Neural Network

DATE: __/__/__
PAGE:

UNit 5

paradigms of Associative memory

Mathematic Pattern

pattern term is used in context of neural networks to mean a set of activations across a pool of units (neurons)

ANN combines biological principles with advanced statistics to solve problems in domains such as pattern recognition and game-play.

pattern →

Pattern is the quantitative description of an object, event.

Pattern can be classified as spatial or temporal pattern.

1) spatial :- Example are pictures, weather, map, video, games

2) Temporal :- Example are speech signal, ECG etc. These pattern involve the ordered sequence of data appeared in time.

* pattern classification :-

The main goal of pattern classification is to assign a physical object, event or phenomenon to one of the prespecified classes

Good Write

DATE: __/__/____
PAGE: _____

for example
We classify various objects into different category like living, non-living thing etc.

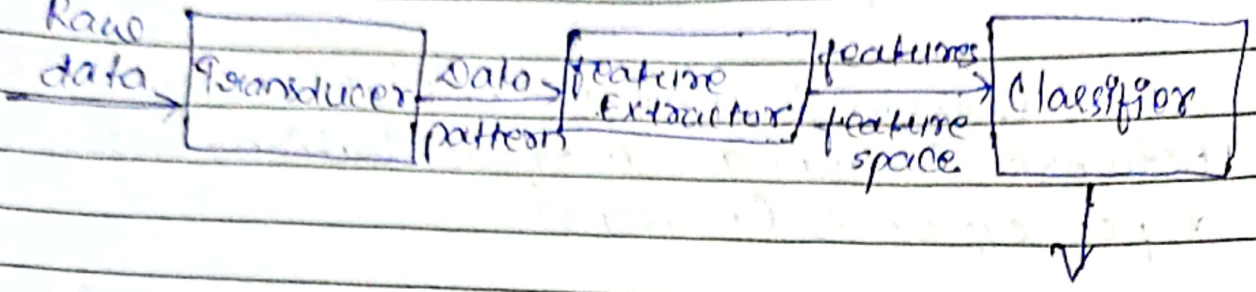


fig:- Block diagram of classification sytle.

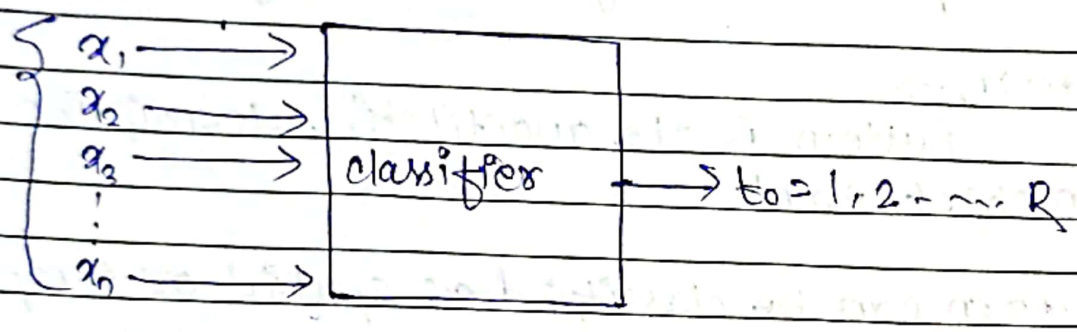


fig:- General diagram of pattern classifier.

→ Transducer generally converts one form of energy to another form of energy.
Here the input is raw data space $x = [x_1, x_2 \ldots x_n]$ & O/P is transducer is dia in pattern space that belong to a certain category that is

$$t_o = 1, 2 \ldots R$$

converted data at the o/p can be compressed. Compressed data is called feature.

→ classifier at the end, convert or map the feature to the particular class

General concept of Associative Memory

It is a storehouse of associated patterns which are encoded in some form.

- An associative memory is a content-addressable structure that maps a set of input patterns to a set of o/p patterns

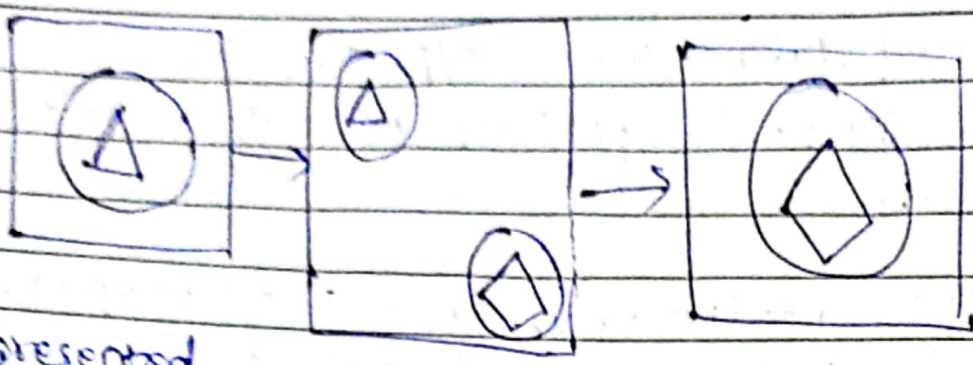
content addressable memory structure is a kind of memory structure that enables the recollection of data based on the intensity of similarity between the input pattern and the pattern stored in the memory. This kind of memory is robust & fault-tolerant.

They are of two types

1) Auto-associative memory

It recovers a previously stored pattern that most closely relates to the current pattern. auto associative correlator.

DATE: __/__/____
PAGE: _____



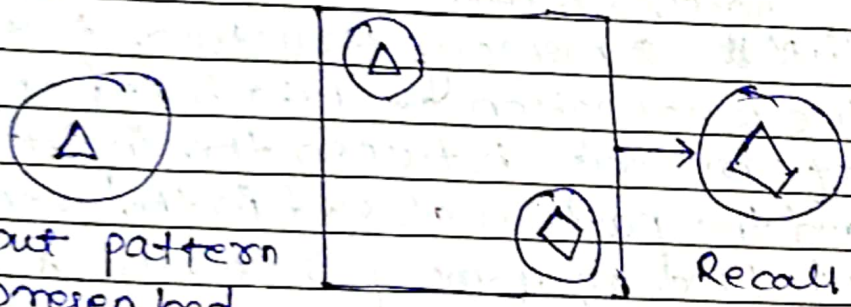
presented distorted pattern.

Recall of perfect pattern.

Hetero-associative memory:-

The recovered patter is generally different from the input pattern not only in type and format but also in content.

Also known as hetero-associative correlator.



input pattern presented

Recall of associated pattern

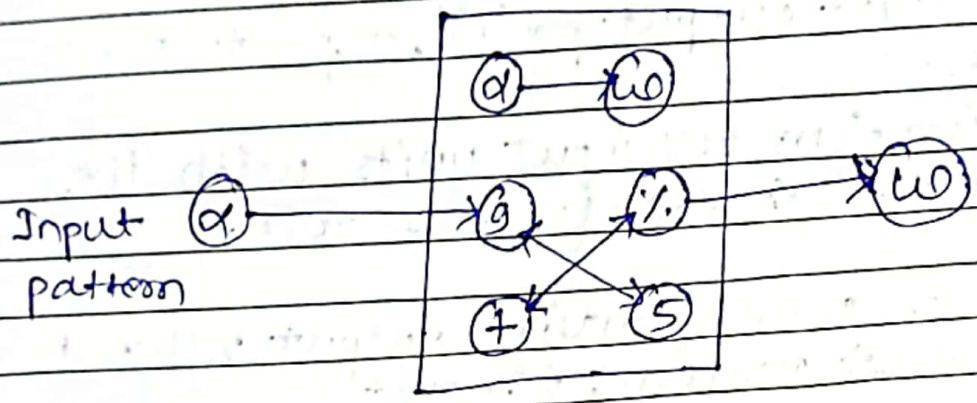
Neural Associative Memory are usually used to implement these memory model.

These models follows distinct architecture to memorize data.

DATE: / /
PAGE:

Working of Associative Memory

Associative memory is a depository of associated pattern which in some form. If the depository is triggered with a pattern the associated pattern pair appear at the output. The input could be an exact or partial representation of a stored pattern.



If the memory is produced with an input pattern, may say α, the associated pattern w is recovered automatically

- Encoding or memorization.
- Errors & noise
- Performance Measure (memory capacity & content-addressability mutually orthogonal)

(performing highly parallel & distributed computation)

Hebbian learning:-

It was proposed by Donald O Hebb. It is one of the first and also easiest learning rule in the neural network used for pattern classification. It's a single layer NN having one input layer with n units and one output layer with

Good Write

DATE: __/__/____
PAGE: _____

one unit.

The rule works by updating the weights between neurons in the neural network for each training sample.

Hebbian Learning Rule Algo:-

1. Set all weights to zero, $w_i = 0$ for $i = 1$ to n and bias to zero.

2. for each input vector, s(input vector): t (target output pair), repeat step 3 to 5.

3. Set activations for input units with the input vector $x_i = s_i$ for $i = 1$ to n.

4. set the corresponding output value to the output neuron; i.e. $y = t$

5. Updata weight and bias by applying Hebb rule for all $i = 1$ to n.

$$w_i(new) = w_i(old) + x_i y$$
$$b(new) = b(old) + y$$

practice Numerical for it.

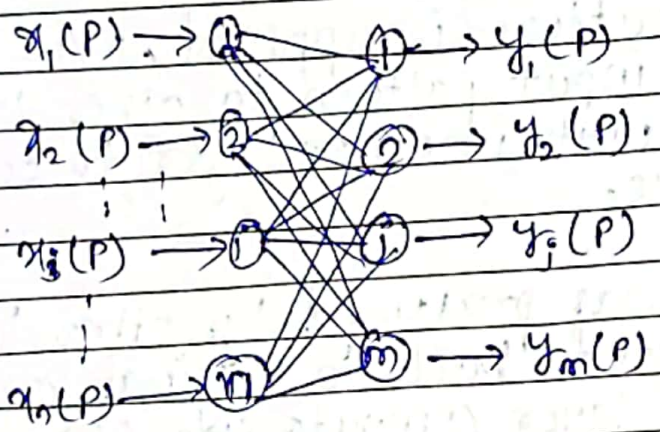
DATE: / /
PAGE:

Bidirectional Associative Memory

BAM is a hetero associative, content addressable memory consists of 2 layers. For an input pattern, it returns another pattern which is potentially of a different size.

BAM Architecture

When BAM accepts an input of n dimensional vector X from set A then the model recalls m-dimensional vector Y from set B. Similarly when Y is treated as input, the BAM recall X.



(a) forward direction.

BAM model can perform both auto & hetero associative recall of stored information

Weight matrix can be calculated as

$W_K = x_K^T y_K$ (to store a single associated patterns.

DATE: __/__/____
PAGE: _____

$W = \alpha \sum_{K=1}^{P} x_K$ (to store simultaneously several associated pattern pair)

This process is generally called storage or encoding.

After encoding N/w can be used for decoding.

The decoding process is as follows

1) Input layer can be applied, either on x layer or on the y layer depending on direction of propagation.

2) When input pattern is applied, N/w will propagate the input pattern to other layer allowing the units in other layer to compute their o/p value.

3) Pattern that was produced by other layer is then propagated back to original layer & in original layer compute the o/p value.

4) This process is repeated until further propagation & computation do not result in a change in the associations of units in both layer where final pattern pair is one of the stored associated pattern pair.

5) Final pattern pair that will be produced by the N/w depend on initial pattern pair &

Good Write

and connection weight matrix.

* Addition & Deletion of pattern pairs.

- Given a set of pattern pair (x_i, y_i) for $i = 1, 2, \ldots n$ & set of correlation matrix M.
- a new pair (x', y') can be added.
- an existing pair (x_j, Y_j) can be deleted from memory model.

Addition

Add a new pair (x', y') to existing correlation matrix M, then new matrix M_{new} is given by

$$M_{new} = X_1^T Y_1 + x_2^T Y_2 + \cdots x_n^T Y_n + x'^T Y'$$

Deletion

Subtract the matrix corresponding to an existing pair (x_j, Y_j) from matrix M then new correlation Matrix is given by

$$M_{new} = M - (X_j^T Y_j)$$

* The addition & deletion of information is similar to the functioning of the system as a human memory exhibiting learning & forgetfulness.

Good Write

DATE: __/__/____
PAGE: _____

* Discrete BAM:-

N/w propagate an input pattern X to Y layer and Y layer compute the net input using

Input is calculated using

$$y_i = \sum_{i=1}^{m} x_i w_{ij} \qquad i = 1, 2 \ldots m \qquad j = 1, 2 \ldots m$$

O/P

$$y_i(t+1) = \begin{cases} +1 & \text{if } y_i > 0 \\ y_i(t) & \text{if } y_i = 0 \\ -1 & \text{if } y_i < 0 \end{cases}$$

pattern produced on Y layer is then propagated back to X layer & X layer compute values.

Input

$$x_j = \sum_{i=1}^{n} y_i w_{ij}$$

O/P

$$x_j(t+1) = \begin{cases} +1 & \text{if } x_j > 0 \\ x_j(t) & \text{if } x_j = 0 \\ -1 & \text{if } x_j < 0 \end{cases}$$

Good Write

DATE: __/__/____
PAGE: _____

Energy function for OBAM

$$E = -\sum_{i=1}^{m}\sum_{j=1}^{n} x_i w_{ij} y_j$$

* continuous BAM

In this, the units use hyperbolic tangent O/P function. The units in X-layer have an extra external input I_i while units in Y-layer have an extra external input J_j

for $i = 1, 2, \ldots m$ & $j = 1, 2 \ldots m$

These extra input lead to a modification in the computation of net input

$$x_i = \sum_{j=1}^{n} y_j w_{ij} + I_i$$

$$y_j = \sum_{j=1}^{m} x_i w_{ij} + J_j$$

Limitation of Bidirectional Associative Memory

- Storage capacity of the BAM: stored number of associations should not be exceeds the number of neuro
- Incorrect convergence

Always the closet association may not to be produced by BAM.

Good Write

DATE: __/__/____
PAGE: _____

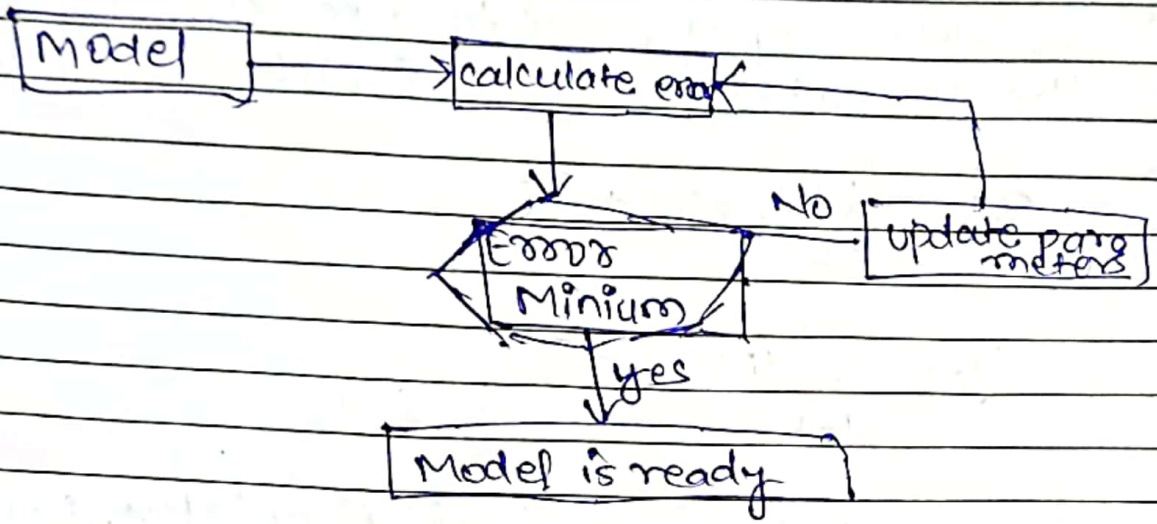
Back-propagation

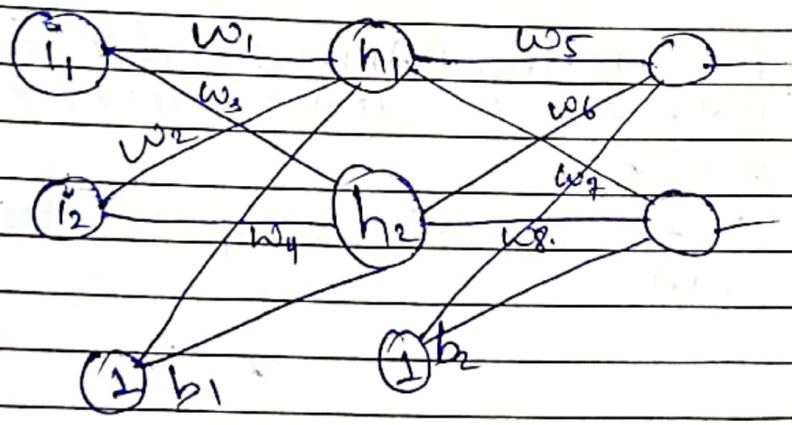
Backpropagation Network is multi layer feed forward N/w with diff. transfer function in ANN.

It is kind of gradient descent technique with backward error propagation. Its a supervised learning and an implementation of delta learning rule.

gradient descent (differentiation)

gradient descent method is used to update the weight by reducing the error function





Good Write

DATE: __/__/____
PAGE: _____

Credit Assignment Problem

Neural network face the "credit assignment problem" in situation in which only incomplete performance evaluations are available. The credit assignment problem is that a network should assign credit or blame for its behaviour according to the contribution to the network performance.

If any distributed system or any distributed ~~system~~ network is there then it is a very common practice to give some credits or blame to each and every internal decision so that it can be reflected on the overall outcomes of the networks.

→ Internal decisions are dependent on many actions. Among such, some actions are very prominent to get the outcome of internal decision. In this case credit assignment is divided into two parts:-

i) Temporal:- If we have given credit or blame for outcomes of the internal decision then this is called temporal credit assignment problem.

ii) Structural:-
If we have given credit or blame for actions of the internal decision, then its is called structural credit assignment.

DATE : __/__/____
PAGE : _____

temporal :- to get the selected outcomes of the internal decisions.

⇒ structural :- to get the selected actions of the internal decision.

DATE: / /
PAGE:

Generalized delta rule.

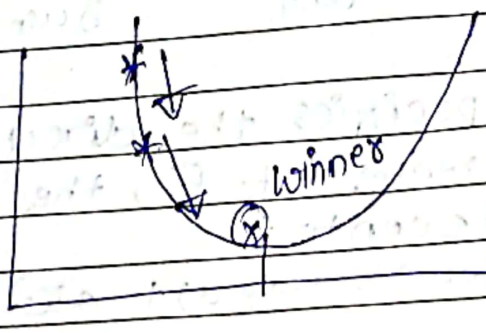
linearly separable and non-linearly separable.

perceptron rules fails for non-linearly separable data. And here comes the delta rule.

Here delta rule tries to converge the solution towards a best-fit approximation to the target concept.

The key idea behind the delta rule is to use gradient descent to search the hypothesis space of possible weight vectors to find the weights that best fit the training examples.

Gradient descent



$O(\vec{x}) = \vec{w} \cdot \vec{x}$

way to define error to derive a weight learning rule for linear units.

$$E\vec{w} \equiv \frac{1}{2} \sum (t_d - o_d)^2$$

t_d = desired output

o_d = actual output.

DATE: __/__/____
PAGE: ____

D is the set of training example

* How can we calculate the direction of steepest descent along the error surface.

The direction can be found by computing the derivative of E with respect to the each component of the vector $\vec{w}$.

The vector derivative is called gradient of E with respect to ($\vec{w}$)

$$\Delta E(\vec{w})$$

$$\Delta E(\vec{w}) = \left[\frac{dE}{\partial w_0}, \frac{dE}{\partial w_1} \cdots \frac{dE}{\partial w_n}\right]$$

Such gradient specifies the direction of steepest increase of E, the training rule gradient decent is

$$w(\text{new}) = w(\text{old}) + \Delta\vec{w}$$

where $\Delta\vec{w} = -\eta \Delta E(\vec{w})$

η = learning rate, it determines the step size in the gradient descent search.

$\Delta E(\vec{w})$ = derivative of error with respect to each component of weight vector.

Good Write

The negative sign is present because we want to move the weight vector in the direction that decrease E (error)

$\Delta E(\vec{w}) = \frac{dE}{dw_i}$ where i is the respective weights

We have to get the value of the given derivative term: $\frac{dE}{\partial w_i}$

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{w_i} \frac{1}{2} \sum (t_d - o_d)^2$$

$$= \frac{1}{2} \sum \frac{1}{dw_i} (t_d - o_d)^2$$

$$= \frac{1}{2} \sum (t_d - x_i \cdot w_i)^2$$

$$= \frac{1}{2} \sum 2 (t_d - o_d) \cdot \frac{d}{dw_i} (t_d - x_i \cdot w_i)$$

$$= \sum (t_d - o_d) (-x_{id})$$

$$= -\sum (t_d - o_d)(x_{id})$$

Good Write

DATE: __/__/____
PAGE: _____

$$\Delta w_{ji} = \eta \sum_{d \in D} (t_d - o_d) \cdot x_{id}$$

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

Back propagating Algorithm.

- To derive the equation for weight updation, stochastic gradient descent rule is used.

- For each training example d, every weight w_{ji} is updated by adding Δw_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

Firstly Error is calculated.

$$E_d(\vec{w}) = \frac{1}{2} \sum (t_k - o_k)$$

Notations used:

x_{ji} = the i^{th} input to unit j

w_{ji} = the weight associated with the ith input to unit j

Good Write

$net_j = \sum w_{ji} x_{ji}$

o_j = the output computed by unit j.

t_j = the target output for unit j.

σ = the sigmoid function.

outputs = the set of units in the final layer of the network

Downstream (j) = the set of units whose immediate inputs include the output of unit j.

To begin, notice that weight w_{ji} can influence the rest of the network only through net_j. Therefore applying chain rule as,

$$\frac{\partial E_d}{\partial w_{ij}} = \frac{\partial E_d}{\partial w_{ij}} = \frac{\partial E_d}{\partial net_j} \times \frac{\partial net_j}{\partial w_{ij}}$$

$$= \frac{\partial E_d}{\partial net_j} \times x_{ji}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ji}$$

DATE: __/__/____
PAGE: _____

$net_j = \sum w_{ji} x_{ji}$

o_j = the output computed by unit j.

t_j = the target output for unit j.

σ = the sigmoid function.

outputs = the set of units in the final layer of the network

Downstream(j) = the set of units whose immediate inputs include the output of unit j.

To begin, notice that weight w_{ji} can influence the rest of the network only through net_j. Therefore applying chain rule as,

$$\frac{\partial E_d}{\partial w_{ij}} = \frac{\partial E_d}{\partial w_{ij}} = \frac{\partial E_d}{\partial net_j} \times \frac{\partial net_j}{\partial w_{ij}}$$

$$= \frac{\partial E_d}{\partial net_j} \times x_{ji}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ji}$$

Good Write

DATE: __/__/____
PAGE: _____

To derive a convenient expression for $\frac{\partial E_d}{\partial net_j}$

We consider two cases in turn:

- Case 1, where unit j is an output unit for the network, and
- case 2; where unit j is an internal unit of the network

Case 1 :-

j as the output unit weight.

- Just as w_{ji} can influence the rest of the network only through net_j, net_j can influence the network through O_j. Here, we can invoke the chain rule again to write,

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial O_j} \times \frac{\partial O_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial O_j} = \frac{\partial}{\partial O_j} \frac{1}{2} \sum (t_k - O_k)^2$$

$$= \frac{1}{2} 2(t_j - O_j) \frac{\partial (t_j - O_j)}{\partial O_j}$$

DATE: __/__/____
PAGE: _____

$$= -(t_j - o_j)$$

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial (net_j)}$$

$$\frac{\partial \sigma(y)}{\partial (x)} = \sigma(y)(1-\sigma(y))$$

$$= \sigma(net_j)(1 - \sigma(net_j))$$

$$= o_j(1 - o_j)$$

$$\boxed{\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)\, o_j(1 - o_j)}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$$= -\eta \frac{\partial E_d}{\partial net_j} x_{ji}$$

$$\delta_j = (t_j - o_j)\, o_j(1 - o_j)$$

$$\Delta w_{ji} = \eta\, \delta_j\, x_{ji}$$

Good Write

DATE: __/__/____
PAGE: _____

Case 2:-

Modify the weights at hidden layer.

$$\frac{\partial E_d}{\partial net_j} = \sum_{K \in Downstream(j)} \frac{\partial E_d}{\partial net_K} \times \frac{\partial net_K}{\partial net_j}$$

$$= \sum_{K \in Downstream(j)} -\delta_K \frac{\partial net_K}{\partial net_j}$$

$$= \sum_{K \in Downstream(j)} -\delta_K \frac{\partial net_K}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$= \sum_{K \in Downstream} -\delta_K w_{Kj} \frac{\partial o_j}{\partial net_j}$$

$$= \sum_{K \in Downstream} -\delta_K w_{Kj} o_j(1-o_j)$$

$$\frac{\partial net_K}{\partial o_j} = \frac{\partial x_{Kj} \cdot w_{Kj}}{\partial o_j} = \frac{\partial o_j w_{Kj}}{\partial o_j}$$

$$\frac{\partial o_j}{\partial (net_j)} = o_j(1-o_j)$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ji}$$

$$\Delta w_{ji} = \eta o_j(1-o_j) \sum \delta_K w_{Kj} x_{ji}$$

Kolmogorov Theorem

Chapman-Kolmogorov equation was derived independently by both the British mathematician Sydney Chapman and the Russian mathematician Andrey Kolmogorov

Suppose $X_n = n = 0, 1, 2 \ldots$ is a homogenous Markov chain then,

$$P_{ij}^{(m+n)} = \sum P_{ik}^{(m)} P_{kj}^{(n)}$$

P_{ij} = conditional probability that the Markov chain goes from state i to j in m+n stem.

$P_{ik}^{(m)}$ = conditional probabilities that of reaching intermediary state K in m steps.

$P_{kj}^{(n)}$ = conditional probabilities of K reaching to j in steps n.

Good Write

n-step transition probability.

$$P_{ij}^{(n)} = P(X_{m+n} = j \mid X_m = i)$$

That is

$$P(X_{m+n} = j \mid X_0 = i)$$

$$= \sum_k P(X_m = k \mid X_0 = i)\, P(X_{m+n} = j \mid X_m = k)$$

proof

The (i,j)th entry of the matrix $P^{(m+n)}$ is given by,

$$P_{ij}^{(m+n)} = P(X_{m+n} = j \mid X_0 = i)$$

$$= \sum_k P(X_{m+n} = j, X_m = k \mid X_0 = i)$$

$$= \sum_k \frac{P(X_{m+n} = j, X_m = k, X_0 = i)}{P(X_0 = i)}$$

$$= \sum_k \frac{P(X_{m+n} = j, X_m = k, X_0 = i)}{P(X_m = k, X_0 = i)} \cdot \frac{P(X_m = k, X_0 = i)}{P(X_0 = i)}$$

DATE: __/__/____
PAGE: _____

Use

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$= \sum_k P(X_{m+n} = j \mid X_m = k; X_0 = i) \cdot P(X_m = k \mid X_0 = i)$$

$$= \sum_k P(X_{m+n} = j / X_m = k) \cdot P(X_m = k \mid X_0 = i)$$

by Markov property

Mathematically the Markov property is stated as

$$P(X_{t+1} = j \mid X_t = i, X_{t-1} = i, \ldots X_0 = i_0) = P(X_{t+1} = j \mid X_t = i)$$

for all $t = 1, 2, 3 \ldots$ and for all states.

the future depends only on the present, not the past behaviour.

$$= \sum_k P_{kj}^{(n)} P_{ik}^{(m)}$$

$$= \sum_k P_{ik}^{(m)} P_{jk}^{(n)}$$

Good Write

In the matrix notation, it can be written as,

$$P^{(m+n)} = P^{(m)} P^{(n)} \quad \text{or} \quad P(m+n) = P(m)\,P(n)$$

Its differential equation.

$$P_{ij}(m+n) = \frac{\partial}{\partial n} P_{ij}(m+n)$$

$$= \sum_k P_{ik}(m) \frac{\partial}{\partial n} P_{kj}(n)$$

At $n=0$, we have.

$$P_{ij}'(m) = \sum_k P_{ik}(m)\, a_{kj}$$

where $a_{kj} = \frac{d}{dn} P_{kj}(n)$ at $n=0$

Or in matrix notation

$$P'(m) = P(m)\,A$$

i.e. $$\frac{d}{dn} P = PA$$

Let $\{X_n, n \geq 0\}$ be a homogeneous Markov chain with TPM (transition probability matrix) $P = (p_{ij})$ and n-step TPM $P^{(n)} = p_{ij}^{(n)}$ where

$$p_{ij}^{(n)} = P(X_n = j \mid X_0 = i) \text{ & } p_{ij}^{(1)} = p_{ij}$$

Then the following properties hold:

(1) $P^{(m+n)} = P^{(m)} P^{(n)}$

(2) $P^{(n)} = P^n$, i.e., n-steps TPM: $P^{(n)}$ is equal to the n^{th} power of the one-step TPM P.

(2) we have to prove by induction

By defination.

we have,

$P = P^{(1)}$

Thus the assertion $P^{(n)} = P^n$ is true for $n = 1$

Assume $P^{(m)} = P^m$ for some positive integer m.

Then by using result $P^{(m+n)} = P^{(m)} P^{(n)}$ (ie Chapman-Kolmogorov equat

Good Write

DATE: __/__/____
PAGE: _____

We have.

$$P^{(m+1)} = P^{(m)}(P^{(1)}) = P^{(m)}P = P^{m+1}$$

Thus the assertion $P^{(n)} = P^n$ is also true for $m+1$.

Hence prove.

The TPM of the Markov chain with three states 1, 2, 3 is

$$P = \begin{bmatrix} 0.1 & 0.5 & 0.4 \\ 0.6 & 0.2 & 0.2 \\ 0.3 & 0.4 & 0.3 \end{bmatrix}$$

Compute 2-step TPM

$$P^2 = P \cdot P$$

$$= \begin{bmatrix} 0.43 & 0.31 & 0.26 \\ 0.24 & 0.42 & 0.34 \\ 0.36 & 0.35 & 0.29 \end{bmatrix}$$

$$P = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1/2 & 1/2 & 0 \end{bmatrix} \end{array}$$

$$P^2 = P \cdot P$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \end{bmatrix}$$

* Recurrent Autoassociative Memory :-

In order to produce an improved association, there is a need for suppressing the output noise at memory output in linear associative memory. This can be done by recycling of output to input & by thresholding the O/P.

→ The repetive process of recycling of the output to the input through the N/w can be performed by a Recurrent Neural Networks (RNN).

Similar as linear associative memory. But it has feedback that is dynamic.

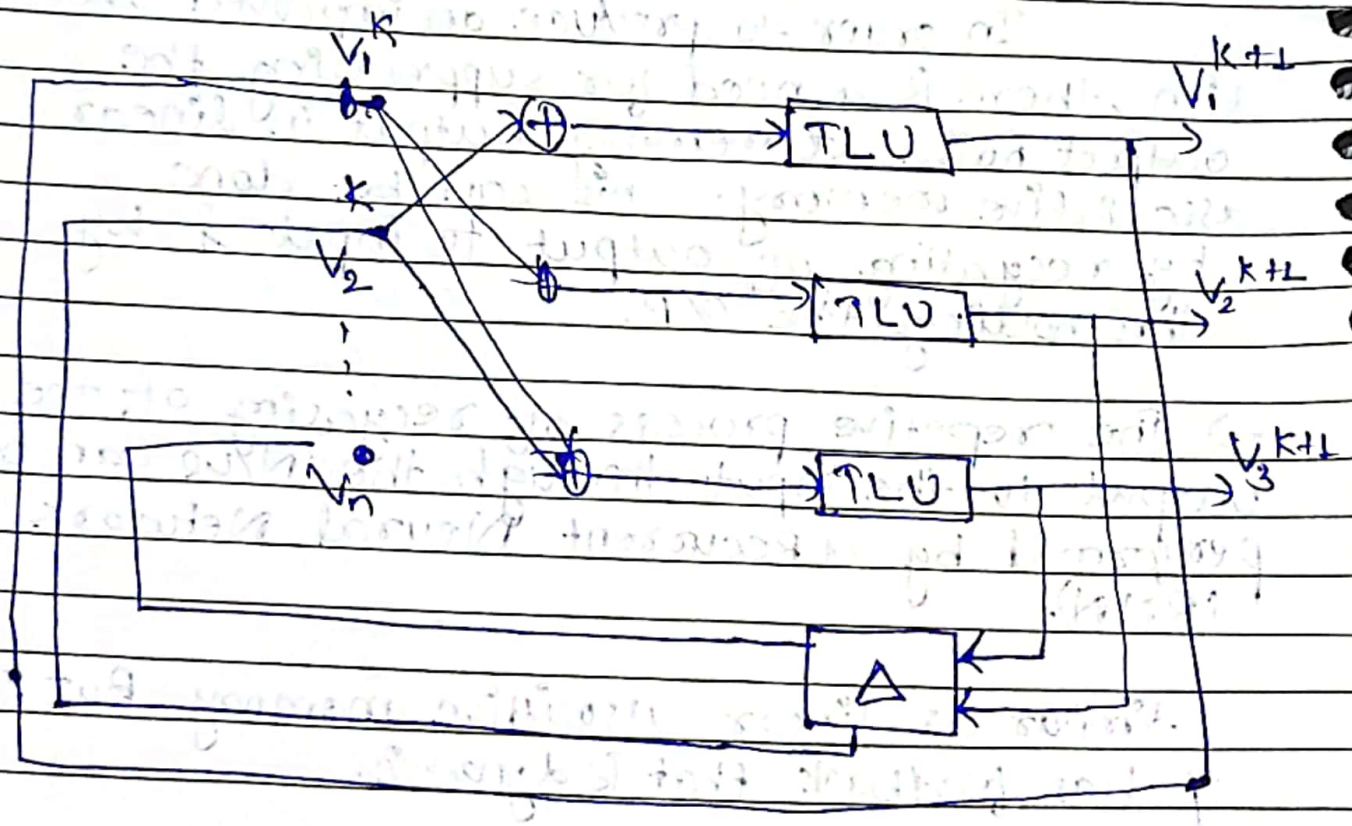
DATE: __/__/____
PAGE: _____

2 modes.

Asynchronous Synchronous.

① Under the asynchronous update mode, only one neuron is allowed to compute or change states at a time, and then all output are delayed by time u. that is produced by unit delay element in the feed back loop.

② In synchronous mode.



Good Write

Explanation of figure dots represent the neuron TLU-threshold logic unit & these are used because associative memories are updated in discrete time.

$\Delta \longrightarrow$ unit delay in time.

Storage & Retrieval Algorithm: storage algorithm is called encoding & retrieval algorithm is called decoding process.

Given: P bipolar binary vector $\{s^{(1)}, s^{(2)} \ldots s^{(P)}\}$

where s^m is $n \times 1$ for $m = 1, 2 \ldots P$

Initializing vector V^0 is $n \times 1$

Storage Algorithm:-

Step 1: weight matrix W is $(n \times n)$:

$W \leftarrow 0, \; m \leftarrow 1$

Step 2: vector s^m is stored, therefore the storage algo for calculating the weight matrix is

$$W \leftarrow s^m s^{m(t)} - I$$

or, $$w_{ij} = (1 - s_{ij}) \sum_{m=1}^{P} s_i^{(m)} s_j^{(m)}$$

($s_{ij} \rightarrow$ kronecker function)

DATE: __/__/____
PAGE: _____

Step 3: if $m < p$ then $m \leftarrow m + 1$ & go to step 2 otherwise go to step 4.

Step 4: Storage is complete, output vector W.

Retrieval Algorithm:-

Step 1: Cycle counter K is initialized $K \leftarrow 1$. Within the cycle counter i is initialized $i \leftarrow 1$ & o/p is initialize as $V \leftarrow V^0$.

Step 2: Integers $1, 2 \dots n$ are arranged in an ordered random sequence $\alpha_1 \alpha_2 \dots \alpha_n$ if randomized sequence of updation is not used then $\alpha_1 = 1$, $\alpha_2 = 2$, $\alpha_n = n$ or $\alpha_i = i$ at each update cycle.

Step 3: Neuron i is updated by computing $V^{new}_{\alpha_i}$

$$net_{\alpha_i} = \sum_{j=1}^{n} W_{\alpha_i j} V_j$$

Step 4: If $i < n$ then $i \leftarrow i + 1$ & go to step 3, otherwise go to step 5

Step 5: If $V^{new}_{\alpha_j} = V_{\alpha_i}$ for $i = 1, 2 \dots n$

Hebb rule.

Algo:-

step 1:- Initialize the weights for each input vectors $w_i = 0$ for $i = 0$ to n (number of neurons) and $b = 0$.

step 2: for each input vector $s:t$ (target output pair), repeat steps from 3 to 5.

step 3: set activation for input unit in input vector $x_i = s_i$ for $i = 0$ to n

set 4: set the corresponding output value to the output neuron. $y = t$

set 5: Update the weights & bia using Hebb rule

$$\Delta w = x_i y$$

x_i = input vector

y = output value.

$$\Delta b = y$$

We will repeat the process for each training sample.

Good Write

And function

Input			output	Δw_1	Δw_2	Δb	w_1	w_2	B
x_1	x_2	B	y	$x_1 y_j$	$x_2 y_j$		0		
-1	-1	1	-1	1	1	-1	1	0	-1
-1	1	1	-1	1	-1	-1	2	-1	-2
1	-1	1	-1	-1	1	-1	1	0	-3
1	1	1	1	1	1	1	2	1	-2

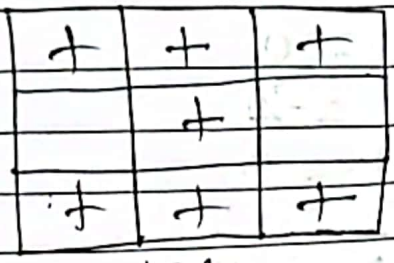
Initial weight = 0.

x_1	x_2	B	y	$x_1 y_j$	$x_2 y_j$	Δb	w_1	w_2	B
1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	-1	1	-1	0	2	0
-1	1	1	-1	1	-1	-1	1	1	-1
-1	-1	1	-1	1	1	-1	2	2	-2

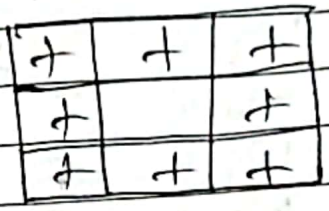
DATE: __/__/____
PAGE: _____

Hopfield

Using Hebb Rule find weight required to perform the following classification of given input pattern '+' symbol represent the value 1 and empty squares indicates -1. Consider 'I' belong to the member of class 1 so have target value 1) & 'O' does not belong to the member of class (so has target value -1)



'I'



'O'

Sol?

	Inputs								Target	Target
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	
I	1	1	1	-1	1	-1	1	1	1	1
O	1	1	1	1	-1	1	1	1	1	-1

Set the Initial weight to 0.

i.e $w_1 = w_2 = w_3 = w_4 = w_5 = w_6 = w_7 = w_8 = w_9 = 0$

and $b = 0$

Now, updating the weights

$w_1(new) = w_1(old) + x_1 y = 1$

$w_2(new) = 0 + 1 = 1$

$w_3(new) = 0 + 1 = 1$

$w_4(new) = 0 + -1 = -1$

$w_5(new) = 0 + (+1) = 1$

$w_6(new) = 0 + -1 = -1$

$w_7(new) = 0 + 1 = 1$

$w_8(new) = 0 + 1 = 1$

$w_9(new) = 0 + 1 = 1$

$b = y = 1$

Good Write

DATE: __/__/____
PAGE: _____

new weight vector is,

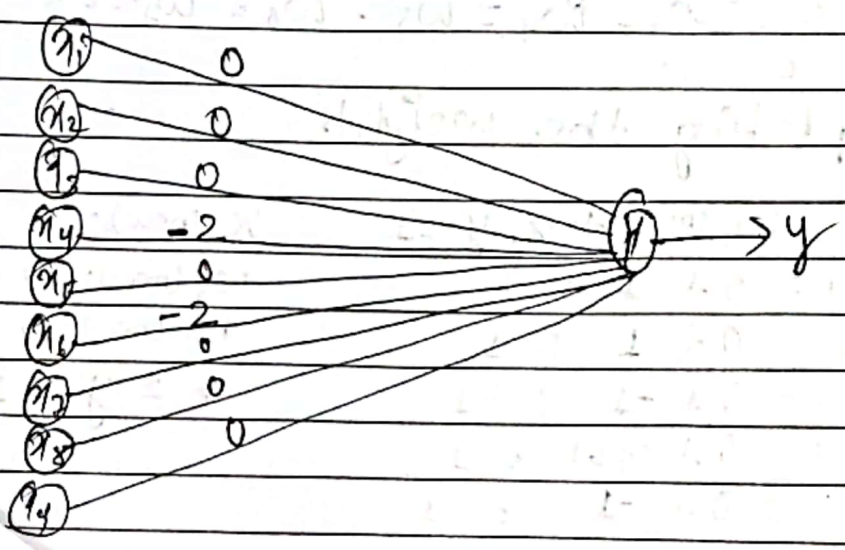
$w = [1, 1, 1, -1, 1, -1, 1, 1, 1]$

$b = 1$

Again

$w_1(new) = 1 + (-1) = 0$
$w_2(new) = 1 + (-1) = 0$
$w_3 \; " = 1 + (-1) = 0$
$w_4 \; " = -1 + (1) = -2$
$w_5 \; " = 1 + (-1) = 0$
$w_6 \; " = -1 + (-1) = -2$
$w_7 \; " = 1 + (-1) = 0$
$w_8 \; " = 1 + (-1) = 0$
$w_9 \; " = 1 + (-1) = 0$

$b = \Delta b + y$
$= 1 - 1$
$= 0$



Good Write

Hopfield Network

The Hopfield Neural Networks, invented by Dr John J. Hopfield consists of one layer 'n' fully connected recurrent neurons.

Discrete Hopfield Network

It is fully interconnected neural network where each unit is connected to every other unit. It behaves in a discrete manner i.e it gives finite distinct output generally of two types.

- Binary (0/1)
- Bipolar (-1/1)

The network has symmetrical weights with no self-connection

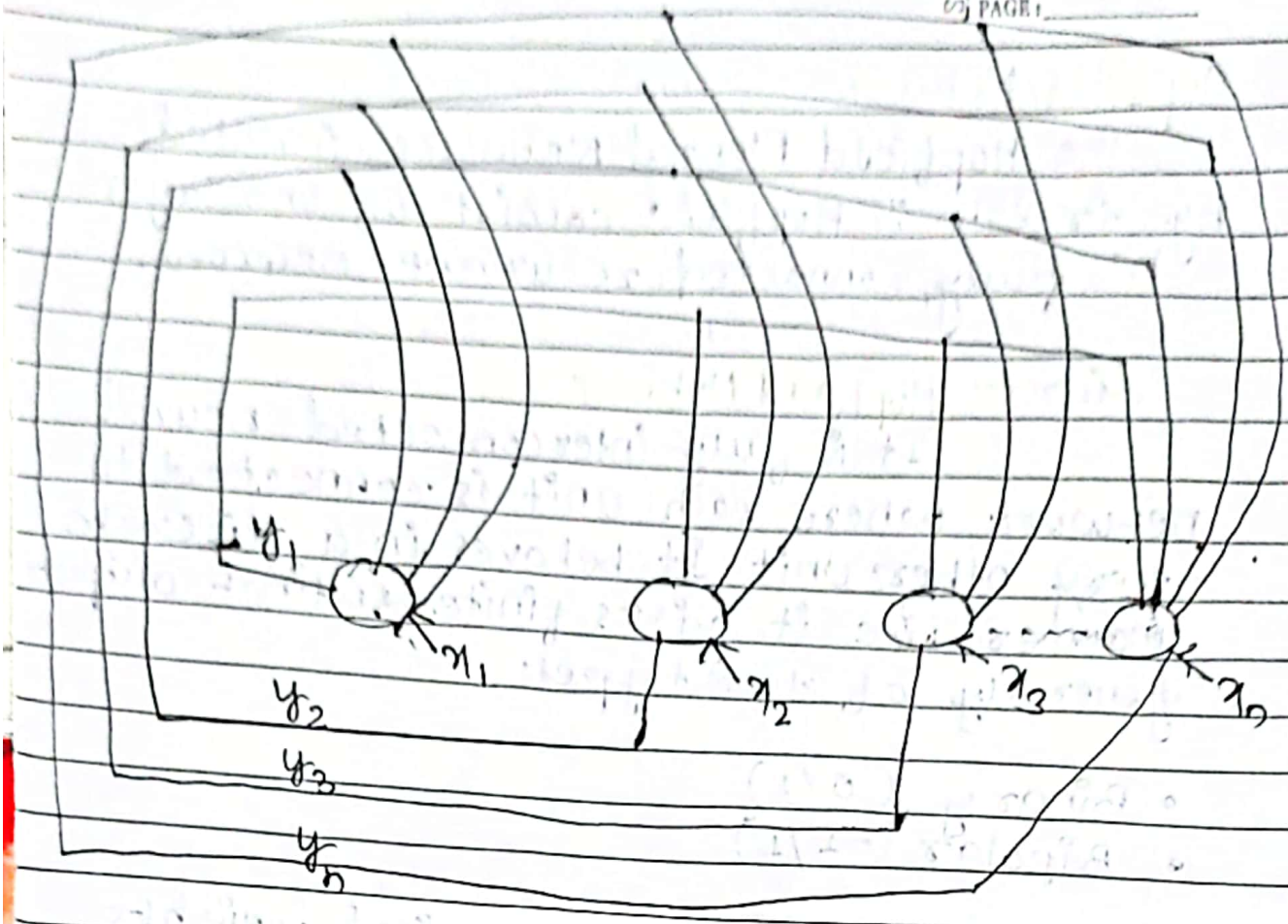
i.e

$w_{ij} = w_{ji}$ & $w_{ii} = 0$

Structure & Architecture

- Each neuron has an inverting and a non-inverting output
- Being fully connected, the output of each neuron is an output to all other neurons but not self.

Good Write



Here $x_1\ x_2 \ldots x_n$ → Input to the n given neurons.

$y_1\ y_2\ y_3 \ldots y_n$ ↔ Output obtained from the n given neurons.

W_{ij} → weight associated with the connection between the i^{th} & j^{th} neuron

Training Algorithm

For storing a set of input pattern $S(p)$ [$p = 1$ to p] where,

$$S(p) = S_1(p) \ldots\ldots S_i(p)$$

... Sn(p), the weight matrix is given by

for binary pattern

$$w_{ij} = \sum_{p=1}^{P} [2s_i(p) - 1][2s_j(p) - 1]$$

w_{ij} for all $i \neq j$

for bipolar

$$w_{ij} = \sum_{p=1}^{P} [s_i(p) s_j(p)]$$

where $w_{ij} = 0$ for all $i = j$

(i.e weights here have no self connections)

Testing Algorithm

Step 1 - Initialize weights (w_{ij}) to store patterns (using training algorithm)

Step 2 - For each input vector y_i, perform steps 3-8.

Step 3 - Make initial activators of the network equal to the external input vector x.

$y_i = x_i$ (for $i = 1$ to n

Good Write

DATE: __/__/____
PAGE: _____

Step 4:- for each vector y_i, perform step 5-8.

Step 5: Calculate the total input of the network y_{in} using the equation

$$y_{in_i} = x_i + \sum_j [y_j w_{ji}]$$

step 6: Apply activation over the total input to calculate the output as per the equation below.

$$y_i = \begin{cases} 1 & \text{if } y_{in} > \theta_i \\ y_i & \text{if } y_{in} = \theta_i \\ 0 & \text{if } y_{in} < \theta_i \end{cases}$$

Where θ is normally taken as 0.

Setp 7:- Now feedback the obtained output y_i to all other units. Thus the activation vectors are updated.

Step 8:- Test the network for convergence.

Good Write

Training

During training of the Discrete hopfield network, the weight will be updated. Here we can take binary input as well bipolar inputs. The weight updation can be done by the following ways.

Here $S(p)$

where S = set of input patterns

$p = 1$ to P

For binary inputs.

$$w_{ij} = \sum_{1}^{P} [2s_i(p) - 1][2s_j(p) - 1] \quad \text{for } i \neq j$$

$$w_{ij} = 0$$

For bipolars.

$$w_{ij} = \sum_{1}^{P} [s_i(p)][s_j(p)] \quad \text{for } i \neq j$$

Testing of the algorithm.

Step 1: Initialize all the weight that can be obtained from Hebbian principle

Step 2: For each input vectors y_i, repeat or perform steps from 3-7.

Good Write

DATE: __/__/____
PAGE: _____

step 3: Make the initial activators equal to the external input vectors.
i.e $y_i = x_i$ for all $i = 1$ to n.

step 4: For all weight that need to be updated. perform 5 to 7.

step 5: Calculate the net input of all the set of input vectors.

$y_{in_i} = x_i + \sum_j (w_{ji} x y_j)$

step 6: Activate all the y_{in} using the below equation

$$y = \begin{cases} 1 & y_{in} > \theta \\ y_{in} & y_{in} = \theta \\ 0 & y_{in} < \theta \end{cases}$$

where $\theta =$

step 7:- Feed back to obtained output.

step 8:- Test it for convergence.

Good Write

Numerical:

Consider the following problem. We are required to create discrete Hopfield N/w with bipolar representation of input vector as [1 1 1 -1] or [1 1 1 0]. is stored in the network. Test the hopfield n/w with missing entries in the first & second component of the stored vector (i.e. [0 0 1 0]

$$w_{ij} = \sum s^T(p)\, t(p)]$$

$$= \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

and the weights with no self connecti

$$w_{ij} = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

DATE: __/__/____
PAGE: _____

Step 3. As per the question the the input vector x with missing entries $x = [0\ 0\ 1\ 0]$ ($[x_1, x_2, x_3, x_4]$)

- Make $y_i = x = [0, 0, 1, 0]$

$= (y_1, y_2, y_3, y_4)$

Now

Choosing unit y_i for updating its activation.

Take ith column of the weight matrix for calculation.

$$y_{in_i} = x_i + \sum_{j=1}^{4} [y_j w_{j_i}]$$

$$= 0 + [0\ 0\ 1\ 0] \begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

$$= 0 + 0 + 0 + 1 + 0$$

$$= 1$$

Applying activation since $y_{in_i} >= 0$

$\therefore y_1 = 1$

Good Write

giving feed back to other units,

$y = [1\ 0\ 1\ 0]$

which is not equal to the input vector.

$x = [1\ 1\ 1\ 0]$

Hence no convergence.

Now, for next we have.

$y = [1\ 0\ 1\ 0]$

$y_{in_3} = x_3 + \sum_{j=1}^{4} [y_j w_{j3}]$

$= 1 + [1\ 0\ 1\ 0] \begin{bmatrix} 1 \\ 1 \\ 0 \\ -1 \end{bmatrix}$

$= 1 + (1 + 0 + 0 + 0)$

$= 2$

Since $y_{in} > 0 \quad \therefore y_3 = 1$

Now,

giving feedback to other units we get $y = [1\ 0\ 1\ 0]$

DATE: __/__/____
PAGE: _____

which is not equal to input vector
$x = [1 \ 1 \ 1 \ 0]$
Hence no.

$$y_{in_4} = x_4 + \sum_{j=1}^{4} [y_j \cdot w_{j4}]$$

$$= 0 + [1 \ 0 \ 1 \ 0] \begin{bmatrix} -1 \\ -1 \\ -1 \\ 0 \end{bmatrix}$$

$$= -2$$

which is less than 0.

$\therefore y_4 = 0$

giving feed back to other.

$\therefore y = [1 \ 0 \ 1 \ 0]$
not equal to the $x = [1 \ 1 \ 1 \ 0]$

$$y_{in_2} = x_2 + \sum_{j=1}^{4} [y_j \cdot w_{j2}]$$

$$= 1 + [1 \ 0 \ 1 \ 0] \begin{bmatrix} 1 \\ 0 \\ 1 \\ -1 \end{bmatrix}$$

$$= 1 + [1 + 0 + 1 + 0]$$

$$= 3$$

$\therefore y_2 = 1$

Good Write

giving feedback to the other units. we get.

$y_i = [1\ 1\ 1\ 0]$

which is equal to input vector.

Hence convergence with vector x.

Continuous Hopfield Network

Here the time parameter is treated as a continuous variable. So, instead of getting binary/bipolar outputs, we can obtain values that lie between 0 & 1.

It can be used to solve constrained optimization and associative memory problems.

$V_i = g(u_i)$

where

V_i = output from the continuous hopfield network.

u_i = internal activity of a node in continuous hopfield network

DATE: __/__/____
PAGE: _____

Energy function evaluation.

Energy function E_f

$$= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j w_{ij} - \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} \theta_i y_i$$

It determines the stability of discrete Hopfield network

Condition

In a stable network, whenever the state of node changes, the above energy function will decrease.

$$\Delta E_y = E_f(y_i^{(k+1)}) - E_f(y_i^{(k)})$$

$$= -\left(\sum_{j=1}^{n} w_{ij} y_i^{(k)} + x_i - \theta_i\right)\left(y_i^{(k+1)} - y_i^{(k)}\right)$$

$$= -(net_i)\,\Delta y_i$$

$$\Delta y_i = y_i^{(k+1)} - y_i^{(k)}$$

The change depends on the fact that only one unit can update its activation at a time.

Good Write

Continuous.

$$E_f = \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ i \neq j}}^{n} y_i y_j w_{ij} - \sum_{i=1}^{n} x_i y_i + \frac{1}{\lambda}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} g_{ri} \int_0^{y_i} g^{-1}(y)\,dy$$

Here λ is gain parameter and g_{ri} input conductance.

Discrete.

$$E_f = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j w_{ij} - \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} \theta_i y_i$$

In a stable condition, the energy also get decreases.

$$\Delta E_f = E_f(y_i^{k+1}) - E_f(y_i^k)$$

$$= -\left(\sum_{j=1}^{n} w_{ij} y_i^k + x_i - \theta_i\right)(y_i^{k+1} - y_i^k)$$

$$= -(net_i)\Delta y_i$$

Good Write

for continuous

$$E = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j w_{ij} - \sum_{i=1}^{n} x_i y_i + \frac{1}{\lambda}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} g_{ri} \int_0^{y_i} a^{-1}(y)\,dy$$

where λ is a gain parameter and g_{ri} is input conductance.

$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} y_i y_j - \sum_{i=1}^{n} x_i y_i + \frac{1}{\lambda}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} g_{ri} \int_0^{y_i} a^{-1}(y)\,dy$$

$$= -\left(\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j w_{ij} - \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} \theta_i y_i\right)$$

Good Write

DATE: / /
PAGE:

1) Storage (learning)
weight matrix is calculated between m pairs of patterns, and stored in synaptic weights of the networks.

$$W = \sum_{m=1}^{M} X_m Y_m^T$$

2) Testing

$$Y_m = \text{sign}(W \cdot X_m)$$

$$X_m = \text{sign}(W Y_m)$$

3) Retrieval: for unknown vector X (a corrupted or incomplete version of a pattern from set A or B) to the BAM and retrieve previously stored.

$$X \neq X_m$$

- Initialize the BAM

$$X(0) = X, \quad p = 0$$

- calculate the BAM output at iteration p:

$$Y_p = \text{sign}(W^T X_p)$$

- Update the input vector

$$X(p+1) = \text{sign}(W Y_p)$$

Good Write