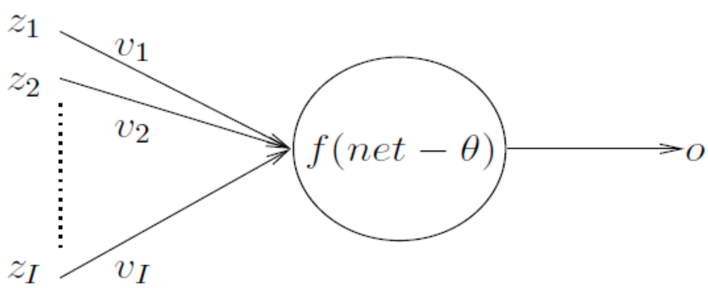
ARTIFICIAL NEURAL NETWORKS

The Artificial Neuron

- An AN receives a vector of I input signals,either from the environment or from other ANs.

$$\mathbf{z} = (z_1, z_2, \cdots, z_I)$$

- To each input signal, z_i, is associated a *weight*, v_i, to strengthen or deplete the input signal.
- The AN computes the *net* input signal, and uses an activation function f_{AN} to compute the output signal, o, given the *net* input.
- The strength of the output signal is further influenced by a threshold value, θ, also referred to as the *bias*.



Calculating the Net Input Signal

The *net* input signal to an AN is usually computed as the weighted sum of all input signals,

$$net = \sum_{i=1}^{I} z_i v_i$$

or by using the product,

$$net = \prod_{i=1}^{I} z_i^{v_i}$$

Activation Functions

The function f_{AN} receives the *net* input signal and *bias*, and determines the output of the neuron. This function is referred to as the *activation function*.

- There are several types of activation functions:

1- Linear Function:

$$f_{AN}(net - \theta) = \lambda(net - \theta)$$

whereλ is a constant and it represents the slope of the function.

2- Step Function:

$$f_{AN}(net - \theta) = \begin{cases} \gamma_1 & \text{if } net \geq \theta \\ \gamma_2 & \text{if } net < \theta \end{cases}$$

The step function produces one of two scalar output values, depending on the value of the threshold θ.

- Usually, a *binary* output is produced for which $\gamma_1 = 1$ and $\gamma_2 = 0$; a *bipolar* output is also sometimes used where $\gamma_1 = 1$ and $\gamma_2 = -1$.

3- Ramp Function:

$$f_{AN}(net - \theta) = \begin{cases} \gamma & \text{if } net - \theta \geq \epsilon \\ net - \theta & \text{if } -\epsilon < net - \theta < \epsilon \\ -\gamma & \text{if } net - \theta \leq -\epsilon \end{cases}$$

The ramp function is a combination of the linear and step functions.

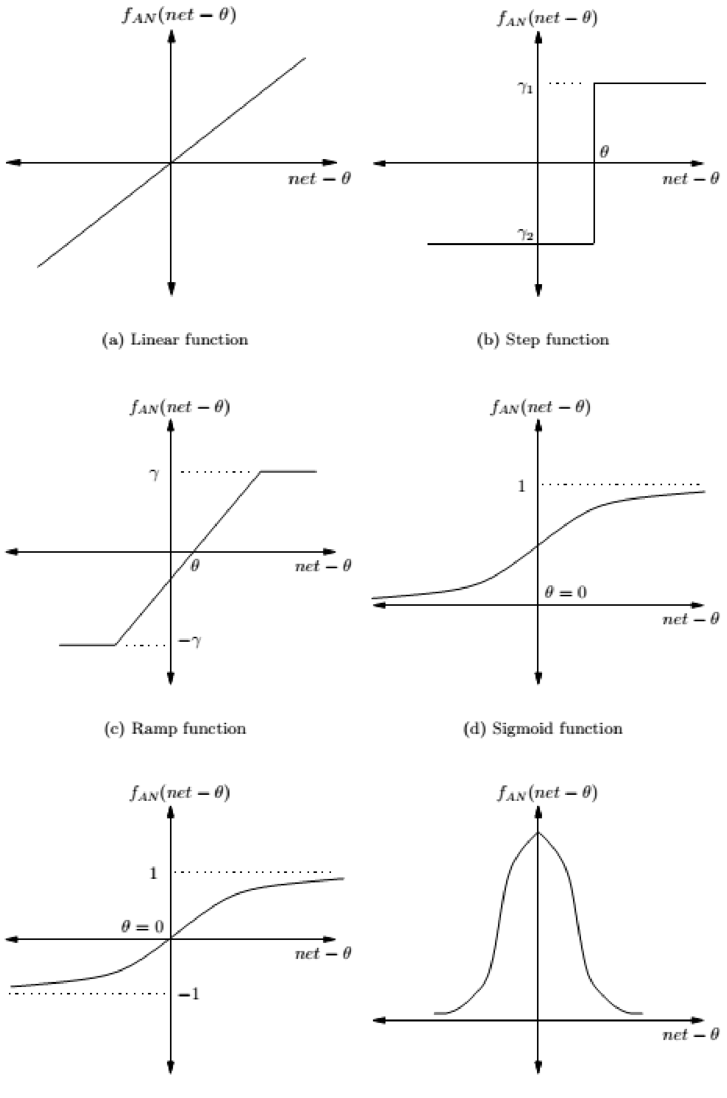
4- Sigmoid Function:

$$f_{AN}(net - \theta) = \frac{1}{1 + e^{-\lambda(net - \theta)}}$$

The sigmoid function is a continuous version of the ramp function, with $f_{AN}(net) \in (0, 1)$. The parameter λ controls the steepness of the function. Usually, $\lambda = 1$.

5- Hyperbolic tangent:

$$f_{AN}(net - \theta) = \frac{2}{1 + e^{-\lambda(net - \theta)}} - 1$$



(a) Linear function (b) Step function

(c) Ramp function (d) Sigmoid function

(e) Hyperbolic tangent function (f) Gaussian function

Artificial Neuron Geometry

- Single neurons can be used to realize linearly separable functions without any error.
- Linear reparability means that the neuron can separate the space of *I*-dimensional input vectors yielding an above-threshold response from those having a below-threshold response by an *I*-dimensional *hyperplane*.
- The hyperplane separates the input vectors for which have:

$$\sum_i z_i v_i - \theta > 0$$

from the input vectors for which have:

$$\sum_i z_i v_i - \theta < 0$$

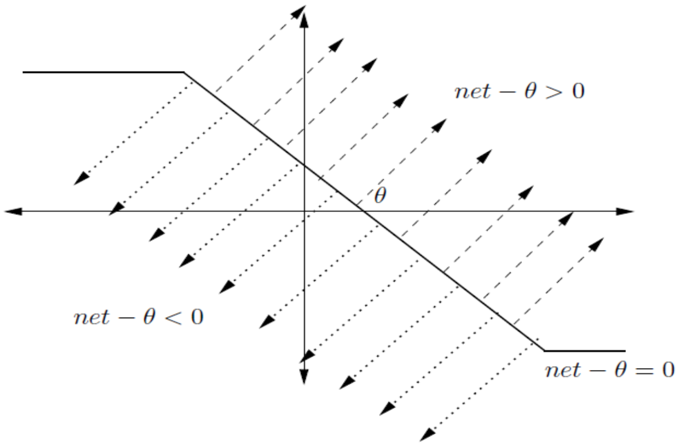
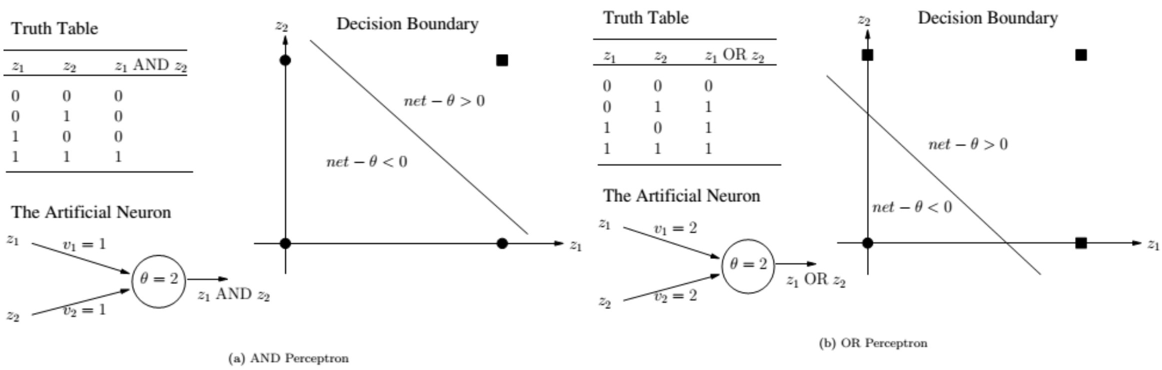


Figure 2.3 Artificial Neuron Boundary

Below we shows how two Boolean functions, AND and OR, can be implemented using a single *perceptron*. These are examples of linearly separable functions. For such simple functions, it is easy to manually determine values for the bias and the weights.



- The XOR Boolean function is <u>not a separable function</u>. Thus a single *perceptron*cannot implement this function.
- So, a layered NN of several neurons is required.
- For example, the XOR function requires two input units, two hidden units and one output unit.

Augmented Vectors

- To simplify learning equations, the input vector is augmented to include an additional input unit, z_{I+1}, referred to as the *bias* unit.
- The value of z_{I+1} is always -1, and the weight v_{I+1} serves as the value of the threshold. The net input signal to the AN is then calculated as:

$$\begin{aligned} net &= \sum_{i=1}^{I} z_i v_i - \theta \\ &= \sum_{i=1}^{I} z_i v_i + z_{I+1} v_{I+1} \\ &= \sum_{i=1}^{I+1} z_i v_i \end{aligned}$$

Where,

$$\theta = z_{I+1} v_{I+1} = -v_{I+1}.$$

In the case of the step function, an input vector yields an output of 1 when $\sum_{i=1}^{I+1} z_i v_i \geq 0$, and 0 when $\sum_{i=1}^{I+1} z_i v_i < 0$

Artificial Neuron Learning

- How can the v_i and θ values be computed? The answer is through *learning*.
- The AN learns the best values for the v_i and θ from the given data.
- Learning consists of adjusting weight and threshold values until a certain criterion (or several criteria) is (are) satisfied.
- There are three main types of learning:
  - ❖ **Supervised learning**, where the neuron (or NN) is provided with a data set consisting of input vectors and a target (desired output) associated with each input vector. This data set is referred to as the *training set*. The aim of supervised training is then to adjust the weight values such that the error between the real output, $o = f(net - \theta)$, of the neuron and the target output, t, is minimized.

x1	x2	x3	target
1	2	3	3
4	1	4	2
3	0	-1	-1
2	2	4	2
8	2	1	1

Patterns

Inputs

Supervised learning can be summarized as follows:

- Initially, set all the weights to some random values
- Repeat (for many epochs):
  - a) Feed the network with an input from one of the examples in the training set
  - b) Compute the error between the output of the network and the desired output
  - c) Correct the error by adjusting the weights of the nodes
- Until the error is sufficiently small

- **Unsupervised learning**, where the aim is to discover patterns or features in the input data with no assistance from an external source. Many unsupervised learning algorithms basically perform a clustering of the training patterns.
- **Reinforcement learning**, where the aim is to reward the neuron (or parts of a NN) for good performance, and to penalize the neuron for bad performance.

Learning Rules

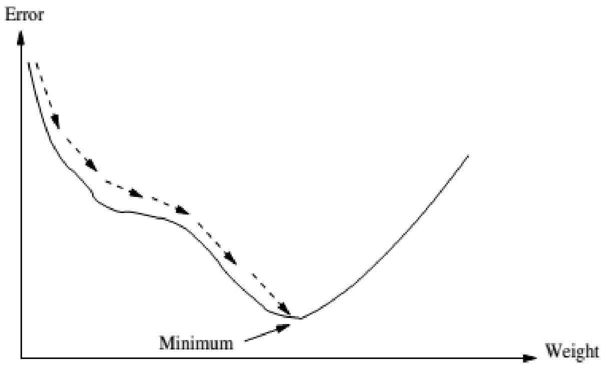
1. Gradient Descent GD Learning Rule

- It is usually used to learn the neurons.
- GD requires the definition of an error (or objective) function to measure the neuron's error in approximating the target.
- This rule use the sum of squared errors:

$$\mathcal{E} = \sum_{p=1}^{P_T} (t_p - o_p)^2$$

where t_p and o_p are respectively the target and actual output for the *p-th* pattern, and PT is the total number of input-target vector pairs (patterns) in the training set.

- The aim of GD is to find the weight values that minimize ε.



- Given a single training pattern, weights are updated using:

$$v_i(t) = v_i(t-1) + \Delta v_i(t)$$

with

$$\Delta v_i(t) = \eta(-\frac{\partial \mathcal{E}}{\partial v_i})$$

where

$$\frac{\partial \mathcal{E}}{\partial v_i} = -2(t_p - o_p)\frac{\partial f}{\partial net_p} z_{i,p}$$

and η is the learning rate.

- The calculation of the partial derivative of f with respect to net_p (the *net* input for pattern p) presents a problem for all discontinuous activation functions, such as the step and ramp functions; $z_{i,p}$ is the i-th input signal corresponding to pattern p.
- The Widrow-Hoff learning rule presents a solution for the step and ramp functions, while the generalized delta learning rule assumes continuous functions that are at least once differentiable.

2. Widrow-Hoff Learning Rule

For the Widrow-Hoff learning rule, assume that $f = net_p$, then $\frac{\partial f}{\partial net_p} = 1$, giving:

$$\frac{\partial \mathcal{E}}{\partial v_i} = -2(t_p - o_p) z_{i,p}$$

Weights are then updated using

$$v_i(t) = v_i(t-1) + 2\eta(t_p - o_p)z_{i,p}$$

❖ This rule, also referred to as the ***least-means-square*** (LMS) algorithm, was one of the first algorithms used to train layered neural networks with multiple adaptive linear neurons.

3. Generalized Delta Learning Rule

- The generalized delta learning rule is a generalization of the Widrow-Hoff learning rule that assumes differentiable activation functions.
- Assume that the sigmoid function is used. Then,

$$\frac{\partial f}{\partial net_p} = o_p(1 - o_p)$$

giving

$$\frac{\partial \mathcal{E}}{\partial v_i} = -2(t_p - o_p)o_p(1 - o_p)z_{i,p}$$

4. Error-Correction Learning Rule

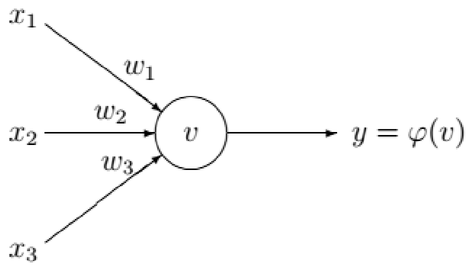
- Weights are only adjusted when the neuron responds in error. That is, only when $(t_p - o_p) = 1$ or $(t_p - o_p) = -1$, are weights adjusted using equation:

$$v_i(t) = v_i(t-1) + 2\eta(t_p - o_p)z_{i,p}$$

Assignments

1- Explain why the threshold θ is necessary. What is the effect of θ, and what will the consequences be of not having a threshold?

2- Suppose the following single neuron:



Suppose that the weights corresponding to the three inputs have the following values:

$$\boxed{\begin{array}{lcr} w_1 & = & 2 \\ w_2 & = & -4 \\ w_3 & = & 1 \end{array}}$$

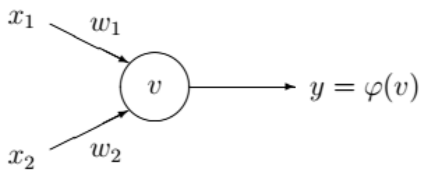
and the activation of the unit is given by the step-function:

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Calculate what will be the output value y of the unit for each of the following input patterns:

Pattern	P_1	P_2	P_3	P_4
x_1	1	0	1	1
x_2	0	1	0	1
x_3	0	1	1	1

3- Suppose you have the following single neuron:



And you have the AND function table:

x_1 :	0	1	0	1
x_2 :	0	0	1	1
x_1 AND x_2 :	0	0	0	1

if the weights are w1= 1 and w2= 1 and the activation function is:

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 2 \\ 0 & \text{otherwise} \end{cases}$$

Then, test how the neural AND function works

4- Suppose you have four patterns:

z1	z2	z3	t
1	2	4	2
2	3	3	2
3	4	2	3
2	2	1	1

and the weights (2,1,0,-1), $\eta = 1, \theta = -1, \lambda = 1$, use the sigmoid function , andgeneralized delta learning rule, then compute:

1- The output for each pattern.
2- The total error.
3- The new weight values.