_/_/_

# SECTION - B
## UNIT - I

# Associative Memory

An associative memory is a brain like distributed memory that learns by association. Association takes one form → auto ~~and~~ or hetero.

- In auto-association, a NN is required to store a set of patterns by repeatedly presenting them to network. The network is subsequently presented with a partial description or distorted version of an original pattern stored in it. and the task is to retrieve or recall that particular pattern.

- Hetero-association differs in way → an arbitrary set of input patterns is paired with another arbitrary set of output patterns.

* Auto-association uses unsupervised learning.
* Hetero- " " supervised " .

Let $x_k$ denote a key pattern applied to an associative memory & $y_k$ denote a memorised pattern.
The pattern association performed by the network is described as →

$$x_k \rightarrow y_k \qquad k = 1, 2, 3, \ldots, q$$

where q is no. of input patterns

In auto-associative memory, the input & output spaces have same dimensionality but in hetero-associative memory, it may or may not be the case.

$y_k = x_k$ $\qquad\qquad$ $y_k \neq x_k$

## OPERATION

2 Phases

1. The ~~associative~~ storage phase : where training happens in accordance with the mapping.

$$x_k \rightarrow y_k \qquad k = 1, 2, 3, \ldots, q$$

2. The recall / retrieval phase : involves retrieval of a memorised pattern.

In auto-associative

Let $x$ represent a noisy / distorted version of $x_j$, consider response $y$ is produced. For perfect recall, $y = y_j$ but if it is not then memory is said to have made an error in recall.

## CAPACITY (STORAGE) OF NETWORK

# Storage Phase

$$x_k = [x_{k1}, x_{k2}, ---, x_{km}]^T$$

$$y_k = [y_{k1}, y_{k2}, ---, y_{km}]^T$$

$$\underline{y_k = W(k)\, x_k} \qquad k = 1, 2, 3, ---, q$$

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

$$\underline{y_{ki} = [w_{i1}(k) \quad w_{i2}(k) \quad --- \quad w_{im}(k)]} \begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{km} \end{bmatrix}$$

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

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

$$M_k = M_{k-1} + W(k)$$

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

$$\hat{M} = \begin{bmatrix} y_1 & y_2 & \cdots & y_q \end{bmatrix} \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_q^T \end{bmatrix}$$

$$\hat{M} = Y X^T$$

→ correlation matrix memory using outer product rule (Hebbian learning)

$$\hat{M}_k = \hat{M}_{k-1} + y_k x_k^T \qquad k = 1, 2, 3, \ldots, q$$

$$X = [x_1, x_2, x_3, \ldots, x_q]$$
$$Y = [y_1, y_2, y_3, \ldots, y_q]$$

# RECALL PHASE

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

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

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

$$y = Y_j + V_j$$

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

$$E_k = x_k^T x = 1 \qquad \text{where } k = 1, 2, 3, \ldots, q$$

$$\cos(x_k, x_j) = \frac{x_k^T x_j}{\|x_k\| \|x_j\|}$$

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

$$\cos(x_k, x_j) = x_k^T x_j$$

$$V_j = \sum_{\substack{k=1 \\ k \neq j}}^{m} \cos(x_k, x_j) y_k$$

CONDITION FOR PERFECT RECALL $\rightarrow$ $\cos(x_k, x_j) = 0$

i.e. $x_k$ and $x_j$ are orthogonal

# Radial-Basis Function Networks (RBF networks)

In RBF networks, we take a completely different approach by viewing the design of a neural network as a curve fitting problem in a high-dimensional space.
Acc. to this new point, learning is equivalent to finding a surface in a multi-dimensional space that provides a best fit to the training data with the criteria for best fit being measured in some statistical sense.
Correspondingly, generalization is equivalent to the use of this multi-dimensional surface to interpolate the test data. Such a view point is the motivation behind RBF networks.

In the context of a NN, the hidden units provide a set of functions that constitute an arbitrary basis for the input patterns when they are expanded into the hidden space. These are called radial-basis functions.

The construction of a RBF network in its most basic form involves 3 layers.

- Input Layer → Made of source nodes that connect network to its environment.
- Hidden Layer → Applies non-linear transformation from input space to hidden space. In most applications, the hidden space is of high dimensionality.
- Output Layer → It is linear that supplies response of network to activation pattern applied to input layer.

* The mathematical justification for the rational of a non-linear transformation followed by a linear transformation is due to COVER'S THEOREM.
It states a pattern classification problem cast in a high dimensional space is more likely to be linear separable than in low dimensional space.

# MATHEMATICAL MODEL

Consider a family of surfaces where each naturally divides an input space into 2 regions.
Let X denote a set of N patterns.

$$x_1, x_2, \dots, x_N$$

each of which is assigned to one to the two classes $X_1$ or $X_2$.
This binary partition of the points is said to be separable wrt family of surfaces if a surface exists that separates the points in class $X_1$ from those in class $X_2$.
For each pattern, $x \in X$, define a vector made up of a set of real valued functions →

$$\{\phi_i(x) \mid i = 1, 2, \dots, m_1\}$$

$$\phi(x) = [\phi_1(x), \phi_2(x), \dots, \phi_{m_1}(x)]$$

Suppose that pattern $x$ is a vector in $m_0$ dimensional input space. The vector $\phi(x)$ then maps points in $m_0$ dim. input space into corresponding points in a new space of dimension $m_1$.

We refer to $\phi_i(x)$ as hidden function.
Correspondingly, the space spanned by the set of hidden functions is riffered to as hidden space / feature space.

The binary partion $X_1$, $X_2$ of $X$ is said to be $\Phi$-separable if there exists a $m_1$ dimensional vector $w$ such that →

$$w^T \Phi(x) > 0 \qquad x \in X_1$$
$$w^T \Phi(x) < 0 \qquad x \in X_2$$

Hyperplane / separating surface → $w^T \Phi(x) = 0$
in hidden space.

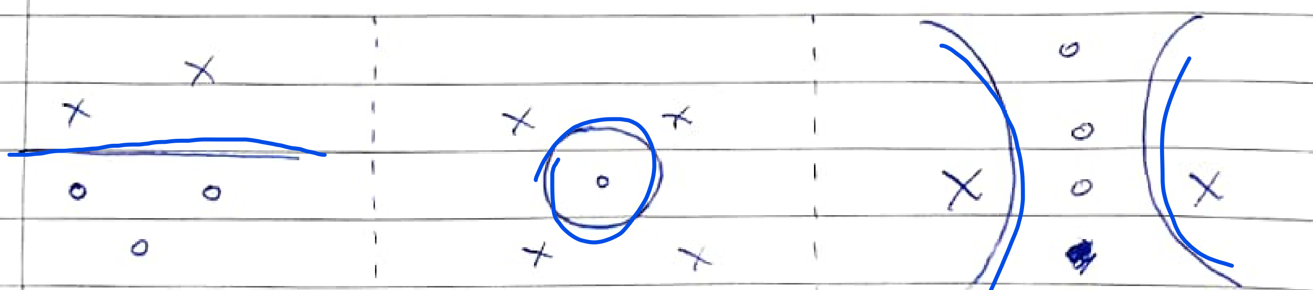
Consider a natural class of mappings obtained by using a linear combination of r-wise products of the pattern vector coordinates. The separating surfaces corresponding to such mappings are referred to as $r^{th}$ order rational varieties.
A rational variety of order r in a space of dim. $m_0$ is described by an $r^{th}$ degree homogenous eq$^n$ in the coordinates of input vector $x$ given by →

$$\sum_{0 \le i_1 \le i_2 \le \dots \le i_r \le m_0} a_{i_1 i_2 i_3 \dots i_r} x_{i_1} x_{i_2} \dots x_{i_r} = 0$$

where $x_i$ is the $i^{th}$ component of input vector $x$.

The $r^{th}$ order product of entries $x_i$ is called a monomial. For an input space of dim. $m_0$, there are $\frac{(m_0 - r)!}{m_0!\, r!}$ monomials

The separating surfaces described by above eq$^n$, are →
- Hyperplanes (1$^{st}$ order)
- Quadrices (2$^{nd}$ order)
- Hyperspheres (quadrices with certain linear constraints on coefficients)



Linearly separable | Spherically separable | Quadrically separable

Let $m_0$ denote the dim. of input space, then in the overall fashion, the network represents a map from $m_0$ dimensional input space to a single dim. output space given as : $S : R^{m_0} \rightarrow R^1$

We can think of this map as a hypersurface graph which is a subset of $(m_0+1)$ dim. space. →

$$\Gamma \subset R^{m_0+1}$$

The training and generalisation phase of the learning process can be viewed as →

1. The training phase constitutes the optimization of a fitting procedure for surface $\Gamma$ based on known data points presented to network in form of input / output patterns.

2. The generalisation phase is synonymous with interpolation b/w the data points, with the interpolation being performed along the constrained surface generated by the fitting procedure as the optimum approximation to the true surface $\Gamma$.

/ /

Now we are having a multi-variable interpolation problem in high dim. space. It can be stated as →

INTERPOLATION PROBLEM:

Given a set of N different points $\{x_i \in R^{m_0} \mid i = 1, 2, \ldots N\}$ and a corresponding set of N real numbers $\{d_i \in R^1 \mid i = 1, 2, \ldots N\}$. Find a function $F : R^N \to R^1$ which satisfies interpolation condition $F(x_i) = d_i$ where $i = 1, 2, \ldots, N$.

For strict interpolation, the interpolating surface is constrained to pass through all the training data points.

The RBF ~~techniqu~~ techniques consist of choosing a function that is of form →

$$F(x) = \sum_{i=1}^{N} w_i \, \phi(\|x - x_i\|)$$

$\phi(\|x - x_i\|)$ is set of N arbitrary RBFs.

$x_i \in R^{m_0}$ are centers to the RBFs

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1N} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2N} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ \vdots \\ d_N \end{bmatrix}$$

$$\phi w = X$$

$$\boxed{w = \phi^{-1}(X)}$$

There is large class of RBFs that is covered by Micchelli's theorem and considers given forms →

1. Multiquadrise

$$\phi(r) = \sqrt{r^2+c^2}$$

2. Inverse multiquadrise

$$\phi(r) = \frac{1}{\sqrt{r^2+c^2}}$$

3. Gaussian

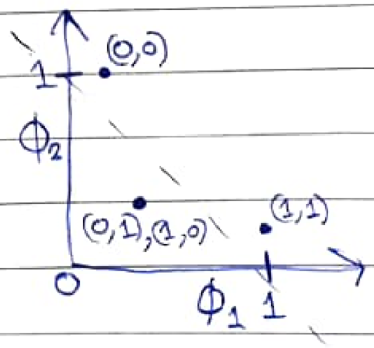
$$\phi(r) = e^{\left(\frac{-r^2}{2\sigma^2}\right)}$$

- Solution to XOR using RBF

$$\phi_1(x) = e^{\|x-t_1\|^2} \qquad t_1 = [1 \;\; 1]^T$$

$$\phi_2(x) = e^{-\|x-t_2\|^2} \qquad t_2 = [0 \;\; 0]^T$$

| $x$ | $\phi_1(x)$ | $\phi_2(x)$ |
|---|---|---|
| (0,0) | 0.1353 | 1 |
| (0,1) | 0.3678 | 0.3678 |
| (1,0) | 0.3678 | 0.3678 |
| (1,1) | 1 | 0.1353 |




Supervised learning is illposed hyperspace reconstruction.

| * RBF | MLP |
| --- | --- |
| • Single hidden layer<br>• Computational nodes | • 1 or more hidden layer |

- Computational nodes of MLP share a common model. For RBF, the computational nodes are quite different from others and serve a different purpose.
- The argument of activation function of each hidden unit in RBF network computes Eucledian norm whereas activation fn of hidden unit in MLP computes the inner product.
- MLPs construct global approximations and RBFs using exponential decaying localised non linear functions construct local approximations.

- Learning strategies in RBF

1. Fixed centers selected at random
2. Self organized selection of centers
3. Supervised selection of centers

* Global order can arise from local interactions.
* Modifications in synaptic weights tend to self-amplify.
* Limitation of resources leads to competition among synapses and thus the selection of most fittest at the expense of others.
* Modifications in synaptic weights tend to co-operate.
* Order and structure in the activation patterns represent redundant information that is acquired by network in form of knowledge.

Basic philosophy of PCA

Let X denote a m-dimensional vector representing

We assume the vector X to have 0 mean.

$$E[X]=0$$

Let q denote unit vector also of dimension m onto which vector X is to be projected. This projection is defined by eq$^n$ →

$$A = X^T q = q^T X$$

$$\|q\| = (q^T q)^{1/2} = 1$$

$$E[A] = q^T E[X] = 0$$

$$\sigma^2 = E[A^2]$$
$$\sigma^2 = E[(q^T X)(X^T q)]$$
$$\sigma^2 = q^T E[X X^T] q$$
$$\boxed{\sigma^2 = q^T R q}$$

where $R = E[XX^T]$

↳ corelation matrix

$$\boxed{\sigma^2 = \phi(q)}$$

$$\boxed{Rq = \lambda q}$$

$\lambda$ is the scaling factor representing the eigen value of R

Matrix form →

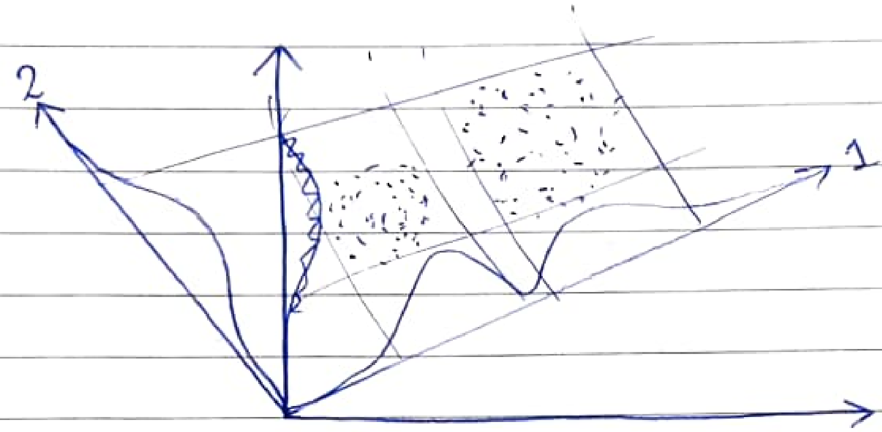
$$Q^T R Q = \Lambda$$

$$R = \sum_{i=1}^{m} \lambda_i q_i q_i^T$$

The important findings from mathematical analysis reveal →

1. Eigenvectors of corelation matrix ~~are~~ pertaining to zero mean random vector X, define the unit vectors $q_j$ representing principal directions along which $\phi(q)$ have the extremal values. ~~The asso~~

2. The associated eigen values define the extremal values of $\Phi(q)$.




With $m$ possible solutions for unit vector $q$, we find that there are $m$ possible projections of data. →

$$a_j = q_j^T X = X^T q_j \quad : j = 1, 2, \ldots, m$$

$$a = [a_1 \ a_2 \ \ldots \ a_m]^T$$

$$a = [X^T q_1 \ X^T q_2 \ \ldots \ X^T q_m]^T$$

$$a = Q^T X$$

$$X = Qa$$

$$X = \sum_{j=1}^{m} a_j q_j$$

$$\hat{X} = \sum_{j=1}^{\ell} a_j q_j$$

$$\hat{X} = \begin{bmatrix} q_1 & q_2 & \cdots & q_l \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_l \end{bmatrix}$$

$$l \le m$$

so, $$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_l \end{bmatrix} = \begin{bmatrix} q_1^T \\ q_2^T \\ \vdots \\ q_l^T \end{bmatrix} X$$

~~ENCODER~~

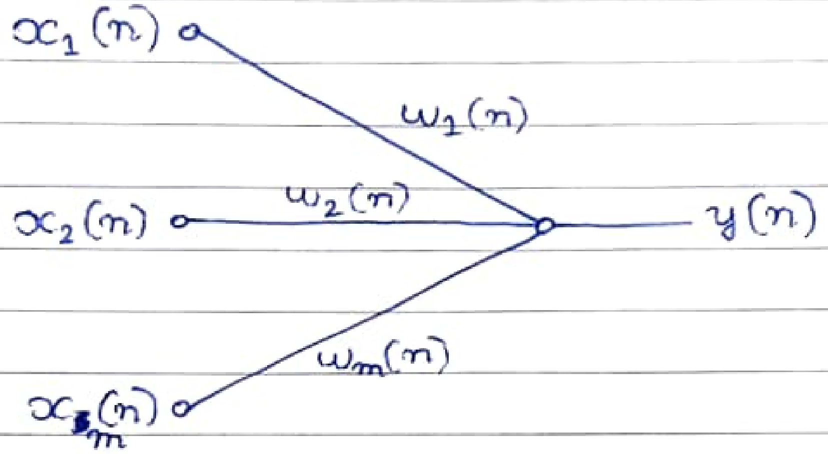
ENCODER

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix} \Rightarrow \begin{bmatrix} q_1^T \\ q_2^T \\ \vdots \\ q_l^T \end{bmatrix} \Rightarrow \cancel{\begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_l \end{bmatrix}} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_l \end{bmatrix}$$

DECODER

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_l \end{bmatrix} \cancel{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_l \end{bmatrix}} \Rightarrow \begin{bmatrix} q_1 & q_2 & \cdots & q_l \end{bmatrix} \Rightarrow \begin{bmatrix} \hat{X}_1 \\ \hat{X}_2 \\ \vdots \\ \hat{X}_l \end{bmatrix}$$

$$y = \sum_{i=1}^{m} w_i x_i$$




$$w_i(n+1) = w_i(n) + \eta\, y(n)\, x_i(n) \qquad : i = 1, 2, 3, \ldots, m$$

$$w_i(n+1) = w_i(n) + \eta\, y(n) \left[ x_i(n) - y(n) w_i(n) \right]$$

$$y(n) = X^T(n) W(n) = W^T(n) X(n)$$

$$W(n+1) = W(n) + \eta \left[ X(n) X^T(n) W(n) - W^T(n) X^T(n) W^2(n) \right]$$