Learning Processes

Sanjay Singh

Department of Information and Communication Technology
Manipal Institute of Technology, MAHE, Manipal-576104, INDIA
sanjay.singh@manipal.edu

January 24, 2019

Introduction

- Primary property of neural networks: **learn** from its environment, and **improve** its performance through learning
- Iterative adjustment of synaptic weights
- Definition of learning due to Mendel and McClaren[1]
  Learning is a process by which the free parameters of a neural network are adapted through a process of stimulating by the environment in which the network is embedded. Type of learning is determined by the manner in which the parameter changes take place.

[1] J. M. Mendel and R. W. McLaren. "A Prelude to Neural Networks". In: ed. by Jerry M. Mendel. Upper Saddle River, NJ, USA: Prentice Hall Press, 1994. Chap. Reinforcement-learning Control and Pattern Recognition Systems, pp. 287–318. ISBN: 0-13-147448-0.

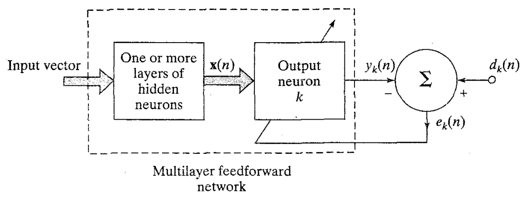
Learning

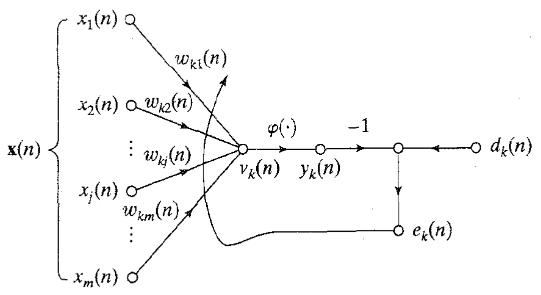
Sequence of events in neural network learning:

- NN is **simulated** by the environment
- NN **undergoes changes** in its free parameters as a result of this stimulation
- NN **responds in a new way** to the environment because of the changes that have occurred in its internal structure

A prescribed set of well-defined rules for the solution of the learning problem is called a **learning algorithm.**
The manner in which a NN relates to the environment dictates the **learning paradigm** that refers to a **model** of environment operated on by the NN

Error Correction Learning





- Input $X(n)$, output $y_k(n)$, and desired response or target output $d_k(n)$
- Error signal $e_k(n) = d_k(n) - y_k(n)$
- $e_k(n)$ actuates a control mechanism that gradually adjust the synaptic weights, to minimize the **cost function**

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

- When synaptic weights reach a steady state, learning is stopped

Error Correction Learning

- Delta rule or Widrow-Hoff rule, with **learning rate** η:

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

- With that, we can update the weights:

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

- There is a strong theoretical reason for doing this, which we'll discuss later

Memory-Based Learning

- All (or most) past experiences are explicitly stored, as input-output pairs $\{(x_i, d_i)\}_{i=1}^{N}$
- Consider two classes $\mathcal{C}_1, \mathcal{C}_2$
- Given a new input x_{test}, determine class based on local nbh of x_{test}
  - Criterion used for determining the nbh
  - Learning rule applied to the nbh of the input, within the set of training examples

Memory-Based Learning

- A set of instances observed so far:

$$X = \{x_1, x_2, \ldots, x_n\}$$

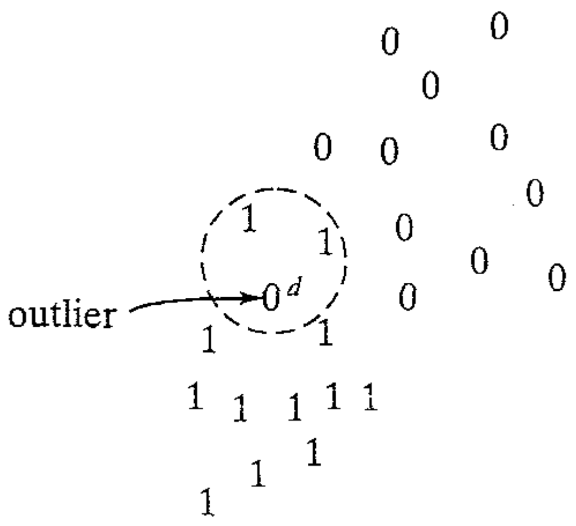
- Nearest neighbor $x'_N \in X$ of x_{test}:

$$\min_i d(x_i, x_{test}) = d(x'_N, x_{test})$$

  where $d(.,.)$ is the Euclidean distance
- x_{test} is classified as the same class as x'_N
- Cover and Hart[2] studied nnbh as tool for pattern classification with following assumptions:
  - The classified examples are iid
  - The sample size is infinitely large

[2] T. Cover and P. Hart. "Nearest Neighbor Pattern Classification". In: *IEEE Trans. Inf. Theor.* 13.1 (1967), pp. 21–27.

Memory-Based Learning: K-Nearest Neighbor



- Identify k classified patterns that lie nearest to the test vector x_{test} for some integer k
- Assign x_{test} to the class that is most frequently represented by the k neighbors (use majority vote)
- KNN acts like an averaging device

Hebbian Learning

- Donald Hebb's postulate of learning appeared in his book, The Organization of Behavior:
  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.
- Hebbian synapse
  - 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.

Hebbian Synapse

Characteristics of a Hebbian synapse:

- Time-dependent mechanism
- Local mechanism
- Interactive mechanism
- Correlative/conjunctive mechanism

Strong evidence for Hebbian plasticity in the Hippocampus (brain region)

Classification of Synaptic Modification

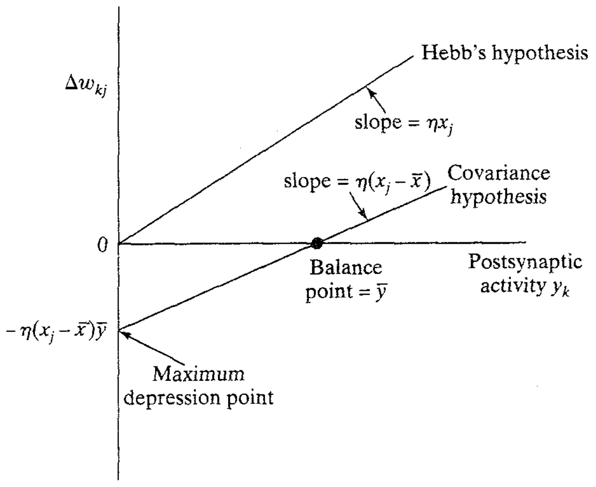
Type	Positively correlated	Negatively correlated
Hebbian	Strengthen	Weaken
Anti-Hebbian	Weaken	Strengthen
Non-Hebbian	X	X

Mathematical Models of Hebbian Modifications

- Consider a synaptic weight w_{kj} of neuron k with presynaptic and postsynaptic signals x_j and y_k
- General form the adjustment applied to the synaptic weight w_{kj} at time step n

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

- Hebb's hypothesis (Hebbian learning): $\Delta w_{kj}(n) = \eta y_k(n) x_j(n)$



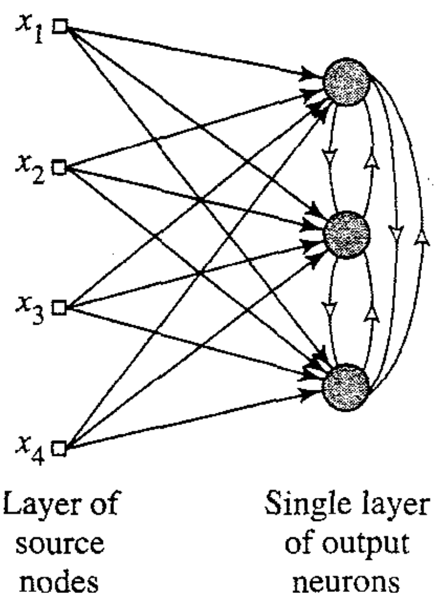
Covariance Rule

- Sejnowski (1977) introduced covariance hypothesis to overcome the limitation of Hebb's hypothesis
- $\Delta w_{kj}(n) = \eta(x_j - \bar{x})(y_k - \bar{y})$
- Convergence to a nontrivial state, which is reached when $x_k = \bar{x}$ or $y_j = \bar{y}$
- Prediction of both synaptic potentiation and depression
- Observations:
  - weight enhanced when both pre-and post snaptic activities are above average
  - weight depressed if
  1. $x_j > \bar{x}$ and $y_k < \bar{y}$ or
  2. $y_k > \bar{y}$ and $x_j < \bar{x}$

Competitive Learning

- Output neurons compete with each other for a chance to become active
- Highly suited to discover statistically salient features (that may aid in classification)
- Basic elements of competitive learning are:
  1. A set of same type of neuron with randomly distributed synaptic weight, so they respond differently to a given set of inputs
  2. A limit imposed on the strength of each neuron
  3. Competition mechanism, to choose one winner: winner-takes-all neuron

Competitive Learning



- Single layer, feedforward excitatory, and lateral inhibitory connections

- Winner selection

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

- Limit: $\sum_j w_{kj} = 1 \quad \forall k$
- Adaptation:

$$\Delta w_{kj} = \begin{cases} \eta(x_j - w_{kj}) & winner = k \\ 0 & \text{otherwise} \end{cases}$$

- Weight vector $W_k = (w_{k1}, w_{k2}, \ldots, w_{kn})$ is moved to toward the input vector

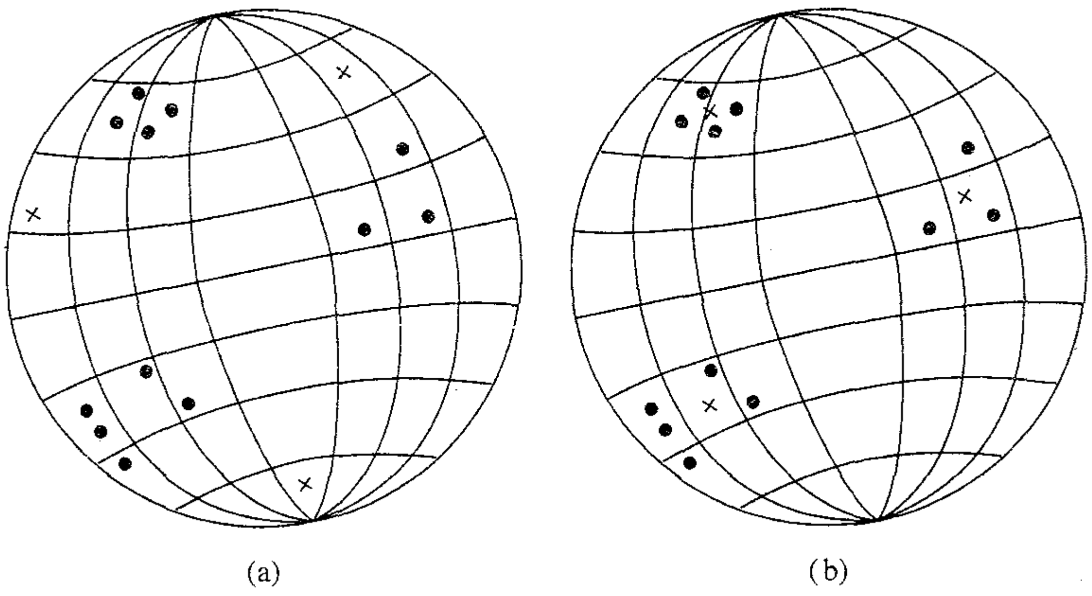


Figure 1: Geometric interpretation of the competitive learning process.

Boltzmann Learning

- Stochastic learning algorithm rooted in statistical mechanics
- ANN designed on the basis of Boltzmann learning rule is called Boltzmann machine
- In Boltzmann machine, the neuron constitutes a recurrent structure, and operate in a binary manners (on/off state denoted by +1/-1)
- M/c is characterized by an energy function E:

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

- Activation:
  - Choose a random neuron k
  - Flip state with a probability (given temperature T)

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

Boltzmann Machine

Neurons of Boltzmann machine are partition into two function groups:

- Visible neurons-can be affected by the environment
- Hidden neuron:isolated

Modes of operation

- Clamped: visible neuron states are fixed by environmental input and held constant
- Free-running: all neurons are allowed to update their activity freely

Boltzmann Machine: Learning and Operation

- Learning:
  - Correlation of activity during clamped condition ρ_{kj}^{+}
  - Correlation of activity during free-running condition ρ_{kj}^{-}
  - Weight update: $\Delta w_{kj} = \eta(\rho_{kj}^{+} - \rho_{kj}^{-}),\ j \neq k$
- Train weights w_{kj} with various clamping input patterns
- After training is completed, present new clamping input that is a partial input of one of the known vectors
- Let it run clamped on the new input (subset of visible neurons), and eventually it will complete the pattern

$$Correl(x, y) = \frac{Cov(x, y)}{\sigma_x \sigma_y}$$

Learning Paradigms

How neural networks relate to their environment

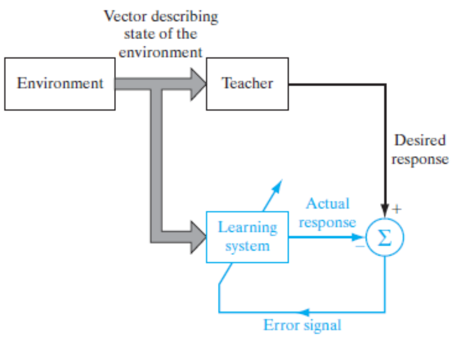
- Credit assignment problem
- Learning with a teacher
- Learning without a teacher

Credit-Assignment Problem

- How to assign **credit** or **blame** for overall outcomes to each of the internal decision made by a learning machine
- In many cases, the outcomes depends on a **sequence of actions**
  - Assignment of credit for outcomes of actions (**temporal credit-assignment problem**): When does a particular action deserve credit
  - Assignment of credit for actions to internal decisions (**structural credit-assignment problem**): assign credit to **internal structures** of actions generated by the system

Credit-assignment problem routinely arises in neural network learning. Which neuron, which connection to credit or blame?

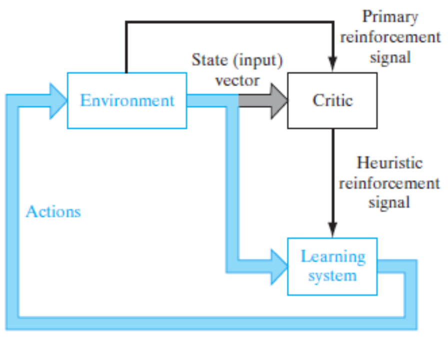
Learning with a teacher



- Also known as supervised learning
- Teacher has knowledge, represented as input-output examples. Environment is unknown to the NN
- NN tries to emulate the teacher gradually
- Error-correction learning is one way to achieve this
- Error-performance surface or error surface

To be used when, ”I know how to classify this data, I just need you(the classifier) to do it.”

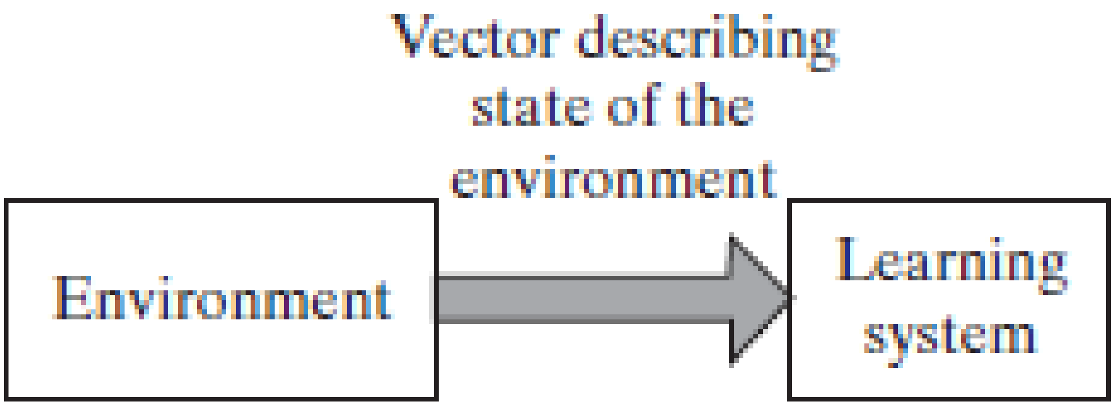
Learning without a teacher: Reinforcement Learning



- Learning input-output mapping through continued interaction with the environment
- **Actor-critic**:critic converts **primary reinforcement signal** into high-quality **heuristic reinforcement signal**
- Goal is to optimize the **cumulative cost** of actions

To be used when, ”I have no idea how to classify this data, can you classify this data and I’ll give you a reward if it’s correct or I’ll punish you if it’s not.”

Learning without a teacher: Unsupervised Learning



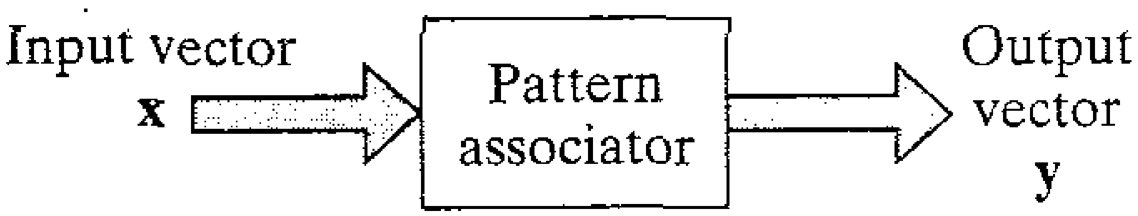
- Learn based on **task-independent measure** of the quality of representation
- Internal representations for encoding features of the input space
- Competetive learning rule needed, such as **winner-takes-all**

To be used when, ”I have no idea how to classify this data, can you(the algorithm) create a classifier for me?”

Learning Tasks

- Pattern association
- Pattern recognition
- Function approximation

Pattern Association

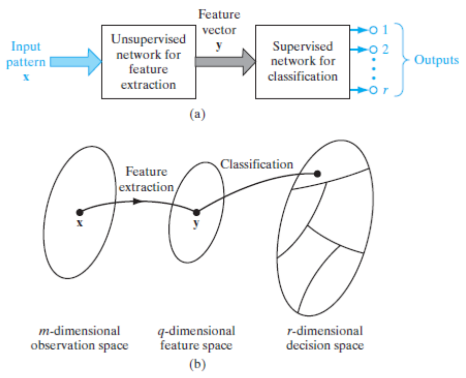


- Associative memory: brainlike distributed memory that learns association
- Pattern association (x_k : key patterns, y_k : memorized pattern)

$$x_k \to y_k, k = 1, 2, \ldots, q$$

  - **autoassociation** ($x_k = y_k$):given partial or corrupted version of stored pattern and retrieve the original
  - **heteroassociation** ($x_k \neq y_k$): learn arbitrary pattern pairs and retrieve them
- Relevant issues: storage capacity vs accuracy

Pattern Recognition


Input pattern x
Unsupervised network for feature extraction
Feature vector y
Supervised network for classification
1
2
r
Outputs
(a)
Feature extraction
Classification
x
y
m-dimensional observation space
q-dimensional feature space
r-dimensional decision space
(b)

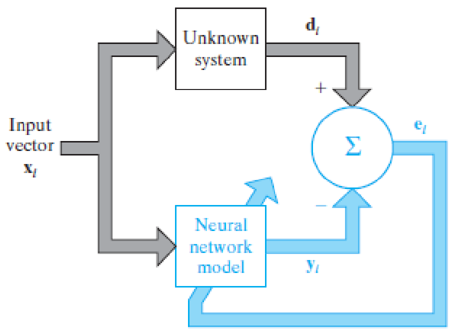
- Mapping between input pattern and a prescribed number of classes (categories)
- Two general types:
  - Feature extraction (observation space to feature space), then classification (feature space to decision space)
  - Single step (observation space to decision space)

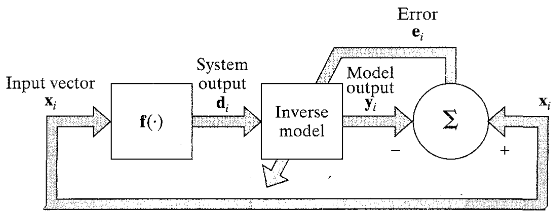
Function Approximation

- Nonlinear input-output mapping: $d = f(x)$ for an unknown f
- Given a set of labeled examples $\mathcal{T} = \{(x_i, d_i)\}_{i=1}^{N}$, estimate $F(.)$ such that

$$||F(x) - f(x)|| < \epsilon, \ \forall x$$

Function Approximation: System Identification and Inverse System Modeling




Error $\mathbf{e}_i$
Input vector $\mathbf{x}_i$
$\mathbf{f}(\cdot)$
System output $\mathbf{d}_i$
Inverse model
Model output $\mathbf{y}_i$
Σ
−
+
$\mathbf{x}_i$

- System identification: learn function of an unknown system

$$d = f(x)$$

- Inverse system modeling: learn inverse function

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

Memory

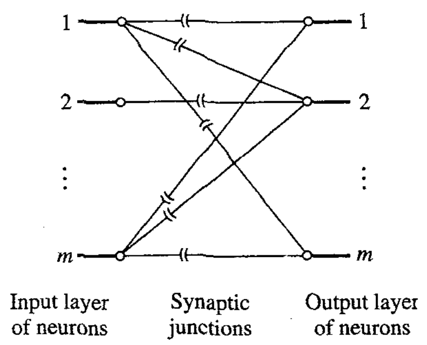
- In neurobiological context, **memory** referes to relatively enduring neural alterations induced by the interaction of an organism with its environment
- Memory needs to be **accessible** by the nervous system to influence behavior
- Activity patterns need to be stored through a learning process
- Types of memory: **short-term** and **long-term**

Memory

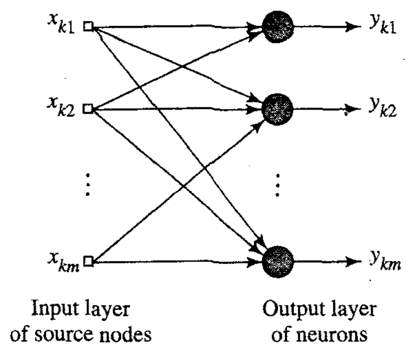
Associative memory offers following characteristics:

- Memory is distributed
- Both the stimulus (key) pattern and response (stored) pattern of an associative memory consists of data vectors
- Information is stored in memory by setting up a spatial pattern of neural activities across a large number of neurons
- Information contained in a stimulus not only determines its storage location in memory but also an address for its retrieval
- Memory exhibits a high degree of resistance to noise and damage of diffusive kind
- Interaction between individual patterns stored in memory

Associative Memory



(a) Associative memory model component of a nervous system



(b) Associative memory model using artificial neurons

- q pattern pairs: (x_k, y_k) for $k = 1, 2, \dots, q$
- Input (key vectors) $x_k = [x_{k1}, x_{k2}, \dots, x_{km}]^T$
- Output (memorized vector) $y_k = [y_{k1}, y_{k2}, \dots, y_{km}]^T$
- Association of key vector x_k with memorized vector y_k can be written in matrix form as

$$y_k = W(k)x_k, \quad k = 1, 2, \dots, q$$

Associative Memory

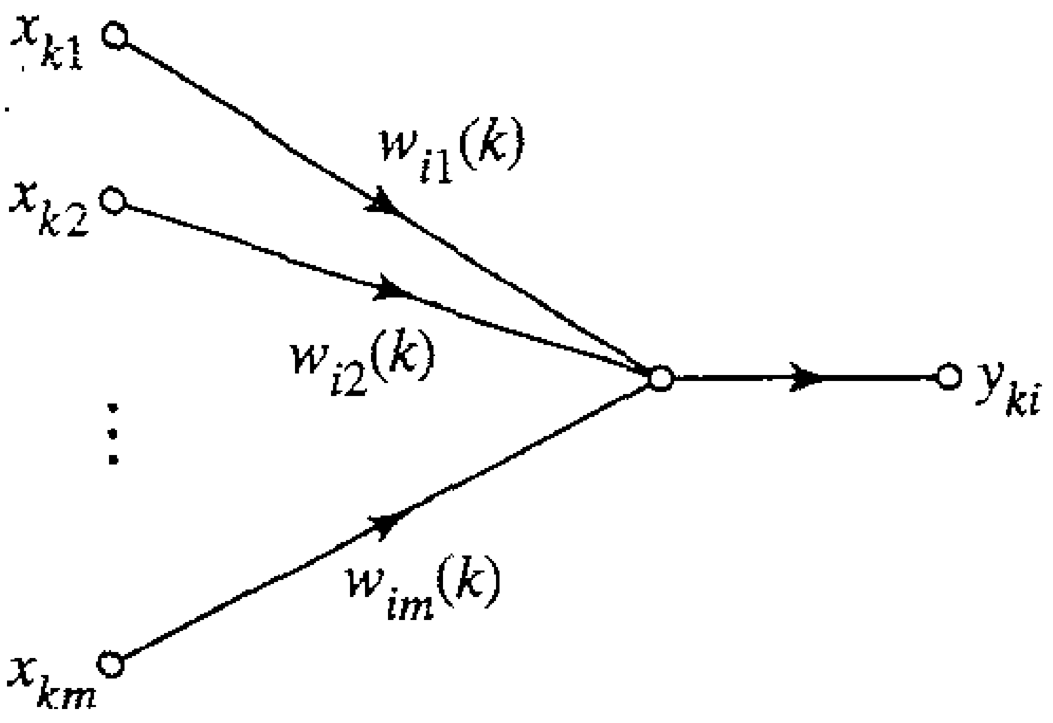


Figure 2: SFG of a linear neuron labeled i

Associative Memory

- $W(k)$ is a weight matrix determined solely by the input-output pair (x_k, y_k)
- $y_{ki} = \sum_{j=1}^{m} w_{ij}(k) x_{kj}, \quad i = 1, 2, \ldots, m$
- $y_{ki} = [w_{i1}(k), \ldots, w_{im}(k)] \begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{km} \end{bmatrix}, \ i = 1, 2, \ldots, m$
- $\begin{bmatrix} y_{k1} \\ y_{k2} \\ \vdots \\ y_{km} \end{bmatrix} = \begin{bmatrix} w_{11}(k) & w_{12}(k) & \ldots & w_{1m}(k) \\ w_{21}(k) & w_{22}(k) & \ldots & w_{2m}(k) \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1}(k) & w_{m2}(k) & \ldots & w_{mm}(k) \end{bmatrix} \begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{km} \end{bmatrix}$

Associative Memory

- With a single $W(k)$, we can only represent one mapping $(x_k \rightarrow y_k)$. For all pairs $(x_k, y_k)(k = 1, 2, \ldots, q)$ we need q such weight matrices

$$y_k = W(k)x_k, k = 1, 2, \ldots, q$$

- One strategy is to combine all $W(k)$ into a single memory matrix M

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

- Memory matrix M contains a piece of every input-output pair of activity patterns presented to memory
- $M_k = M_{k-1} + W(k), \quad k = 1, 2, \ldots, q \qquad M_0 = [0]$

Correlation Matrix Memory

- With q pairs (x_k, y_k), we can construct a candidate memory matrix that stores all q mappings as:

$$\hat{M} = \sum_{k=1}^{q} y_k x_k^T$$

, where $y_k x_k^T$ represents a outer product of vectors that results in a matrix, i.e.,

$$(y_k x_k^T)_{ij} = y_{ki} x_{kj}$$

- Outer product is an estimate of the weight matrix $W(k)$ that maps the output pattern y_k onto the input pattern x_k
- $\hat{M} = [y_1, y_2, \ldots, y_q] \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_q^T \end{bmatrix} = YX^T$

where $X = [x_1, x_2, \ldots, x_q]$ and $Y = [y_1, y_2, \ldots, y_q]$

Correlation Matrix Memory: Recall

- Let $\hat{M}$ denote memory matrix of an associative memory, and x_j be picked at random and reapplied as stimulus to the memory yielding response

$$y = \hat{M}x_j$$

- Since $\hat{M} = \sum_{k=1}^{q} y_k x_k^T$,

$$y = \sum_{k}^{q} y_k x_k^T x_j = \sum_{k=1}^{q} (x_k^T x_j) y_k$$

- $y = (x_j^T x_j) y_j + \sum_{\substack{k=1 \\ k \neq j}}^{q} (x_k^T x_j) y_k$

Correlation Matrix Memory: Recall

- Let each key patterns $x_1, x_2, \ldots, x_q$ be normalized to have unit energy

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

- Now memory response becomes $y = y_j + \underbrace{\sum_{\substack{k=1 \\ k \neq j}}^{q} (x_k^T x_j) y_k}_{\text{Noise term}}$
- Noisy term arises due to crosstalk between key vector x_j and all other key vectors stored in memory
- Noise component is responsible for making errors on recall

Correlation Matrix Memory: Recall

- $cos(x_k, x_j) = \dfrac{x_k^T x_j}{||x_k|| ||x_j||}$
- $||x_k||$ denotes the Euclidean norm of vector x_k, defined as

$$||x_k|| = (x_k^T x_k)^{1/2} = E_k^{1/2}$$

- Since, the key vectors are normalized to have unit energy, $cos(x_k, x_j) = x_k^T x_j$
- $v_j = \sum_{\substack{k=1 \\ k \neq j}} cos(x_k, x_j) y_k$
- Memory associates perfectly, if the key vectors forms an orthonormal set, i.e.,

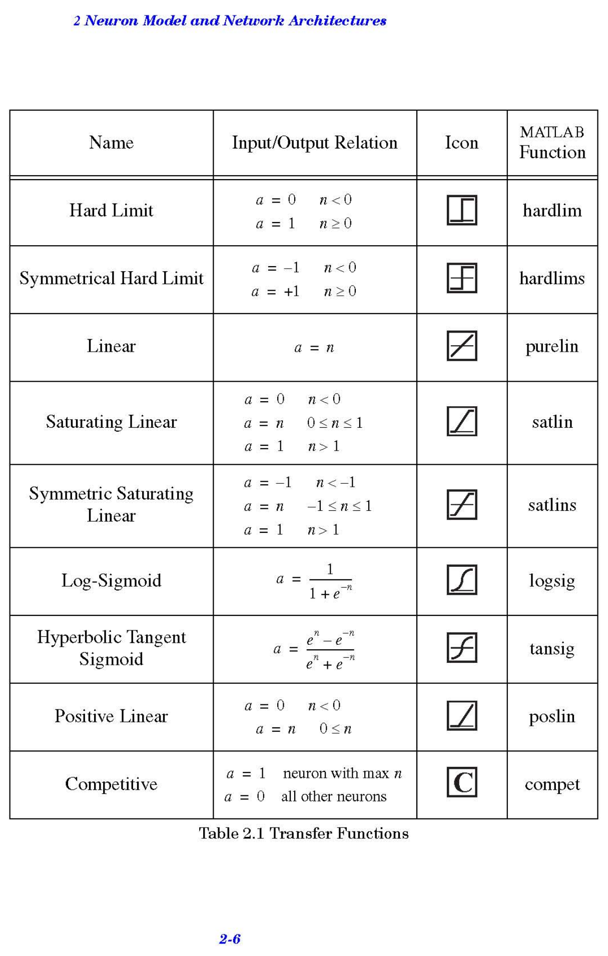
$$x_k^T x_j = \begin{cases} 1 & k = j \\ 0 & k \neq j \end{cases}$$

Correlation Matrix Memory: Recall

- What is the largest number of patterns that can be reliably stored in $\hat{M}$?
- Capacity is closely related with the rank of the matrix $\hat{M}$. The rank means the number of linearly independent column vectors (or row vectors) in the matrix
- Linear independence means a linear combination of the vectors can be zero only when the coefficients are all zero:

$$c_1 x_1 + c_2 x_2 + \ldots + c_n x_n = 0$$

  only when $c_i = 0, \forall i$
- Let a rectangular matrix $A_{l \times m}$, then rank(A), $r \leq min(l, m)$
- Since $\hat{M}_{m \times m}$, hence the rank of the memory matrix is limited by the dimensionality m
- Number of patterns that be reliably stored in a correlational matrix memory can never exceed the input space dimensionality


2 Neuron Model and Network Architectures

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 Transfer Functions

2-6

Figure 3: Activation function types

Summary

- Learning techniques:
  - Error-Correction Learning: $\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$
  - Hebb's hypothesis: $\Delta w_{kj}(n) = \eta y_k(n) x_j(n)$
  - Covariance hypothesis: $\Delta w_{kj}(n) = \eta(x_j - \bar{x})(y_k - \bar{y})$
  - Competitive learning:

$$\Delta w_{kj}(n) = \begin{cases} \eta(x_j - w_{kj}) & k \text{ wins} \\ 0 & otherwise \end{cases}$$

  - Boltzmann learning: $\Delta w_{kj} = \eta(\rho_{kj}^{+} - \rho_{kj}^{-}), \quad j \neq k$
- Learning paradigms:
  - Learning with a teacher: supervised learning
  - Learning without a teacher: reinforcement learning
  - Learning without a teacher: unsupervised learning
- Memory associates perfectly if the key vectors form an orthonormal set
- The number of patterns that can be reliably stored in a correlation matrix memory can never exceed the input space dimensionality