

Neural Networks

Chapter 2 Learning Process
by Alexander Shultz

“Learning is a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. The type of leaning is determined by the manner in which the parameter changes take place”

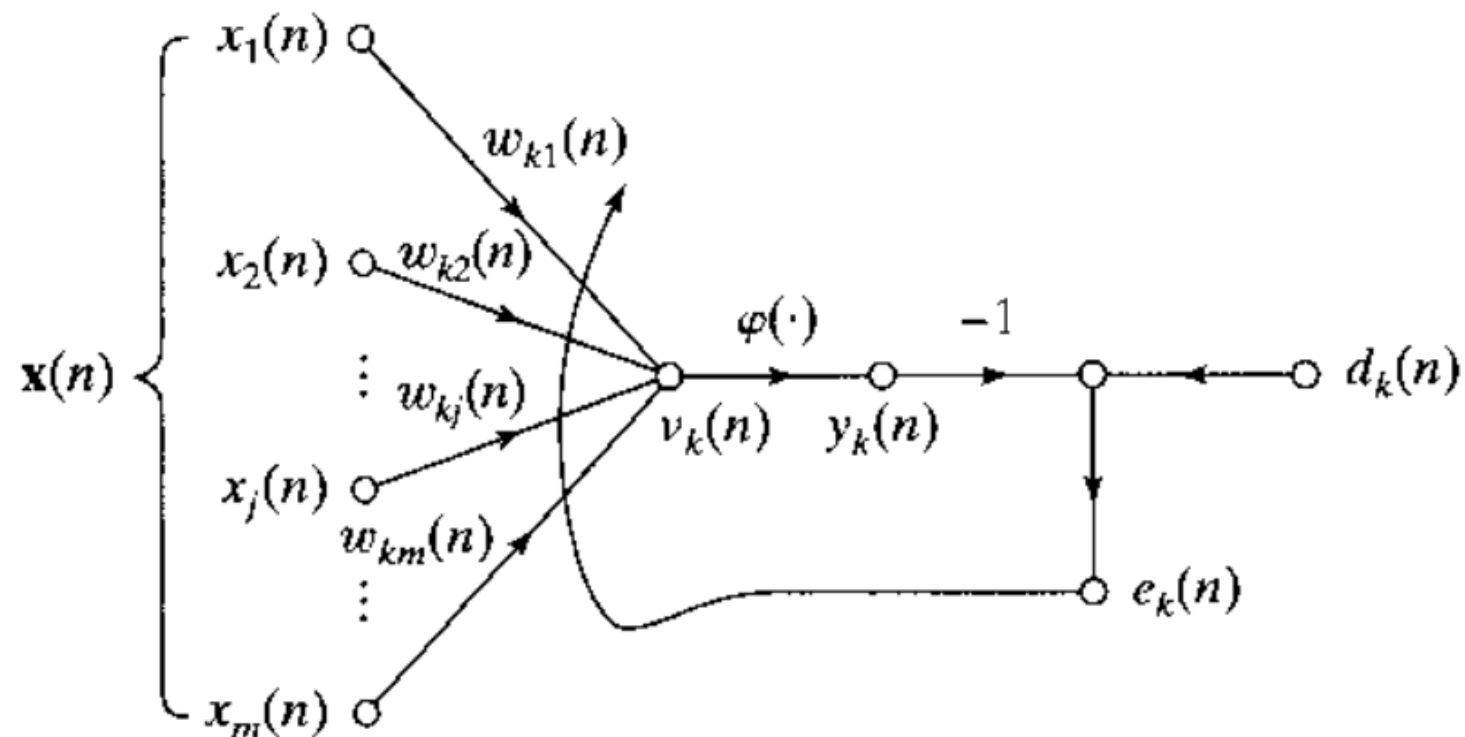
–Mendel and McClaren(1970)

Error-correction learning

$$e_k(n) = d_k(n) - y_k(n)$$

desired response

error signal



Error-correction learning

This objective is achieved by minimizing a cost function or index of performance

$$\mathcal{E}(n) = \frac{1}{2} e_k^2(n)$$

the instantaneous value of the error energy

Widrow-Hoff rule

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$$

$$w_{kj}(n + 1) = w_{kj}(n) + \Delta w_{kj}(n)$$

η plays a key role in determining the performance of error-correction learning in practice

Memory-based Learning

all of the past experiences are explicitly stored in a large memory of correctly classified input-output examples

$$\{(\mathbf{x}_i, d_i)\}_{i=1}^N$$

when classification of a test vector \mathbf{x}_{test} (not seen before) is required the algorithm respond by retrieving and analyzing the training data in a ‘local neighborhood’ of \mathbf{x}_{test}

1. Criterion used for defining the local neighborhood of \mathbf{x}_{test}
2. Learning applied to the training examples in the local neighborhood of \mathbf{x}_{test}

neighbor rule

$$\mathbf{x}'_N \in \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$$

$$\min_i d(\mathbf{x}_i, \mathbf{x}_{\text{test}}) = d(\mathbf{x}'_N, \mathbf{x}_{\text{test}})$$

This rule is independent of the underlying distribution responsible for generating the training examples

Cover and Hart's study of the rule

assumptions:

The classified examples (x_i, d_i) are independently and identically distributed

The sample size N is infinitely large

conclusion:

probability of classification error incurred by the nearest neighbor rule is bounded above by twice the Bayes probability, which means half the classification information in a training set of infinite size is contained in the nearest neighbor

Variant: k-nearest neighbor classifier

- 1. Identify the k classified pattern that lie nearest to the test vector x_{test} for some integer k**
- 2. Assign x_{test} to the class that is most frequently represented in the k nearest neighbors to x_{test}**

“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 changes take place in one or both cells such that A’s efficiency as one of the cells firing B, is increased.”

–Hebb

Hebbian Learning

If two neurons on either side of a synapse are activated simultaneously, then the strength of that synapse is selectively increased

If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated

Time-dependent

Local mechanism

Interactive

Conjunctive mechanism

Hebbian, anti-Hebbian, non-Hebbian

Mathematical Models of Hebbian Modifications

$$\Delta w_{kj}(n) = F(y_k(n), x_j(n))$$

Hebb's hypothesis

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n)$$

repeated application of the input signal x_j leads to an increase in y_k and therefore exponential growth that finally drives the synaptic connection into saturation

Covariance hypothesis

$$\Delta w_{kj} = \eta (x_j - \bar{x})(y_k - \bar{y})$$

- **a presynaptic activation w_{kj} is enhanced if there are sufficient levels of presynaptic activation**
- **Synaptic weight is depressed if they are not synchronous**

Competitive Learning

only a single output neuron is active at any one time

- A set of neurons that are all the same except for some randomly distributed synaptic weights, and which therefore respond differently to a given set of input pattern
- A limit imposed on the 'strength' of each neuron
- A mechanism that permits the neuron to compete for the right to respond to a given subset of inputs, such that only one output neuron, or only one neuron per group, is active at a time. The neuron that wins the competition is called a winner-takes-all neuron

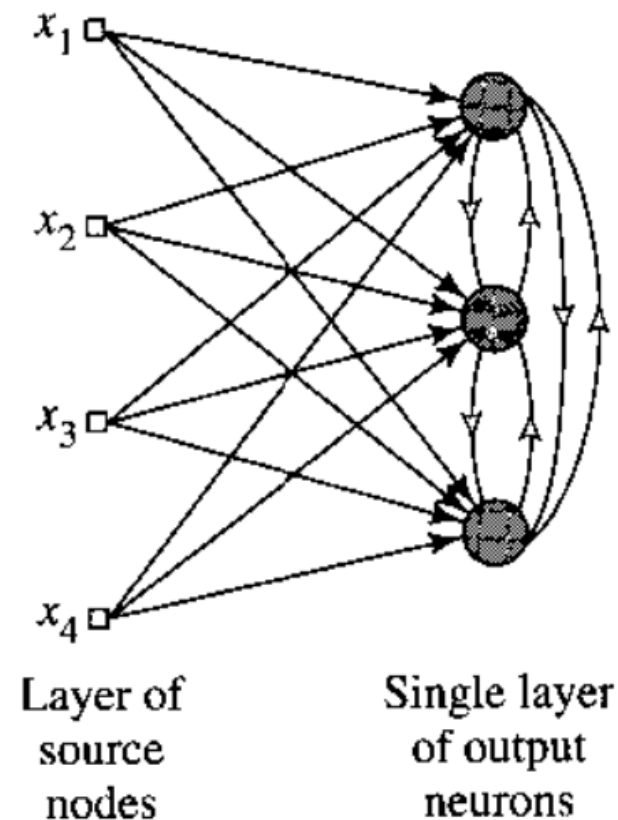


FIGURE 2.4 Architectural graph of a simple competitive learning network with feedforward (excitatory) connections from the source nodes to the neurons, and lateral (inhibitory) connections among the neurons; the lateral connections are signified by open arrows.

$$y_k = \begin{cases} 1 & \text{if } v_k > v_j \text{ for all } j, j \neq k \\ 0 & \text{otherwise} \end{cases}$$

each neuron is allotted a fixed amount of synaptic weight, which is distributed among its input nodes

$$\sum_j w_{kj} = 1 \quad \text{for all } k$$

If a particular neuron wins the competition, each input node of that neuron relinquishes some proportion of its synaptic weight, and the weight relinquished is then distributed equally among the active input nodes.

$$\Delta w_{kj} = \begin{cases} \eta(x_j - w_{kj}) & \text{if neuron } k \text{ wins the competition} \\ 0 & \text{if neuron } k \text{ loses the competition} \end{cases}$$

This rule has the overall effect of moving the synaptic weight w_k of winning neuron k toward the input pattern x

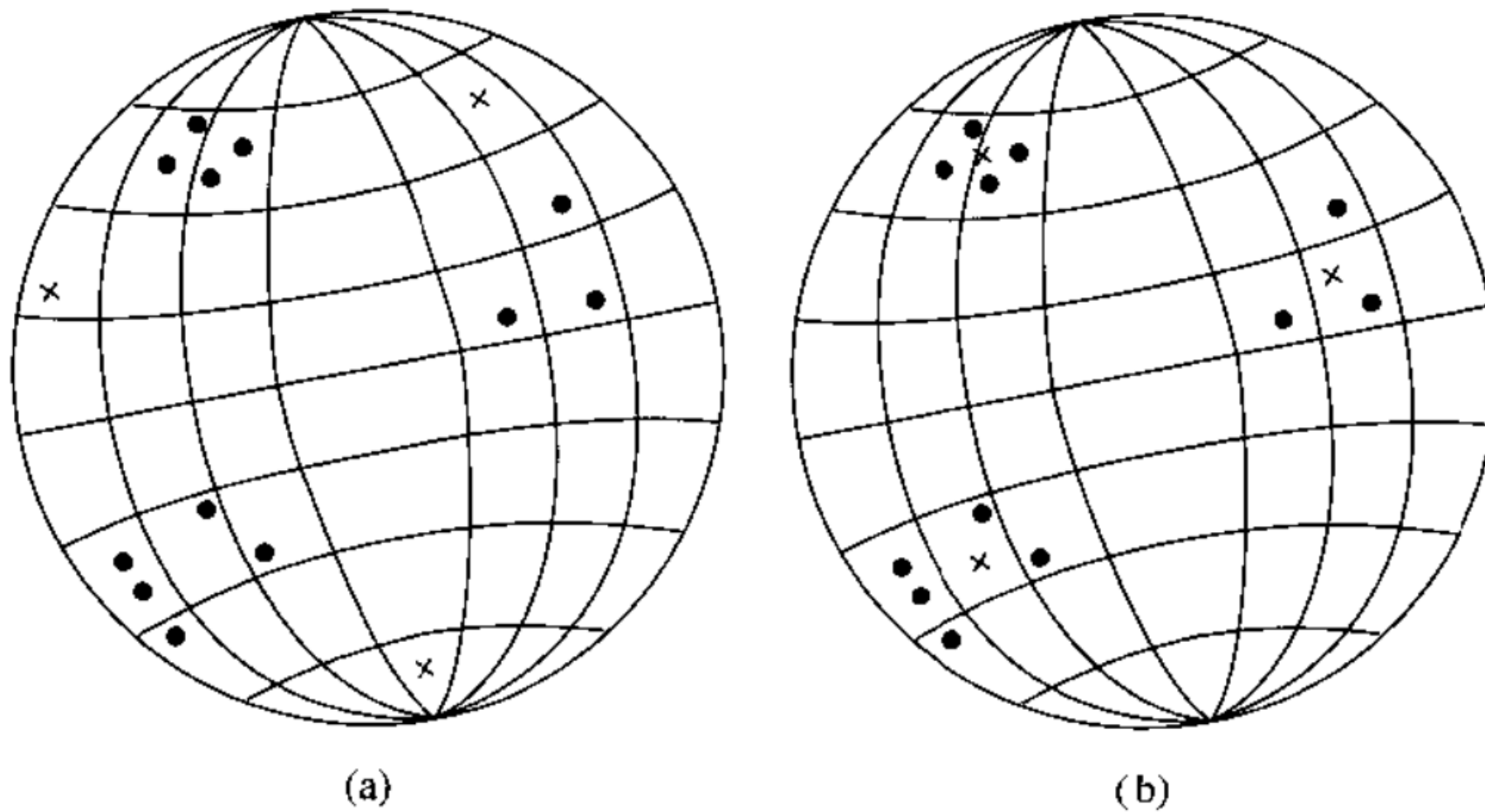


FIGURE 2.5 Geometric interpretation of the competitive learning process. The dots represent the input vectors, and the crosses represent the synaptic weight vectors of three output neurons. (a) Initial state of the network. (b) Final state of the network.

$$\sum_j w_{kj}^2 = 1 \quad \text{for all } k$$

Boltzmann Learning

The Boltzmann learning rule, name in honor of Ludwig Boltzmann, is a stochastic learning algorithm derived from ideas rooted in statistical mechanics.

In a Boltzmann machine the neurons constitute a recurrent structure, and they operate in a binary manner(+1, -1)

The machine is characterized by an energy function, E, determined by the particular states occupied by the individual neurons of the machine

$$E = -\frac{1}{2} \sum_j \sum_{\substack{k \\ j \neq k}} w_{kj} x_k x_j$$

The fact that $j \neq k$ means simply that none of the neurons in the machine has self-feedback


The machine operates by choosing a neuron at random — — for example k — — at some step of learning process, then flipping the state of neuron k from x_k to $-x_k$ at some temperature T with propability

$$P(x_k \rightarrow -x_k) = \frac{1}{1 + \exp(-\Delta E_k/T)}$$

δE is the energy change resulting from such a flip

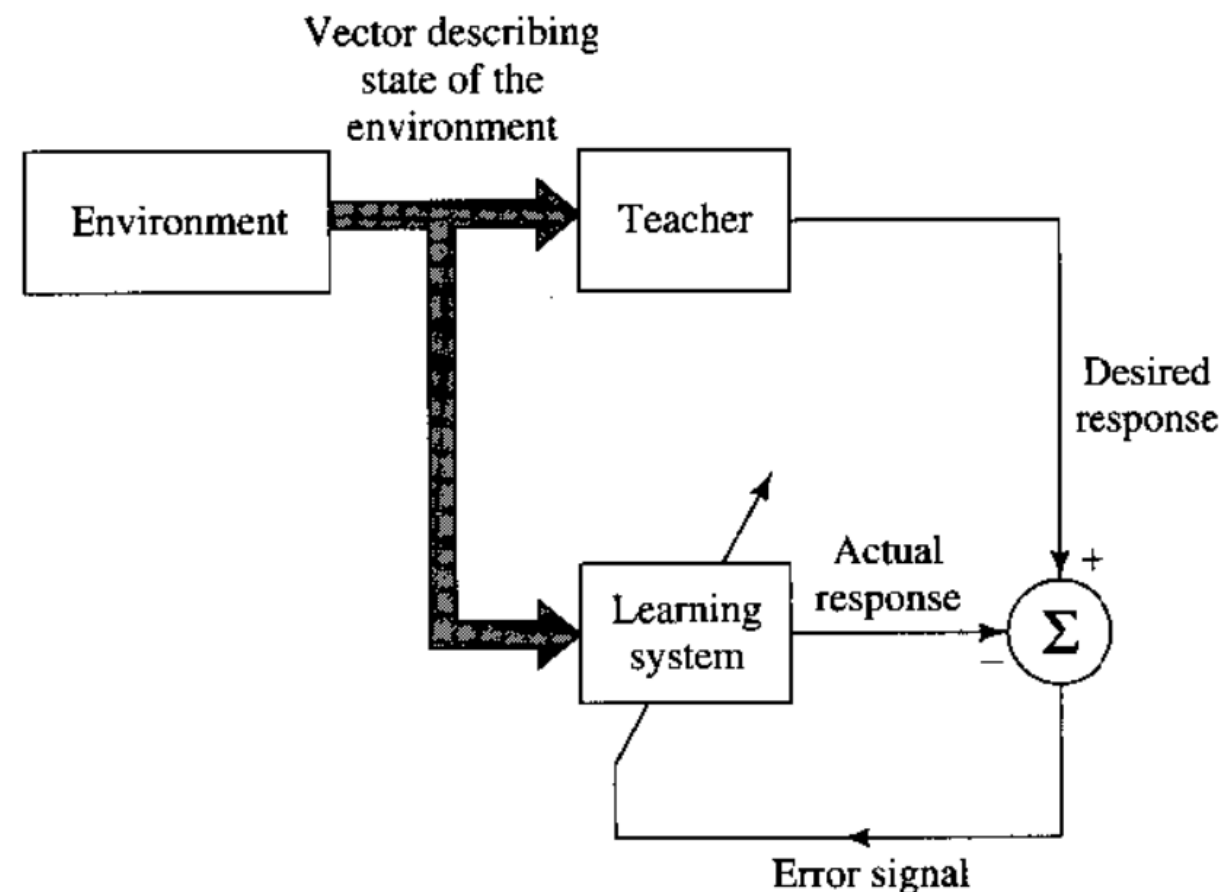
If this rule is applied repeatedly, the machine will reach thermal equilibrium

- **Clamped condition, in which the visible neurons are all clamped onto specific states determined by the environment**
- **Free-running condition, in which all the neurons are allowed to operate freely.**

$$\Delta w_{kj} = \eta(\rho_{kj}^+ - \rho_{kj}^-), \quad j \neq k$$


the correlation between the states of neuron j and k with the network in its clamped condition

Learning with a teacher

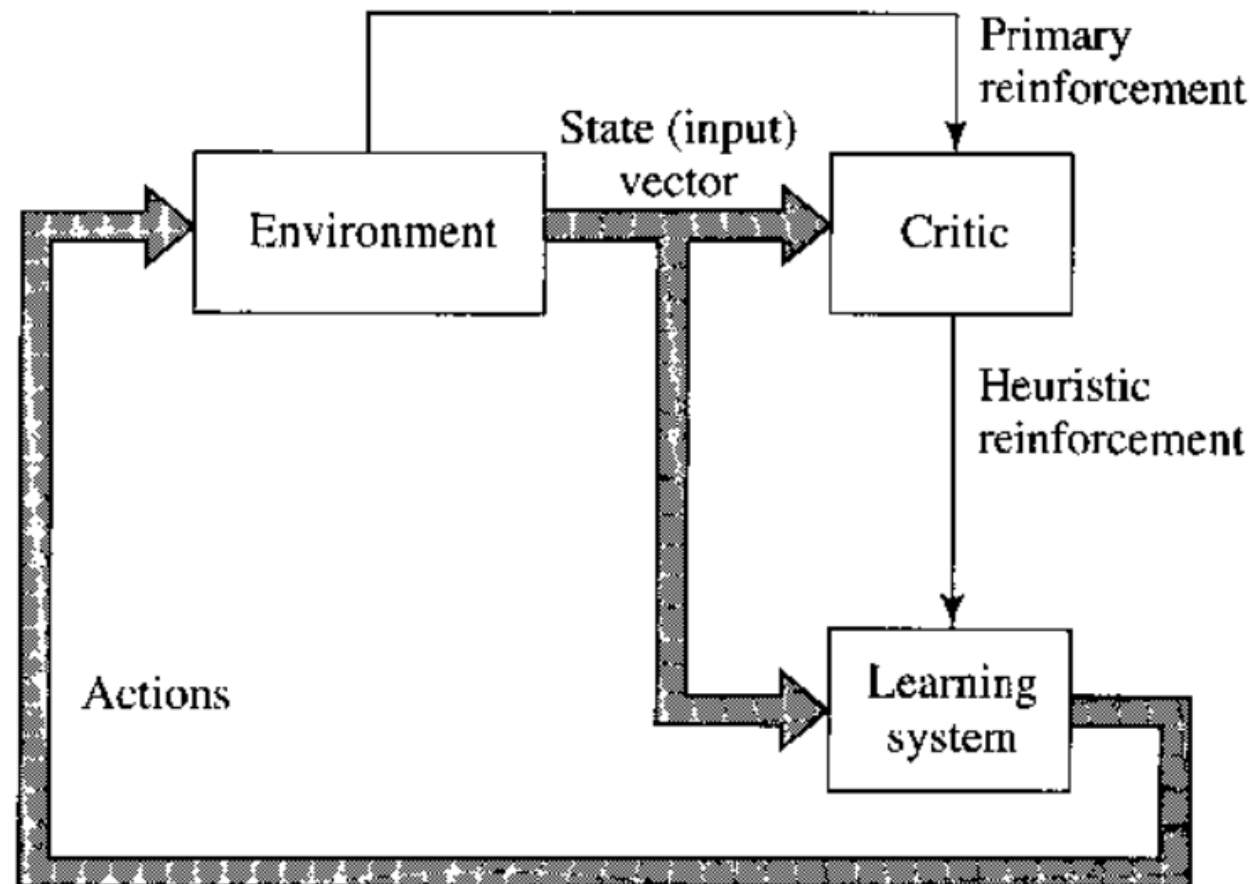


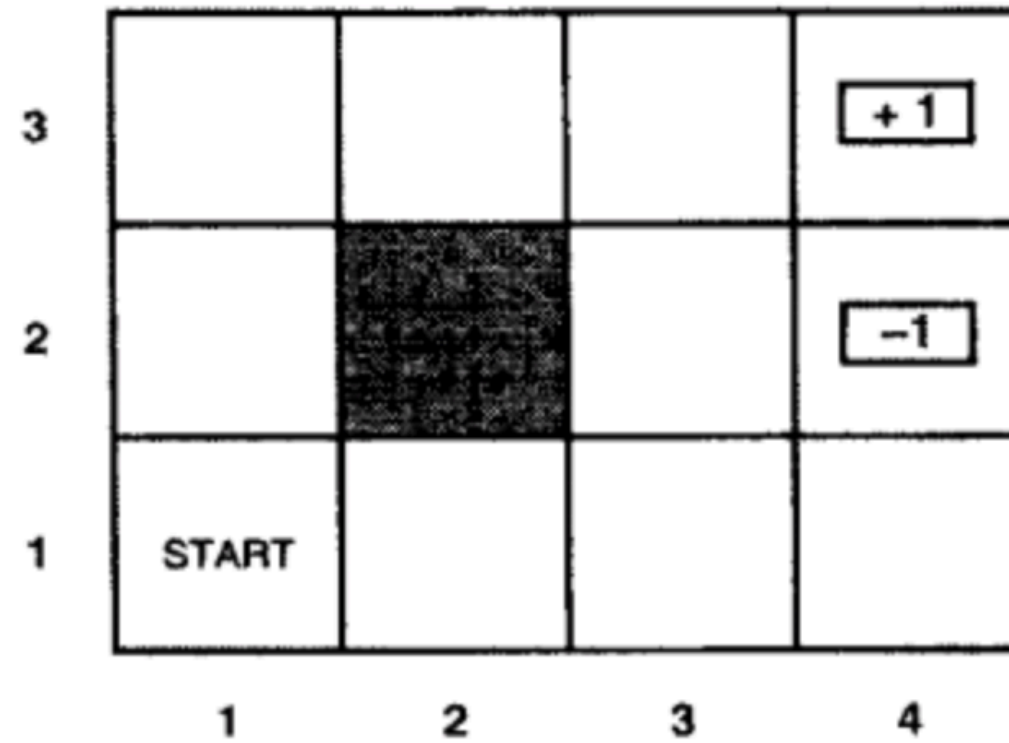
teacher: have knowledge of the environment being represented by a set of input-output examples

mean-square error or the sum of squared errors over the training sample, defined as a function of the free parameters of the system to form the error-performance surface and to spot the point with the lowest value

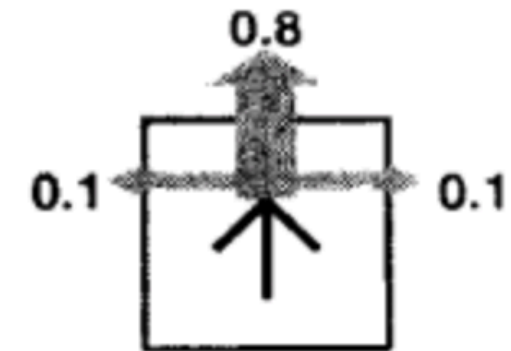
Learning without a teacher

1. Reinforcement learning/ Neurodynamic programming





(a)



(b)

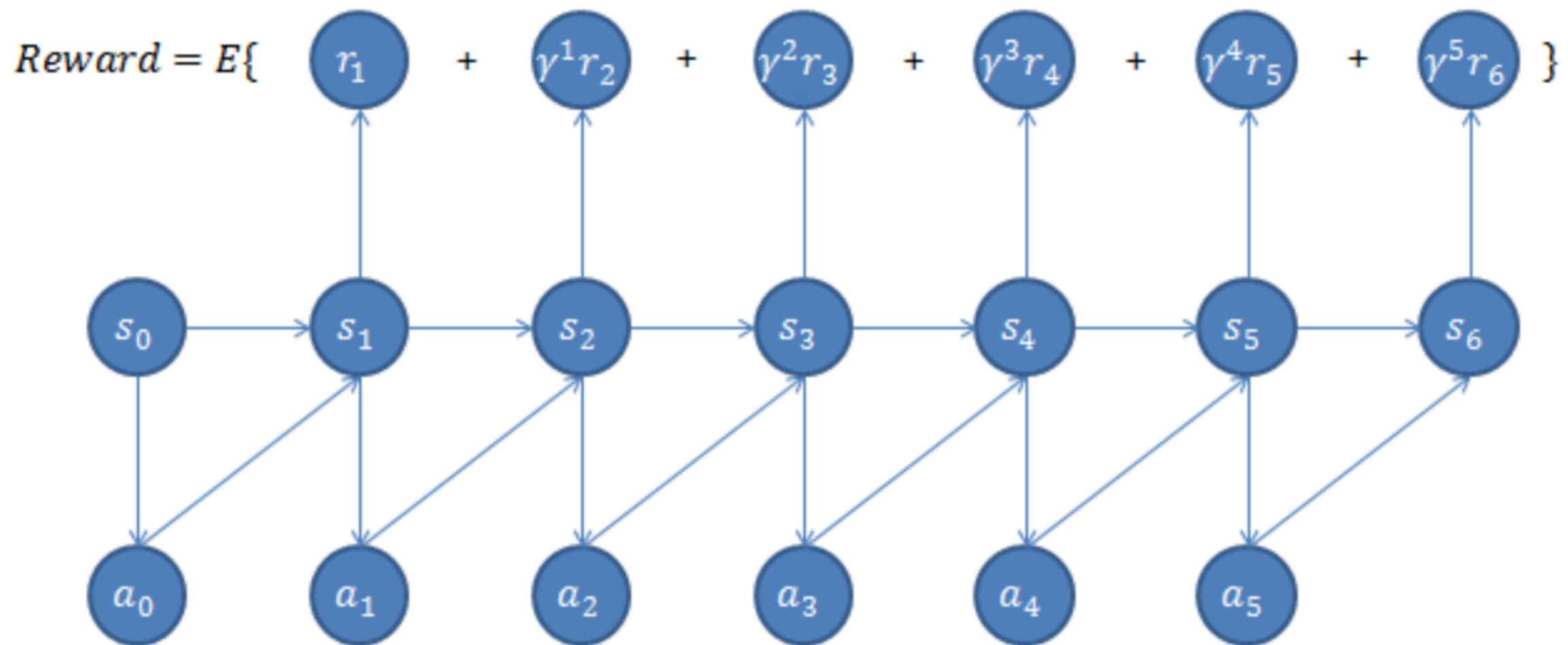
Markov Decision Processes, MDP

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} \dots$$

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots$$

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots$$

$$Reward = E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots]$$



$$V^\pi(s) = E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi]$$

Bellman Equations

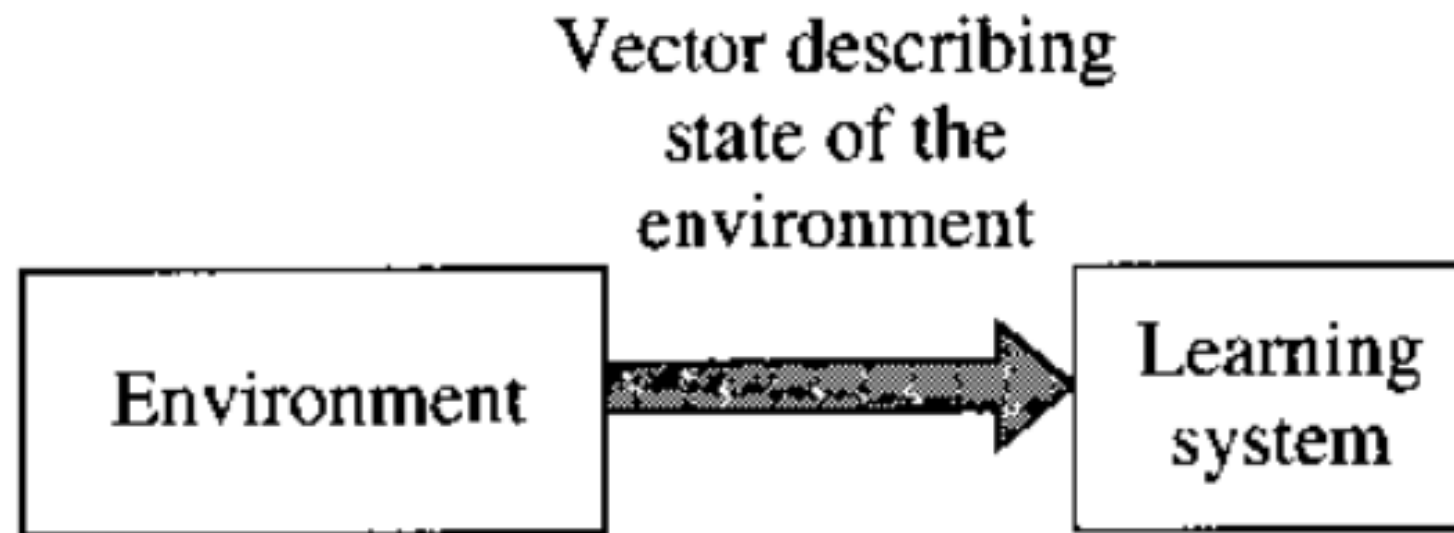
$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s')$$

$$V^*(s) = \max_{\pi} V^\pi(s)$$

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^*(s')$$

$$\pi^*(s) = \arg \max_{a \in A} \sum_{s' \in S} P_{sa}(s') V^*(s')$$

Unsupervised Learning



Learning Tasks

Pattern Association

Autoassociation:

a neural network is required to store a set of patterns(vectors) by repeatedly presenting them to the network

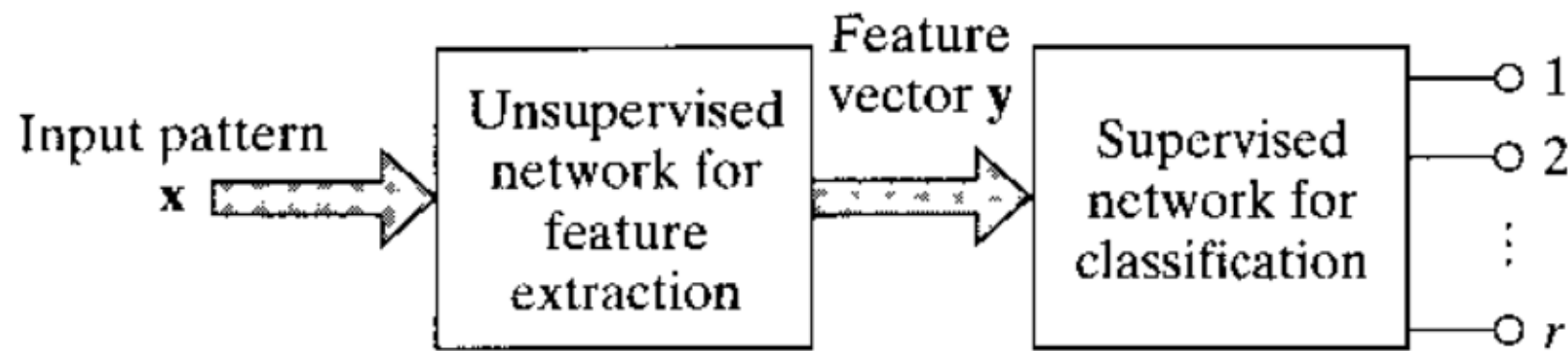
**The network is subsequently presented a partial description or distorted version of an original pattern stored in it, and the task is to recall that particular pattern
(unsupervised learning)**

Heteroassociation

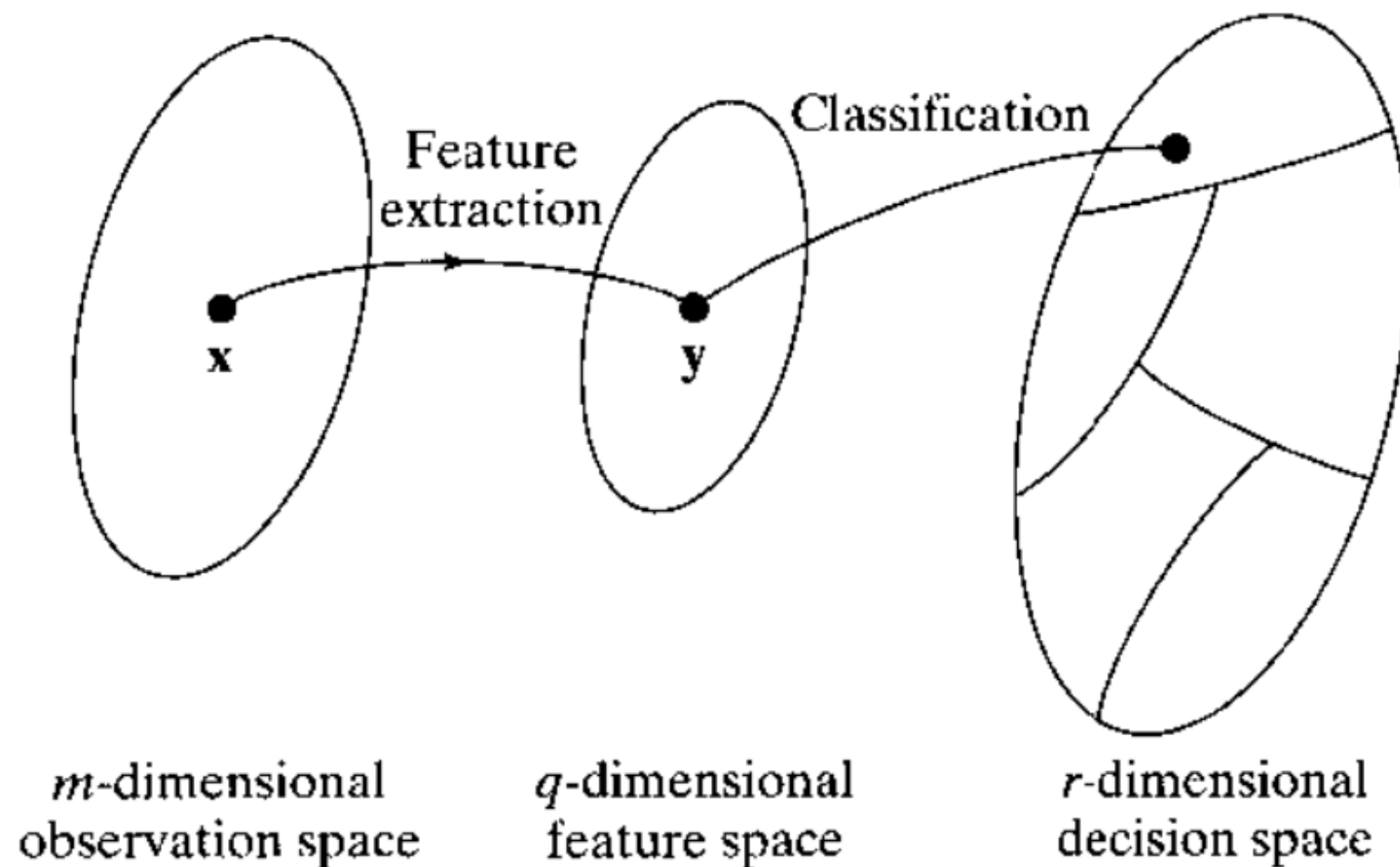
**differs from autoassociation in that an arbitrary input is paired with another output pattern
(supervised learning)**

$$\mathbf{x}_k \rightarrow \mathbf{y}_k, \quad k = 1, 2, \dots, q$$

Pattern Recognition



(a)



Function Approximation

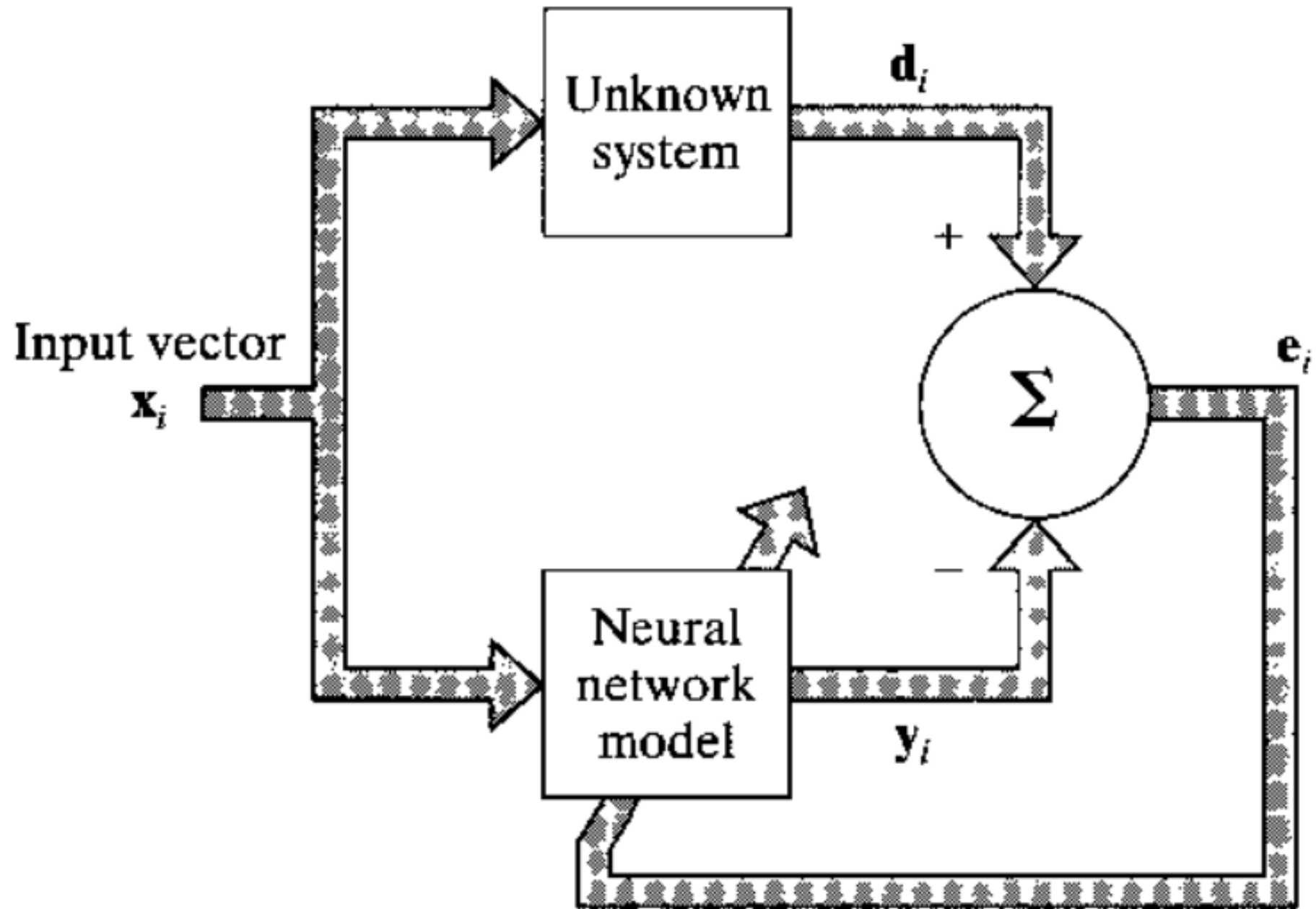
$$\mathbf{d} = \mathbf{f}(\mathbf{x})$$

$$\mathcal{T} = \{(\mathbf{x}_i, \mathbf{d}_i)\}_{i=1}^N$$

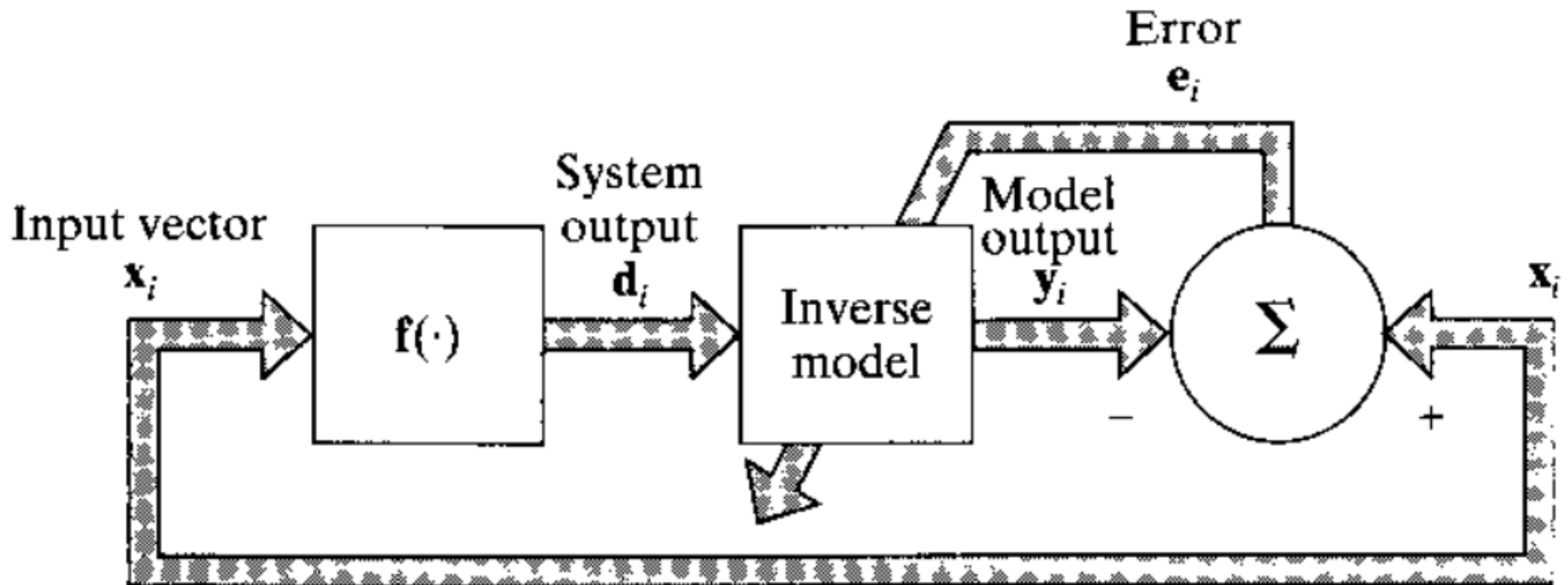
the requirement is to design a neural network that approximates the unknown function $\mathbf{f}(\cdot)$ such that the function $\mathbf{F}(\cdot)$ describing the input-output mapping actually realized by the network closed to $\mathbf{f}(\cdot)$

$$\|\mathbf{F}(\mathbf{x}) - \mathbf{f}(\mathbf{x})\| < \epsilon \quad \text{for all } \mathbf{x}$$

System identification

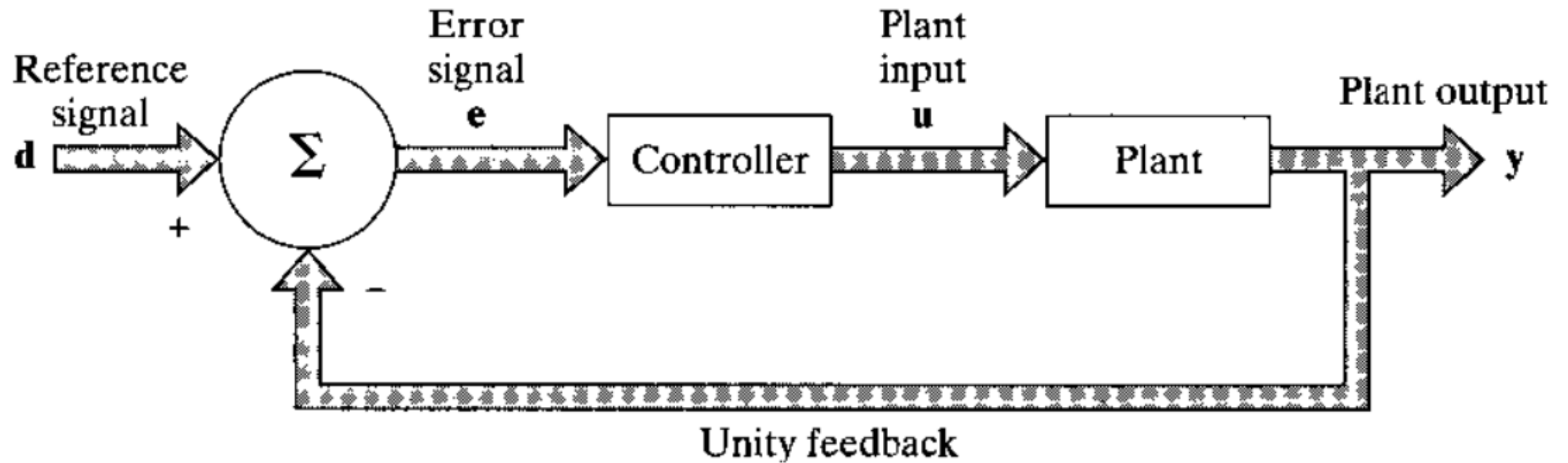


Inverse system



$$\mathbf{x} = \mathbf{f}^{-1}(\mathbf{d})$$

Control



Filtering

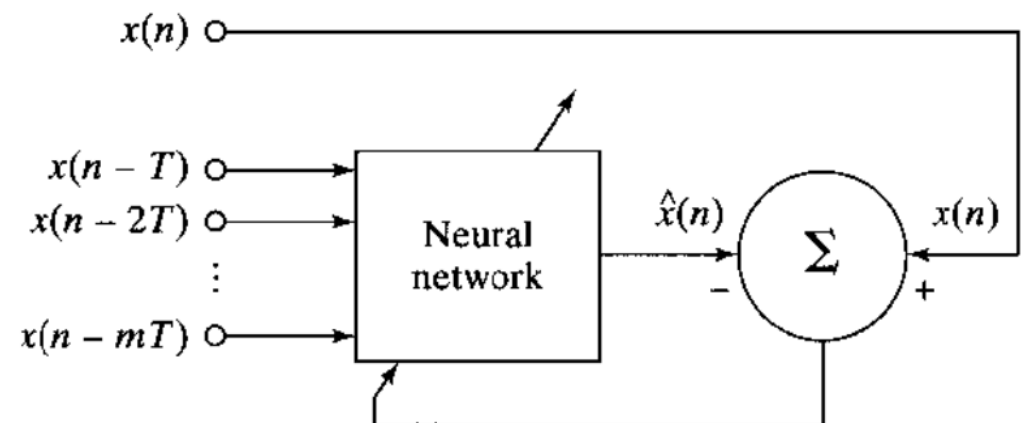
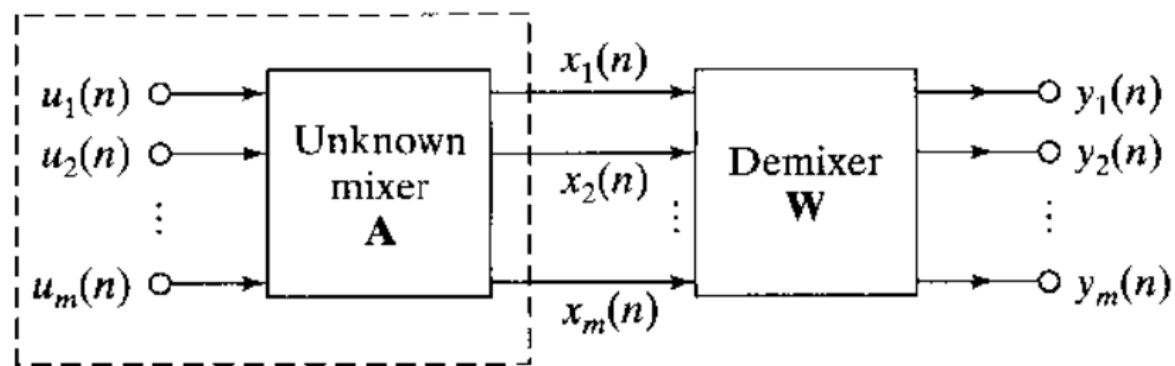
filter: a device or algorithm used to extract information about a prescribed quantity of interest from a set of noisy data

three basic tasks:

Filtering: extraction of information about a quantity of interest at discrete time n by using data measured up to and including time n

Smoothing: is much more accurate than filtering, we are able to use data obtained not only up to time n but also after time n

Prediction. This task is the forecasting side of information processing



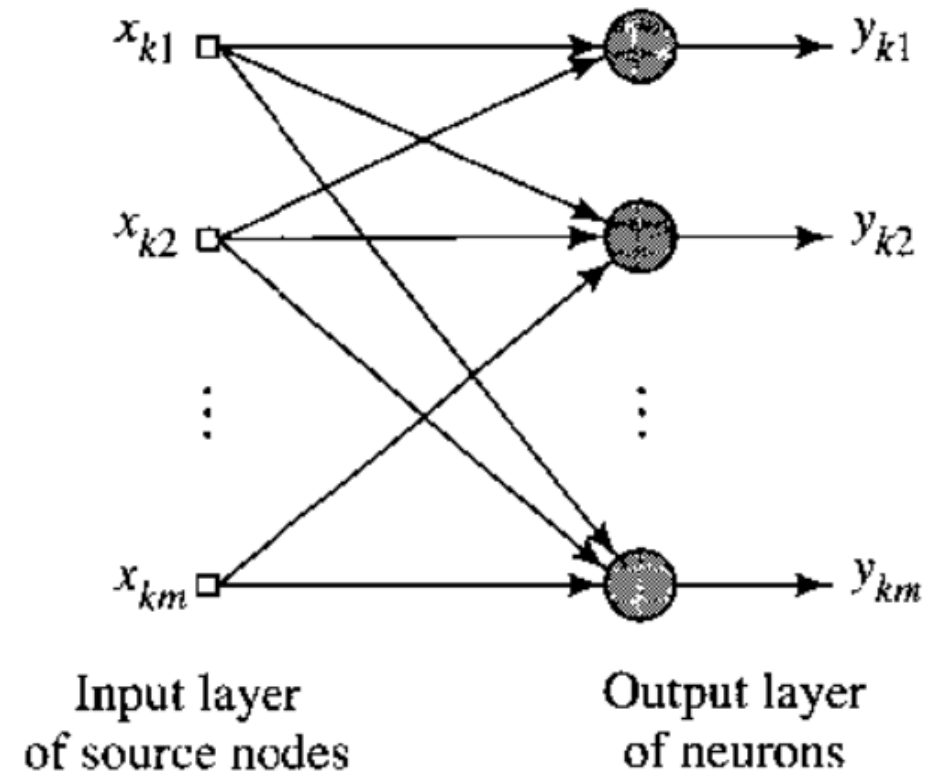
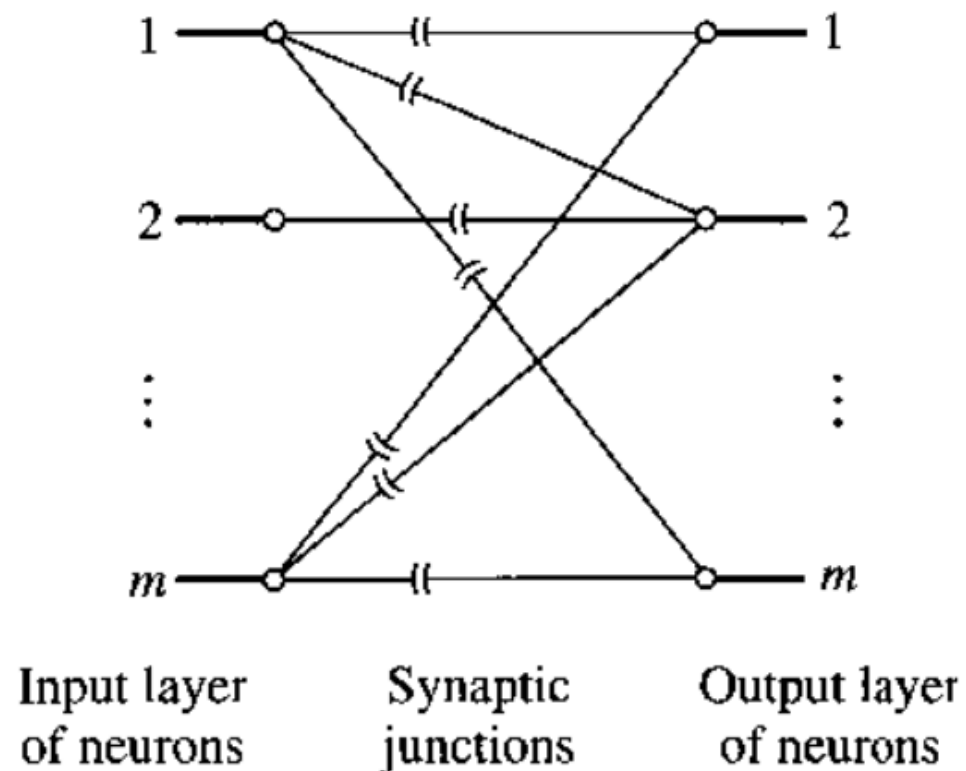
Memory

distributed memory

the basic issue of interest is the simultaneous or near-simultaneous activities of many different neurons

The neural activities form a spatial pattern inside the memory that contains information about the stimuli. The memory is therefore said to perform a distributed mapping that transforms an activity pattern in the input space into another activity pattern in the output space

examples



we assume that the neural networks are linear

an activity pattern \mathbf{x}_k occurs in the input layer of the network and that an activity pattern \mathbf{y}_k occurs simultaneously in the output layer. The issue we wish to consider here is that of learning from the association between the pattern \mathbf{x}_k and \mathbf{y}_k

$$\mathbf{x}_k = [x_{k1}, x_{k2}, \dots, x_{km}]^T$$

$$\mathbf{y}_k = [y_{k1}, y_{k2}, \dots, y_{km}]^T$$

the association of key vector \mathbf{x}_k with memorized vector \mathbf{y}_k may be described in the matrix form as:

$$\mathbf{y}_k = \mathbf{W}(k)\mathbf{x}_k, \quad k = 1, 2, \dots, q$$

$\mathbf{W}(k)$ is a weight matrix determined solely by the input-output pair

$$y_{ki} = \sum_{j=1}^m w_{ij}(k)x_{kj}, \quad i = 1, 2, \dots, m$$

$$y_{ki} = [w_{i1}(k), w_{i2}(k), \dots, w_{im}(k)] \begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{km} \end{bmatrix}, \quad i = 1, 2, \dots, m$$

$$\begin{bmatrix} y_{k1} \\ y_{k2} \\ \vdots \\ y_{km} \end{bmatrix} = \begin{bmatrix} w_{11}(k) & w_{12}(k) & \dots & w_{1m}(k) \\ w_{21}(k) & w_{22}(k) & \dots & w_{2m}(k) \\ \vdots & \vdots & \vdots & \vdots \\ w_{m1}(k) & w_{m2}(k) & \dots & w_{mm}(k) \end{bmatrix} \begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{km} \end{bmatrix}$$

The individual presentations of the q pairs of associated pattern $\mathbf{x}_k - \mathbf{y}_k$, $k=1, 2, 3, \dots, q$, produce corresponding values of the individual matrix, $\mathbf{W}(1), \mathbf{W}(2), \dots, \mathbf{W}(q)$

$$\mathbf{M} = \sum_{k=1}^q \mathbf{W}(k)$$

$$\mathbf{M}_k = \mathbf{M}_{k-1} + \mathbf{W}(k), \quad k = 1, 2, \dots, q$$

we may postulate $\hat{\mathbf{M}}$, denoting an estimate of the memory matrix \mathbf{M}

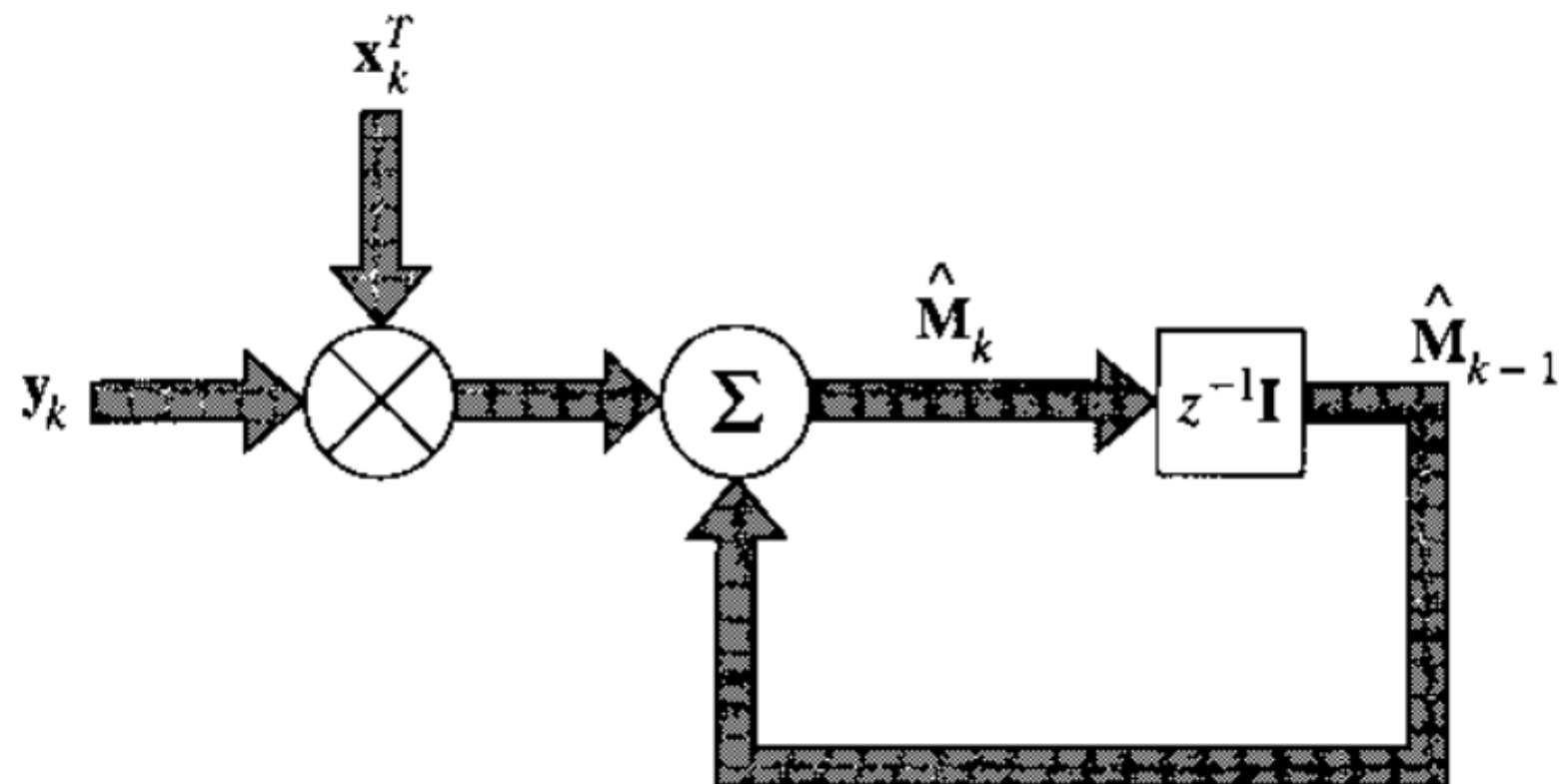
$$\hat{\mathbf{M}} = \sum_{k=1}^q \mathbf{y}_k \mathbf{x}_k^T$$

A typical term of the outer product $\mathbf{Y}_k \mathbf{X}_k^T$ is written as $y_{ki} x_{kj}$, — — — Hebbian learning

$$\hat{\mathbf{M}} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_q] \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_q^T \end{bmatrix}$$

$$= \mathbf{Y}\mathbf{X}^T$$

$$\hat{\mathbf{M}}_k = \hat{\mathbf{M}}_{k-1} + \mathbf{y}_k \mathbf{x}_k^T, \quad k = 1, 2, \dots, q$$



Recall

$$\mathbf{y} = \hat{\mathbf{M}} \mathbf{x}_j$$

$$\mathbf{y} = \sum_{k=1}^m \mathbf{y}_k \mathbf{x}_k^T \mathbf{x}_j$$

$$= \sum_{k=1}^m (\mathbf{x}_k^T \mathbf{x}_j) \mathbf{y}_k$$

$$\mathbf{y} = (\mathbf{x}_j^T \mathbf{x}_j) \mathbf{y}_j + \sum_{\substack{k=1 \\ k \neq j}}^m (\mathbf{x}_k^T \mathbf{x}_j) \mathbf{y}_k$$

Let each of the key pattern $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_q$ be normalized to have unit energy

$$E_k = \sum_{l=1}^m x_{kl}^2$$

$$= \mathbf{x}_k^T \mathbf{x}_k$$

$$= 1, \quad k = 1, 2, \dots, q$$

$$\mathbf{y} = \mathbf{y}_j + \mathbf{v}_j$$

$$\mathbf{v}_j = \sum_{\substack{k=1 \\ k \neq j}}^m (\mathbf{x}_k^T \mathbf{x}_j) \mathbf{y}_k$$

The first term on the right-hand side represents the desired response \mathbf{y}_j ; the second term is a noise vector that arises because crosstalk between the key vector and all the other key vectors stored in memory.

$$\cos(\mathbf{x}_k, \mathbf{x}_j) = \frac{\mathbf{x}_k^T \mathbf{x}_j}{\|\mathbf{x}_k\| \|\mathbf{x}_j\|}$$

$$\cos(\mathbf{x}_k, \mathbf{x}_j) = \mathbf{x}_k^T \mathbf{x}_j$$

We may then redefine the noise vector

$$\mathbf{v}_j = \sum_{\substack{k=1 \\ k \neq j}}^m \cos(\mathbf{x}_k, \mathbf{x}_j) \mathbf{y}_k$$

We now see that if the key vectors are orthogonal

$$\cos(\mathbf{x}_k, \mathbf{x}_j) = 0, \quad k \neq j$$

In this case the memory associates perfectly

$$\mathbf{x}_k^T \mathbf{x}_j = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases}$$

What is then the limit on the storage capacity of the associative memory?

the answer lies in the rank of the memory matrix

$$r \leq \min(l, m)$$

thus the number of patterns that can be reliably stored in a correlation matrix memory

can never exceed the input space dimensionality

In real-life situations, we often find that the key patterns presented to an associative memory are neither orthogonal nor highly separated from each other

$$\{\mathbf{x}_{\text{key}}\}: \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_q$$

$$\{\mathbf{y}_{\text{mem}}\}: \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_q$$

$$\mathbf{x}_k^T \mathbf{x}_j \geq \gamma \quad \text{for } k \neq j$$

if the key pattern in this set have the form


$$\mathbf{x}_j = \mathbf{x}_0 + \mathbf{v}$$


\mathbf{v} is a stochastic vector, it is likely that the memory will recognize \mathbf{x}_0 and associate with it a vector \mathbf{y}_0 rather than any of the actual pattern pairs used to train it in the first place

statistical nature of the learning process

we are not interested in the evolution of the weight vector w as a neural network is cycled through a learning algorithm, we instead focus on the deviation between a target function $f(x)$ and the actual function $F(x, w)$ realized by the neural network

\mathbf{X} — consisting of a set of independent variables, and D representing a dependent variable


$$\{\mathbf{x}_i\}_{i=1}^N$$


$$\{d_i\}_{i=1}^N$$

$$\mathcal{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^N$$



training sample

$$D = f(\mathbf{X}) + \epsilon$$

$$D = f(\mathbf{X}) + \epsilon$$

$$E[\epsilon|\mathbf{x}] = 0$$

$$f(\mathbf{x}) = E[D|\mathbf{x}]$$

The expectational error ϵ is uncorrelated with the regression function $f(\mathbf{X})$

$$E[\epsilon f(\mathbf{X})] = 0$$

The purpose of this second model, based on a neural network, is to encode the empirical knowledge represented by the training sample

$$\mathcal{T} \rightarrow \mathbf{w}$$

Let the actual response of the neural network, produced in response to the input vector \mathbf{x} , be denoted by the random variable

$$Y = F(\mathbf{X}, \mathbf{w})$$

minimize the cost function

$$\mathcal{E}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (d_i - F(\mathbf{x}_i, \mathbf{w}))^2$$

the use of this equation as the cost function implies weights of the network are performed over the entire set of training examples rather than on an example-by-example basis

$$\mathcal{E}(\mathbf{w}) = \frac{1}{2} E_{\mathcal{T}}[(d - F(\mathbf{x}, \mathcal{T}))^2]$$

average operator

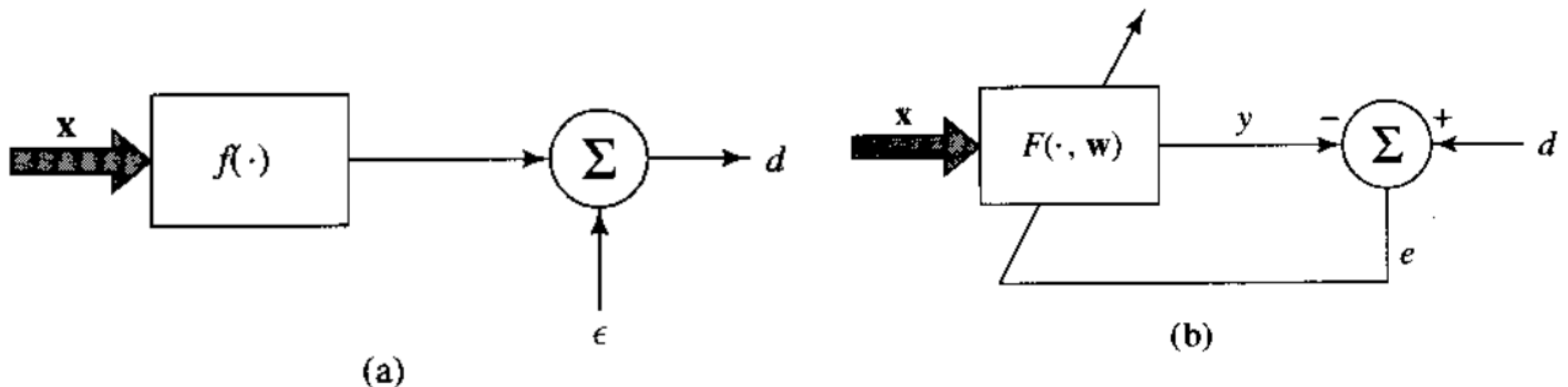
the statistical expectation operator E acts on the whole ensemble of random variables \mathbf{X} and \mathbf{D} , which includes \mathcal{T} as a subset

$$\mathcal{E}(\mathbf{w}) = \frac{1}{2} E_{\mathcal{T}}[(d - F(\mathbf{x}, \mathcal{T}))^2]$$

$$\begin{aligned} d - F(\mathbf{x}, \mathcal{T}) &= (d - f(\mathbf{x})) + (f(\mathbf{x}) - F(\mathbf{x}, \mathcal{T})) \\ &= \epsilon + (f(\mathbf{x}) - F(\mathbf{x}, \mathcal{T})) \end{aligned}$$

$$\mathcal{E}(\mathbf{w}) = \frac{1}{2} E_{\mathcal{T}}[\epsilon^2] + \frac{1}{2} E_{\mathcal{T}}[f(\mathbf{x}) - F(\mathbf{x}, \mathcal{T})]^2 + E_{\mathcal{T}}[\epsilon(f(\mathbf{x}) - F(\mathbf{x}, \mathcal{T}))]$$

The expectational error e is uncorrelated with the regression function $f(\mathbf{x})$
 The expectational error pertains to the regressive model in a , whereas the approximating function $F(\mathbf{x}, \mathbf{w})$ pertains to the neural network model of b



$$\mathcal{E}(\mathbf{w}) = \frac{1}{2}E_{\mathcal{T}}[\epsilon^2] + \frac{1}{2}E_{\mathcal{T}}[f(\mathbf{x}) - F(\mathbf{x}, \mathcal{T}))^2] + E_{\mathcal{T}}[\epsilon(f(\mathbf{x}) - F(\mathbf{x}, \mathcal{T}))]$$

$$\mathcal{E}(\mathbf{w}) = \frac{1}{2}E_{\mathcal{T}}[\epsilon^2] + \frac{1}{2}E_{\mathcal{T}}[(f(\mathbf{x}) - F(\mathbf{x}, \mathcal{T}))^2]$$



independent of the weight vector \mathbf{w}

$$L_{\text{av}}(f(\mathbf{x}), F(\mathbf{x}, \mathbf{w})) = E_{\mathcal{T}}[f(\mathbf{x}) - F(\mathbf{x}, \mathcal{T}))^2]$$

$$f(\mathbf{x}) = E[D|\mathbf{x}]$$

$$L_{\text{av}}(f(\mathbf{x}), F(\mathbf{x}, \mathbf{w})) = E_{\mathcal{T}}[(E[D|\mathbf{X} = \mathbf{x}] - F(\mathbf{x}, \mathcal{T}))^2]$$

$$E[D|\mathbf{X} = \mathbf{x}] - F(\mathbf{x}, \mathcal{T}) = (E[D|\mathbf{X} = \mathbf{x}] - E_{\mathcal{T}}[F(\mathbf{x}, \mathcal{T})]) + (E_{\mathcal{T}}[F(\mathbf{x}, \mathcal{T})] - F(\mathbf{x}, \mathcal{T}))$$

$$L_{\text{av}}(f(\mathbf{x}), F(\mathbf{x}, \mathcal{T})) = B^2(\mathbf{w}) + V(\mathbf{w})$$

$$B(\mathbf{w}) = E_{\mathcal{T}}[F(\mathbf{x}, \mathcal{T})] - E[D|\mathbf{X} = \mathbf{x}]$$



the inability of the neural network defined by the function $F(\cdot)$ to accurately approximate $f(\cdot)$

$$V(\mathbf{w}) = E_{\mathcal{T}}[(F(\mathbf{x}, \mathcal{T}) - E_{\mathcal{T}}[F(\mathbf{x}, \mathcal{T})])^2]$$

the inadequacy of the information contained in the training sample

Statistical Learning Theory

In this section we continue the statistical characterization of neural networks by describing a learning theory that addresses the fundamental issue of how to control the generalization ability of a neural network in mathematical terms

1. **Environment.** The environment is stationary, supplying a vector \mathbf{x} with a fixed but unknown cumulative distribution function $F_{\mathbf{x}}(\mathbf{x})$

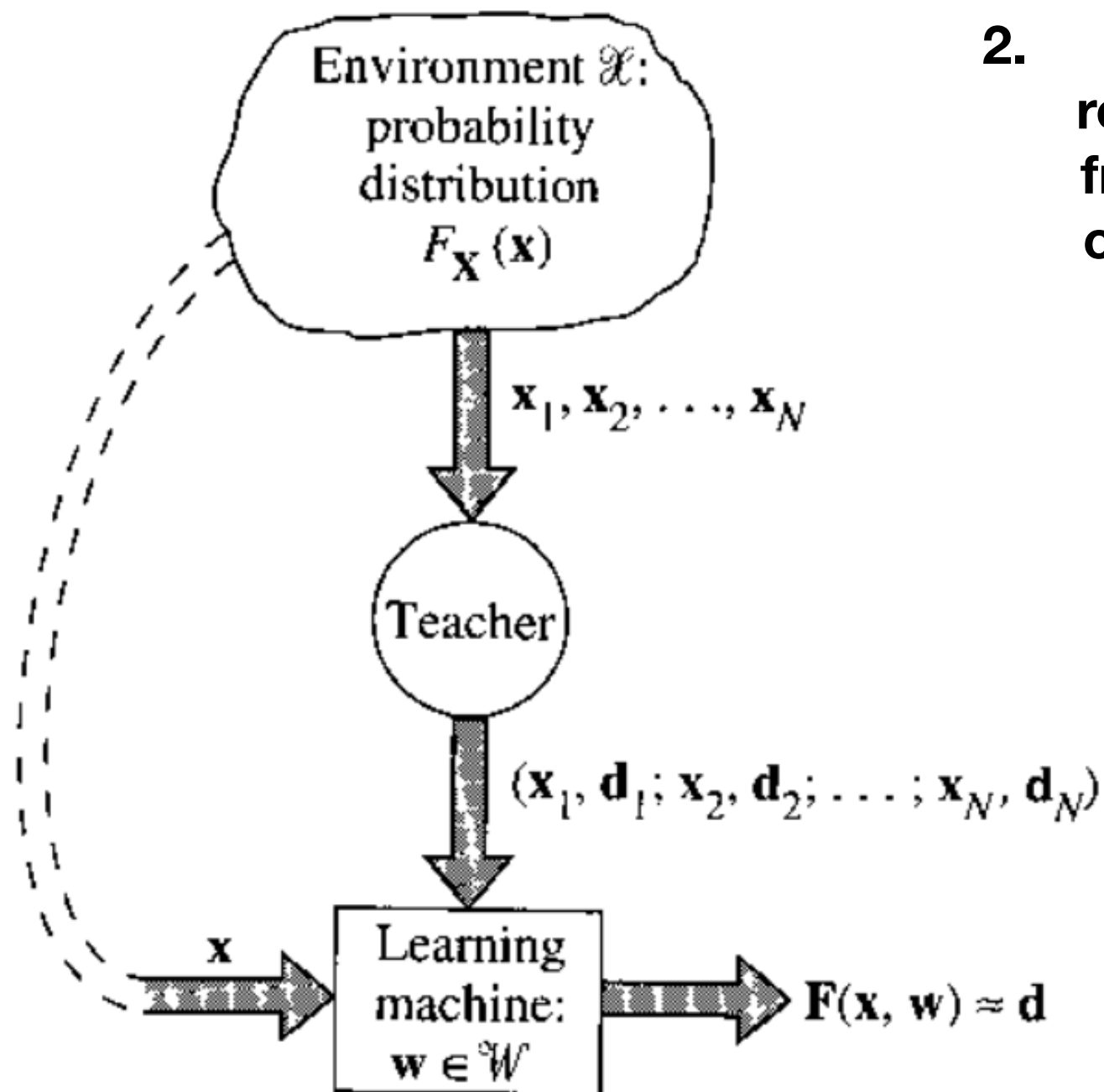
2. **Teacher.** The teacher provides a desired response d for every input vector \mathbf{x} received from the environment, in accordance with a conditional cumulative distribution function $F_{\mathbf{x}}(\mathbf{x}|d)$ that is also fixed but unknown

$$d = f(\mathbf{x}, v)$$

v is a noise term, permitting the teacher to be “noisy”

3. **Learning machine.** The learning machine is capable of implementing a set of input-output mapping functions described by

$$y = F(\mathbf{x}, \mathbf{w})$$



$$\mathcal{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^N$$

$$L(d, F(\mathbf{x}, \mathbf{w})) = (d - F(\mathbf{x}, \mathbf{w}))^2$$

$$R(\mathbf{w}) = \int L(d, F(\mathbf{x}, \mathbf{w})) dF_{\mathbf{x},D}(\mathbf{x}, d)$$

How to minimize it $F_{\mathbf{x},d}$ is unknown?

Some Basic Definitions

Convergence in probability

Consider a sequence of random variables a_1, a_2, \dots, a_N

$$P(|a_N - a_0| > \delta) \xrightarrow{P} 0 \quad \text{as } N \rightarrow \infty$$

Supremum and infimum

Empirical risk functional

$$R_{\text{emp}}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L(d_i, F(\mathbf{x}_i, \mathbf{w}))$$

Strict Consistency

$$\mathcal{W}(c) = \left\{ \mathbf{w}: \int L(d, F(\mathbf{x}, \mathbf{w})) \geq c \right\}$$

$$\inf_{\mathbf{w} \in \mathcal{W}(c)} R_{\text{emp}}(\mathbf{w}) \xrightarrow{P} \inf_{\mathbf{w} \in \mathcal{W}(c)} R(\mathbf{w}) \quad \text{as } N \rightarrow \infty$$

Principle of Empirical Risk Minimization

$$R_{\text{emp}}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L(d_i, F(\mathbf{x}_i, \mathbf{w}))$$

This function differs from $R(w)$ in two desirable ways

- 1. It does not depend on the unknown distribution function $F_{x,d}(x,d)$ in an explicit sense**
- 2. In theory it can be minimized with respect to the weight vector w**

Let w_{emp} and $F(x, w_{\text{emp}})$ denote the weight vector and the corresponding mapping that minimize the empirical risk functional Rw)