Neural Network

1 Introduction

1.1 History

- Early Period: Hermann von Helmholtz, Ernst Mach, Ivan Pavlov
- Modern Period: Warren McCulloch, Walter Pitts, Donald Hebb, Frank Rosenblatt, Bernard Widrow and Ted Hoff, Teuvo Kohonen, James Anderson, Stephen Grossberg, John Hopfield, David Rumelhart, James McClelland

1.2 Applications

- Aerospace, Automotive, Banking, Defense, Electronics, Entertainment, Financial, Insurance, Manufacturing, Medical, Oil and Gas, Robotics, Speech, Securities, Telecommunications, Transportation
- Groups of Applications
  - Pattern Recognition; Image, Optical Character Recognition (OCR), Speech, Sensors Pattern, etc.
  - Pattern Recall
  - Classification; Unsupervised Learning
  - Function Estimation

1.3 Biological Inspiration

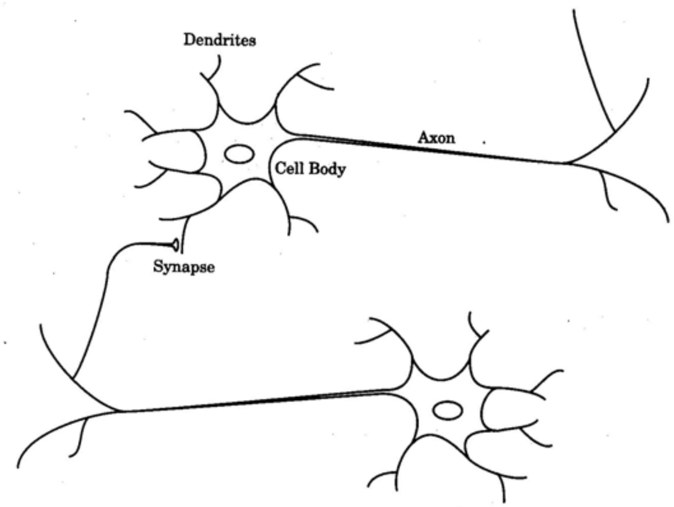

Figure 1.3-1 Schematic Drawing of Biological Neurons

- Dendrites, Cell Body, Axon
- Synapse; Constant, Learning
- Human Brain: 10^{11} Neurons, 10^{4} Connections/Neuron
- Simple Function -> Parallel Computation

2 Neuron Model and Network Architectures

2.1 Neuron Model

2.1.1 Single-Input Neuron

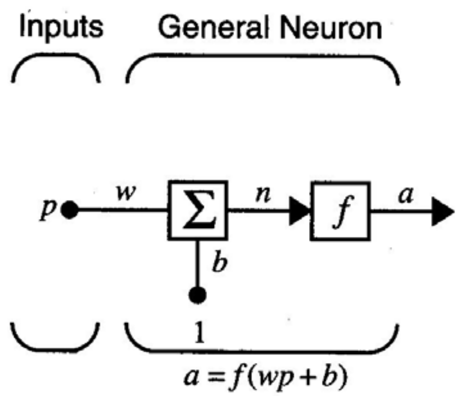

Figure 2.1.1-1 Single-Input Neuron

$$a = f(wp+b) \qquad (2.1.1\text{-}1)$$

p	:	scalar input	
w	:	weight	(corresponding to the strength of a synapse)
b	:	bias, offset	
Σ	:	summer	(corresponding to a part of the cell body)
n	:	summer output, net input	
f	:	transfer function, activation function	(corresponding to a part of the cell body)
a	:	scalar neuron output	(corresponding to signal on the axon)

2.1.2 Transfer Functions

Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins
Log-Sigmoid	$a = \frac{1}{1+e^{-n}}$		logsig
Hyperbolic Tangent Sigmoid	$a = \frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$		tansig
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n$		poslin
Competitive	$a = 1$ neuron with max n $a = 0$ all other neurons	C	compet

Table 2.1.2-1 Transfer Functions

2.1.3 Multiple-Input Neuron

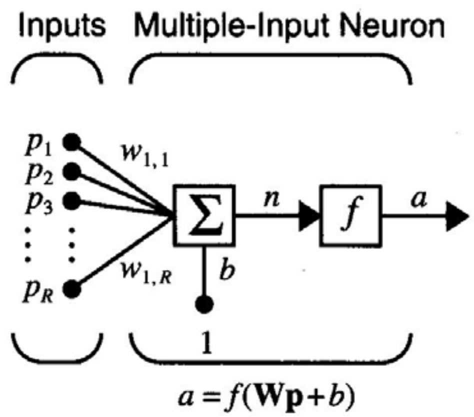

Figure 2.1.3-1 Multiple-Input Neuron

- Number of inputs (R) depends on amount of information.
- $w_{i,j}$: Weight linking between output (neuron) no i with input no j

$$n = w_{1,1}p_1+w_{1,2}p_2+\ldots+w_{1,R}p_R+b \quad (2.1.3\text{-}1)$$

$$a = f(n) \quad (2.1.3\text{-}2)$$

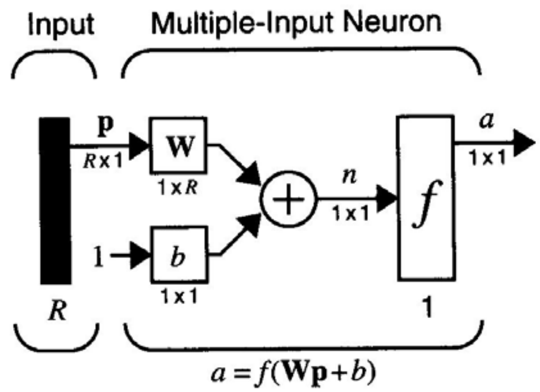

Figure 2.1.3-2 Neuron with *R* Inputs, Abbreviated Notation

2.2 Network Architectures

2.2.1 A Layer of Neurons

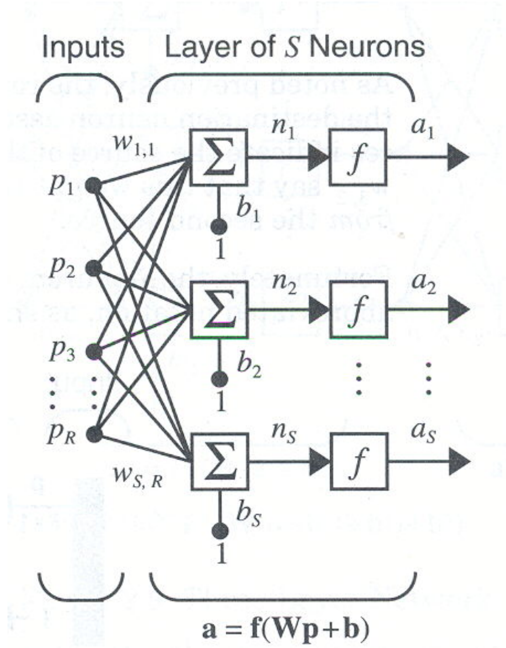

Figure 2.2.1-1 Layer of *S* Neurons

- Number of outputs (S) depends on amount of desired outputs.

$$n_i = w_{i,1}p_1 + w_{i,2}p_2 + \ldots + w_{i,R}p_R + b_i \quad (2.2.1\text{-}1)$$

$$\mathbf{n} = \mathbf{Wp} + \mathbf{b} \quad (2.2.1\text{-}2)$$

$$\mathbf{a} = f(\mathbf{Wp} + \mathbf{b}) \quad (2.2.1\text{-}3)$$

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{bmatrix} \quad (2.1.3\text{-}4)$$

$$\mathbf{n} = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_S \end{bmatrix}, \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_S \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_S \end{bmatrix} \quad (2.1.3\text{-}5)$$

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \quad (2.2.1\text{-}6)$$

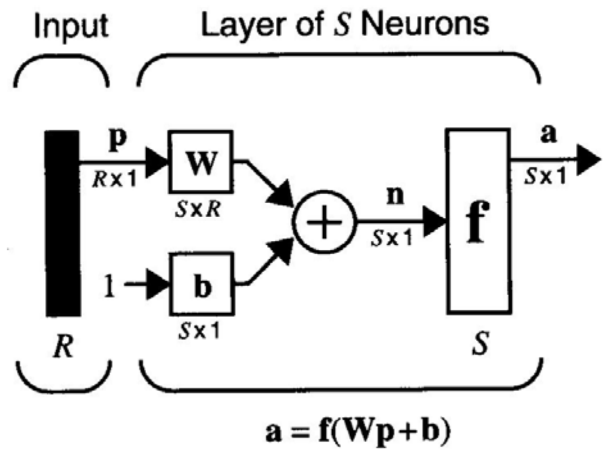

Figure 2.2.1-2 Layer of *S* Neurons, Abbreviated Notation

2.2.2 Multiple Layers of Neurons

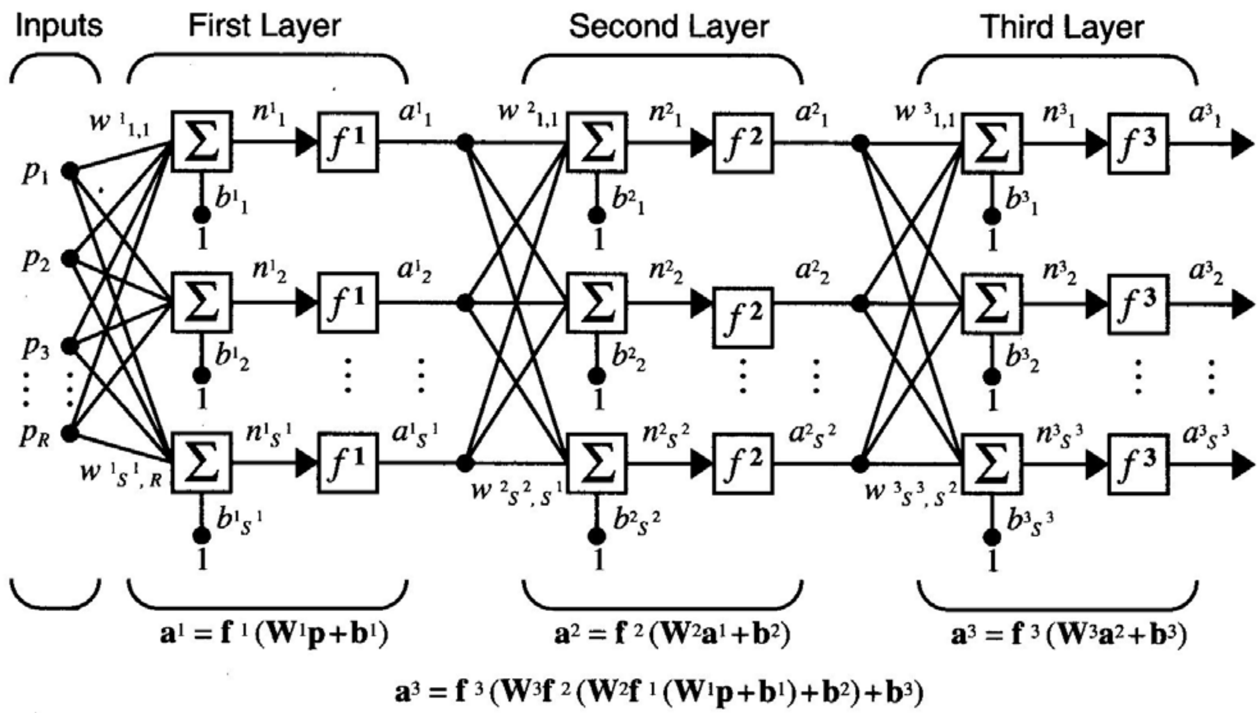

Figure 2.2.2-1 Three-Layer Network

- Number of neurons in the hidden layer (S^m) depends on complexity of the problem.

$$\mathbf{a}^m = f^m(\mathbf{W}^m\mathbf{a}^{m-1}+\mathbf{b}^m) \quad (2.2.2\text{-}1)$$

$$\mathbf{a}^3 = f^3(\mathbf{W}^3 f^2(\mathbf{W}^2 f^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3) \quad (2.2.2\text{-}2)$$

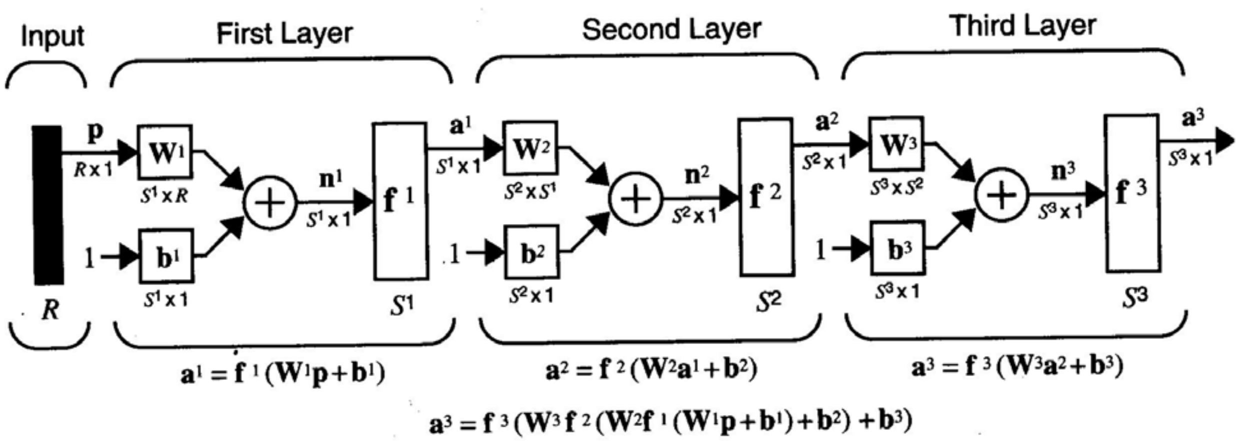

Figure 2.2.2-2 Three-Layer Network, Abbreviated Notation

2.2.3 Recurrent Networks

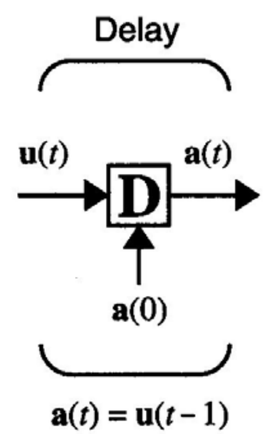

Figure 2.2.3-1 Delay Block

$$\mathbf{a}(t) = \mathbf{u}(t-1) \quad (2.2.3\text{-}1)$$

$$\mathbf{a}(t) = \mathbf{a}(0);\ t = 0 \quad (2.2.3\text{-}2)$$

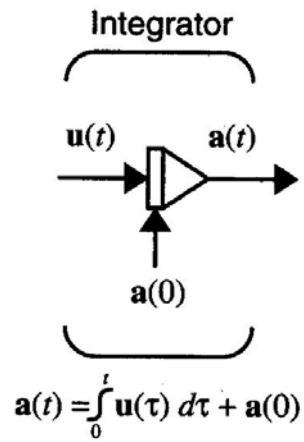

Figure 2.2.3-2 Integrator Block

$$\mathbf{a}(t) = \int_0^t \mathbf{u}(\tau)d\tau + \mathbf{a}(0) \tag{2.2.3-3}$$

$$\mathbf{a}(t) = \mathbf{a}(0);\ t = 0 \tag{2.2.3-4}$$

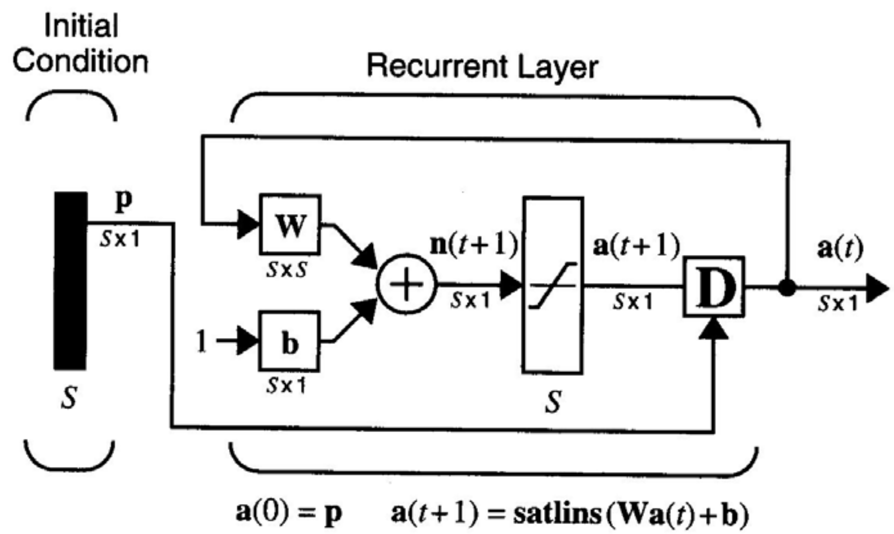

Figure 2.2.3-3 Recurrent Network

$$\mathbf{a}(t+1) = \text{satlins}(\mathbf{W}\mathbf{a}(t)+\mathbf{b}) \tag{2.2.3-5}$$

$$\mathbf{a}(0) = \mathbf{p} \tag{2.2.3-6}$$

3 An Illustrative Example

Fruits : Apple, Orange

Sensor Inputs : Shape (1-Round, -1-Elleptical)

Texture (1-Smooth, -1-Rough)

Weight (1-More Than One Pound, -1-Less Than One Pound)

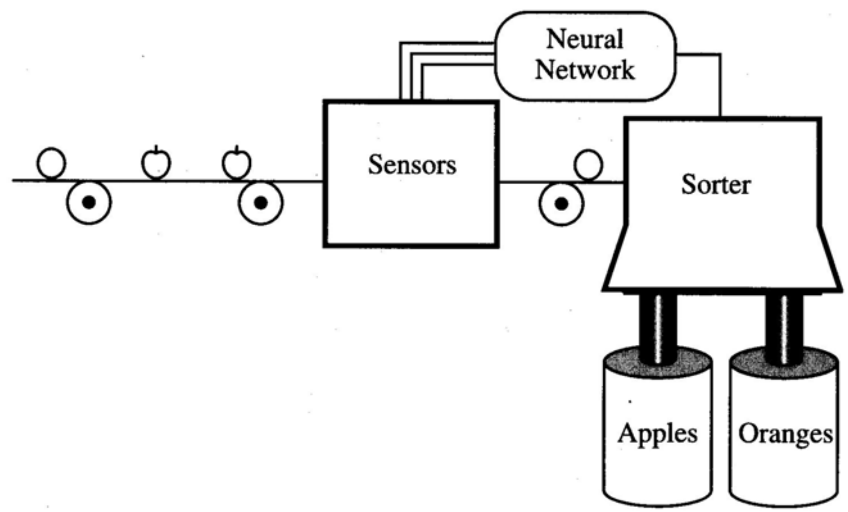

Figure 3-1 Neural Network Controlled Fruits Sorter

$$\mathbf{p} = \begin{bmatrix} shape \\ texture \\ weight \end{bmatrix} \tag{3-1}$$

A prototype orange (round, rough, less than 1 pound)

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \tag{3-2}$$

A prototype apple (round, smooth, less than 1 pound)

$$\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \tag{3-3}$$

3.1 Perceptron

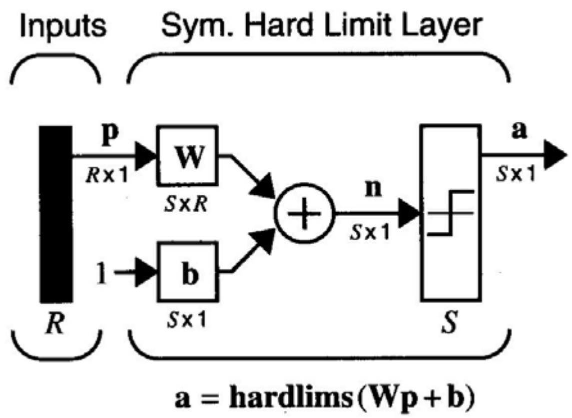

Figure 3.1-1 Single-Layer Perceptron

$$a = hardlims\left(\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix}\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + b\right) \quad (3.1\text{-}1)$$

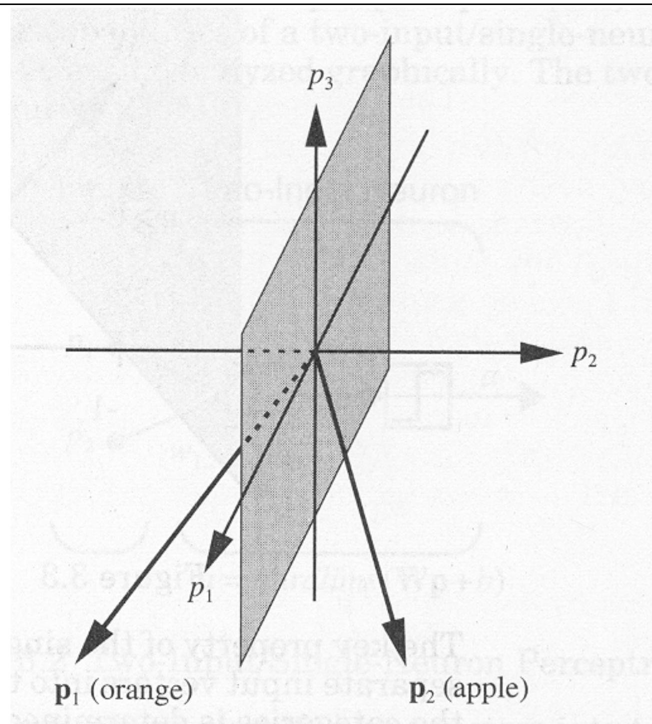

Figure 3.1-2 Prototype Vectors

$$p_2 = 0 \tag{3.1-2}$$

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + 0 = 0 \tag{3.1-3}$$

$$\mathbf{W} = [0 \quad 1 \quad 0],\ b = 0 \tag{3.1-4}$$

For orange (round, rough, less than 1 pound),

$$a = hardlims\left([0 \quad 1 \quad 0]\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0 \right) = -1 \text{ (orange)} \tag{3.1-5}$$

For apple (round, smooth, less than 1 pound),

$$a = hardlims\left([0 \quad 1 \quad 0]\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + 0 \right) = 1 \text{ (apple)} \tag{3.1-6}$$

For unidentified fruit (elliptical, rough, less than 1 pound)

$$\mathbf{p} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \tag{3.1-7}$$

$$a = hardlims\left([0 \quad 1 \quad 0]\begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + 0 \right) = -1 \text{ (orange)} \tag{3.1-8}$$

3.2 Hamming Network

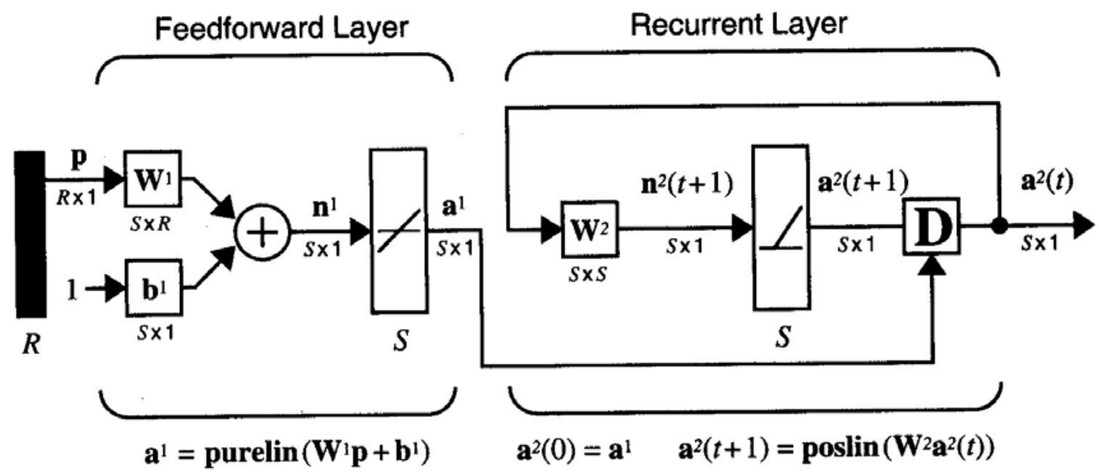

Figure 3.2-1 Hamming Networks

- R : No of Inputs, S : No of Prototypes = No of Neurons in Layer 1 = No of Neurons in Layer 2

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \tag{3.2-1}$$

$$\mathbf{b}^1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \tag{3.2-2}$$

$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{p}+\mathbf{b}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix}\mathbf{p} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T\mathbf{p}+3 \\ \mathbf{p}_2^T\mathbf{p}+3 \end{bmatrix} \tag{3.2-3}$$

$$\mathbf{a}^2(0) = \mathbf{a}^1 \text{ (initial condition)} \tag{3.2-4}$$

$$\mathbf{a}^2(t+1) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(t)) \tag{3.2-5}$$

$$\mathbf{W}^2 = \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix} \tag{3.2-6}$$

Where ε is some number less than 1/(*S*-1), and *S* is the number of neurons in the recurrent layer.

$$\mathbf{a}^2(t+1) = \mathbf{poslin(}\begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix}\mathbf{a}^2(t)) = \mathbf{poslin}\left(\begin{bmatrix} a_1^2(t)-\varepsilon a_2^2(t) \\ a_2^2(t)-\varepsilon a_1^2(t) \end{bmatrix}\right) \tag{3.2-7}$$

For orange (round, rough, less than 1 pound),

$$a^1 = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} (3+3) \\ (1+3) \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix} \tag{3.2-8}$$

$$\mathbf{a}^2(1) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(0)) = \mathbf{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\begin{bmatrix} 6 \\ 4 \end{bmatrix}\right) = \mathbf{poslin}\left(\begin{bmatrix} 4 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 4 \\ 1 \end{bmatrix} \tag{3.2-9}$$

$$\mathbf{a}^2(2) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(1)) = \mathbf{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\begin{bmatrix} 4 \\ 1 \end{bmatrix}\right) = \mathbf{poslin}\left(\begin{bmatrix} 3.5 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 3.5 \\ 0 \end{bmatrix} \tag{3.2-10}$$

For apple (round, smooth, less than 1 pound),

$$a^1 = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} (1+3) \\ (3+3) \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix} \quad (3.2\text{-}11)$$

$$\mathbf{a}^2(1) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(0)) = \mathbf{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 6 \end{bmatrix} \right) = \mathbf{poslin}\left(\begin{bmatrix} 1 \\ 4 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ 4 \end{bmatrix} \quad (3.2\text{-}12)$$

$$\mathbf{a}^2(2) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(1)) = \mathbf{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} \right) = \mathbf{poslin}\left(\begin{bmatrix} -1 \\ 3.5 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 3.5 \end{bmatrix} \quad (3.2\text{-}13)$$

For unidentified fruit (elliptical, rough, less than 1 pound)

$$a^1 = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} (1+3) \\ (-1+3) \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix} \quad (3.2\text{-}14)$$

$$\mathbf{a}^2(1) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(0)) = \mathbf{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \end{bmatrix} \right) = \mathbf{poslin}\left(\begin{bmatrix} 3 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \quad (3.2\text{-}15)$$

$$\mathbf{a}^2(2) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(1)) = \mathbf{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \end{bmatrix} \right) = \mathbf{poslin}\left(\begin{bmatrix} 3 \\ -1.5 \end{bmatrix} \right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \quad (3.2\text{-}16)$$

3.3 Hopfield Network

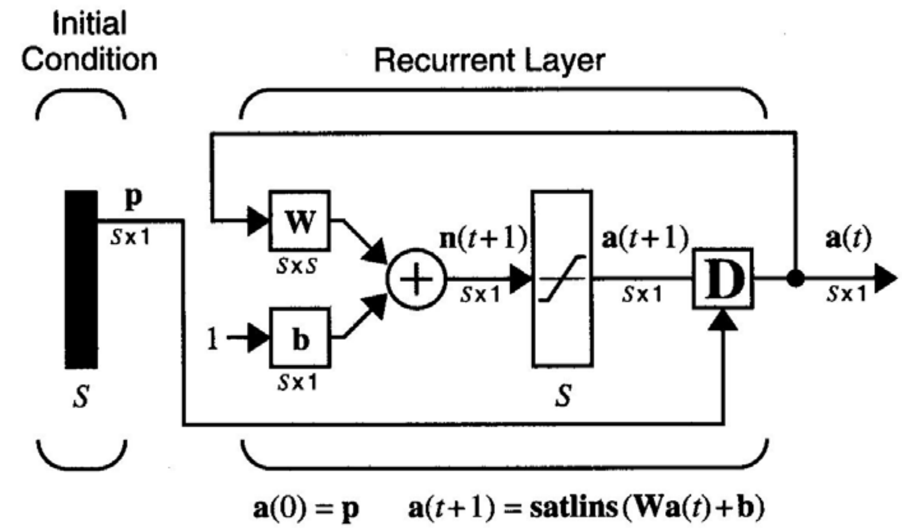

Figure 3.3-1 Hopfield Network

- S : No of Inputs = No of Neurons in Layer 1 = No of Outputs

$$\mathbf{a}(0) = \mathbf{p} \tag{3.3-1}$$

$$\mathbf{a}(t+1) = \mathbf{satlins}(\mathbf{Wa}(t)+\mathbf{b}) \tag{3.3-2}$$

$$\mathbf{W} = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 1.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0.9 \\ 0 \\ -0.9 \end{bmatrix} \quad (3.3\text{-}3)$$

$$\begin{bmatrix} a_1(t+1) \\ a_2(t+1) \\ a_3(t+1) \end{bmatrix} = \mathbf{satlins}\left(\begin{bmatrix} 0.2a_1(t)+0.9 \\ 1.2a_2(t) \\ 0.2a_3(t)-0.9 \end{bmatrix} \right) \quad (3.3\text{-}4)$$

For orange (round, rough, less than 1 pound),

$$\mathbf{a}(0) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \mathbf{a}(1) = \mathbf{satlins}\begin{bmatrix} 1.1 \\ -1.2 \\ -1.1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \mathbf{a}(2)\ \mathbf{satlins}\begin{bmatrix} 1.1 \\ -1.2 \\ -1.1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \mathbf{a}(3) = \mathbf{satlins}\begin{bmatrix} 1.1 \\ -1.2 \\ -1.1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \quad (3.3\text{-}5)$$

For apple (round, smooth, less than 1 pound),

$$\mathbf{a}(0) = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \mathbf{a}(1) = \mathbf{satlins}\begin{bmatrix} 1.1 \\ 1.2 \\ -1.1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \mathbf{a}(2) = \mathbf{satlins}\begin{bmatrix} 1.1 \\ 1.2 \\ -1.1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \mathbf{a}(3) = \mathbf{satlins}\begin{bmatrix} 1.1 \\ 1.2 \\ -1.1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \quad (3.3\text{-}6)$$

For unidentified fruit (elliptical, rough, less than 1 pound)

$$\mathbf{a}(0) = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}, \mathbf{a}(1) = \mathbf{satlins}\begin{bmatrix} 0.7 \\ -1.2 \\ -1.1 \end{bmatrix} = \begin{bmatrix} 0.7 \\ -1 \\ -1 \end{bmatrix}, \mathbf{a}(2) = \mathbf{satlins}\begin{bmatrix} 1.04 \\ -1.2 \\ -1.1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \mathbf{a}(3) = \mathbf{satlins}\begin{bmatrix} 1.1 \\ -1.2 \\ -1.1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \quad (3.3\text{-}7)$$

3.4 Examples of Neural Network Application

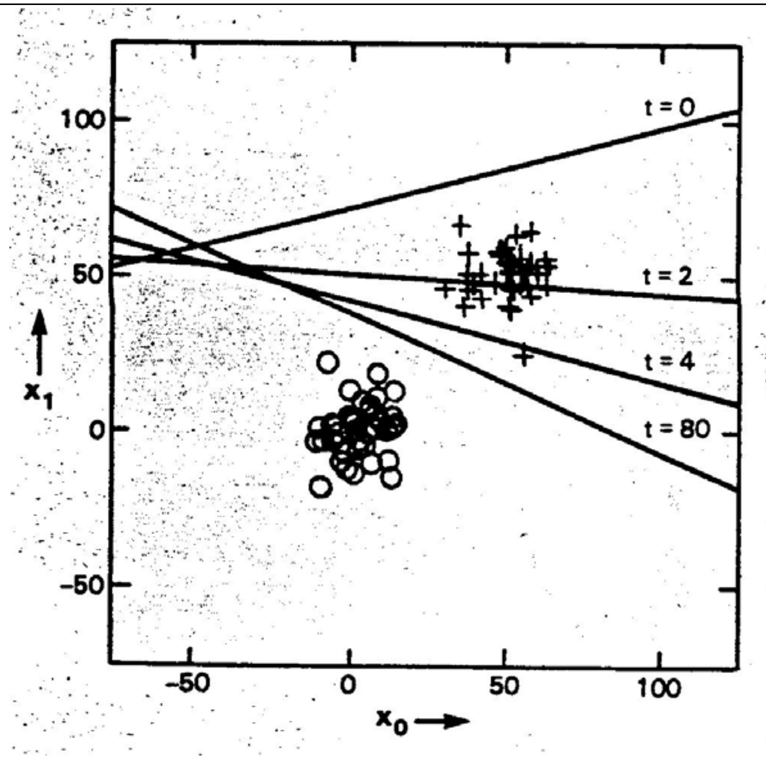

Figure 3.4-1 An Example of the decision boundaries formed by the perceptron convergence procedure with two classes. Samples from class A are represented by circles and samples from class B by crosses. Lines represent decision boundaries after trials where errors occurred and weights were adapted.

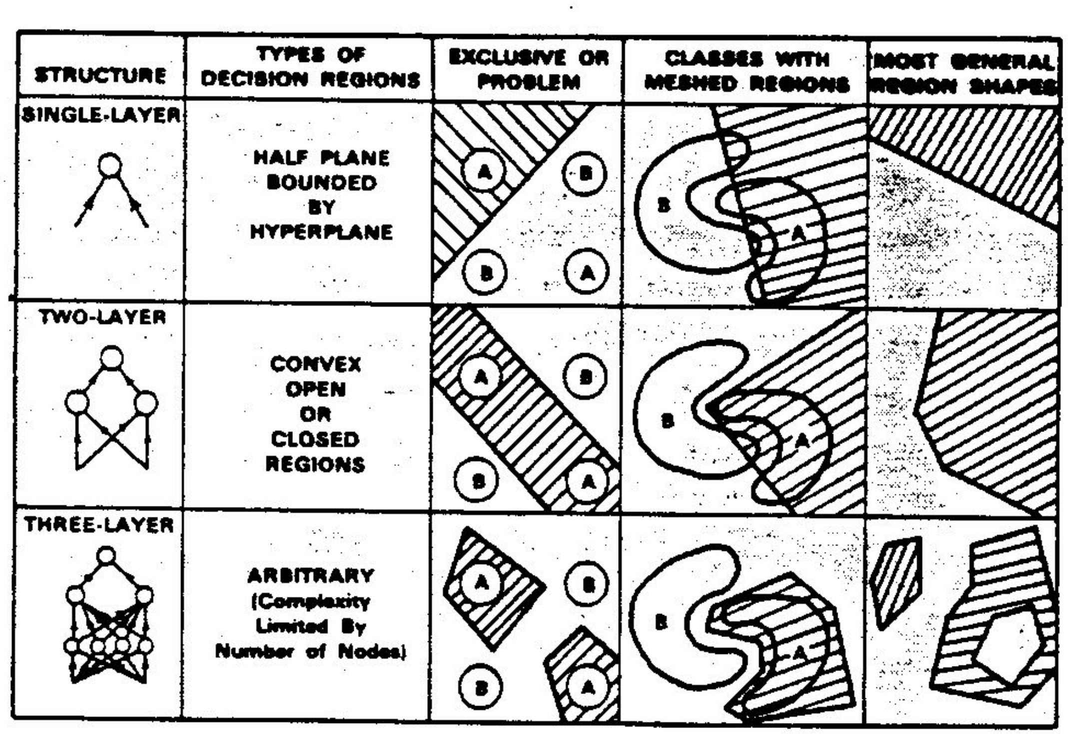

Figure 3.4-2 Types of decision regions that can be formed by single- and multi-layer perceptrons with one and two layers of hidden units and two inputs. Shading denotes decision regions for class A. Smooth closed contours bound input distributions for classes A and B. Nodes in all nets use hard limiting nonlinearities.

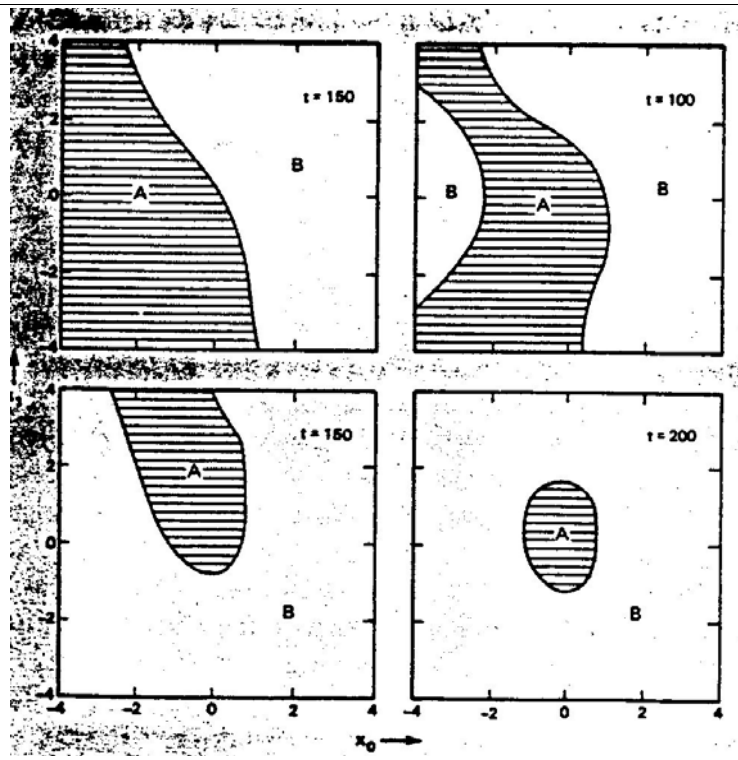

Figure 3.4-3 Decision regions after 50, 100, 150, and 200 trials generated by a two layer perceptron using the back-propagation training algorithm. Inputs from classes A and B were presented on alternate trials. Samples from class A were distributed uniformly over a circle of radius 1 centered at the origin. Samples from class B were distributed uniformly outside the circle. The shaded area denotes the decision region for class A.

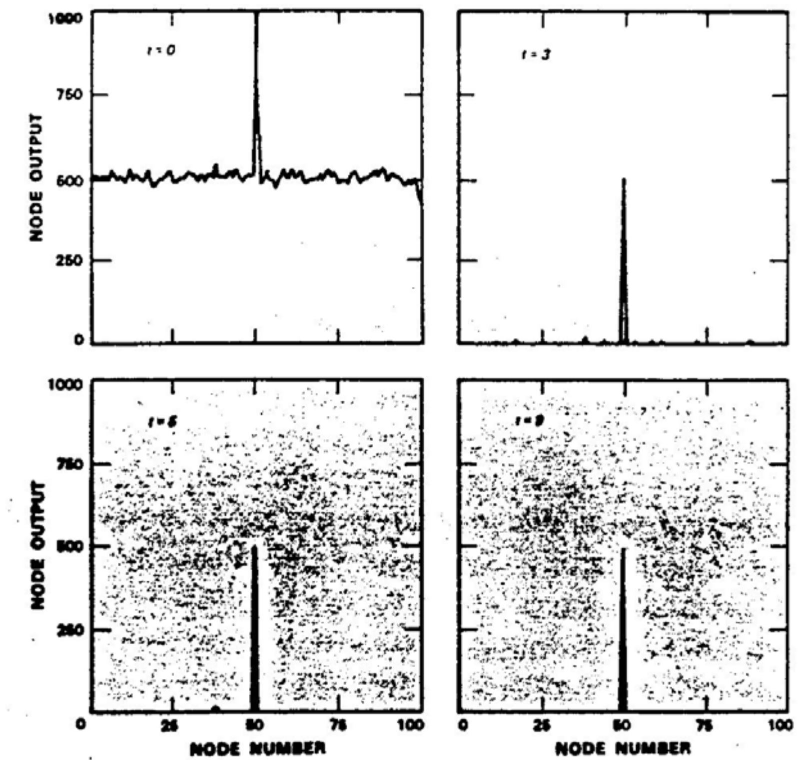

Figure 3.4-4 Node outputs for a Hamming net with 1,000 binary inputs and 100 output nodes or classes. Output values of all 100 nodes are presented at time zero and after 3, 6, and 9 iterations. The input was the exemplar pattern corresponding to output node 50

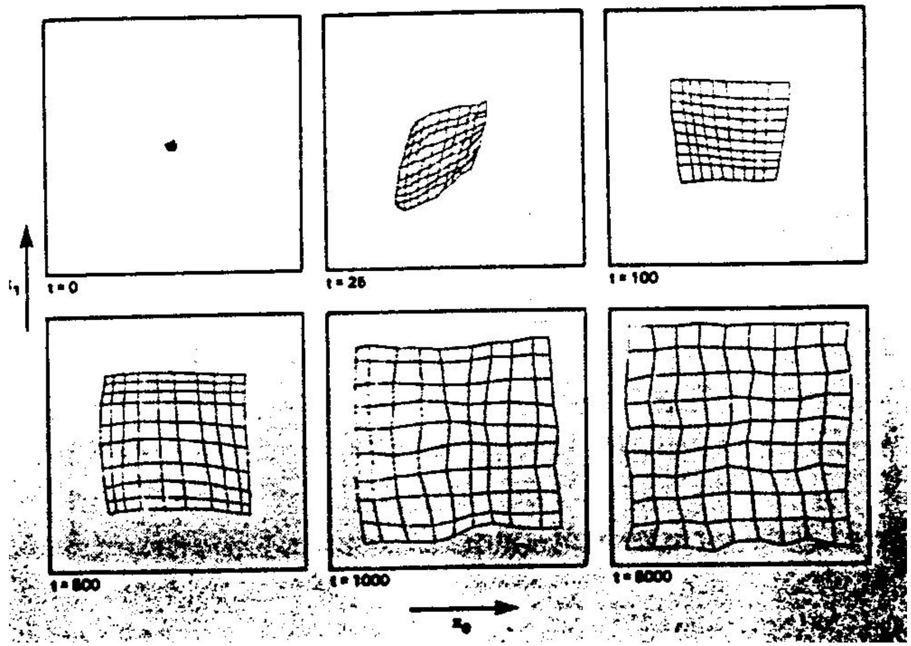

Figure 3.4-5 Weights to 100 output nodes from two input nodes as a feature map is being formed. The horizontal axis represents the value of the weight from input x_0 and the vertical axis represents the value of the weight from input x_1. Line intersections specify the two weights for each node. Lines connect weights for nodes that are nearest neighbors. An orderly grid indicates that topologically close nodes code inputs that are physically similar. Inputs were random, independent, and uniformly distributed over the area shown.

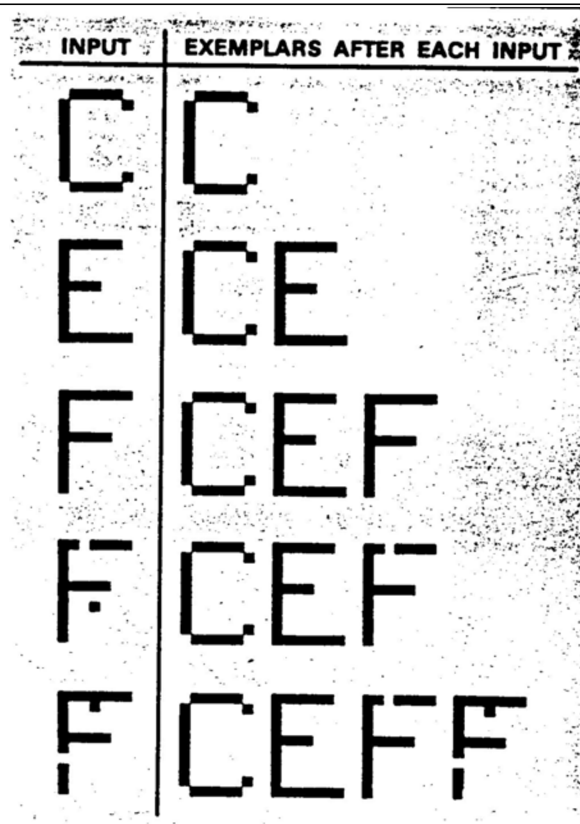

Figure 3.4-6 An example of the behavior of the Carpenter Grossberg net for letter patterns. Binary input patterns on the left were applied sequentially starting with the upper “C” pattern. Exemplars formed by top-down connection weights after each input was presented are shown at the right.

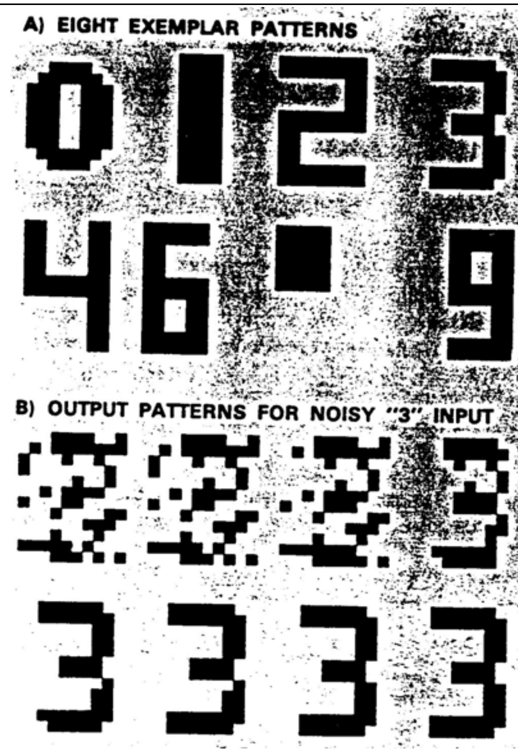

Figure 3.4-7 An example of the behavior of a Hopfield net when used as a content-addressable memory. A 120 node net was trained using the eight exemplars shown in (A). The pattern for the digit "3" was corrupted by randomly reversing each bit with a probability of 0.25, and then applied to the net at time zero. Outputs at time zero and after the first seven iterations are shown in (B).

4 Perceptron Learning Rule

- Supervised Learning: Perceptron, Back-Propagation, Supervised Hebb, $\{\mathbf{p}_1, \mathbf{t}_1\}$, $\{\mathbf{p}_2, \mathbf{t}_2\}$, …, $\{\mathbf{p}_Q, \mathbf{t}_Q\}$, where $\mathbf{p}_Q$ is an input and $\mathbf{t}_Q$ is the corresponding correct (target) output.
- Unsupervised Learning: Grossberg, Unsupervised Hebb
- Reinforcement Learning: Q, Bayesian

4.1 Perceptron Architecture

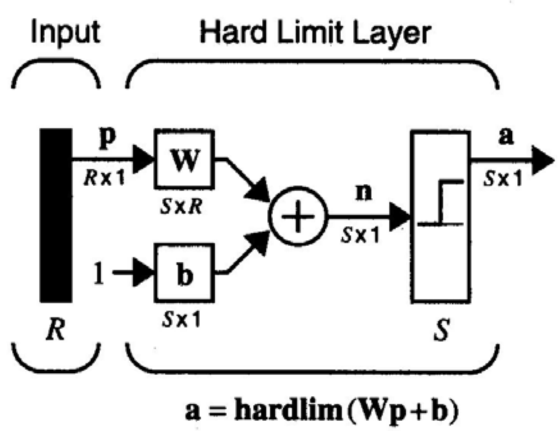

Figure 4.1-1 Perceptron Network

$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp}+\mathbf{b}) \tag{4.1-1}$$

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \tag{4.1-2}$$

$${}_{i}\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix} \tag{4.1-3}$$

$$\mathbf{W} = \begin{bmatrix} {}_{1}\mathbf{w}^{T} \\ {}_{2}\mathbf{w}^{T} \\ \vdots \\ {}_{s}\mathbf{w}^{T} \end{bmatrix} \tag{4.1-4}$$

$$a_i = hardlim(n_i) = hardlim({}_{i}\mathbf{w}^{T}\mathbf{p} + b_i) \tag{4.1-5}$$

$$a = hardlim(n) = \begin{cases} 1 & if\ (n \geq 0) \\ 0 & otherwise \end{cases} \tag{4.1-6}$$

4.1.1 Single-Neuron Perceptron

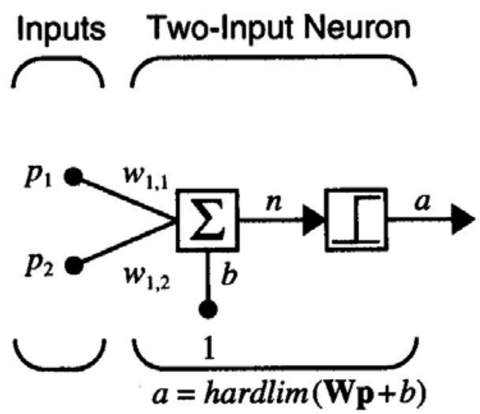

Figure 4.1.1-1 Two-Input/Single-Output Perceptron

$$a = hardlim(n) = hardlim(\mathbf{W}\mathbf{p}+b) = hardlim({}_1\mathbf{w}^T\mathbf{p}+b) = hardlim(w_{1,1}p_1+ w_{1,2}p_2+b) \quad (4.1.1\text{-}1)$$

The decision boundary is determined by the input vectors for which the net input n is zero

$$n = {}_1\mathbf{w}^T\mathbf{p}+b = w_{1,1}p_1+ w_{1,2}p_2+b = 0 \quad (4.1.1\text{-}2)$$

Example:

$$w_{1,1} = 1,\ w_{1,2} = 1,\ b = -1 \quad (4.1.1\text{-}3)$$

$$n = {}_1\mathbf{w}^T\mathbf{p}+b = w_{1,1}p_1+ w_{1,2}p_2+b = p_1+ p_2-1 = 0 \quad (4.1.1\text{-}4)$$

Intersection points on the axis,

$$p_2 = -\frac{b}{w_{1,2}} = -\frac{-1}{1} = 1 \quad \text{if } p_1 = 0 \tag{4.1.1-5}$$

$$p_1 = -\frac{b}{w_{1,1}} = -\frac{-1}{1} = 1 \quad \text{if } p_2 = 0 \tag{4.1.1-6}$$

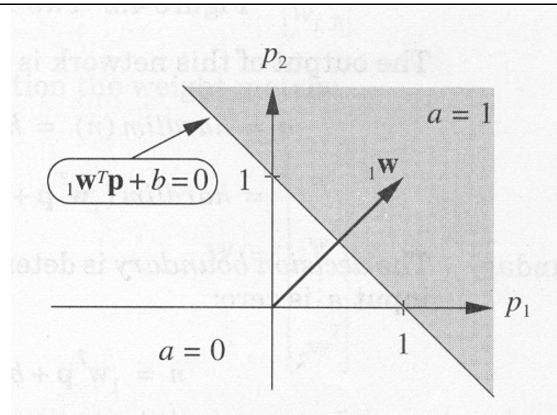

Figure 4.1.1-2 Decision Boundary for Two-Input Perceptron

For the input $\mathbf{p} = [2 \;\; 0]^T$

$$a = hardlim({}_1\mathbf{w}^T\mathbf{p}+b) = hardlim\left([1 \;\; 1]\begin{bmatrix}2\\0\end{bmatrix} - 1\right) = 1 \tag{4.1.1-7}$$

The weight vector ${}_1\mathbf{w}$ is always orthogonal to decision boundary and points toward the region where the neuron output is 1.

For a simple logic function AND gate, the input/target pairs for the AND gate

$$\left\{\mathbf{p}_1 = \begin{bmatrix}0\\0\end{bmatrix}, t_1 = 0\right\} \left\{\mathbf{p}_2 = \begin{bmatrix}0\\1\end{bmatrix}, t_2 = 0\right\} \left\{\mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix}, t_3 = 0\right\} \left\{\mathbf{p}_4 = \begin{bmatrix}1\\1\end{bmatrix}, t_4 = 1\right\} \quad (4.1.1\text{-}8)$$

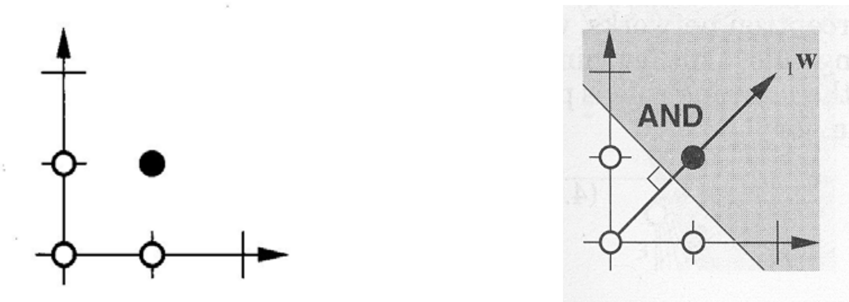

Figure 4.1.1-3 AND Gate Problem

Select a weight vector which points 45°,

$$_1\mathbf{w} = \begin{bmatrix}2\\2\end{bmatrix} \quad (4.1.1\text{-}9)$$

$$_1\mathbf{w}^T\mathbf{p} + b = 0 \quad (4.1.1\text{-}10)$$

Select $\mathbf{p} = [1.5 \;\; 0]^T$, a point on the decision boundary,

$$_1\mathbf{w}^T\mathbf{p} + b = \begin{bmatrix}2 & 2\end{bmatrix}\begin{bmatrix}1.5\\0\end{bmatrix} + b = 3 + b = 0 \Rightarrow b = -3 \quad (4.1.1\text{-}11)$$

4.2 Perceptron Learning Rule

Perceptron Learning Rule: Supervised Learning Rule

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\} \tag{4.2-1}$$

4.2.1 Test Problem

$$\left\{\mathbf{p}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1\right\} \left\{\mathbf{p}_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0\right\} \left\{\mathbf{p}_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, t_3 = 0\right\} \tag{4.2.1-1}$$

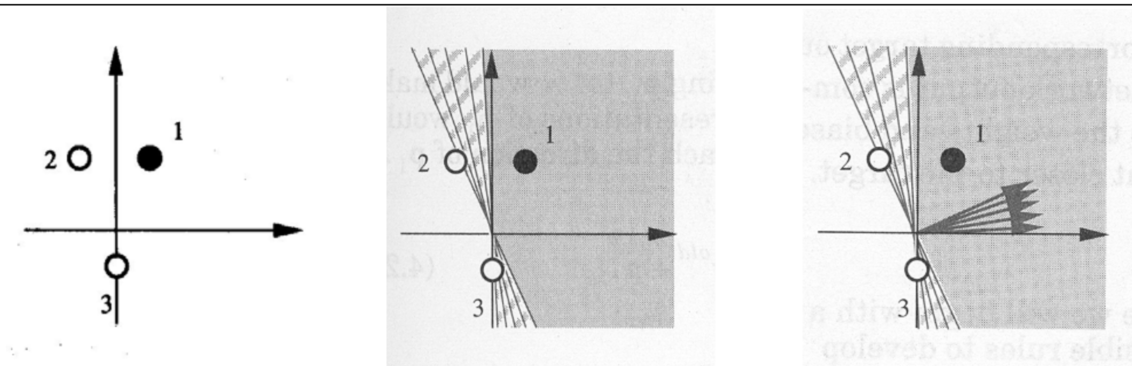

Figure 4.2.1-1 The Input/Target Pairs

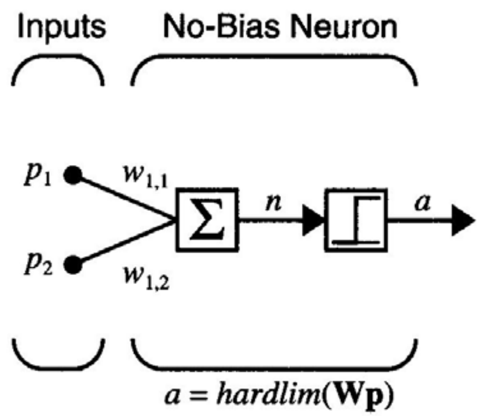

Figure 4.2.1-2 Test Problem Network

4.2.2 Constructing Learning Rules

Select an arbitrary initial weight vector,

$$ {}_1\mathbf{w}^T = [1.0 \;\; -0.8] \qquad (4.2.2\text{-}1)$$

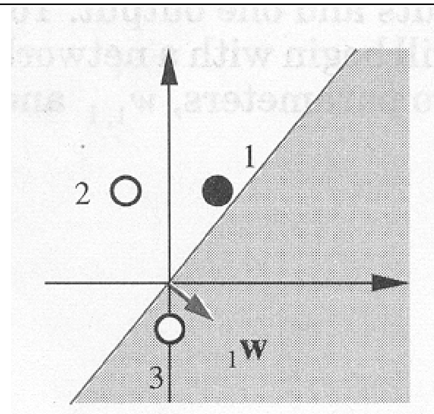

Figure 4.2.2-1 Decision Boundary by Random Weight Vector

With $\mathbf{p}_1$:

$$ a = hardlim({}_1\mathbf{w}^T\mathbf{p}_1) = hardlim\left(\begin{bmatrix}1.0 & -0.8\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix}\right) = hardlim(-0.6) = 0 \qquad (4.2.2\text{-}2)$$

$$ \text{If } t = 1 \text{ and } a = 0, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p} \qquad (4.2.2\text{-}3)$$

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} \qquad (4.2.2\text{-}4) $$

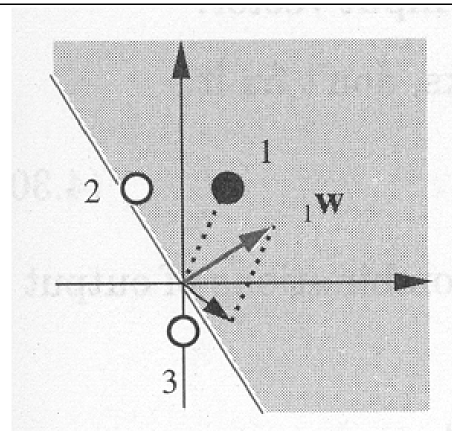

Figure 4.2.2-2 Altering of Weight Vector

With $\mathbf{p}_2$

$$ a = hardlim({}_1\mathbf{w}^T\mathbf{p}_2) = hardlim\left(\begin{bmatrix} 2.0 & 1.2 \end{bmatrix}\begin{bmatrix} -1 \\ 2 \end{bmatrix}\right) = hardlim(0.4) = 1 \qquad (4.2.2\text{-}5) $$

If $t = 0$ and $a = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$ - $\mathbf{p}$ (4.2.2-6)

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} \qquad (4.2.2\text{-}7) $$

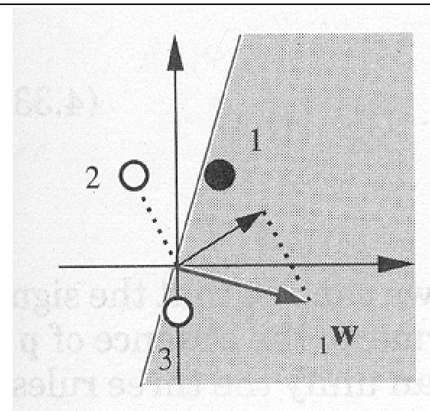

Figure 4.2.2-3 Modified Weight Vector

With $\mathbf{p}_3$

$$a = hardlim({}_1\mathbf{w}^T\mathbf{p}_3) = hardlim\left(\begin{bmatrix}3.0 & -0.8\end{bmatrix}\begin{bmatrix}0\\-1\end{bmatrix}\right) = hardlim(0.8) = 1 \quad (4.2.2\text{-}8)$$

$$_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_3 = \begin{bmatrix}3.0\\-0.8\end{bmatrix} - \begin{bmatrix}0\\-1\end{bmatrix} = \begin{bmatrix}3.0\\0.2\end{bmatrix} \quad (4.2.2\text{-}9)$$

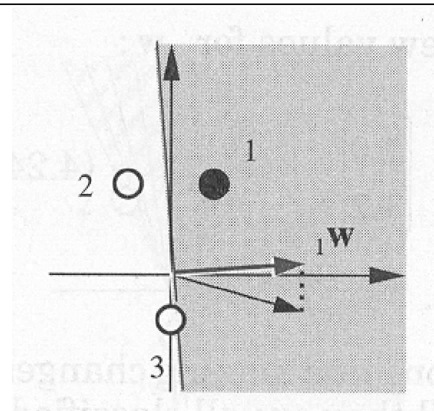

Figure 4.2.2-4 Weight Vector and Decision Boundary after 3 Input/Target Pairs

$$\text{If } t = a, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} \quad (4.2.2\text{-}10)$$

4.2.3 Unified Learning Rule

$$e = t - a. \quad (4.2.3\text{-}1)$$

$$\text{If } e = 1, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}. \quad (4.2.3\text{-}2)$$

$$\text{If } e = -1, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}. \quad (4.2.3\text{-}3)$$

$$\text{If } e = 0, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}. \quad (4.2.3\text{-}4)$$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t-a)\mathbf{p} \quad (4.2.3\text{-}5)$$

$$b^{new} = b^{old} + e \quad (4.2.3\text{-}6)$$

4.2.4 Training Multiple-Neuron Perceptrons

$$ {}_i\mathbf{w}^{new} = {}_i\mathbf{w}^{old} + e_i\mathbf{p} \tag{4.2.4-1}$$

$$ b_i^{new} = b_i^{old} + e_i \tag{4.2.4-2}$$

$$ \mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T \tag{4.2.4-3}$$

$$ \mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e} \tag{4.2.4-4}$$

4.2.5 Limitations

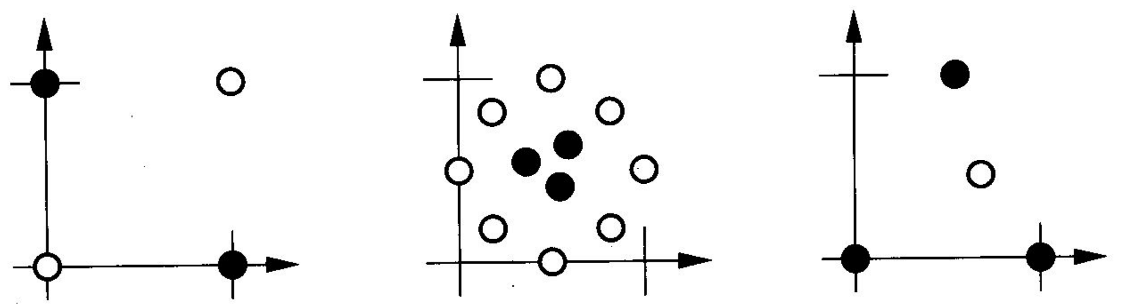

Figure 4.2.5-1 Linearly Inseparable Problems

5 Signal and Weight Vector Spaces

5.1 Linear Vector Spaces

Definition: A **linear vector space**, X, is a set of elements (vectors) defined over a scalar field, F, that satisfies the following conditions:

1. An operation called vector addition is defined such that if $x \in X$ (x is an element of X) and $y \in X$, then $x+y \in X$.
2. $x+y = y+x$.
3. $(x+y)+z = x+(y+z)$.
4. There is a unique vector $0 \in X$, called the zero vector, such that $x+0=x$ for all $x \in X$.
5. For each vector $x \in X$ there is a unique vector in X, to be called $-x$, such that $x + (-x) = 0$.
6. An operation, called multiplication, is defined such that for all scalars $a \in F$, and all vectors $x \in X$, $ax \in X$.
7. For any $x \in X$, $1x = x$ (for scalar 1).
8. For any two scalars $a \in F$ and $b \in F$, and any $x \in X$, $a(bx) = (ab)x$.
9. $(a+b)x = ax+bx$.
10. $a(x+y) = ax+ay$.

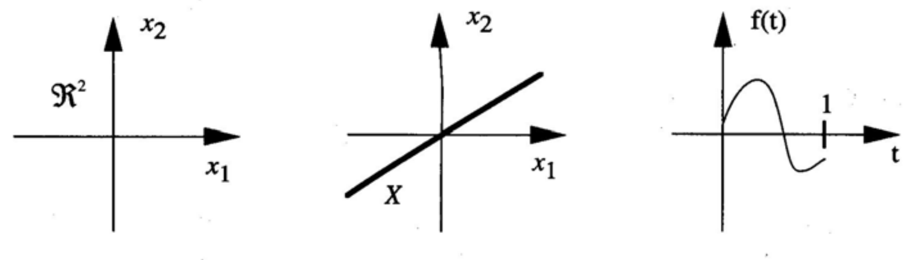

Figure 4.1-1 Linear Vector Spaces

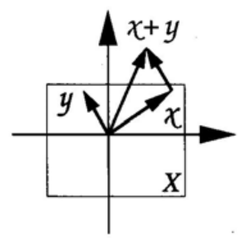

Figure 4.1-2 Non-Linear Vector Space

- Examples of the vector spaces are two-dimensional Euclidean space, polynomials of degree less than or equal to 2, continuous functions defined on the interval [0, 1].
- For subset of two-dimensional Euclidean space, some are vector spaces, e.g., straight line. Some are not vector spaces, e.g., box area at the origin.

5.2 Linear Independence

Definition: Consider n vectors $\{x_1, x_2, ..., x_n\}$. If there exist n scalars $a_1, a_2, ..., a_n$, at least one of which is nonzero, such that

$$a_1x_1+a_2x_2+\ldots+a_nx_n = 0, \tag{5.2-1}$$

then the $\{x_i\}$ are **linearly dependent**.

Definition: If $a_1x_1+a_2x_2+\ldots+a_nx_n = 0$ implies that each $a_i = 0$, then $\{x_i\}$ is a set of **linearly independent** vectors.

Examples:

$$x_1 = i = \begin{bmatrix}1\\0\\0\end{bmatrix}, x_2 = j = \begin{bmatrix}0\\1\\0\end{bmatrix}, x_3 = k = \begin{bmatrix}0\\0\\1\end{bmatrix} \tag{5.2-2}$$

$a_1x_1+a_2x_2+a_3x_3= 0$, only when $a_1= a_2 = a_3 = 0$, thus $\{x_1, x_2, x_3\}$ are linearly independent.

$$x_1 = 1+t+t^2 = \begin{bmatrix}1\\1\\1\end{bmatrix}, x_2 = 2+2t+t^2 = \begin{bmatrix}2\\2\\1\end{bmatrix}, x_3 = 1+t = \begin{bmatrix}1\\1\\0\end{bmatrix} \tag{5.2-3}$$

$a_1x_1+a_2x_2+a_3x_3= 0$, not only when $a_1= a_2 = a_3 = 0$, but also when $a_1= 1$, $a_2 = -1$, $a_3 = 1$, thus $\{x_1, x_2, x_3\}$ are linearly dependent.

5.3 Spanning a Space

Definition: Let X be a linear vector space and let $\{x_1, x_2, ..., x_m\}$ be a subset of general vectors in X. This subset **spans** X if and only if for every vector $x \in X$ there exist scalars $a_1, a_2, ..., a_m$ such that

$$x = a_1x_1+a_2x_2+\ldots+a_mx_m \tag{5.3-1}$$

Definition: The **dimension** of a vector space is determined by the minimum number of vectors it takes to span the space.

Definition: A **basis set** for X is a set of linearly independent vectors that spans X. Any basis set contains the minimum number of vectors required to span the space.

- The dimension of X is therefore equal to the number of elements in the basis set.
- Any vector space can have many basis sets, but each one must contain the same number of elements.

Examples: Let X be polynomial degree less than 2

$$\{x_1 = 1, x_2 = t, x_3 = t^2\} \text{ is a basis set of } X \tag{5.3-2}$$

$$\{x_1 = 2, x_2 = 2t, x_3 = 2t^2\} \text{ is a basis set of } X \tag{5.3-3}$$

$$\{x_1 = 1, x_2 = t, x_3 = t^2, x_4 = 2\} \text{ spans } X \text{ but not a basis set of } X \tag{5.3-4}$$

5.4 Inner Product

Definition: Any scalar function of x and y can be defined as an **inner product**, (x,y), provided that the following properties are satisfied:

1. $(x,y) = (y,x)$.
2. $(x,ay_1+by_2) = a(x,y_1)+b(x,y_2)$.
3. $(x,x) \geq 0$, where equality holds if and only if x is the zero vector.

The standard inner product for vectors in R^n

$$(x,y) = \mathbf{x}^T\mathbf{y} = x_1y_1+ x_2y_2+\ldots+ x_ny_n, \tag{5.4-1}$$

Example:

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, y = \begin{bmatrix} 3 \\ 4 \end{bmatrix}, (x,y) = \begin{bmatrix} 1 & 2 \end{bmatrix}\begin{bmatrix} 3 \\ 4 \end{bmatrix} = (1\times 3) + (2\times 4) = 11 \tag{5.4-2}$$

5.5 Norm

Definition: A scalar function $||x||$ is called a **norm** if it satisfies the following properties:

1. $||x|| \geq 0$.
2. $||x||= 0$ if and only $x = 0$.
3. $||ax|| = |a|\ ||x||$ for scalar a.
4. $||x+y|| \leq ||x||+||y||$.

There are many functions that would satisfy these conditions. One common norm based on the inner product

$$||x|| = (x,x)^{1/2} \tag{5.5-1}$$

For Euclidean spaces, R^n,

$$\|\mathbf{x}\| = (\mathbf{x}^T\mathbf{x})^{1/2} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}. \tag{5.5-2}$$

For vector spaces of dimension greater than two, the angle θ between two vectors x and y

$$\cos\theta = \frac{(x, y)}{\|x\|\|y\|} \tag{5.5-3}$$

Example: The angle between $\begin{bmatrix}1\\2\end{bmatrix}$ and $\begin{bmatrix}2\\1\end{bmatrix}$,

$$\theta = \arccos\left(\frac{\begin{bmatrix}1 & 2\end{bmatrix}\begin{bmatrix}2\\1\end{bmatrix}}{\sqrt{1^2+2^2}\sqrt{2^2+1^2}}\right) = \arccos\left(\frac{4}{5}\right) \tag{5.5-4}$$

5.6 Orthogonality

Definition: Two **vectors** $x,y \in X$ are said to be **orthogonal** if $(x,y) = 0$.

Definition: A **vector** $x \in X$ is **orthogonal** to a **subspace** X_1 if x is orthogonal to every vector in X_1. This is typically represented as $x \perp X_1$.

Definition: A **subspace** X_1 is **orthogonal** to a **subspace** X_2 if every vector in X_1 is orthogonal to every vector in X_2. This is represented by $X_1 \perp X_2$.

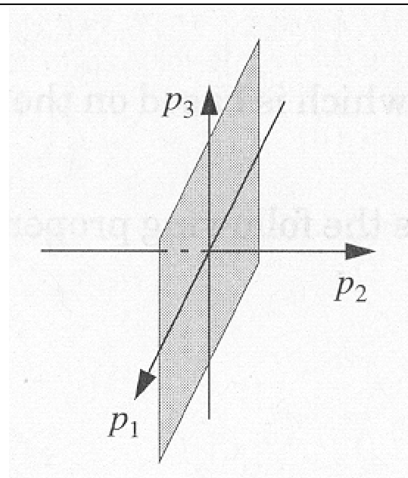

Figure 5.6-1 p_1, p_3 plane is a subspace of R^3, which is orthogonal to the p_2 axis

5.6.1 Gram-Schmidt Orthogonalization

- Gram-Schmidt orthogonalization is used to convert non-orthogonal basis set to orthogonal basis set.

Procedure: From non-orthogonal n independent vectors $y_1, y_2, \ldots, y_n$ to n orthogonal vectors $v_1, v_2, \ldots, v_n$,

The first orthogonal vector is chosen to be the first independent vector:

$$v_1 = y_1 \qquad (5.6.1\text{-}1)$$

To obtain the second orthogonal vector we use y_2, but subtract off the portion of y_2 that is in the direction of y_1.

$$v_2 = y_2 - av_1 \qquad (5.6.1\text{-}2)$$

av_1 is the projection of y_2 on the vector v_1.

$$(v_1,v_2) = (v_1, y_2-av_1) = (v_1, y_2)-a(v_1,v_1) = 0 \qquad (5.6.1\text{-}3)$$

$$a = \frac{(v_1, y_2)}{(v_1, v_1)} \qquad (5.6.1\text{-}4)$$

For the kth step,

$$v_k = y_k - \sum_{i=1}^{k-1} \frac{(v_i, y_k)}{(v_i, v_i)} v_i \qquad (5.6.1\text{-}5)$$

Example: $y_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, $y_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$,

$$v_1 = y_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{5.6.1-6}$$

$$v_2 = y_2 - \frac{(v_1, y_2)}{(v_1, v_1)} v_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \frac{\begin{bmatrix} 2 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix}}{\begin{bmatrix} 2 & 1 \end{bmatrix}\begin{bmatrix} 2 \\ 1 \end{bmatrix}} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 1.6 \\ 0.8 \end{bmatrix} = \begin{bmatrix} -0.6 \\ 1.2 \end{bmatrix} \tag{5.6.1-7}$$

- $\mathbf{v}_1$ and $\mathbf{v}_2$ can be converted to a set of orthonormal (orthogonal and normalized) vectors by dividing each vector by its norm.

$$\text{Normalized } v_1 = \frac{1}{\sqrt{5}} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2/\sqrt{5} \\ 1/\sqrt{5} \end{bmatrix} \tag{5.6.1-8}$$

$$\text{Normalized } v_2 = \frac{1}{\sqrt{(-0.6)^2 + (1.2)^2}} \begin{bmatrix} -0.6 \\ 1.2 \end{bmatrix} = \begin{bmatrix} -0.6/\sqrt{1.8} \\ 1.2/\sqrt{1.8} \end{bmatrix} \tag{5.6.1-9}$$

Example: $y_1 = \begin{bmatrix}1\\1\\1\end{bmatrix}, y_2 = \begin{bmatrix}1\\2\\3\end{bmatrix}, y_3 = \begin{bmatrix}1\\3\\2\end{bmatrix},$

$$v_1 = y_1 = \begin{bmatrix}1\\1\\1\end{bmatrix} \qquad (5.6.1\text{-}10)$$

$$v_2 = y_2 - \frac{(v_1, y_2)}{(v_1, v_1)} v_1 = \begin{bmatrix}1\\2\\3\end{bmatrix} - \frac{\begin{bmatrix}1 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\2\\3\end{bmatrix}}{\begin{bmatrix}1 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}}\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}-1\\0\\1\end{bmatrix} \qquad (5.6.1\text{-}11)$$

$$v_3 = y_3 - \frac{(v_1, y_3)}{(v_1, v_1)} v_1 - \frac{(v_2, y_3)}{(v_2, v_2)} v_2 = \begin{bmatrix}1\\3\\2\end{bmatrix} - \frac{\begin{bmatrix}1 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\3\\2\end{bmatrix}}{\begin{bmatrix}1 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}}\begin{bmatrix}1\\1\\1\end{bmatrix} - \frac{\begin{bmatrix}-1 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\3\\2\end{bmatrix}}{\begin{bmatrix}-1 & 0 & 1\end{bmatrix}\begin{bmatrix}-1\\0\\1\end{bmatrix}}\begin{bmatrix}-1\\0\\1\end{bmatrix} = \begin{bmatrix}-0.5\\1\\-0.5\end{bmatrix} \qquad (5.6.1\text{-}12)$$

5.7 Vector Expansions

Definition: If a vector space X has a basis set $\{v_1, v_2, ..., v_n\}$, then any $x \in X$ has a unique **vector expansion**

$$x = \sum_{i=1}^{n} x_i v_i = x_1 v_1 + x_2 v_2 + \cdots + x_n v_n \quad (5.7\text{-}1)$$

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T \quad (5.7\text{-}2)$$

For orthogonal basis set ($(v_i,v_j) = 0$, $i \neq j$),

$$(v_j, x) = (v_j, \sum_{i=1}^{n} x_i v_i) = \sum_{i=1}^{n} x_i (v_j, v_i) = x_j (v_j, v_j) \quad (5.7\text{-}3)$$

$$x_j = \frac{(v_j, x)}{(v_j, v_j)} \quad (5.7\text{-}4)$$

Example: For $v_1 = \begin{bmatrix}1\\0\\0\end{bmatrix}, v_2 = \begin{bmatrix}0\\1\\0\end{bmatrix}, v_3 = \begin{bmatrix}0\\0\\1\end{bmatrix}$,

$$\begin{bmatrix}6\\9\\9\end{bmatrix} = \frac{\begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}6\\9\\9\end{bmatrix}}{\begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix}}\begin{bmatrix}1\\0\\0\end{bmatrix} + \frac{\begin{bmatrix}0 & 1 & 0\end{bmatrix}\begin{bmatrix}6\\9\\9\end{bmatrix}}{\begin{bmatrix}0 & 1 & 0\end{bmatrix}\begin{bmatrix}0\\1\\0\end{bmatrix}}\begin{bmatrix}0\\1\\0\end{bmatrix} + \frac{\begin{bmatrix}0 & 0 & 1\end{bmatrix}\begin{bmatrix}6\\9\\9\end{bmatrix}}{\begin{bmatrix}0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix}}\begin{bmatrix}0\\0\\1\end{bmatrix} = 6\begin{bmatrix}1\\0\\0\end{bmatrix} + 9\begin{bmatrix}0\\1\\0\end{bmatrix} + 9\begin{bmatrix}0\\0\\1\end{bmatrix} = 6v_1 + 9v_2 + 9v_3 \qquad (5.7\text{-}5)$$

Example: For $v_1 = \begin{bmatrix}2\\0\\0\end{bmatrix}, v_2 = \begin{bmatrix}0\\2\\0\end{bmatrix}, v_3 = \begin{bmatrix}0\\0\\2\end{bmatrix}$,

$$\begin{bmatrix}6\\9\\9\end{bmatrix} = \frac{\begin{bmatrix}2 & 0 & 0\end{bmatrix}\begin{bmatrix}6\\9\\9\end{bmatrix}}{\begin{bmatrix}2 & 0 & 0\end{bmatrix}\begin{bmatrix}2\\0\\0\end{bmatrix}}\begin{bmatrix}2\\0\\0\end{bmatrix} + \frac{\begin{bmatrix}0 & 2 & 0\end{bmatrix}\begin{bmatrix}6\\9\\9\end{bmatrix}}{\begin{bmatrix}0 & 2 & 0\end{bmatrix}\begin{bmatrix}0\\2\\0\end{bmatrix}}\begin{bmatrix}0\\2\\0\end{bmatrix} + \frac{\begin{bmatrix}0 & 0 & 2\end{bmatrix}\begin{bmatrix}6\\9\\9\end{bmatrix}}{\begin{bmatrix}0 & 0 & 2\end{bmatrix}\begin{bmatrix}2\\0\\0\end{bmatrix}}\begin{bmatrix}0\\0\\2\end{bmatrix} = 3\begin{bmatrix}2\\0\\0\end{bmatrix} + 4.5\begin{bmatrix}0\\2\\0\end{bmatrix} + 4.5\begin{bmatrix}0\\0\\2\end{bmatrix} = 3v_1 + 4.5v_2 + 4.5v_3 \qquad (5.7\text{-}6)$$

Example: For $v_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, v_2 = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}, v_3 = \begin{bmatrix} -0.5 \\ 1 \\ -0.5 \end{bmatrix}$,

$$\begin{bmatrix} 6 \\ 9 \\ 9 \end{bmatrix} = \frac{\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 6 \\ 9 \\ 9 \end{bmatrix}}{\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \frac{\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}\begin{bmatrix} 6 \\ 9 \\ 9 \end{bmatrix}}{\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}}\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} + \frac{\begin{bmatrix} -0.5 & 1 & -0.5 \end{bmatrix}\begin{bmatrix} 6 \\ 9 \\ 9 \end{bmatrix}}{\begin{bmatrix} -0.5 & 1 & -0.5 \end{bmatrix}\begin{bmatrix} -0.5 \\ 1 \\ -0.5 \end{bmatrix}}\begin{bmatrix} -0.5 \\ 1 \\ -0.5 \end{bmatrix} = 8\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + 1.5\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} + 1\begin{bmatrix} -0.5 \\ 1 \\ -0.5 \end{bmatrix} \tag{5.7-7}$$

5.7.1 Reciprocal Basis Vectors

For non-orthogonal basis vectors $\{v_1, v_2, \ldots, v_n\}$, a vector expansion requires the reciprocal basis vectors $\{r_1, r_2, \ldots, r_n\}$.

$$(r_i, v_j) = 0,\ i \neq j \text{ and } (r_i, v_j) = 1,\ i = j \tag{5.7.1-1}$$

$$\mathbf{R}^T\mathbf{B} = \mathbf{I} \tag{5.7.1-2}$$

$$\mathbf{B} = [\mathbf{v}_1\ \ \mathbf{v}_2\ \ \ldots\ \ \mathbf{v}_n] \tag{5.7.1-3}$$

$$\mathbf{R} = [\mathbf{r}_1\ \ \mathbf{r}_2\ \ \ldots\ \ \mathbf{r}_n] \tag{5.7.1-4}$$

$$\mathbf{R}^T = \mathbf{B}^{-1} \tag{5.7.1-5}$$

For a vector expansion,

$$x = x_1 v_1 + x_2 v_2 + \cdots + x_n v_n \tag{5.7.1-6}$$

$$(r_i, x) = x_1 (r_i, v_1) + x_2 (r_i, v_2) + \cdots + x_n (r_i, v_n) = x_i \tag{5.7.1-7}$$

Example: For $v_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, $v_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$,

$$\mathbf{R}^T = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}^{-1} = \begin{bmatrix} 2/3 & -1/3 \\ -1/3 & 2/3 \end{bmatrix}, r_1 = \begin{bmatrix} 2/3 \\ -1/3 \end{bmatrix}, r_2 = \begin{bmatrix} -1/3 \\ 2/3 \end{bmatrix} \tag{5.7.1-8}$$

$$\begin{bmatrix} 0 \\ 3/2 \end{bmatrix} = \left(\begin{bmatrix} 2/3 & -1/3 \end{bmatrix} \begin{bmatrix} 0 \\ 3/2 \end{bmatrix} \right) \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \left(\begin{bmatrix} -1/3 & 2/3 \end{bmatrix} \begin{bmatrix} 0 \\ 3/2 \end{bmatrix} \right) \begin{bmatrix} 1 \\ 2 \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} 2 \\ 1 \end{bmatrix} + 1 \begin{bmatrix} 1 \\ 2 \end{bmatrix} = -\frac{1}{2} v_1 + 1 v_2 \tag{5.7.1-9}$$

6 Linear Transformations for Neural Networks

A transformation consists of three parts:

1. a set of elements $X = \{x_i\}$, called the domain,
2. a set of elements $Y = \{y_i\}$, called the range, and
3. a rule relating each $x_i \in X$ to an element $y_i \in Y$.

Definition: A **transformation** A is **linear** if:

1. for all $x_1, x_2 \in X$, $A(x_1 + x_2) = A(x_1) + A(x_2)$,
2. for all $x \in X$, $a \in R$, $A(ax) = aA(x)$.

6.1 Matrix Representations

$\{v_1, v_2, \ldots, v_n\}$: basis for vector space X

$\{u_1, u_2, \ldots, u_m\}$: basis for vector space Y

$x \in X$ and $y \in Y$

$$x = \sum_{i=1}^{n} x_i v_i \quad \text{and} \quad y = \sum_{i=1}^{m} y_i u_i \tag{6.1-1}$$

A: linear transformation with domain X and range Y (A:$X \to Y$)

$$A(x) = y \tag{6.1-2}$$

$$A\left(\sum_{j=1}^{n} x_j v_j\right) = \sum_{i=1}^{m} y_i u_i \tag{6.1-3}$$

$$\sum_{j=1}^{n} x_j A(v_j) = \sum_{i=1}^{m} y_i u_i \tag{6.1-4}$$

$$A(v_j) = \sum_{i=1}^{m} a_{ij} u_i \tag{6.1-5}$$

$$\sum_{j=1}^{n} x_j \sum_{i=1}^{m} a_{ij} u_i = \sum_{i=1}^{m} y_i u_i \tag{6.1-6}$$

$$\sum_{i=1}^{m} u_i \sum_{j=1}^{n} a_{ij} x_j = \sum_{i=1}^{m} y_i u_i \tag{6.1-7}$$

$$\sum_{i=1}^{m} u_i \left(\sum_{j=1}^{n} a_{ij} x_j - y_i \right) = 0 \tag{6.1-8}$$

$$\sum_{j=1}^{n} a_{ij} x_j = y_i \tag{6.1-9}$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{6.1-10}$$

$$\mathbf{Ax} = \mathbf{y} \tag{6.1-11}$$

6.2 Change of Basis

$A{:}X \to Y$: linear transformation

$\{v_1, v_2, \ldots, v_n\}$: basis for vector space X

$\{u_1, u_2, \ldots, u_m\}$: basis for vector space Y

For $x \in X$

$$x = \sum_{i=1}^{n} x_i v_i \tag{6.2-1}$$

For $y \in Y$

$$y = \sum_{i=1}^{m} y_i u_i \tag{6.2-2}$$

$$A(x) = y \tag{6.2-3}$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{6.2-4}$$

$$\mathbf{Ax} = \mathbf{y} \tag{6.2-5}$$

$A{:}X \rightarrow Y$: linear transformation

$\{t_1, t_2, \ldots, t_n\}$: new basis for vector space X

$\{w_1, w_2, \ldots, w_m\}$: new basis for vector space Y

For $x \in X$

$$x = \sum_{i=1}^{n} x_i' t_i \tag{6.2-6}$$

For $y \in Y$

$$y = \sum_{i=1}^{m} y_i' w_i \tag{6.2-7}$$

$$\begin{bmatrix} a_{11}' & a_{12}' & \cdots & a_{1n}' \\ a_{21}' & a_{22}' & \cdots & a_{2n}' \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}' & a_{m2}' & \cdots & a_{mn}' \end{bmatrix} \begin{bmatrix} x_1' \\ x_2' \\ \vdots \\ x_n' \end{bmatrix} = \begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_n' \end{bmatrix} \tag{6.2-8}$$

$$\mathbf{A}'\mathbf{x}' = \mathbf{y}' \tag{6.2-9}$$

$$t_i = \sum_{j=1}^{n} t_{ji} v_j \tag{6.2-10}$$

$$w_i = \sum_{j=1}^{m} w_{ji} u_j \tag{6.2-11}$$

$$\mathbf{t}_i = \begin{bmatrix} t_{1i} \\ t_{2i} \\ \vdots \\ t_{ni} \end{bmatrix}, \mathbf{w}_i = \begin{bmatrix} w_{1i} \\ w_{2i} \\ \vdots \\ w_{mi} \end{bmatrix} \tag{6.2-12}$$

$$\mathbf{B}_t = [\mathbf{t}_1 \ \ \mathbf{t}_2 \ \ \ldots \ \ \mathbf{t}_n] \tag{6.2-13}$$

$$\mathbf{x} = x_1'\mathbf{t}_1 + x_2'\mathbf{t}_2 + \cdots + x_n'\mathbf{t}_n = \mathbf{B}_t\mathbf{x}' \tag{6.2-14}$$

$$\mathbf{B}_w = [\mathbf{w}_1 \ \ \mathbf{w}_2 \ \ \ldots \ \ \mathbf{w}_m] \tag{6.2-15}$$

$$\mathbf{y} = \mathbf{B}_w\mathbf{y}' \tag{6.2-16}$$

$$\mathbf{A}\mathbf{B}_t\mathbf{x}' = \mathbf{B}_w\mathbf{y}' \tag{6.2-17}$$

$$\left[\mathbf{B}_w^{-1}\mathbf{A}\mathbf{B}_t\right]\mathbf{x}' = \mathbf{y}' \tag{6.2-18}$$

$$\mathbf{A}' = \mathbf{B}_w^{-1}\mathbf{A}\mathbf{B}_t \tag{6.2-19}$$

6.3 Eigenvalues and Eigenvectors

Definition: Consider a linear transformation $A:X \rightarrow X$. Those vectors $z \in X$ that are not equal to zero and those scalars λ that satisfy.

$$A(z) = \lambda z \qquad (6.3\text{-}1)$$

are called **eigenvectors** (z) and **eigenvalues** (λ), respectively.

- An eigenvector of a given transformation represents a direction, such that any vector in that direction, when transformed, will continue to point in the same direction, but will be scaled by the eigenvalue.

$$\mathbf{Az} = \lambda\mathbf{z} \qquad (6.3\text{-}2)$$

$$[\mathbf{A}-\lambda\mathbf{I}]\mathbf{z} = 0 \qquad (6.3\text{-}3)$$

This means that the columns of $[\mathbf{A}-\lambda\mathbf{I}]$ are dependent, and therefore the determinant of this matrix must be zero:

$$|\mathbf{A}-\lambda\mathbf{I}| = 0 \qquad (6.3\text{-}4)$$

6.3.1 Diagonalization

$A{:}X \rightarrow X$: linear transformation

$\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\}$: independent eigenvectors of a matrix **A**

$\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$: eigenvalues of the matrix **A**

$$\mathbf{B} = [\mathbf{z}_1 \quad \mathbf{z}_2 \quad \ldots \ \mathbf{z}_n] \tag{6.3.1-1}$$

$$\mathbf{AB} = \mathbf{A}[\mathbf{z}_1 \quad \mathbf{z}_2 \quad \ldots \ \mathbf{z}_n] = [\lambda_1\mathbf{z}_1 \quad \lambda_2\mathbf{z}_2 \quad \ldots \ \lambda_n\mathbf{z}_n] = [\mathrm{z1} \quad \mathrm{z2} \quad \ldots \mathrm{zn}]\begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} = \mathbf{B}\begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \tag{6.3.1-2}$$

$$\mathbf{B}^{-1}\mathbf{AB} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \tag{6.3.1-3}$$

From

$$\mathbf{A}' = \mathbf{B}_w^{-1}\mathbf{AB}_t \tag{6.3.1-4}$$

- For $A{:}X \rightarrow X$, if both domain and range are changed into independent eigenvectors basis set, the matrix representation is diagonal.

Example:

$$\mathbf{A} = \begin{bmatrix} 2 & -2 \\ -1 & 3 \end{bmatrix} \quad (6.3.1\text{-}5)$$

$$|\mathbf{A}\text{-}\lambda\mathbf{I}| = \begin{vmatrix} 2-\lambda & -2 \\ -1 & 3-\lambda \end{vmatrix} = \lambda^2 - 5\lambda + 4 = (\lambda - 1)(\lambda - 4) = 0 \quad (6.3.1\text{-}6)$$

$$[\mathbf{A}\text{-}\lambda\mathbf{I}]\mathbf{z} = \begin{bmatrix} 2-\lambda & -2 \\ -1 & 3-\lambda \end{bmatrix}\mathbf{z} = 0 \quad (6.3.1\text{-}7)$$

For $\lambda = \lambda_1 = 1$,

$$\begin{bmatrix} 1 & -2 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} z_{11} \\ z_{21} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (6.3.1\text{-}8)$$

$$z_{11} = 2z_{21} \quad (6.3.1\text{-}9)$$

Select

$$\mathbf{z}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad (6.3.1\text{-}10)$$

For $\lambda = \lambda_2 = 4$,

$$\begin{bmatrix} -2 & -2 \\ -1 & -1 \end{bmatrix}\begin{bmatrix} z_{12} \\ z_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (6.3.1\text{-}11)$$

$$z_{12} = -z_{22} \quad (6.3.1\text{-}12)$$

Select

$$\mathbf{z}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{6.3.1-13}$$

$$\mathbf{A}' = \mathbf{B}^{-1}\mathbf{A}\mathbf{B} \tag{6.3.1-14}$$

$$\mathbf{B} = [\mathbf{z}_1 \quad \mathbf{z}_2] = \begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix} \tag{6.3.1-15}$$

$$\mathbf{B}^{-1} = \begin{bmatrix} 1/3 & 1/3 \\ 1/3 & -2/3 \end{bmatrix} \tag{6.3.1-16}$$

$$\mathbf{A}' = \mathbf{B}^{-1}\mathbf{A}\mathbf{B} = \begin{bmatrix} 1/3 & 1/3 \\ 1/3 & -2/3 \end{bmatrix}\begin{bmatrix} 2 & -2 \\ -1 & 3 \end{bmatrix}\begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{6.3.1-17}$$

If $\mathbf{x}' = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$,

$$\mathbf{y}' = \mathbf{A}'\mathbf{x}' = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 8 \end{bmatrix} \tag{6.3.1-18}$$

$$\mathbf{x} = \mathbf{B}_t\mathbf{x}' = \begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ -1 \end{bmatrix} \tag{6.3.1-19}$$

$$\mathbf{y} = \mathbf{B}_w\mathbf{y}' = \begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 8 \end{bmatrix} = \begin{bmatrix} 10 \\ -7 \end{bmatrix} = \mathbf{A}\mathbf{x} = \begin{bmatrix} 2 & -2 \\ -1 & 3 \end{bmatrix}\begin{bmatrix} 4 \\ -1 \end{bmatrix} \tag{6.3.1-20}$$

7 Supervised Hebbian Learning

"*When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.*"

7.1 Linear Associator

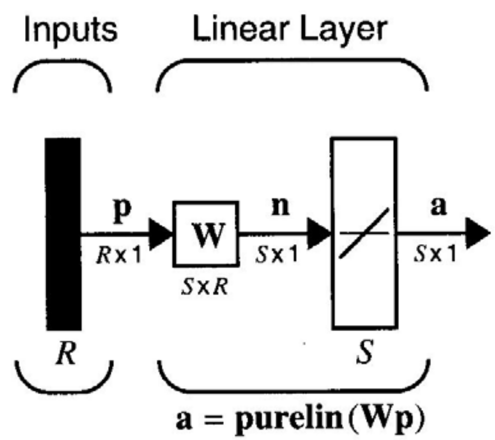

Figure 7.1-1 Linear Associator

$$\mathbf{a} = \mathbf{W}\mathbf{p} \tag{7.1-1}$$

$$a_i = \sum_{j=1}^{R} w_{ij} p_j \tag{7.1-2}$$

- The linear associator is an example of a type of neural network called an associative memory.
- If the network receives an input $\mathbf{p} = \mathbf{p}_q$ then it should produce an output $\mathbf{a} = \mathbf{t}_q$, for $q = 1, 2, \ldots, Q$.
- If the input is changed slightly (i,e., $\mathbf{p} = \mathbf{p}_q + \delta$) then the output should only be changed slightly (i,e., $\mathbf{a} = \mathbf{t}_q + \varepsilon$).

7.2 The Hebb Rule

- If two neurons on either side of a synapse are activated simultaneously, the strength of the synapse will increase.
- The connection (synapse) between input p_j and output a_i is the weight w_{ij}.
- If a positive p_j produces a positive a_i then w_{ij} should increase.

For Hebb's unsupervised learning rule,

$$w_{ij}^{new} = w_{ij}^{old} + \alpha f_i(a_{iq})g_j(p_{jq}) \quad (7.2\text{-}1)$$

p_{jq}: the jth element of the qth input vector $\mathbf{p}_q$;

a_{iq}: the ith element of tie network output when the qth input vector is presented to the network

α: a positive constant, called the learning rate.

For Hebb's simplified unsupervised learning rule,

$$w_{ij}^{new} = w_{ij}^{old} + \alpha a_{iq} p_{jq} \quad (7.2\text{-}2)$$

For Hebb's supervised learning rule with learning rate = 1,

$$w_{ij}^{new} = w_{ij}^{old} + t_{iq} p_{jq} \quad (7.2\text{-}3)$$

t_{iq}: the ith element of the qth target vector $\mathbf{t}_q$

In vector notation,

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{t}_q \mathbf{p}_q^T \quad (7.2\text{-}4)$$

If we assume that the weight matrix is initialized to zero and then each of the Q input/output pairs, $\{\mathbf{p}_1, \mathbf{t}_1\}$, $\{\mathbf{p}_2, \mathbf{t}_2\}$, …, $\{\mathbf{p}_Q, \mathbf{t}_Q\}$, are applied once,

$$\mathbf{W} = \mathbf{t}_1\mathbf{p}_1^T + \mathbf{t}_2\mathbf{p}_2^T + \cdots + \mathbf{t}_Q\mathbf{p}_Q^T = \sum_{q=1}^{Q} \mathbf{t}_q\mathbf{p}_q^T \qquad (7.2\text{-}5)$$

$$\mathbf{W} = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_Q \end{bmatrix} \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_Q^T \end{bmatrix} = \mathbf{T}\mathbf{P}^T \qquad (7.2\text{-}6)$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_Q \end{bmatrix}, \ \mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_Q \end{bmatrix} \qquad (7.2\text{-}7)$$

7.2.1 Performance Analysis

$\mathbf{p}_q$ vectors are orthonormal (orthogonal and unit length),

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \left(\sum_{q=1}^{Q} \mathbf{t}_q \mathbf{p}_q^T\right)\mathbf{p}_k = \sum_{q=1}^{Q} \mathbf{t}_q (\mathbf{p}_q^T \mathbf{p}_k) \tag{7.2.1-1}$$

$$(\mathbf{p}_q^T \mathbf{p}_k) = 1 \text{ when } q = k \text{ and } (\mathbf{p}_q^T \mathbf{p}_k) = 0 \text{ when } q \neq k \tag{7.2.1-2}$$

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \mathbf{t}_k \tag{7.2.1-3}$$

$\mathbf{p}_q$ vectors are normalized but not orthogonal,

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \mathbf{t}_k + \sum_{q \neq k} \mathbf{t}_q (\mathbf{p}_q^T \mathbf{p}_k) \tag{7.2.1-4}$$

$\mathbf{p}_q$ vectors are not normalized and not orthogonal,

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \mathbf{t}_k \left\|\mathbf{p}_k\right\|^2 + \sum_{q \neq k} \mathbf{t}_q (\mathbf{p}_q^T \mathbf{p}_k) \tag{7.2.1-5}$$

7.3 Pseudoinverse Rule

$$\mathbf{WP} = \mathbf{T} \tag{7.3-1}$$

$$\mathbf{T} = \begin{bmatrix}\mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_Q\end{bmatrix},\ \mathbf{P} = \begin{bmatrix}\mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_Q\end{bmatrix} \tag{7.3-1}$$

Not all the cases,

$$\mathbf{W}=\mathbf{TP}^{-1} \tag{7.3-3}$$

Pseudoinverse rule:

$$\mathbf{W} = \mathbf{TP}^{-1} = \mathbf{TP}^{-1}\mathbf{I} = \mathbf{TP}^{-1}(\mathbf{P}^T)^{-1}(\mathbf{P}^T) = \mathbf{T}(\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T = \mathbf{TP}^+ \tag{7.3-4}$$

$\mathbf{P}^+$: the Moore-Penrose pseudoinverse.

The pseudoinverse of a real matrix **P** is the unique matrix that satisfies

$$\mathbf{PP}^+\mathbf{P} = \mathbf{P} \text{ and}$$
$$\mathbf{P}^+\mathbf{PP}^+ = \mathbf{P}^+ \text{ and}$$
$$\mathbf{P}^+\mathbf{P} = (\mathbf{P}^+\mathbf{P})^T \text{ and}$$
$$\mathbf{PP}^+ = (\mathbf{PP}^+)^T \tag{7.3-5}$$

When the number, R, of rows of **P** is greater than the number of columns, Q, of **P**, and the columns of **P** are independent, then the pseudoinverse can be computed by

$$\mathbf{P}^+=(\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T \tag{7.3-6}$$

7.4 Application in Autoassociate Memory

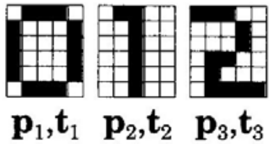

Figure 7.4-1 The Patterns of 0, 1, and 2

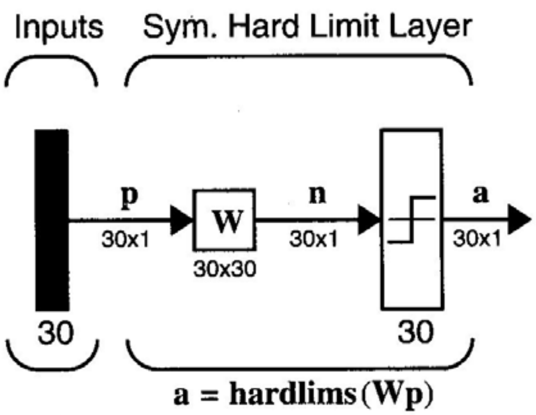

Figure 7.4-2 Autoassociative Network for Digit Recognition

$$\mathbf{p}_1 = [-1\ 1\ 1\ 1\ 1\ -1\ 1\ -1\ -1\ -1\ -1\ 1\ 1\ -1\ \ldots\ 1\ -1]^T \quad (7.4\text{-}1)$$

$$\mathbf{W} = \mathbf{p}_1\mathbf{p}_1^T + \mathbf{p}_2\mathbf{p}_2^T + \mathbf{p}_3\mathbf{p}_3^T \quad (7.4\text{-}2)$$

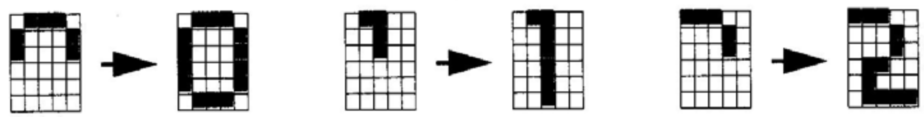

Figure 7.4-3 Recovery of 50% Occluded Patterns

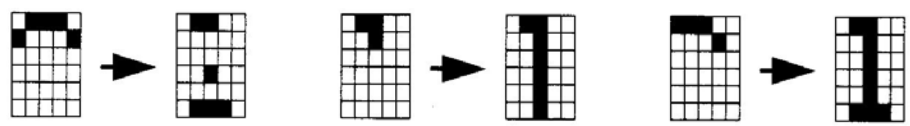

Figure 7.4-4 Recovery of 67% Occluded Patterns

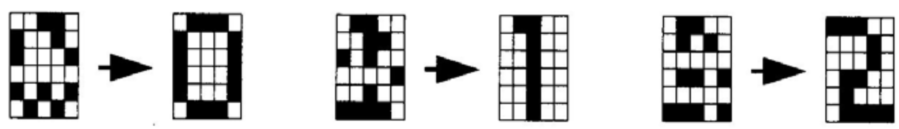

Figure 7.4-5 Recovery of Noisy Patterns

7.5 Variations of Hebbian Learning

For Hebb's general supervised learning rule,

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T \tag{7.5-1}$$

For Hebb's supervised learning rule with decay term,

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T - \gamma \mathbf{W}^{old} = (1-\gamma)\mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T \tag{7.5-2}$$

where γ: decay rate, a positive constant less than one.

For delta rule or Widrow-Hoff algorithm,

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha(\mathbf{t}_q - \mathbf{a}_q)\mathbf{p}_q^T \tag{7.5-3}$$

- The advantage of the delta rule is that it can update the weights after each new input pattern is presented, whereas the pseudoinverse rule computes the weights in one step, after all of the input/target pairs are known. This sequential updating allows the delta rule to adapt to a changing environment.

For Hebb's unsupervised learning rule,

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{a}_q \mathbf{p}_q^T \tag{7.5-4}$$