

Lecture 6.  
Learning (III):  
Associative Learning and  
Principal Component Analysis

# Outline

- Associative Learning:  
Hebbian Learning
- Use Associative Learning to compute Principal Component

# Associative (Hebbian) Learning

- A Learning rule postulated by Hebb (1949) on the associative learning at the cellular level.

$$\Delta w_{kj}(n) = w_{kj}(n+1) - w_{kj}(n) = \eta d_k(n)x_j(n)$$

or  $\Delta w(n) = \eta d_k(n)x(n)$

The change of synaptic weights is proportional to the correlation of the (desired) activation  $d_k(n)$  and the input  $x(n)$ .

- The weight vector  $w(n)$  converges to the cross-correlation of  $d_k(n)$  and  $x(n)$ :

$$w_{kj}(N) = w_{kj}(0) + \eta \sum_{n=0}^N d_k(n)x_j(n) \quad \text{or}$$

$$\mathbf{W}(N) = \mathbf{W}(0) + \eta \sum_{n=0}^N \underline{d}(n)\underline{x}^T(n)$$

# Potential Applications of Associative Learning

- Pattern Retrieval
  - Each pattern (input:  $x$ ) is associated with a label (output:  $d$ ).
  - The association information is stored in weights of a single layer network  $W$ .
  - When a test pattern is presented to the input, the label of a stored pattern that is closest to the test pattern should appear as the output.
  - This is the mechanism of an “associative memory”

# Linear Associative Memory

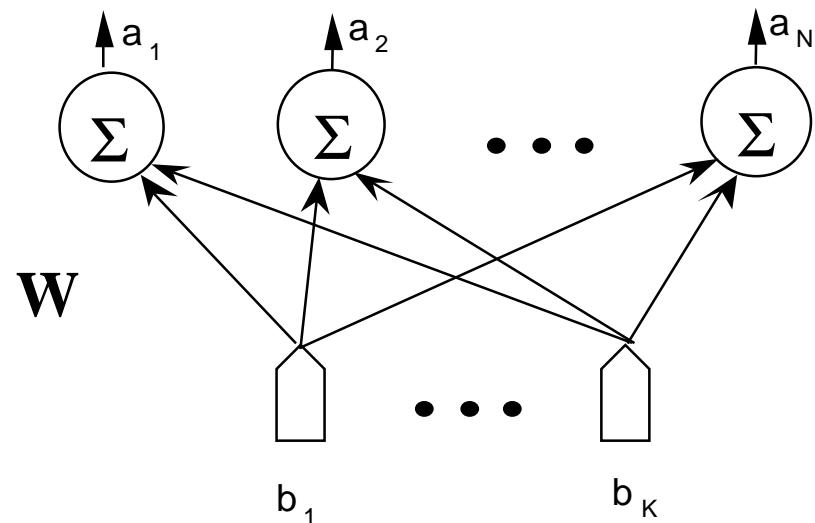
- M pairs of training vectors  
 $\{(\underline{b}_m, \underline{a}_m); m = 1, 2, \dots, M\}$ .
- Weight matrix:

$$\mathbf{W} = \sum_{m=1}^M \underline{a}_m \underline{b}_m^T = \mathbf{A} \cdot \mathbf{B}^T$$

- Retrieval:  $\underline{y} = \mathbf{W} \underline{x}$ .
- Let  $\underline{x} = \underline{b}_k$ ,

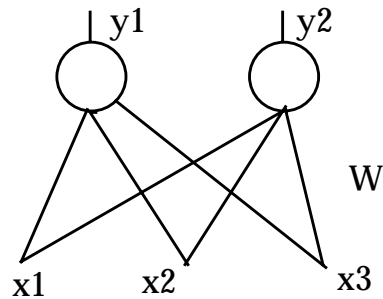
if  $\underline{b}_m^T \underline{b}_k = 0$  for  $m \neq k$ , then

$$\underline{y} = \underline{a}_k (\underline{b}_k^T \underline{b}_k) + \sum_{m=1, m \neq k}^M \underline{a}_m (\underline{b}_m^T \underline{b}_k) = \|\underline{b}_k\|^2 \underline{a}_k \propto \underline{a}_k$$



# Example

- Consider 3 input patterns:  $x(1) = [1 \ 2 \ -1]^T$ ,  $x(2) = [-1 \ 1 \ 1]^T$ ,  $x(3) = [1 \ 0 \ 1]^T$ , and their corresponding 2D labels:  $y(1) = [-1 \ 2]^T$ ,  $y(2) = [2 \ 4]^T$ ,  $y(3) = [1 \ -3]^T$ .
- Compute the cross correlation matrix using the Hebbian learning, with learning rate  $\eta = 1$ :



$$\mathbf{W} = \sum_{i=1}^3 y(i)x^T(i) = \begin{bmatrix} -2 & 0 & 4 \\ -5 & 8 & -1 \end{bmatrix}$$

- Recall: Present  $x(2)$ . Then  $y = W x(2) = [6 \ 12]^T = 3[2 \ 4]^T$  which is proportional to  $y(2)$ !

# Hebbian Learning Modifications

- Problem with Hebbian learning: when both  $d_k(n)$  and  $x_j(n)$  are positive, the weight  $w_{kj}(n)$  will grow unbound.
- Kohonen (1988) proposed to add a forgetting factor:

$$\Delta w_{kj}(n) = d_k(n) x_j(n) - \alpha d_k(n) w_{kj}(n)$$

or  $\Delta w_{kj}(n) = \alpha d_k(n)[c x_j(n) - w_{kj}(n)]$ ; where  $c = \eta/\alpha$

- Analysis: Suppose that  $x_j(n) > 0$  and  $d_k(n) > 0$ . When  $w_{kj}(n)$  increases until  $c x_j(n) < w_{kj}(n)$ ,  $\Delta w_{kj}(n) < 0$ . This, in turn, will reduce  $w_{kj}(n)$  for future  $n$ .
- It can be shown that when  $|\alpha d_k(n)| < 1$ ,  $|w_{kj}(n)|$  will be bounded if  $d_k(n)$  and  $x_j(n)$  are bounded.

# Principal Component Analysis (PCA)

- Principal component: part of signal (feature vectors) which consists of most energy of the signal!
- Also known as *Karhunen-Loeve Transformation*
- A linear transform of n-dim vector  $x$ ,  $Tx$  such that  $\|x - Tx\|^2$  is minimum among all n by n matrices  $T$  whose rank =  $p < n$ .
- Applications:  
Feature dimension reduction  
Data Compression

# PCA Method

Given  $\{x(k); 1 \leq k \leq K\}$ . Let dimension of  $x(k) = n$ .

1. Form a  $n$  by  $n$  Covariance matrix estimate

$$C = (1/K) \sum_{i=1}^K [x(k) - m][x(k) - m]^T \quad m = (1/K) \sum_{i=1}^K x(k)$$

2. Eigenvalue decomposition

$$C = \sum_{i=1}^n \lambda_i v_i v_i^T \quad \begin{array}{l} \lambda_1 > \lambda_2 > \dots > \lambda_n : \text{eigenvalues} \\ \{v_i; i = 1, n\} : \text{eigenvectors} \end{array}$$

3. Principal components

$$y(k) = V_p^T x(k) = [v_1 v_2 \dots v_p]^T x(k), \quad p \leq n.$$

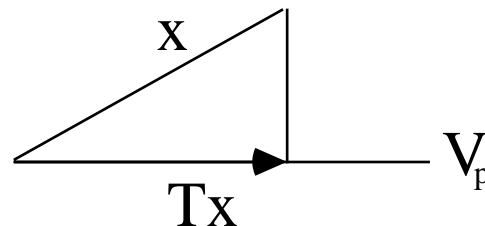
$$Tx(k) = V_p y(k) = V_p V_p^T x(k)$$

# Optimality of Principal Components

- Among all  $n$  by  $n$  matrices  $T'$  whose rank =  $p \leq n$ ,

$$\|x - Tx\|^2 = \sum_{i=p+1}^n \lambda_i \leq \|x - T'x\|^2$$

This is the orthogonality principle:



- $y(k)$  is the  $p$ -dimensional representation of  $x$  in the subspace spanned by columns of  $V_p$ .

$\|y(k)\| = \|Tx(k)\|$  is the length of the shadow of  $x(k)$ .

# Principal Component Network

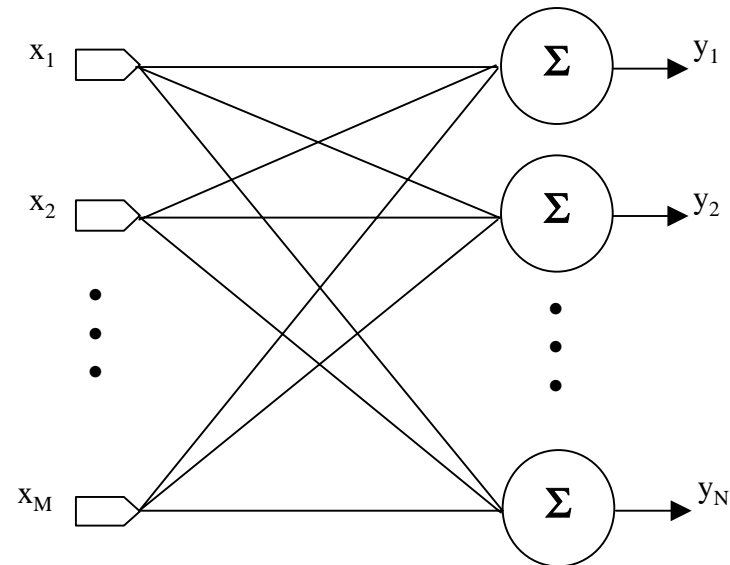
- Linear neuron model:

$$y_n(t) = \sum_{m=1}^M w_{nm}(t) x_m(t)$$

Choose  $\underline{w}$  to maximize

$$E\{\|y\|^2\} = \underline{w}^T E\{\underline{x} \underline{x}^T\} \underline{w} = \underline{w}^T \underline{C} \underline{w}$$

subject to  $\|\underline{w}\|^2 = 1$ .



- Generalized Hebbian learning rule:

$$\Delta w_{nm}(t) = w_{nm}(t+1) - w_{nm}(t) = \eta y_n(t) \left[ x_m(t) - \sum_{k=1}^n w_{km}(t) y_k(t) \right]$$