Artificial Neural Networks

INTRODUCTION TO LEARNING ALGORITHMS IN NEURAL NETWORKS

Topics

1. Introduction to Artificial Neural Networks
   a) Perceptron Learning
   b) Habbian Rule and Supervised Hebbian Learning
   c) Examples

Bayesian Classifier Example

Assume a Bayesian classifier should classify two class of objects. The classifier uses two features (area and perimeter for instance)

We have 5 sample from each class

Estimate a-priori probability

Use

$$P(\omega_i) = \frac{n_i}{\sum_{j=1}^{c} n_j}.$$

Estimate PDF Parameters

Find mean and covariance matrix

Use (1/n-1)sum(x_i-x’)(y_i-y’) for covariance

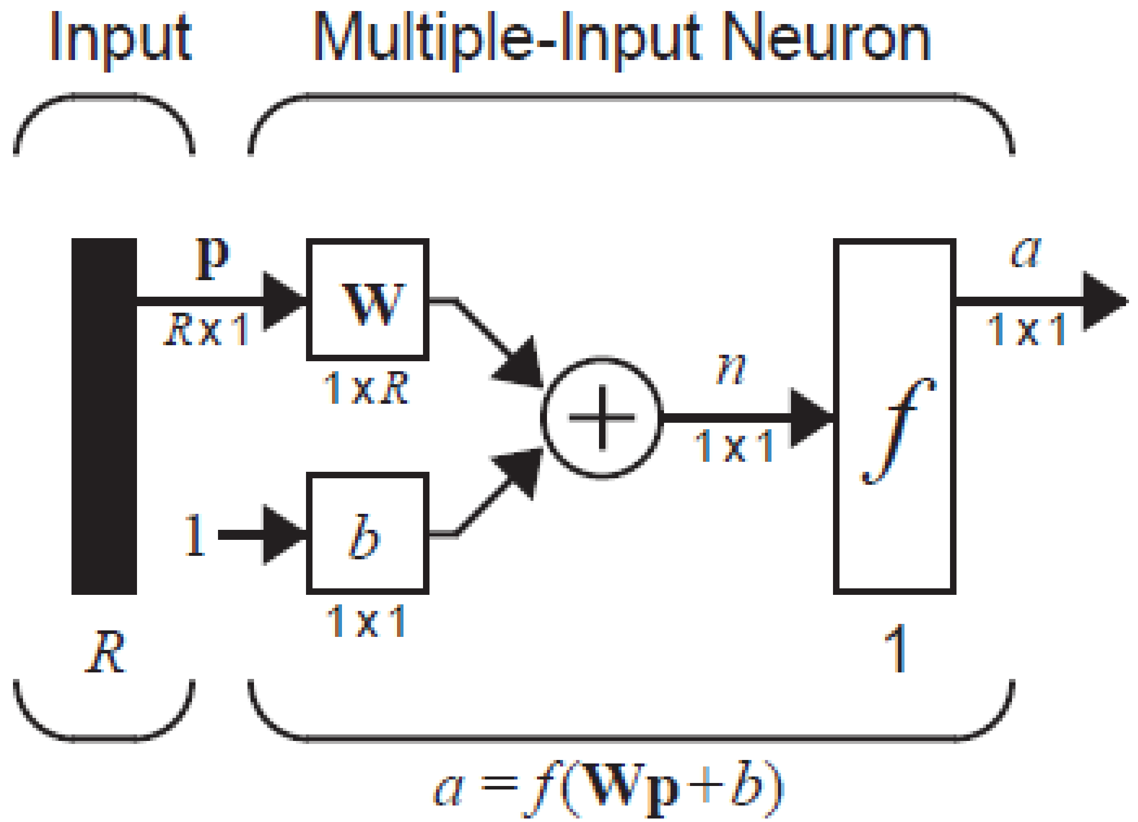
Classify using Bayesian

Use:

$$P(\omega_j / x) = \frac{p(x / \omega_j) P(\omega_j)}{p(x)} = \frac{likelihood \;\times\; prior}{evidence}$$

Neural Networks (recap)

Structure of a Neuron



Matrix Representation of a Multiple Input Neuron

A multiple input neuron can be represented by:

$a = f(WP + b)$

$W=[w_1, w_2, w_3,\ldots, w_R]$

$P=[p_1, p_2, p_3, \ldots, p_R]^T$

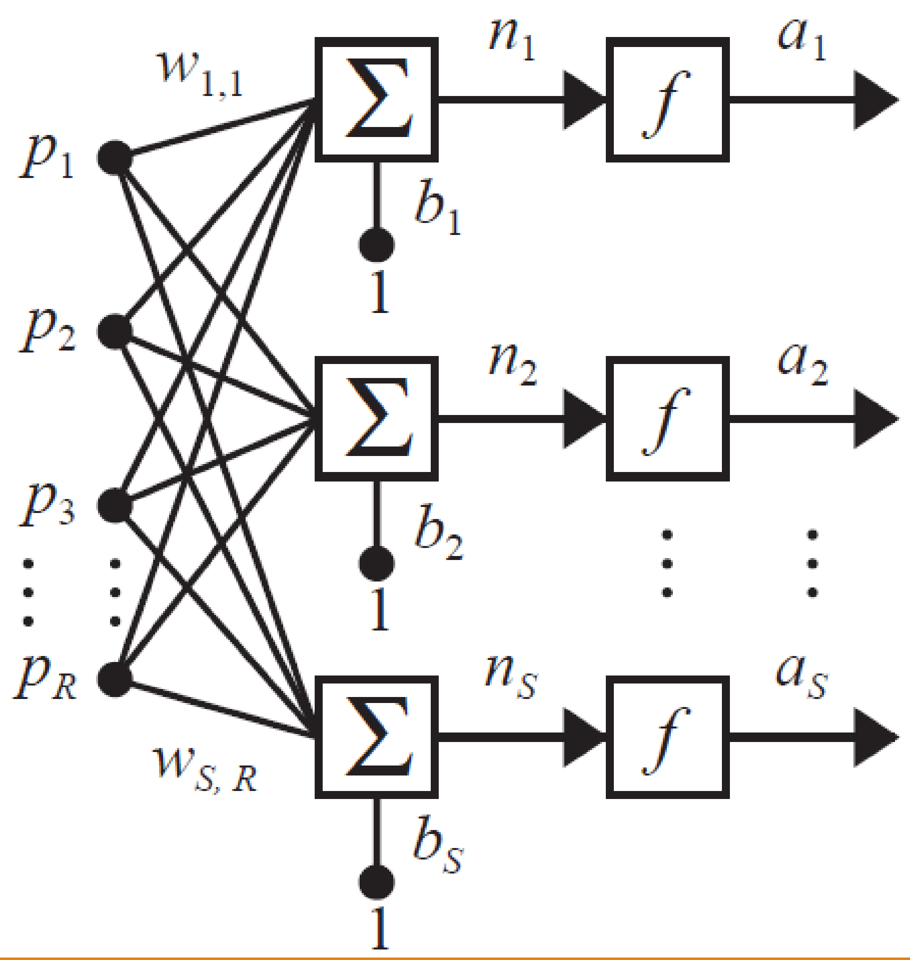
Neural Network Model

An artificial neural network is an information processing model, which is composed of a large number of highly interconnected processing elements (neurons).

Artificial neural networks learn by example.

An artificial neural network is configured for a specific problem

Neural Network Model



Matrix Representation of a Single-Layer Neural Network

A single layer neural network can be represented by:

a = f(WP + b)

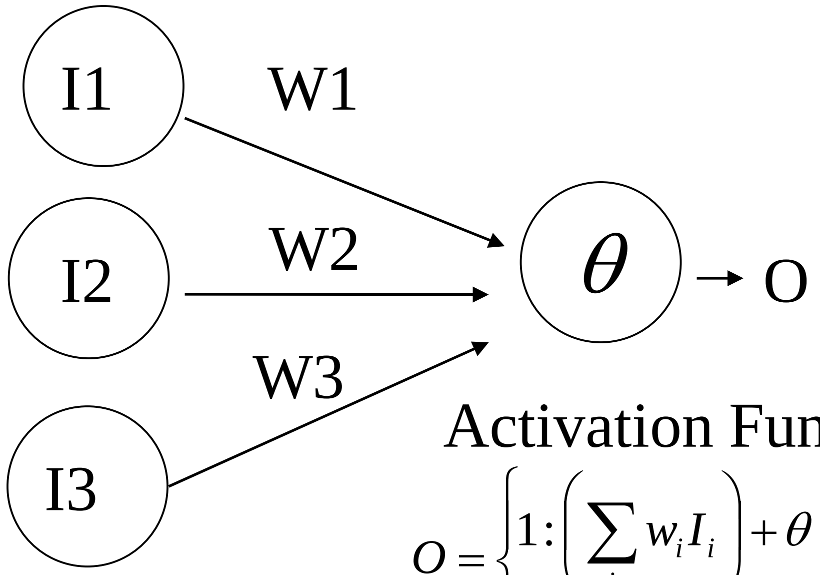
$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_S \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_S \end{bmatrix}$$

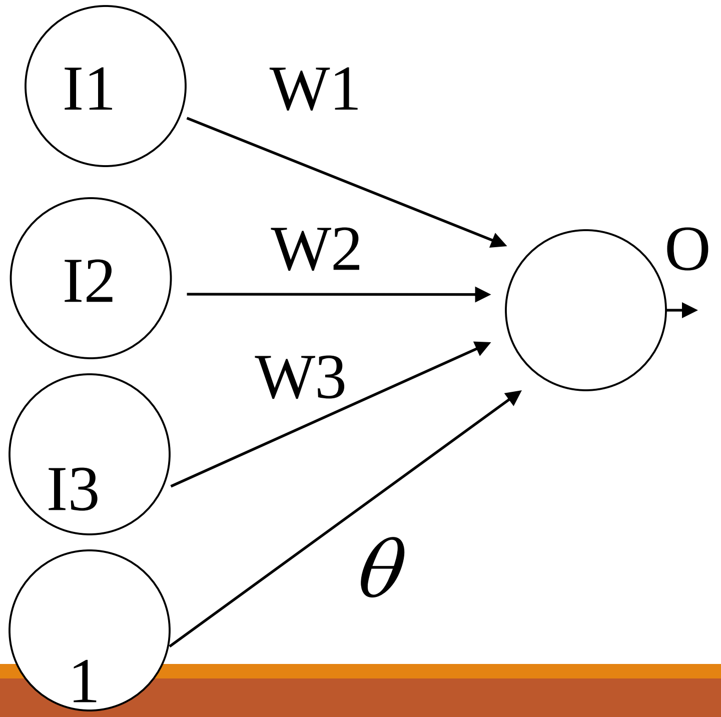
Perceptron

Initial proposal of connectionist networks Rosenblatt, 50’s and 60’s

Essentially a linear discriminant composed of nodes, weights



or



Activation Function

$$O = \begin{cases} 1 : \left(\sum_i w_i I_i \right) + \theta > 0 \\ 0 : otherwise \end{cases}$$

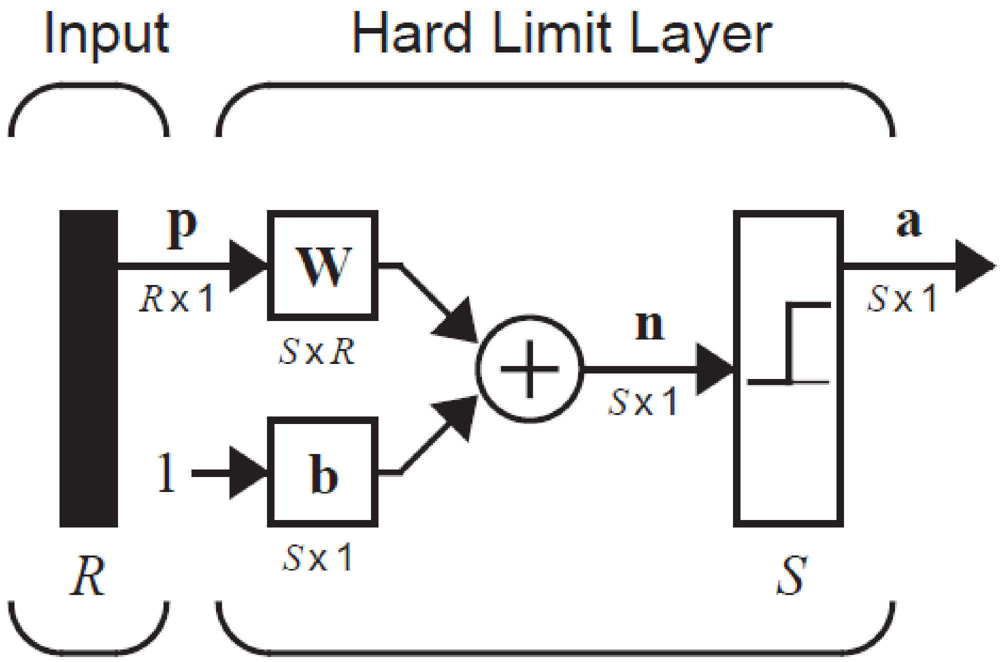
Perceptron

The activation function is a threshold (hard limit) function.

A perceptron a linear discriminant function dividing the input space into two areas.

The boundary is given by $W^T P + b = 0$

Perceptron



$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp}+\mathbf{b})$$

Learning Rules

1. Supervised Learning

   The network is provided with a set of input/target examples

2. Unsupervised Learning

   Only inputs are available. The network learns to cluster the inputs

3. Reinforcement Learning

   The network is provided with input and a set of grades that shows how good its performance is

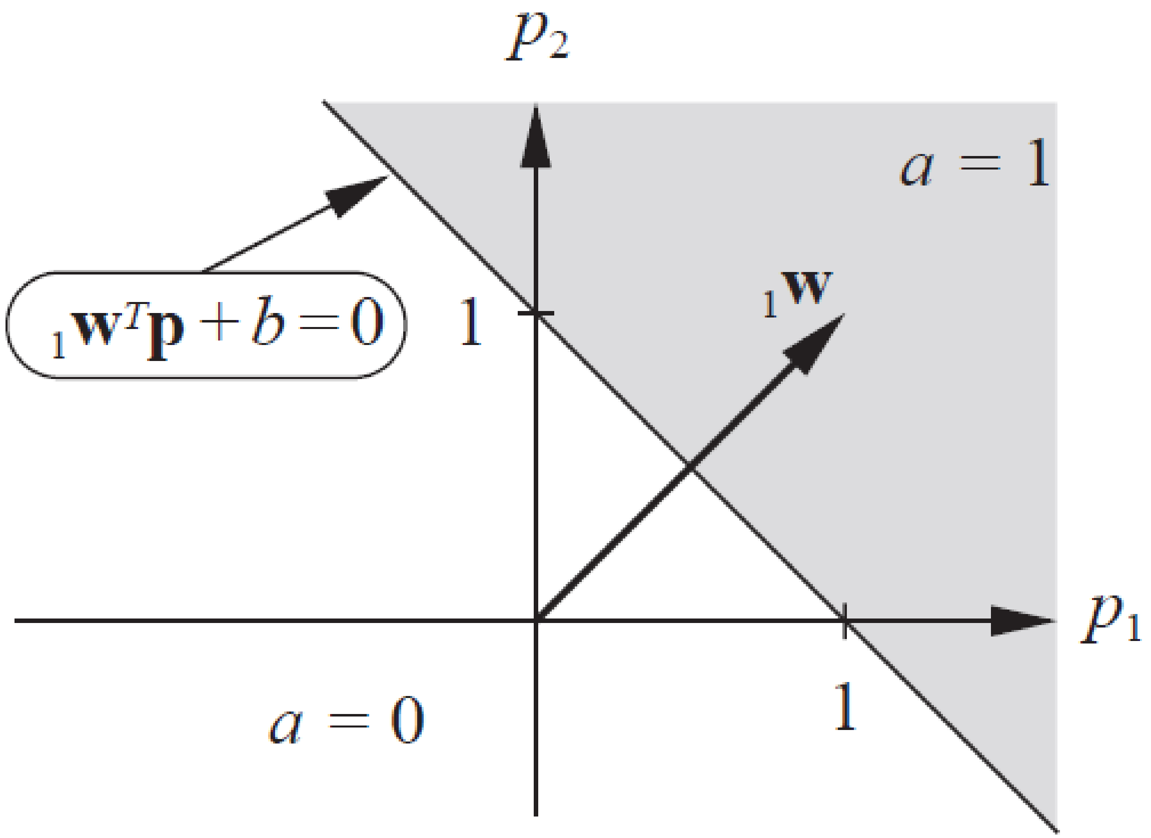
Perceptron Learning

This learning rule is an example of supervised learning, in which the learning rule is provided with a set of examples of proper network behavior.

Perceptron Classifier

Decision boundary is the weighted sum of the input value(s) where the net input is zero.

For example assuming $w_{1,1} = 1$, $w_{1,2} = 1$, $b = -1$

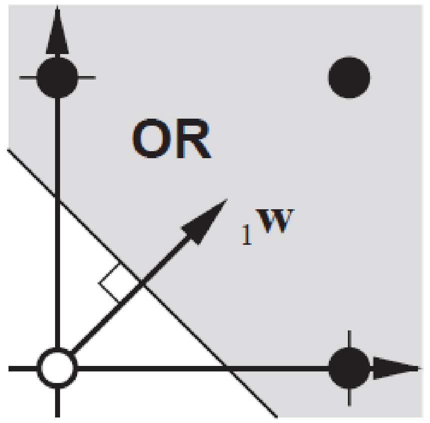


Example

Decision boundary with a single perceptron

Consider OR operation, weight vector is W=[0.5   0.5]

Bias can be found from a point on the decision boundary



Multiple Neuron Perceptron

Each neuron has its own decision boundary

$$W^TP+b = 0$$

This results in 2^S different sub-space

Single Perceptron Learning

Perceptron learn by example (supervised learning)

Assume:

- a is the out put of the perceptron
- P is the input vector
- W is the weight vector
- b is bias
- t is true value

Single Perceptron Learning

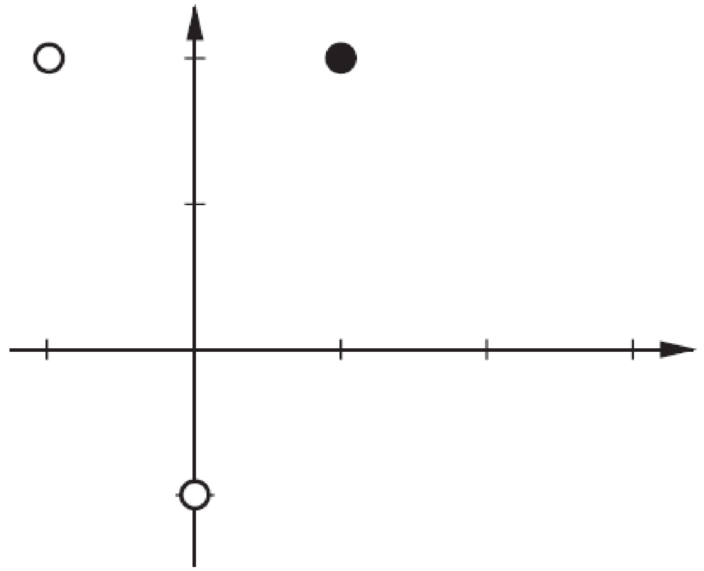
Learning rules:

- If the output is 0 but the expected output is 1 add P to W
- If the output is 1 but the expected output is 0 subtract P from W
- Otherwise (If the output is equal to the expected output) don’t change weights
- Repeat for all inputs until converge
- (The algorithm converges if a solution exists)

Single Perceptron Learning

Example:

Case 1: Without bias

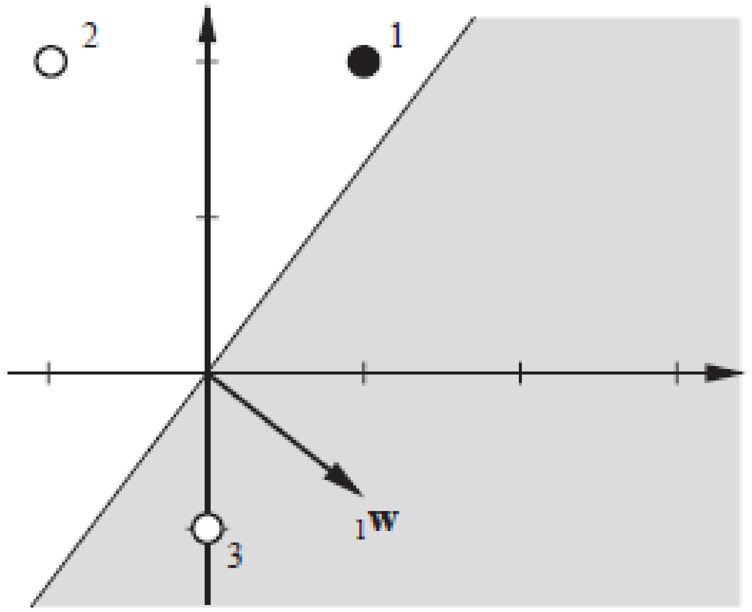


$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\2\end{bmatrix}, t_1 = 1\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}-1\\2\end{bmatrix}, t_2 = 0\right\} \quad \left\{\mathbf{p}_3 = \begin{bmatrix}0\\-1\end{bmatrix}, t_3 = 0\right\}$$

Single Perceptron Learning

Random initial weight:

$$ {}_1\mathbf{w} = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} $$



Trying P1, the output is 0 (incorrect)

Multiple Neuron Perceptron

The multiple neuron perceptron learning rule is a generalization of the single neuron perceptron.

The learning rule is applied to each neuron separately

$${}_i\mathbf{w}^{new} = {}_i\mathbf{w}^{old} + e_i\mathbf{p} .$$

Hebbian Learning

Hebb’s postulate:

When an axon of a neuron is close enough to excite another neuron, and if the first neuron repeatedly takes part in firing the second neuron, the efficiency of the first neuron in firing the second neuron increases

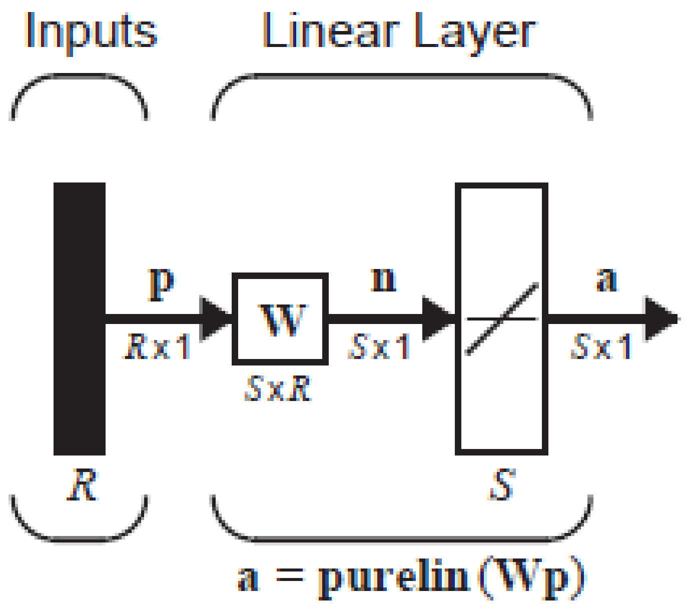
Linear Association

The linear association is an example of a type of neural network called an *associative memory*. The task of an associative memory is to learn Q pairs of prototype input/output vectors:

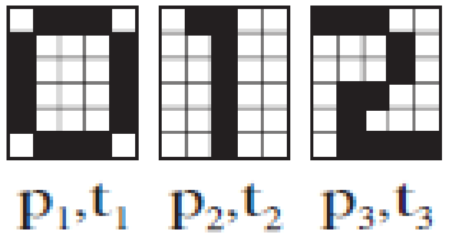
$\{a_1, t_1\}, \{a_2, t_2\}, \ldots . \{a_Q, t_Q\}$

In other words, if the network receives an input $p = a_i$ then it should produce t_i as its output

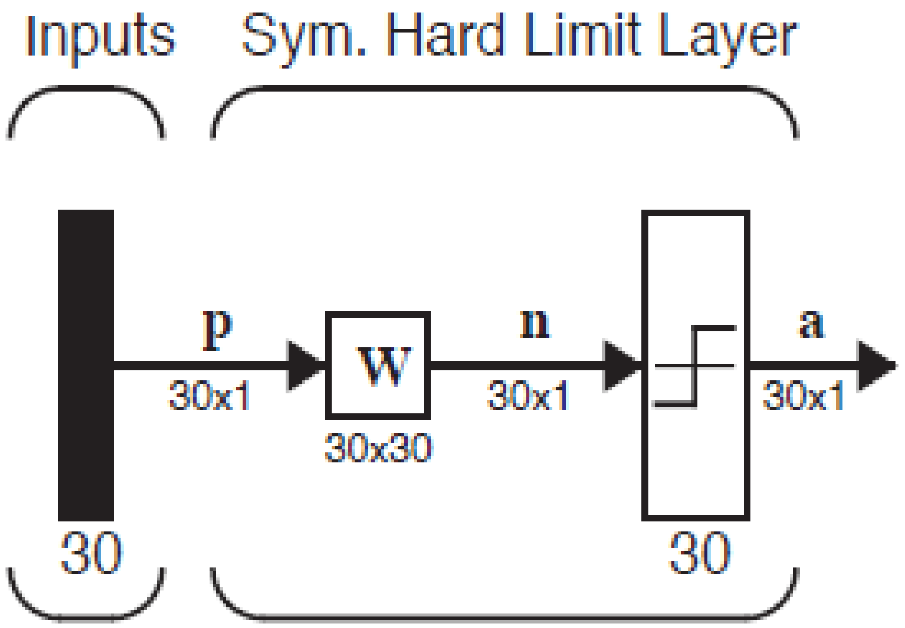
Sample Linear Associator



Example



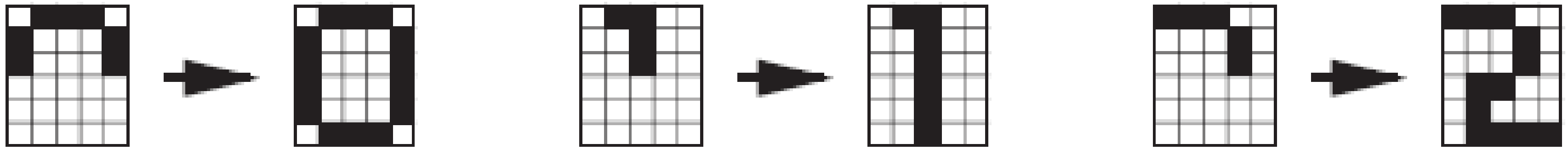
$$\mathbf{p}_1 = \begin{bmatrix} -1 & 1 & 1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & \ldots & 1 & -1 \end{bmatrix}^T$$



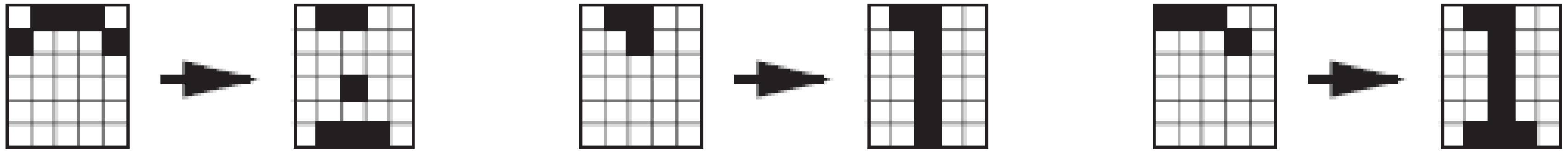
$$\mathbf{a} = \mathbf{hardlims}(\mathbf{W}\mathbf{p})$$

$$\mathbf{W} = \mathbf{p}_1\mathbf{p}_1^T + \mathbf{p}_2\mathbf{p}_2^T + \mathbf{p}_3\mathbf{p}_3^T$$

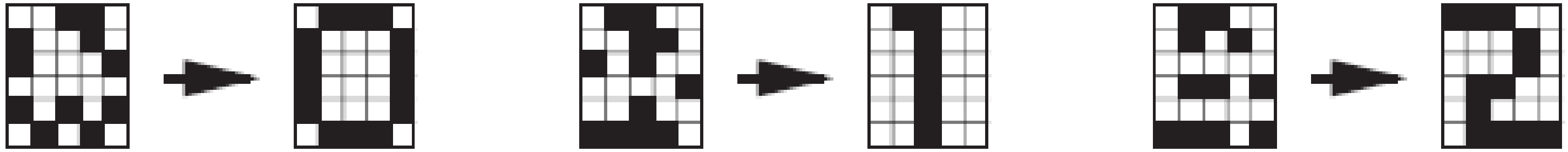
50% Occluded



67% Occluded



Noisy Patterns (7 pixels)



Hebb’s Learning Rule

Hebb’s learning rule states that the weight matrix should be updated by adding true values x input pattern to it

Matrix Form:

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{t}_q \mathbf{p}_q^T$$

Hebb’s Learning Rule

Assuming zero as the initial values of the weights, the updated weight values become:

$W = t_1 p_1^T + t_2 p_2^T + .. + t_Q p_Q^T$

Example

$$\left\{\mathbf{p}_1 = \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right\} \qquad \left\{\mathbf{p}_2 = \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ -0.5 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\}.$$

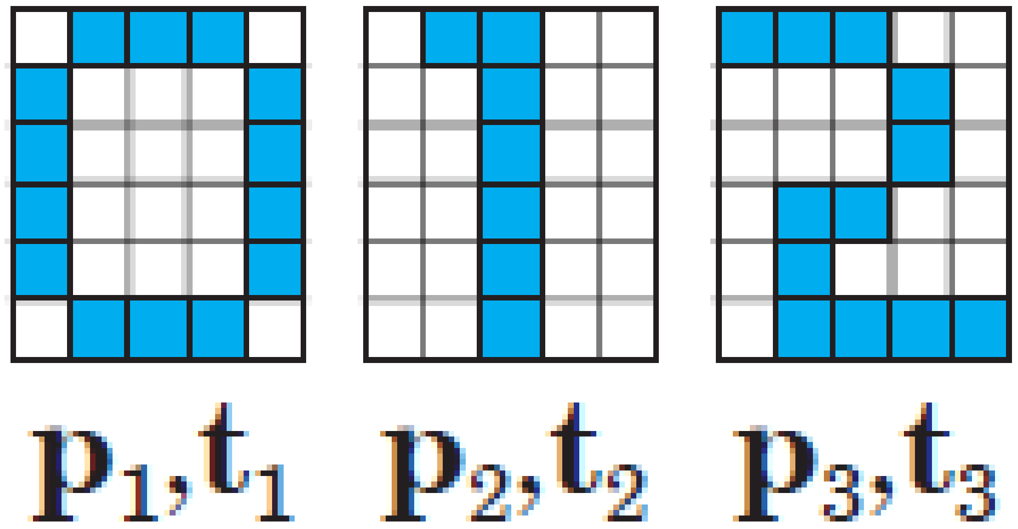
$$\mathbf{W} = \mathbf{T}\mathbf{P}^T = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}.$$

Example

Using autoassociative memory for pattern recognition:

In an *autoassociative memory* the desired output vector is equal to the input vector

Input/output of the network


$\mathbf{p}_1, \mathbf{t}_1$
$\mathbf{p}_2, \mathbf{t}_2$
$\mathbf{p}_3, \mathbf{t}_3$

Assignment

Design and train an associative memory to identify the three patterns of digits 0,1, and 2

Add noise to the input patterns and test the behavior of the network

Use MATLAB for coding

Due date: 15-11-2017