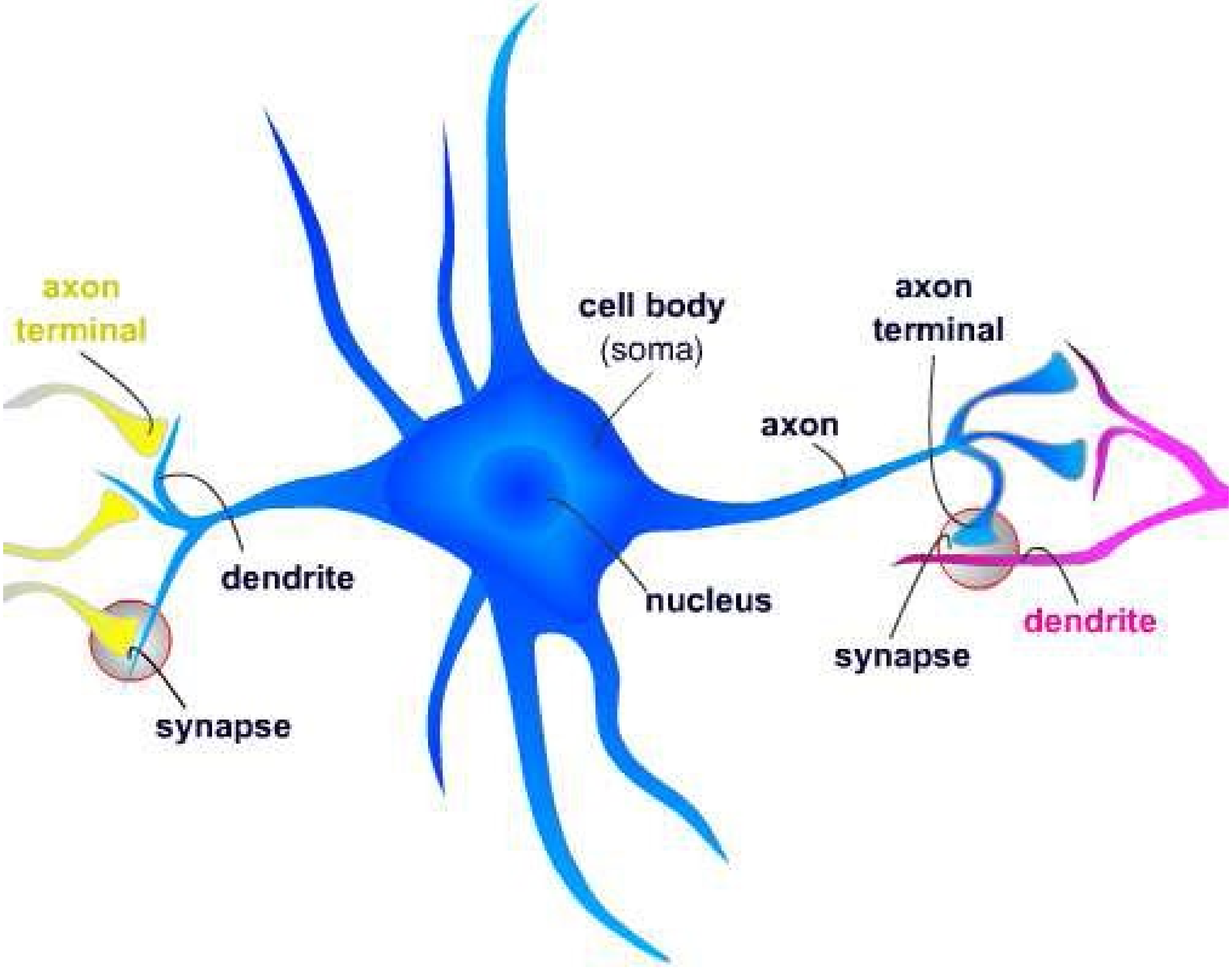
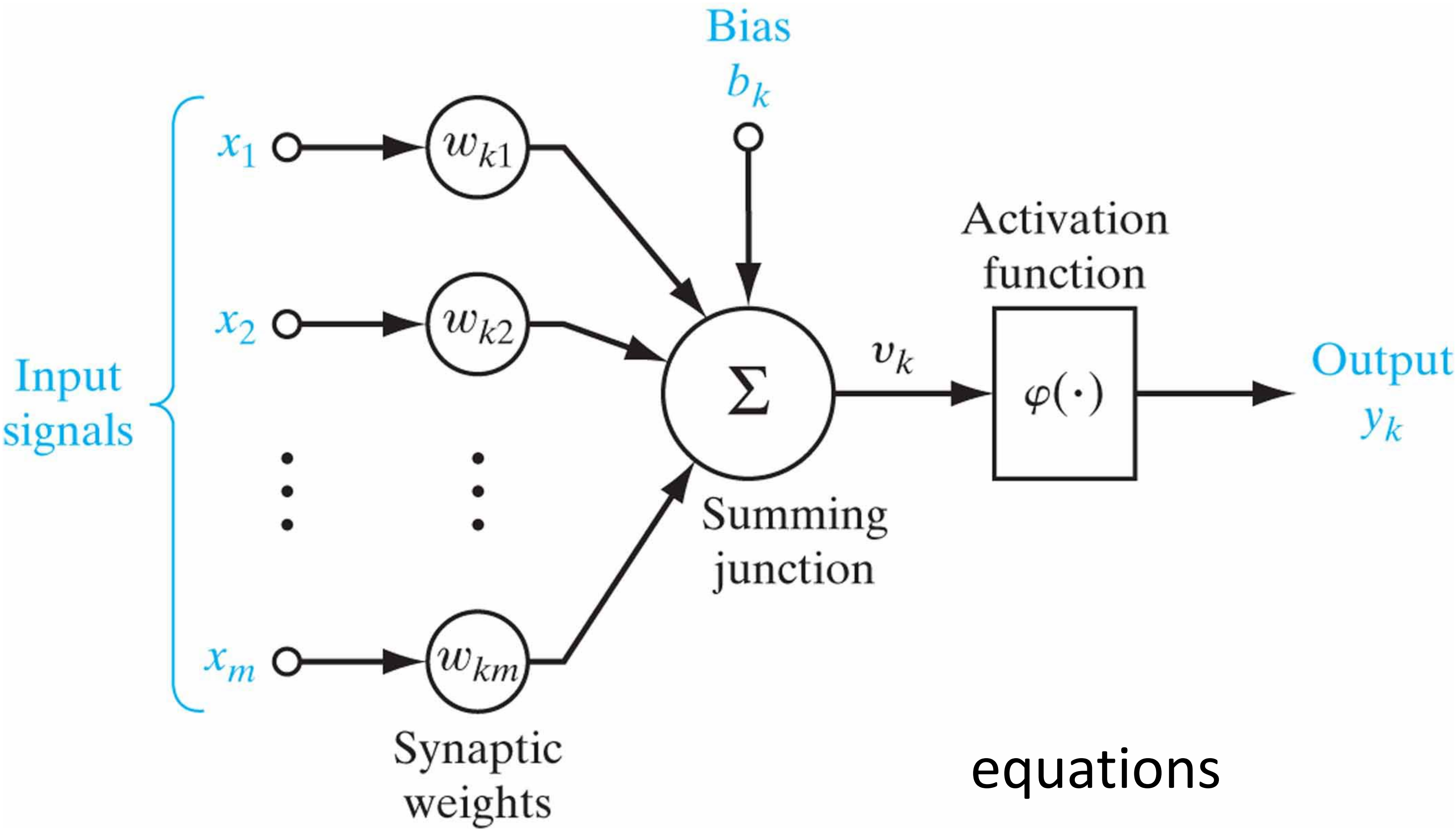
Model of a single neuron:

Perceptron (a nonlinear neuron)

LMS (a linear neuron)

Inspirations from the Brain

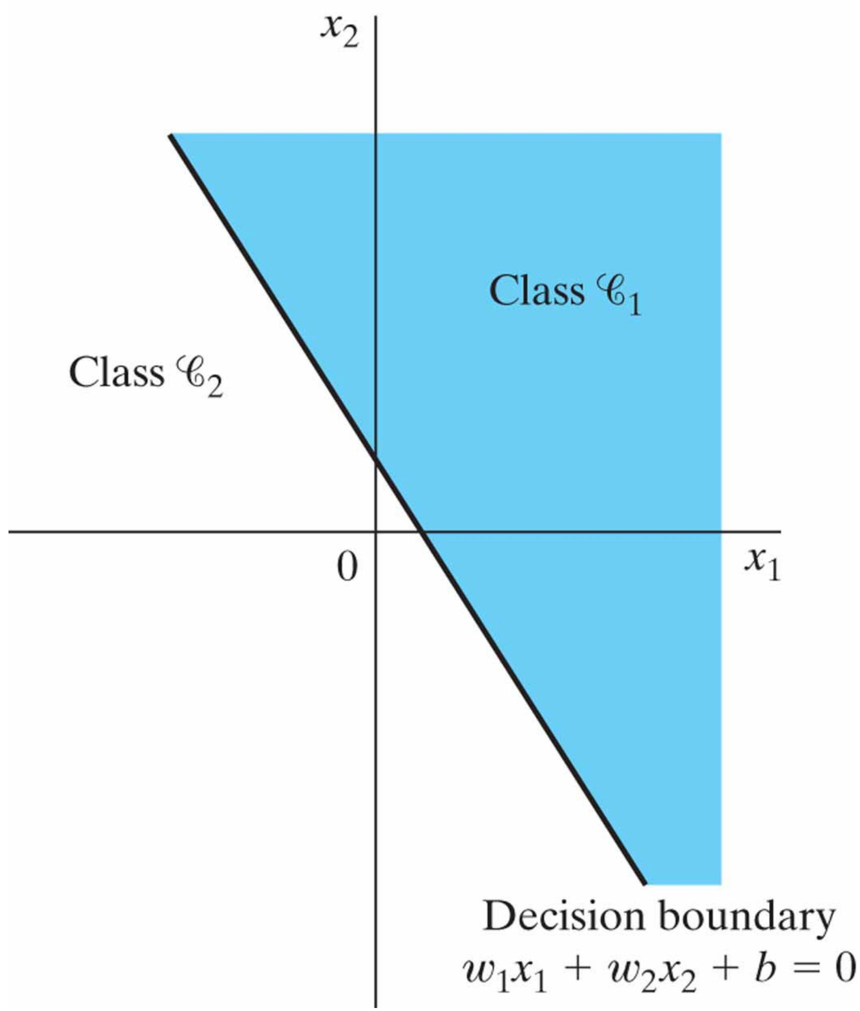




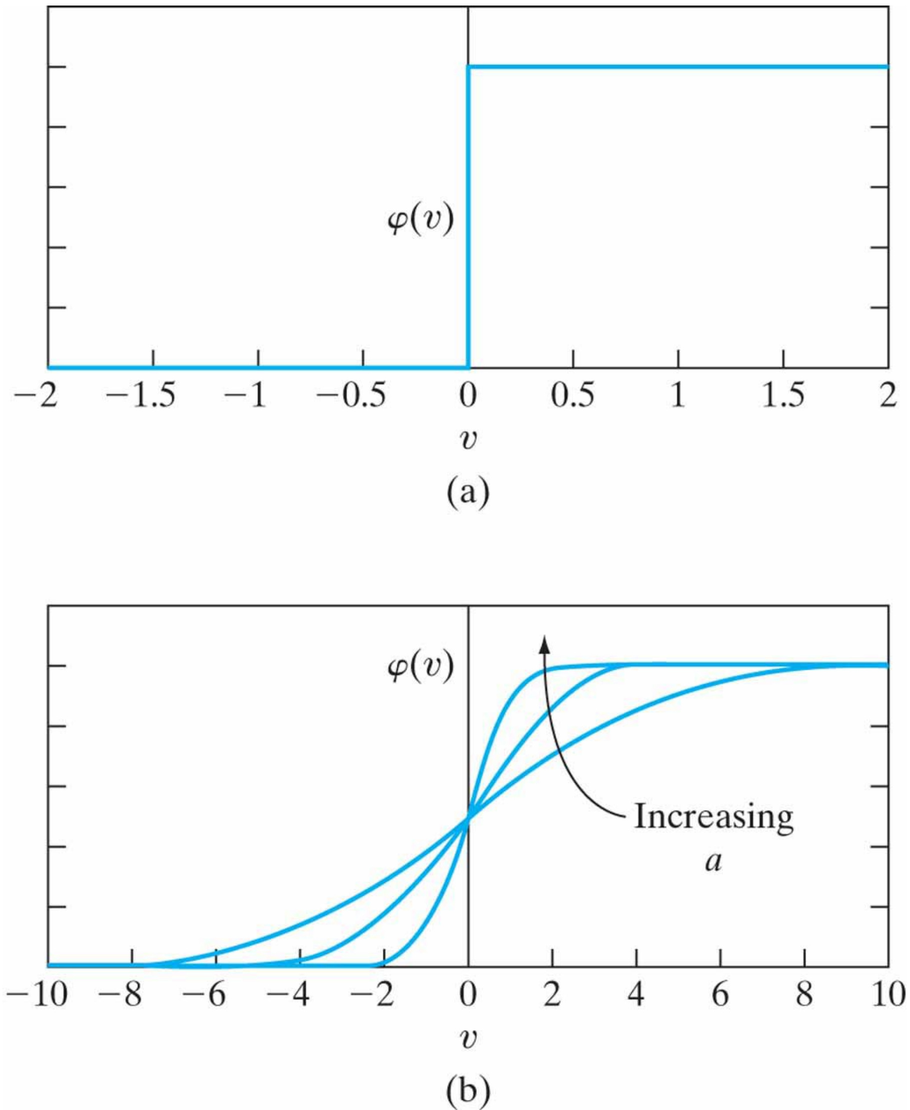
Bias
b_k
x_1
w_{k1}
x_2
w_{k2}
x_m
w_{km}
Input signals
Activation function
v_k
Σ
$\varphi(\cdot)$
Output
y_k
Summing junction
Synaptic weights

equations

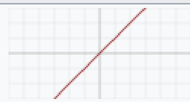
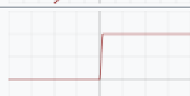
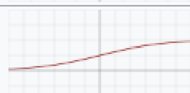
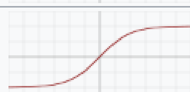
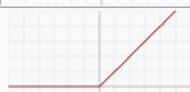
Separating Hyperplane



Nonlinear Model of a Neuron: threshold function



Common types of threshold functions

Name	Plot	Equation	Derivative (with respect to x)	Range
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$ [1]	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$
Rectified linear unit (ReLU)[15]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$

Frank Rosenblatt Mark I Perceptron 1958

TABLE 1.1 Summary of the Perceptron Convergence Algorithm

Variables and Parameters:

$$
\begin{aligned}
\mathbf{x}(n) &= (m+1)\text{-by-1 input vector} \\
&= [+1, x_1(n), x_2(n), \dots, x_m(n)]^T \\
\mathbf{w}(n) &= (m+1)\text{-by-1 weight vector} \\
&= [b, w_1(n), w_2(n), \dots, w_m(n)]^T \\
b &= \text{bias} \\
y(n) &= \text{actual response (quantized)} \\
d(n) &= \text{desired response} \\
\eta &= \text{learning-rate parameter, a positive constant less than unity}
\end{aligned}
$$

1. *Initialization.* Set $\mathbf{w}(0) = \mathbf{0}$. Then perform the following computations for time-step $n = 1, 2, \dots$.
2. *Activation.* At time-step n, activate the perceptron by applying continuous-valued input vector $\mathbf{x}(n)$ and desired response $d(n)$.
3. *Computation of Actual Response.* Compute the actual response of the perceptron as

$$y(n) = \text{sgn}[\mathbf{w}^T(n)\mathbf{x}(n)]$$

where $\text{sgn}(\cdot)$ is the signum function.

4. *Adaptation of Weight Vector.* Update the weight vector of the perceptron to obtain

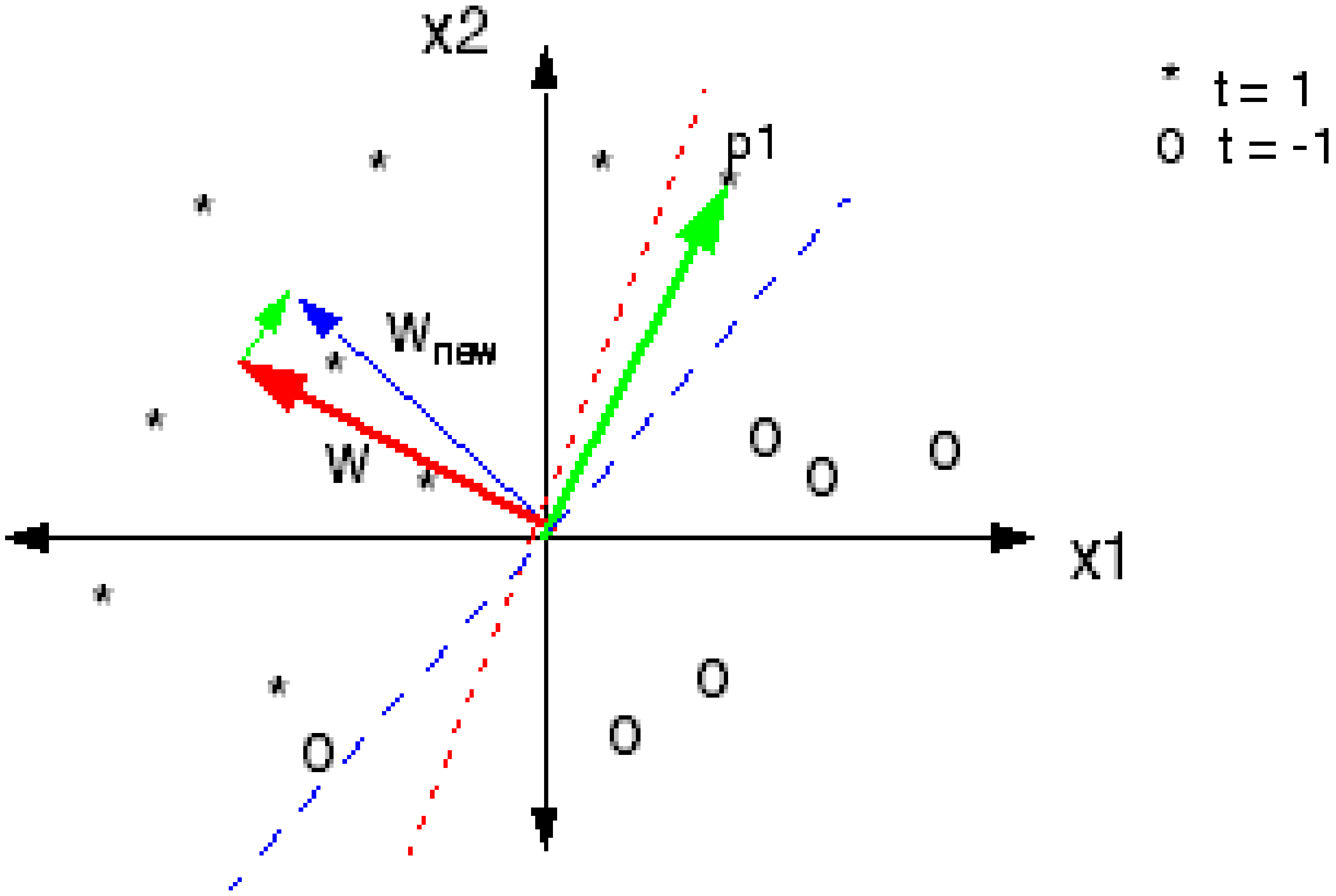
$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n)$$

where

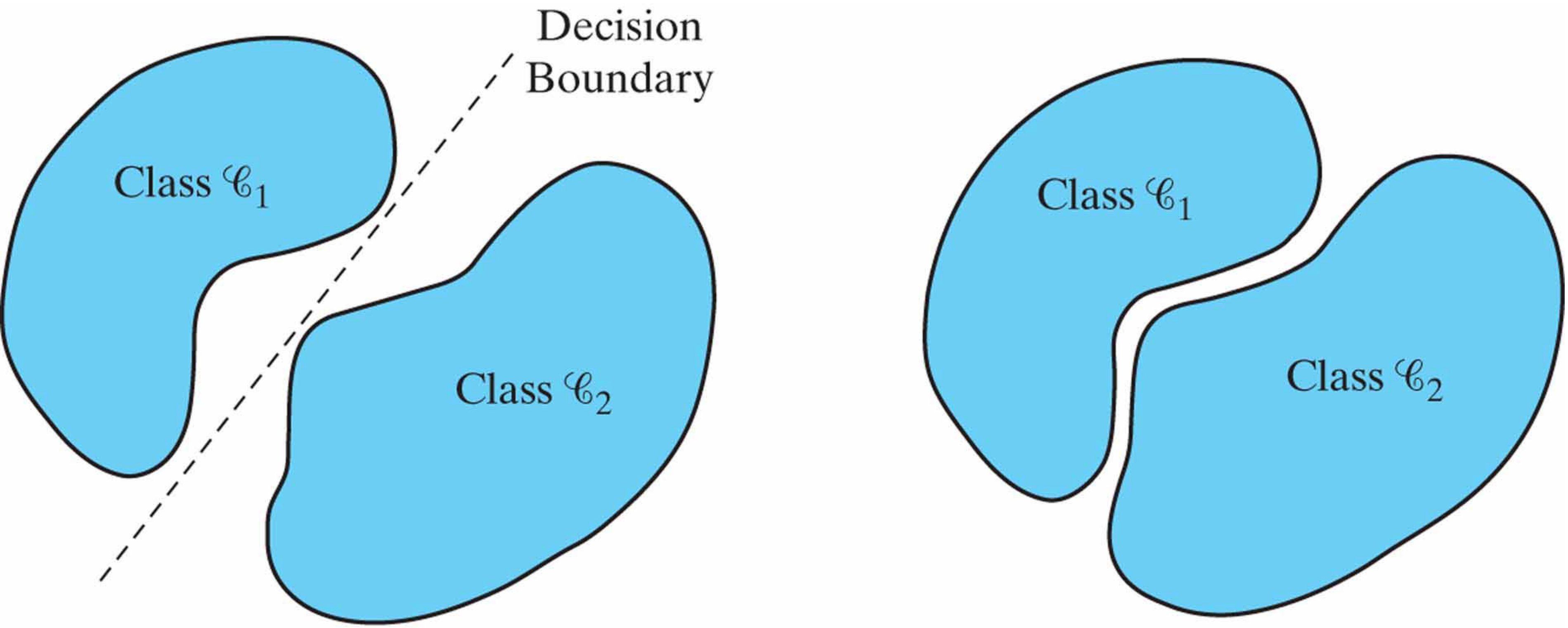
$$d(n) = \begin{cases} +1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1 \\ -1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2 \end{cases}$$

5. *Continuation.* Increment time step n by one and go back to step 2.

Illustration of perceptron updating



(a) A pair of linearly separable patterns.
(b) A pair of non-linearly separable.



(a) (b)

The perceptron convergence theorem

- Outline: Find a lower bound L(k) for $|w|^2$ as a function of iteration k. Then find an upper bound U(k) for $|w|^2$. Then show that the lower bound grows at a faster rate than the upper bound. Since the lower bound can't be larger than the upper bound, there must be a finite k such that the weight is no longer updated. However, this can only happen if all patterns are correctly classified.
- The choice of learning rate η does not matter because it just changes the scaling of w.
- Regardless of w(0), the perceptron is assured of convergence
- $0 < \eta <= 1$, small η is called for to provide stable weight estimates, but large η is associated with fast adaptation and broad exploration of input distribution
- Linearly separable cases, how to address nonlinear problems?

The perceptron model

- Even though a simple model, truly ground-breaking (by Rosenblatt)
  - In a 1958 , the New York Times reported the perceptron to be "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence"
- Only limited capability (single neuron model, solve linearly separable problems only)
- Inspired important and successful models such as Widrow-Hoff's LMS, Vapnik's SVM