Syllabus of Soft Computing

Intoduction to Artificial Neural Networks → ch 1

↳ what is NN, Human Brain. Models of Neurones, Neural Network graph, Feedback, Network Architectures, knowledge representation.

Learning Process — ch 2

↳ Error correction learning, Memory based learning, Heabbial learning, Competitive learning. Boltzmann learning.

Credit Assignment Problems

Learning with a teacher (supervised learning)

Learning without a teacher (un- " ")

Memory

Single Layer Perceptrons — ch 3.

↳ Unconstrained Optimization Techniques

↳ Method of steepest decen

Netoton's method

Gauss Newton's method.

Linear Least square Filter., Least mean square Algorithm., Learning curves, Perceptrons, Perceptron Convergance Theorem

Back Propagation Algorithm – ch-4

Back Propagation Algorithm, XOR problem, Feature Detection, Virtues & Limitations of Back prop. problem. Modification to back prpgn Algorithm.

Radial Basis Function Neural Networks – ch-5

↳ Cover's Theorem, on the seperability of the pattern, Interpolation Problem, Regulization Network. XOR Problem, Comparision of RBF & Multi Layer Perceptron (MLP) Networks.

↳ Introduction to Fuzzy System, Membership function, Fuzzy relation operation, fuzzy If Then rules, Sugeno & Mamdani type systems Adaptive Neuro Fuzzy systems & Training methods. Application of ANN (Artificial Neural Nw) & fuzzy sys. to non-stationary time series prediction, Pattern classification.

References:

1. Neural Network — S. Haykin
   A comprehensive foundation.

2. Neural Networks — Satish kumar.
   A classroom approach

3. Jang, Sun & Mizetani
   Neuro-Fuzzy & Soft computing.
   A computational approach to Learning & machine intelligence

4. T. Hagen, Timuth, Bale.
   Neural Network Design
   Cenage Learning
   — For Numericals problems.

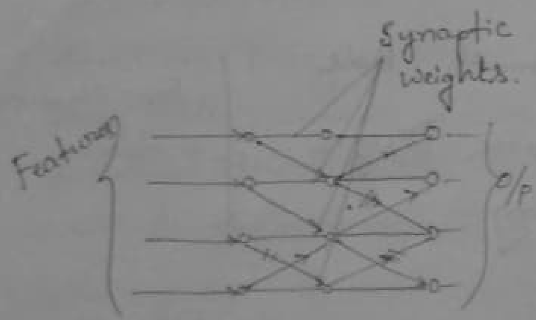
* 1, 3, 4 are important for Fuzzy 3.

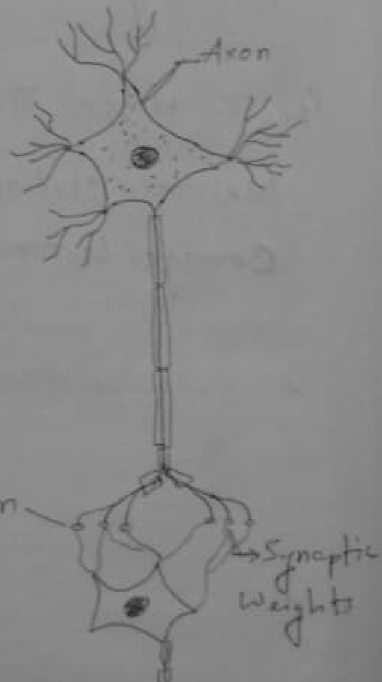
Monday
06.01.14

Neural Network: Is a massively parallel distributed processor made up of simple processing units which is has a natural propensity for storing experimental knowledge & making it available for use. It resembles the brain into two aspects:

[i] knowledge is acquired by the network from its environment thrgh its learning process

[ii] Inter neuron connection strengths known as synaptic weights, are used to store the acquired knowledge.



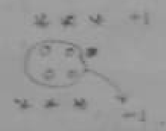


Storing → Learning process
Detection → Training process Dendron

Advantages of the Neural Networks:

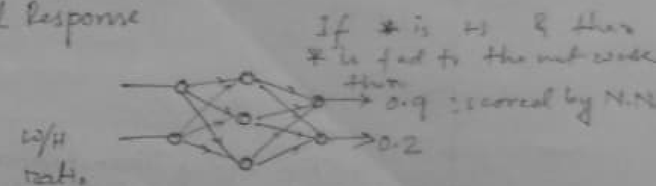
[i] Advantageous compared to the simple prediction system.; Non-linearity.

[ii] Input-output mapping.



[iii] Adaptivity.

[iv] Evidential Response

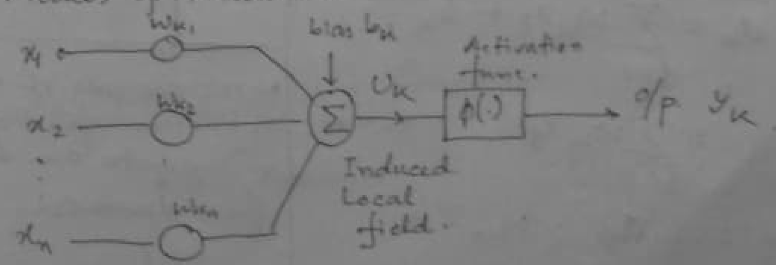


[v] Contextual Information → user dependant.
[vi] Fault tolerence.
[vii] VLSI Implementability.
[viii] Uniformity of analysis & design

Tuesday
07.01.14

#
[ix] Neurobiological Analogy.

* Models of Neurone:

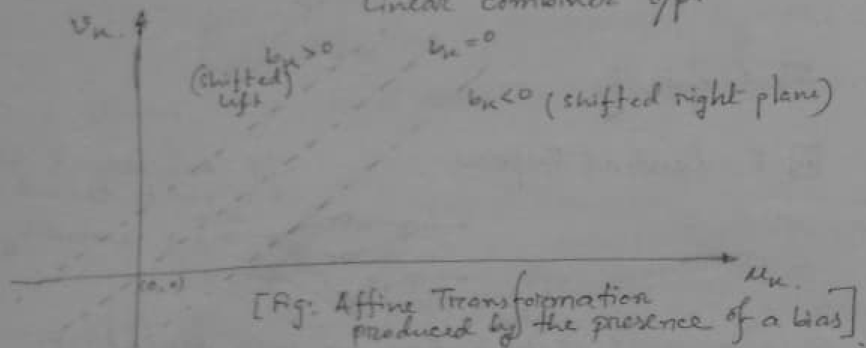


$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

$$y_k = \phi(\underbrace{u_k + b_k}_{v_k})$$

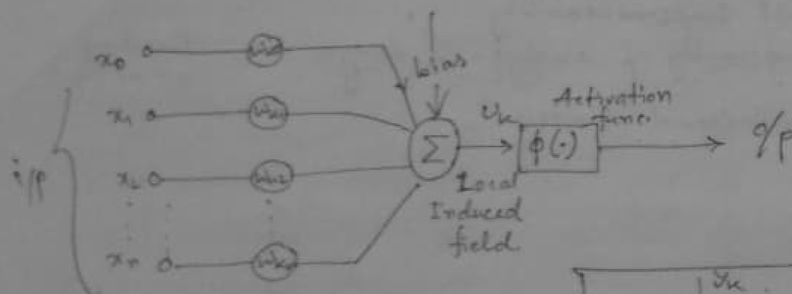
$$= \phi(v_k)$$

$v_k = u_k + b_k$: Induced local field or, Linear combiner o/p.



[Fig: Affine Transformation produced by the presence of a bias]

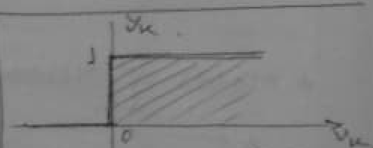
* using b_k as a weight factor:



$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k.$$

$$= \sum_{j=0}^{m} w_{kj} x_j \quad ; x_0 = +1$$

$$y_k = \phi(v_k) \quad ; w_{k0} = b_k$$

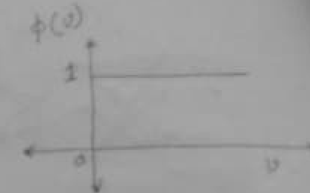


Note: The role of the activation function is to segregate one class from another.

\+ Types of activation functions:

Hard Threshold function

$$\phi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$



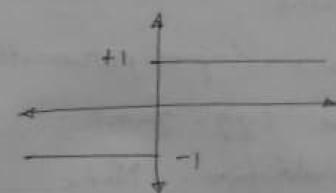
$$y = \begin{cases} 1 & \text{if } v_k \geq 0 \\ 0 & \text{if } v_k < 0 \end{cases}$$

[HardLimiting A.F]

↓

McCulloch - Pits model.

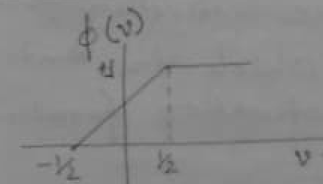
* Symmetric Threshold:



→ all or none.

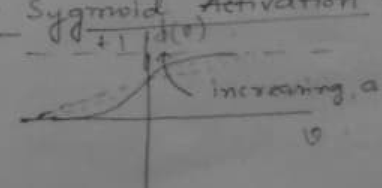
$$\phi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ -1 & \text{if } v < 0 \end{cases}$$

* Piecewise Linear activation function:



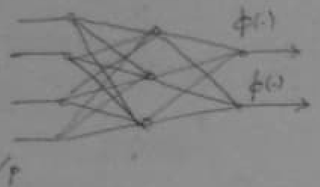
$$\phi(v) = \begin{cases} 1 & v \geq \frac{1}{2} \\ v + \frac{1}{2} & \frac{1}{2} > v > -\frac{1}{2} \\ 0 & v \leq -\frac{1}{2} \end{cases}$$

* Sygmoid Activation function:



$$\phi(v) = \frac{1}{1 + \exp(-av)}$$

10-01-14

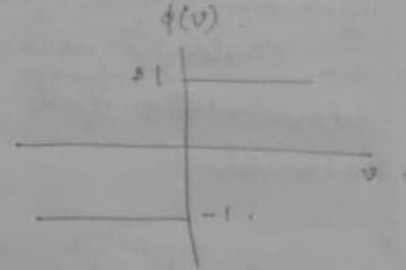


It's defined as a strictly increasing func. that exhibits a graceful balance between linear & non-linear behaviour.

The sygmoid activation func. is defined as $\phi(v) = \frac{1}{1+\exp(av)}$ where a is the slope parameter. As the slope parameter approaches infinite the syg. func. simply becomes an threshold func.. Note that the sygmoid function is differentiable where the threshold func. is not. This func. ranges btwn 0 to +1. It sometimes desirable to have activation func ranges from -1 to +1, which is Anti-symmetric form of activation function.

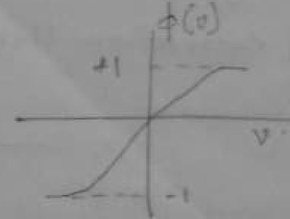
Signum Function:

$$\phi(v) = \begin{cases} 1 & \text{if } v>0 \\ 0 & \text{if } v=0 \\ -1 & \text{if } v<0 \end{cases}$$



Tan-hyperbolic Activation function:

$$\phi = \tanh(av) = \frac{e^{av} - e^{-av}}{e^{av} + e^{-av}}$$



Stochastic Model of Neuron

$P(v) =$

$$v_{ij} = \sum_{i=0}^{N} w_{ij} x_j$$

$$y_{ij} = \phi(v_{ij})$$

$$P(v) = \frac{1}{1+\exp(-v/T)}$$

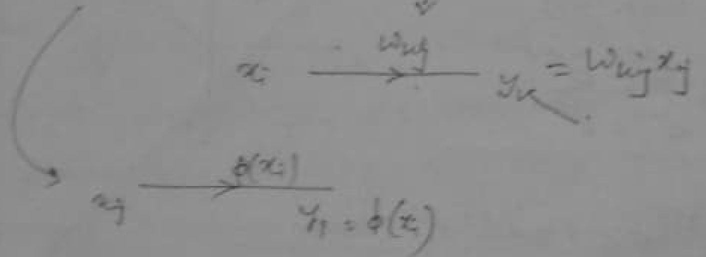
where T is a pseudo temperature which is used to control the noise level & therefore uncertainty of firing.

We should think T as a parameter that controls thermal fluctuations representing the effect of synaptic noise. Note that when $T \to 0$ the stochastic neuron reduces to a noiseless deterministic ~~fett~~ form, namely the McCulloch Pitt model

Neural Network viewed as directed Graph:

Rule 1: [i] Synaptic links.

[ii] Activation links.



Rule 2: A node signals equals the ~~ex~~ algebraic sum of all signals. ~~Entire node via~~ entering the ~~of~~ pertinent node via incoming links.

Rule 3: The signal at a node is transmitted to each outgoing link originating from that node with the transmission entirely independent of the transfer function of the outgoing links.

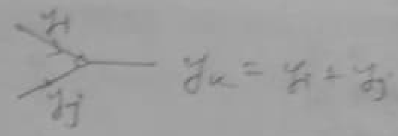


Fig 2

Synaptic divergence or fan out

Fig: 03.

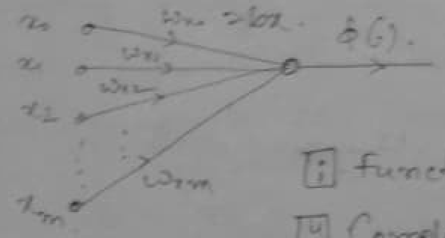


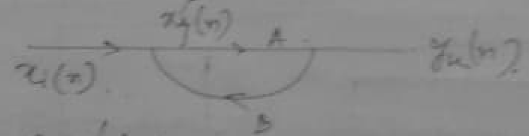
Fig: Total Neural Net Architecture

[i] Functional Description

[ii] Complete "

[iii] Architecture

Feedback:



$$y_k(n) = A\, x_j'(n)$$

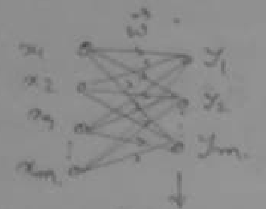
$$x_j'(n) = x_j(n) + B\, y_k(n)$$

Neural Network Architecture:

[i] Single-layer feed forward NN

[ii] Multi- " " " "

[iii] Recurrent NN.

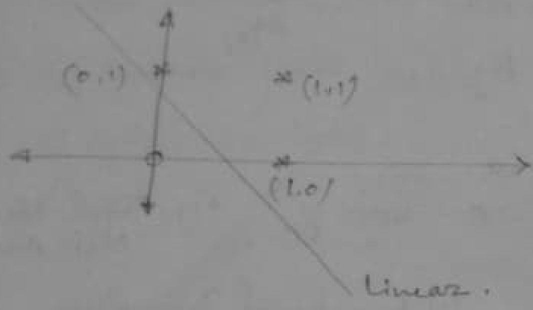


OR Gate

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

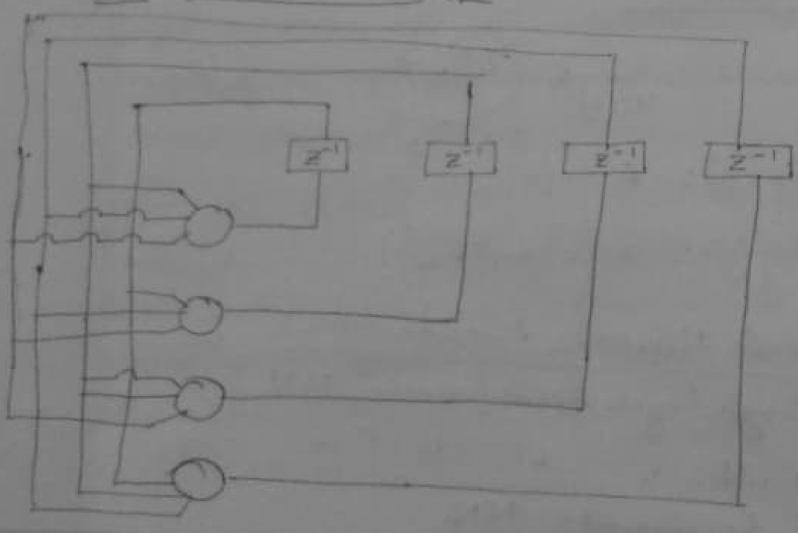
[this is the only layer.]

Fig: Sigle layer feedforward NN.



13.01.14.
Monday.

Recurrent Network:



⇒ A recurrent NN distinguishes itself from a feedforward Neural Network.

14.01.14
Tuesday.

Knowledge Representation:

Rule 01: Similar i/ps from similar classes should usually produce similar representation inside the network, & should therefore ~~the~~ be classified as belonging to the same category.

$$x_i = [x_{i1}, x_{i2}, \dots \dots x_{im}]^T$$

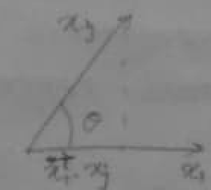
[Eucledian Based Distane measure-ment]

$$d(x_i, x_j) = \|x_i - x_j\|$$

$$= \left[\sum_{k=0}^{m} (x_{ik} - x_{jk})^2\right]^{1/2}.$$

$$\cos\theta = \frac{\vec{x_i} \cdot \vec{x_j}}{\|\vec{x_i}\| \, \|x_j\|}$$

[Dot Product or Inner Product Based distance. measurement]



$$\|x_i\| = \|x_j\| = 1$$

$$d^2(x_i, x_j) = (x_i - x_j)^T (x_i - x_j)$$

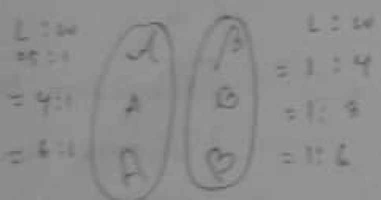
$$= 2 - 2\, x_i^T x_j$$

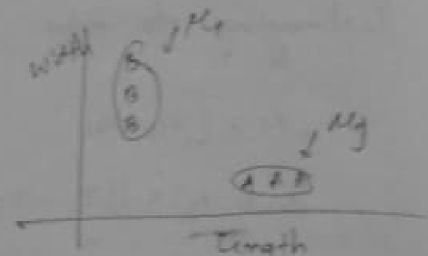
Mahalanobis distance

$$d_{ij} = (x_i - \mu_q) \Sigma^{-1} (x_i - \mu_q)$$

$$\Sigma = E\left[(x_i - \mu_i)(x_i - \mu_i)^T\right].$$

$$\Sigma = E\left[(x_g - \mu_g)(x_g - \mu_g)^T\right].$$





<u>Rule 02:</u> Items to be categorized as separate classes should ~~of~~ be given widely different representations in the network. This rule is exact opposite of Rule 01.

<u>Rule-03:</u> If a particular feature is important then there should be larger num of neurons involved in the representation of that item in the network.

Rule 04. Prior Information & Invariances should be ~~build~~ built into the design of a NN, thereby simplifying the network design by not having to learn them.

* How to build Invariences into NN design?

[i] Invariant by structure.

[ii] " " Training.

[iii] " feature space.

[iv] ~~Invariances by training.~~

Wed.
15.01.14

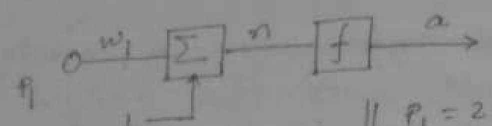
* Tutorial 01

[Q] Input to a single input neuron is 2.0. Its weight is 2.3 & bias/threshold is 3. Then what is the net i/p to the transfer function. Also find out the o/p of the neuron if it has the following transfer function:

[i] hardlimiting [ii] Linear [iii] ~~Constant~~ Sigmoid activation function.

≥ 0 1 < 0 -1

[ii] Linear? [iii] block → $\frac{1}{1+e^{-7.6}} = \phi(7.6)$



$n = w_1 p_1 + b = 7.6$ || $p_1 = 2.0$, $w_1 = 2.3$, $b = 3$

$a = f(n) \cdot f(4.6+3) = 1$

[Q.2] two i/p neurone with the following parameters: $p = [-5 \ 6]^T$, $w = [3 \ 2]$, $b = 1.2$ then calc the neurone o/p for the following transfer function

[i] Symmetric [ii] saturating linear func.

[iii] Hyperbolic tangent / sigmoid

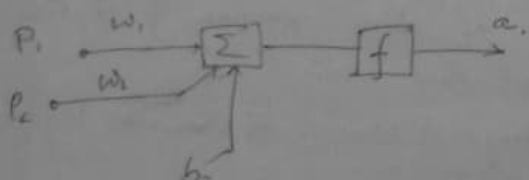
$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$

$n = wp + b$.

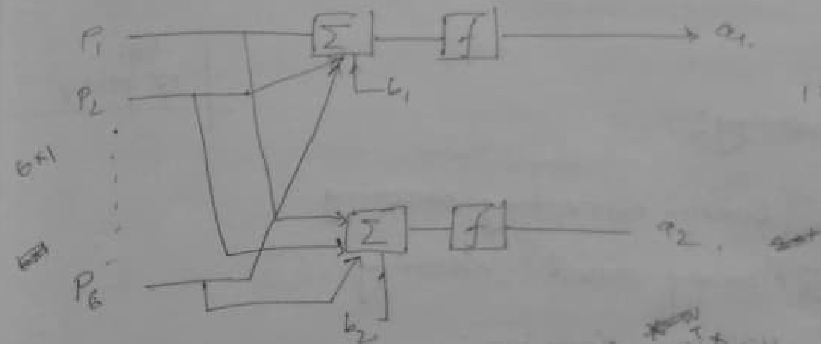
$= -1.8$

$a = f(n)$.

(i) -1 [ii] 0 [iii] $\frac{e^{n} - e^{-n}}{e^{n}+e^{-n}}$.



[Q.3] A single layer neural network has six i/ps & two o/ps. The o/ps are limited to 0 & 1 which is continuous in range. How many neurons are required for the network architecture. What are the dimensions of the weight matrix. What ~~are the~~ kind of transfer functions should be used?



$[w_{11}, \ldots w_{16}]$

$[w_{21}, \ldots w_{26}]$

$[\ ]_{1\times2} = W^T_{2\times6}\,{}_{6\times1}$

weight is 1.3 2.0,

[Q.4] The i/p to a single neurone is and it's bias is 3.0. What possible kind of T.F could this neuron have if it's o/p is [i] 5.6 [ii] 1, [iii] 0.9963 [iv] -1.0

[i] Consider a single i/p neurone, ~~with~~ ~~what~~ where would be o/p to be -1, for i/p < 3.
+1 for i/p > 3.

i) What kind of T.F is required
ii) What bias would you suggest. Did your bias anyway related to the i/p weight. If yes, how?

Fri
17.01.14

Learning:

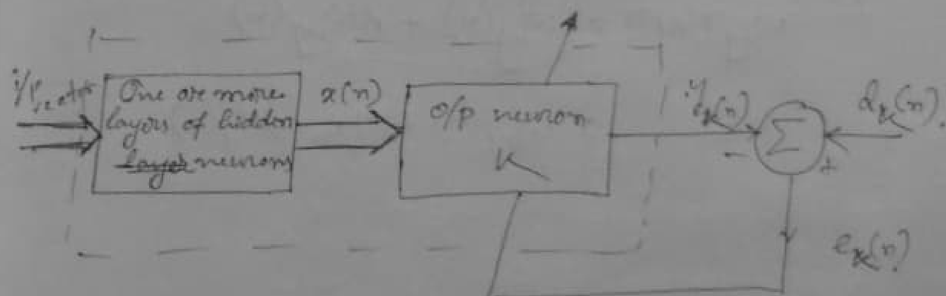
[i] Error-correcting learning.
[ii] Memory Based Learning.
[iii] Hebbian Learning.
[iv] Competitive Learning.
[v] Boltzman learning.

[i] Learning is a process by which the free parameters of a NN are adapted through a process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameters changes take place. The def^n of learning process implies the following sequence of events:

[i] The NN is stimulated by an environment.

[ii] It undergoes the changes in its free parameters as a result of this stimulation.

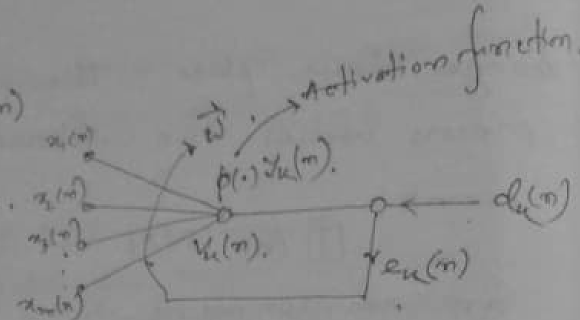
[iii] The NN responds in a new way to the environment because of the changes that have occured to its internal structure.



[MLP- Feed forward NN highlighting the only neuron in the output layer]

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

↳ cost function.



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

↳ cost function.

Maximizing the cost function is the main objective.

Delta rule/ ~~With~~ Widrow-haff rule:

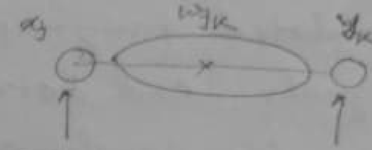
$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$$

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

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

Mon 20.01.14

[ii] ~~#~~ Hebbian Learning:



$$\boxed{\begin{aligned} \Delta w_{kj} &= F(x_j, y_k) \\ &= \eta x_i(n) y_k(n) \end{aligned}}$$

The hebbian learning: Hebbs hypothesis defines/ can be defined by expanding & rephrasing ~~it~~ as a two part rule

[i] If two neurones are either side of a synapse (or connection) are activated simultaneously or synchronously then the strength of synapse is selectively increased

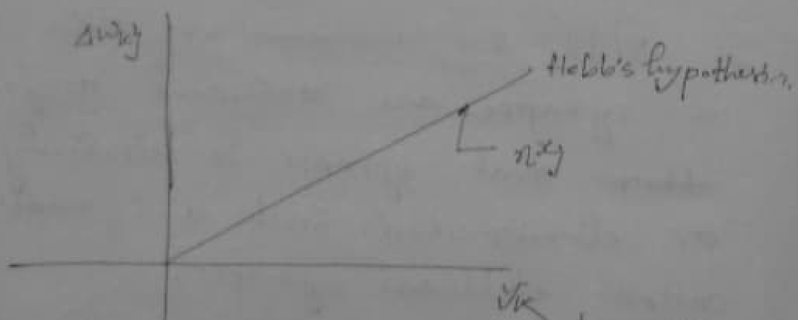
[ii] If 2 neurons on either side of a synapse are activated asynchronously then ~~that~~ that synapse is selectively weakened or eliminated. Such a synapse is called hebbian synapse.

Synaptic modification can be defined as [i] Hebbian [ii] ~~Behebbian~~ Antihebbian. [iii] Non-hebbian.

[i] A hebbian increases strength with positive correlated presynaptic & post synaptic signals. And ~~decreases~~ its strength when ~~big~~ dig signals are either uncorrelated or negatively correlated

[ii] An antihebbian synapse weakens positively correlated pre-synaptic & post-synaptic signal. and strengthen negatively correlated signals.

[iii] On the other hand non-hebbian synapse doesn't involve a hebbian mechanism.



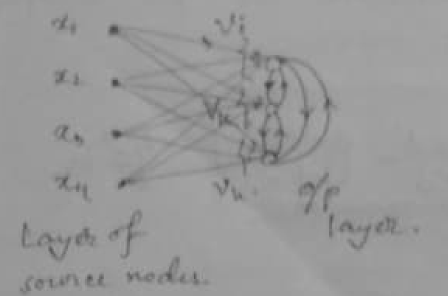
Covariance Hypothesis.

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

$\bar{x}$ = time avg. ~~mean~~ value of the pre-synaptic signals

$\bar{y}$ = post-synaptic.

[iv] Competetive Learning:



$$y_k = \begin{cases} 1 & \text{if this} \\ 0 & \end{cases}$$

$$\sum w_{kj} = 1.$$

→ Here winner takes all

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

[v] Boltzman Learning:

Recurrent structure.

Energy function.

↳ $$E = -\frac{1}{2}\sum_j \sum_k w_{kj} x_k x_j$$

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

In clamped condition in which the visible neurons are clamped onto specific states ~~to~~ whi determined by the environment

condition.
In free running, in which all the neurones are allowed to operate free,

$$\boxed{\Delta w_{kj} = \eta\left(P_{kj}^{+} - P_{kj}^{-}\right)}, \quad j \neq k$$

P_{kj} denotes the corelation b/w neurone k & j with the network in it's clamped condition.

$-P_{kj}$ " " . . in it's free running condition.

21.01.14
Tue

Credit Assignment Problem:

1. Temporal Credit Assignment.
2. Structural Credit Assignment Problem.

[4] Memory: 1. Short term memory.
2. Long term memory.

1. a compilation knowledge representing the current state of the environment. It's called the short-term memory.

2. Any discrepencies between knowledge stored in short term memory & a new state are used to update the short term memory.

Long term memory refers to knowledge stored for a long time or permanently.

* Associative Memory: It has following characteristics: [i] Memory is distributed

[ii] Both the stimulus (key) & the response (stored) patterns of an associative memory consist of data vectors.

[iii] Information stored in memory by setting up a spatial pattern of neural activities across a large number of neurones.

[iv] Information contained in stimulus not only determines it's storage location in memory but also an adress for it's retrieval.

[v] Although neurons do not represent reliable and low noise computing cells, the memory exhibits a high degree of resistance to noise and damage of diffusive kind.

[vi] There may be interaction between individual patterns stored in memory. ~~They are~~ There is therefore the distinct possibility to some errors during recall process.

Friday.
24.01.14.

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

$i = 1, 2, \dots \dots m$

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

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

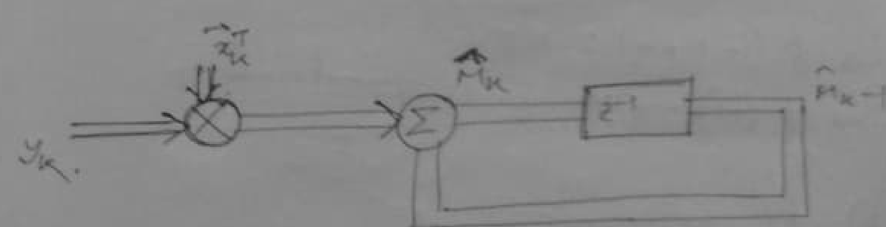
$$\boxed{M_k = M_{k-1} + \Delta W(k)}$$

Correlation Matrix Memory:

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

$$\vec{\hat{M}} = [y_1, y_2 \dots \dots, y_q] \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_q^T \end{bmatrix}$$

$$\boxed{\vec{\hat{M}}_k = \vec{\hat{M}}_{k-1} + \vec{y}_k \vec{x}_k^T}$$



[Fig: Signal Flow Graph.]

Tutorial 2

[Q] Given a 2 i/p neuron, following with matrix & i/p vector given by $w = [3 \quad 2]$ & $p = [-5 \quad 7]^T$, we would like to have output = 0.5

a) Is there any transformation for the o/p if bias is zero.

b) Is there bias that will give the o/p as 0.5 if linear transformation function is used?

c) Is there any bias to give 0.5 if block sigmoid function is used?

d) If symmetrical hard limit TF is used.

Ans a)

$\rightarrow n = wp^T + b = (-5\times3) + (7\times2) + 0 = -1.$

$\hookrightarrow \phi(-1+b) = 0.5$

$\rightarrow \phi(-1) = 0.5 \quad = \dfrac{-\text{symmetrical}}{2}$!

b) $\phi(-1+b) = 0.5$

for linear TF.

$\Rightarrow -1+b = 0.5$

$\therefore b = 1.5$

c) $0.5 = \dfrac{1}{1+e^{(-1+b)}}$

$\Rightarrow 1 + e^{(-1+b)} = 2$

$\Rightarrow e^{(-1+b)} = 1.$

$\therefore b = 1.$

(d) $\phi(-1+b) = 0.5$

$\Rightarrow -1+b \geq 0.5$

$\Rightarrow b \geq 1.5$

[Q] Two layer i/p has 4 i/p & 6 o/p. Range of o/p between 0 & 1. What is the network architecture?

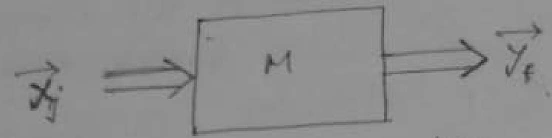
[i] How many neurons are required in each layer

[ii] Dimension of 1st & 2nd weight matrix.

[iii] What kind of TF can be used in each layer?

[iv] Design 2 layer networks.

Recall:



$\boxed{\vec{y} = \vec{M}\vec{x}_j}$ | $\vec{x}_j$ = key pattern

$\vec{y}$ = output response.

$= \sum_{k=1}^{m} (\vec{y}_k \vec{x}_k^T)$ | ; k = no of vectors.

$y_k = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$.

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

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

$\cancel{= y_j + v_j}$

$\boxed{\therefore E_x = \sum_{l=1}^{m} x_{kl}^2 = \vec{x}_k^T \vec{x}_k = 1.}$

$v_j = \sum_{\substack{k=1 \\ k \neq j}}^{m} (\vec{x}_k^T \vec{x}_j) y_k$ (noise vector) — (i)

$\boxed{\Rightarrow \vec{y} = \vec{y}_j + \vec{v}_j}$

$\cos(\vec{x}_k, \vec{x}_j) = \dfrac{\vec{x}_k^T \vec{x}_j}{\|x_k^T\| \, \|x_j\|} = \vec{x}_k^T \vec{x}_j$

now putting this into (i).

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

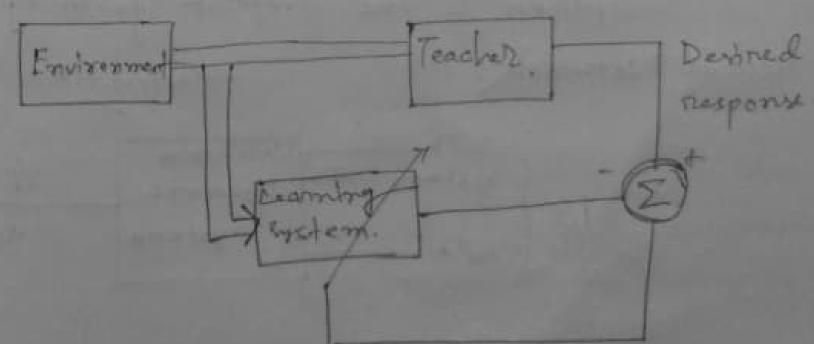
we can get a perfect recall $\Rightarrow \vec{y} = y_j$

For a perfect recall $\vec{y} = \vec{y}_j$

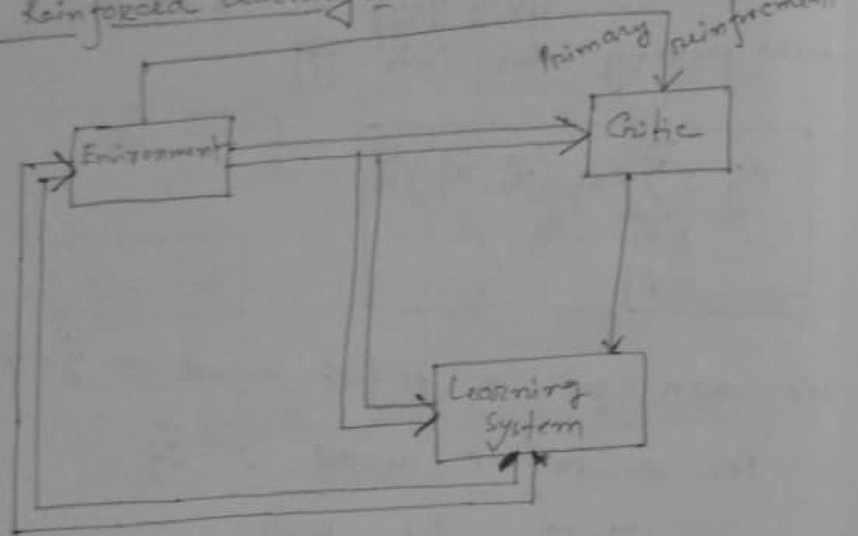
$\vec{x}_k \vec{x}_j = \begin{cases} 1 & k = j \\ 0 & k \neq j \end{cases}$

Q. What is the limit of storage capacity of associative memory. If r is the length of ~~then~~ of dim a rectangular memory matrix $l \times m$ then $r \leq \min(l, m)$.

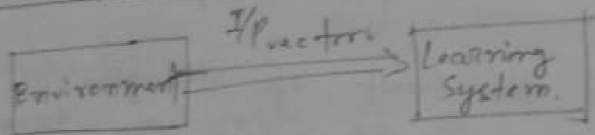
Learning with a teacher (Supervised Learning):



Reinforced Learning:-



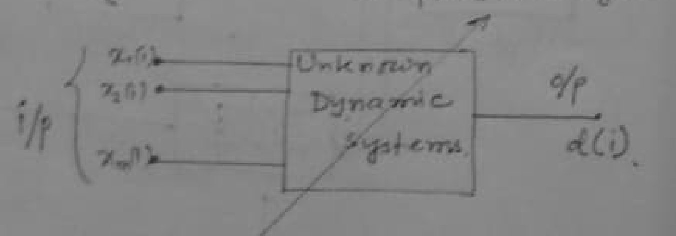
: unsupervised :

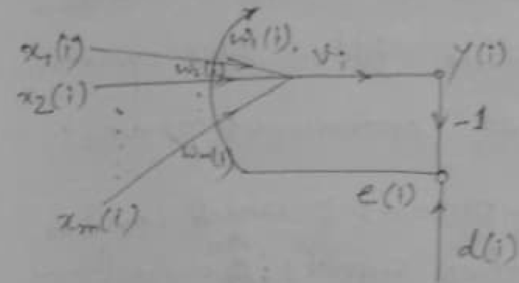


Monday.
27. 01. 14.

Single Layer Perceptron: (SLP):

Perceptron is the simplest form of Neural Network.





[The neural model graph flow figure]

The neural model operates under the influence of an alogotrithm that controls necessary adjustment to the synaptic weights ~~be response to stastical variations in the system behaviours are made on a con~~ of the network with the following points:

[i] The algorithm starts from an arbitary setting of neuron synaptic weights

[ii] Adjustment to the synaptic weights, in response to stastical variation in the system behaviour are made on a continuous basis

[iii] Computations of adjustments to the synaptic weights are completed inside a time interval that is one sampling period long.

The neural model described in here is called adaptive filter. It's operation consists of two continuous process:

(a) Filtering process: which involves computation of two signals: An output denoted by y_i That is produced in response to the m elements to the stimulus vector $x(i)$, an error signal denoted $e(i)$.

(b) Adaptive process which involves automatic adjustments of the synoptic weights of the neurone in accordance with the error signal.

$$y(i) = v(i) = \sum_{k=1}^{\alpha} w_k(i) x_k(i)$$

$$y(i) = \vec{x}^T(i)\, \vec{w}(i)$$

$$e(i) = d(i) - y(i)$$

<u>Unconstrained Optimization Techniques:</u>

$\underbrace{\xi(\vec{w})}_{\text{cost function}} = f(\vec{z})$; to find $\vec{w}^*$

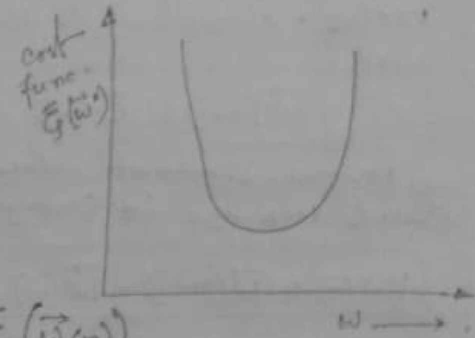
$\xi(\vec{w}^*) < \xi(\vec{w})$

Minimize the cost function $\xi(\vec{w})$ w.r.t $\vec{w}$. The necessary optimization condition is

$\nabla \xi(\vec{w}^*) = 0$

↓

Gradient



$$\xi(\vec{w}(n+1)) < \xi(\vec{w}(n))$$

$\vec{w}(n)$: Old value of weight vector

$\vec{w}(n+1)$: Updated value of weight vector

We hope that every iteration the cost function will minimize

(4) <u>Methods of Steepest Descent</u>: Here the weight vector direction will be opposite to the direction of the ~~vec~~ cost function

$$\boxed{\vec{g} = \nabla \xi(\vec{w})}$$

In this method the successive adjustments to the weight vector $\vec{w}$ are in the direction of steepest descent that is, in a direction opposite to the gradient vector. According to the steepest descent method:

$$\boxed{\vec{w}(n+1) = \vec{w}(n) - \eta \vec{g}(n)}$$

positive constant values called step size or learning rate parameter.

$$\boxed{\Delta w(n) = \vec{w}(n+1) - \vec{w}(n) = -\eta \vec{g}(n)}$$

↳ change of the weight vector.

First-order Taylor series approximation

$$f(x_0 + x) \cong f(x_0) + f'(x_0)x.$$

$$\boxed{\xi(\vec{w}(n+1)) \cong \xi(\vec{w}(n)) + g^T(n)\Delta w(n)}$$

$$\xi(w(n) + \Delta w(n))$$

$$\xi(\vec{w}(n+1)) = \xi(\vec{w}(n)) - \eta g^T(n) g(n)$$

$$\Rightarrow \boxed{\xi(\vec{w}(n+1)) = \xi(\vec{w}(n)) - \eta \|\vec{g}(n)\|^2}$$

Tuesday
28.01.14

Tutorial: 4

Monday
17.02.14.

Apply the steepest decent algorithm to the following cost function:

$$F(x) = x_1^2 + 25 x_2^2$$

The initial condition $x_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$. $\eta = 0.01$.

Find the weight objection up to second iteration:

Ans: $\bar{w}(n+1) = \bar{w}(n) - \eta \bar{g}(n)$.

$g(n)$ = gradient of cost function.

$$g = \nabla F(x).$$

$$= \begin{bmatrix} \frac{\partial F(x)}{\partial x_1} \\ \frac{\partial F(x)}{\partial x_2} \end{bmatrix}_{x=x_0} = \begin{bmatrix} 2x_1 \\ 50x_2 \end{bmatrix}_{x=x_0} = \begin{bmatrix} 1 \\ 25 \end{bmatrix}.$$

↳ $\bar{g}(n) = \begin{bmatrix} 1 \\ 25 \end{bmatrix}$. ; $\eta = 0.01$.

1st iteration:

$$x_1 = x_0 - \eta g.$$

$$= \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - 0.01 \begin{bmatrix} 1 \\ 25 \end{bmatrix} = \begin{bmatrix} 0.49 \\ 0.25 \end{bmatrix}.$$

2nd iteration:

$$x_2 = x_1 - \eta g, \quad g = \begin{bmatrix} \frac{\partial F(x)}{\partial x_1} \\ \frac{\partial F(x)}{\partial x_2} \end{bmatrix}_{x=x_1} = \begin{bmatrix} 0.98 \\ 12.5 \end{bmatrix}.$$

$$= \begin{bmatrix} 0.49 \\ 0.25 \end{bmatrix} - 0.01 \begin{bmatrix} 0.98 \\ 12.5 \end{bmatrix} = \begin{bmatrix} 0.48 \\ 0.125 \end{bmatrix}.$$

Q Apply the newtons method ~~of~~ on the foll cost function ;
$F(x) = x_1^2 + 25x_2^2$ and calculate the weight matrix.

$\Rightarrow$ $F(x) = x_1^2 + 25x_2^2$

$w(n+1) = w(n) - H^{-1} g(n)$. ; H = Hessian matrix.

$$H = \begin{bmatrix} \frac{\partial^2 F(x)}{\partial x_1^2} & \frac{\partial F(x)}{\partial x_1 \partial x_2} \\ \frac{\partial^2 F(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 F(x)}{\partial x_2^2} \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix} \Rightarrow H^{-1} = \frac{1}{100} \begin{bmatrix} 50 & 0 \\ 0 & 2 \end{bmatrix}$$

$$\rightarrow H^{-1} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.02 \end{bmatrix}$$

$x_1 = x_0 - H^{-1} g$, $x_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$.

$$g = \begin{bmatrix} \frac{\partial F(x)}{\partial x_1} \\ \frac{\partial F(x)}{\partial x_2} \end{bmatrix}_{x = x_0} = \begin{bmatrix} 2x_1 \\ 50x_2 \end{bmatrix}_{x = x_0} = \begin{bmatrix} 1 \\ 25 \end{bmatrix}$$

$$x_1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 0.5 & 0 \\ 0 & 0.02 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 25 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Q $F(x) = 5x_1^2 - 6x_1x_2 + 5x_2^2 + 4x_1 + 4x_2$

$\eta = \frac{2}{\lambda_{max}}$; λ = eigen value.

Find the maximum stable learning rate.

$\frac{1}{det}$ [transpose]

steps: 1. Hessian Matrix.
2. Eigen value.
3. $\eta \le \frac{2}{\lambda_{max}}$ $A - \lambda$

$$H = \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix} \Rightarrow H^{-1} = \frac{1}{64} \begin{bmatrix} 10 & 6 \\ 6 & 10 \end{bmatrix}$$

eigen value:

$A - \lambda$

$$\begin{vmatrix} 10-\lambda & -6 \\ -6 & 10-\lambda \end{vmatrix} = 0 \Rightarrow (10-\lambda)^2 = 36.$$

$\Rightarrow 10 - \lambda = \pm 6$.

$\therefore \lambda_1 = 4$

$\lambda_2 = 16$; λ_{max}.

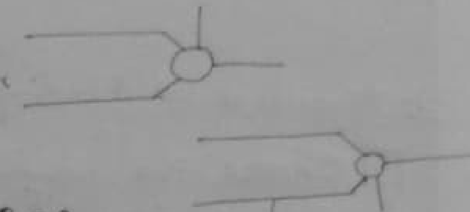
$\Rightarrow \eta \le \frac{2}{16} \Rightarrow \eta \le \frac{1}{8} = 0.125$

Wed
29.01.14

Tutorial: 3.

Use And & OR gate for designing perceptron network. Find the bias & test if according to it's i/p.

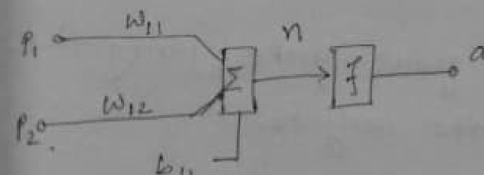
In perceptron rule using separation boundary

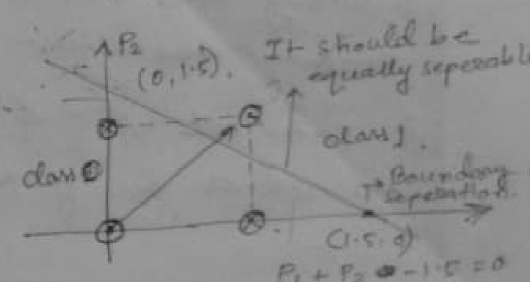


AND

$C_1 \begin{cases} P_1 = 0 \\ P_2 = 0 \end{cases} \quad t_1 = 0$ $C_2 \begin{cases} P_1 = 0 \\ P_2 = 1 \end{cases} \quad t_2 = 0$

$C_3 \begin{cases} P_1 = 1 \\ P_2 = 0 \end{cases} \quad t_3 = 0$ $C_4 \begin{cases} P_1 = 1 \\ P_2 = 1 \end{cases} \quad t_4 = 1$





$n = w_1 P_1 + w_2 P_2 + b = 0$

$w^T p + b = 0$

$w = [2 \ \ 2]$ p = input, $b = ?$

$p^T = [1.5 \ \ 0] \Rightarrow [2 \ \ 2] \begin{bmatrix} 1.5 \\ 0 \end{bmatrix} + b = 0$

1. $t = o/p$

o/p, w, b

Test $\Rightarrow b = -3$

C_1 WTP + b = a

$[2 \ 2] \begin{bmatrix} 0 \\ 0 \end{bmatrix} + (-3)$

$a_1 = f_{har}(-3) = 0$

$a_1 = t_1$

C_2

$[2 \ \ 2] \begin{bmatrix} 0 \\ 1 \end{bmatrix} + (-3)$

$= 2 - 3 = (-1) < 0$

$a_2 = 0$

4) Procedure For Finding Bias:

[i] Choose the weight [synaptic] in the direction of positive targets. Take the i/p values which are on the boundary line [ii]

[iii] $\sum_{i=1}^{n} w_i P_i + b = 0$: condition for the boundary line.

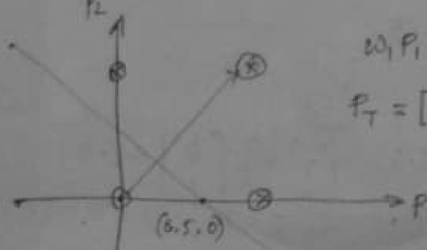
[iv] Putting the value of weight & input in finding the bias.

[v] Test all i/ps of your gate using the bias values & synaptic weights.

OR Gate

$\left\{ P_1 = 0,\ P_2 = 0,\ t_1 = 0 \right\}$ $\left\{ P_1 = 0,\ P_2 = 1,\ t_2 = 1 \right\}$

$\left\{ P_1 = 1,\ P_2 = 0,\ t_3 = 1 \right\}$ $\left\{ P_1 = 1,\ P_2 = 1,\ t_4 = 1 \right\}$.



$w_1 P_1 + w_2 P_2 + b$.

$P_T = [0.5, 0]$

$[1, 1] \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} + b = 0$

$b = -0.5$

Q Consider the classification problem defined below

$\left\{ P_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix},\ t_1 = 1 \right\}$ $\left\{ P_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix},\ t_2 = 1 \right\}$

$\left\{ P_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix},\ t_3 = 0 \right\}$ $\left\{ P_4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ t_4 = 0 \right\}$

a) Design a Single Neuron perceptron to solve this problem.

b) Design a network choosing the synaptic weight that are othogonal to the decision boundary.

b) Test your solution with all above 4 i/p vectors.

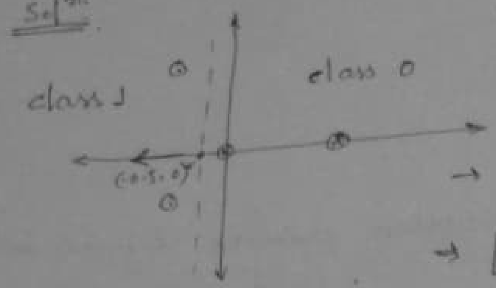
c) classify the following i/p vectors with your solution

$P_5 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$, $P_6 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $P_7 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $P_8 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$.

d) From the above vectors which will be classified in the same way regardless of the solution of the values ~~w~~ w & b.

e) Which vectors may vary depending on the solution & why?

Sol^n.



choose the weight $w = [-1, 0]$

$p = [-0.5, 0]$.

$\rightarrow w^T p + b = 0$

$\rightarrow [-1 \ \ 0]\begin{bmatrix}-0.5 \\ 0\end{bmatrix} + b = 0$

$\Rightarrow b = -0.5$.

Testing the condition:

$\text{hard}\left\{[-1 \ \ 0]\begin{bmatrix}-2 \\ 0\end{bmatrix} - 0.5\right\} = 1 = t_5$.

$t_c = \left\{[-1 \ \ 0]\begin{bmatrix}1 \\ \ \end{bmatrix} - 0.5\right\} = 0$

Friday.
31.01.2014

Gauss Newton Method:

$$\xi(\vec{w}) = \frac{1}{2}\sum_{i=1}^{n} e^2(i)$$

$$e'(i,w) = e(i) + \left[\frac{\delta e(i)}{\delta \vec{w}}\right]^T_{w=w(n)} \left(\vec{w}^{(n+1)} - w(n)\right)$$

$$e'(n,w) = \vec{e}(n) + \vec{J}(n)\left(\vec{w}(n+1) - \vec{w}(n)\right)$$

$$w(n+1) = \arg\min_{\vec{w}} \left\{\frac{1}{2}\|e'(n,w)\|^2\right\}$$

$$\frac{1}{2}\|e'(n,w)\|^2 = \frac{1}{2} e'(n,w) \cdot e^T(n,w).$$

$$= \frac{1}{2}\left[\vec{e}(n) + \vec{J}(n)\left(w(n+1) - w(n)\right)\right]\left[\vec{e}(n) + \vec{J}(n)\left(w(n+1) - w(n)\right)\right]^T$$

$$= \frac{1}{2}\|e(n)\|^2 + e^T(n)\vec{J}(n)\left(w(n+1) - w(n)\right)$$

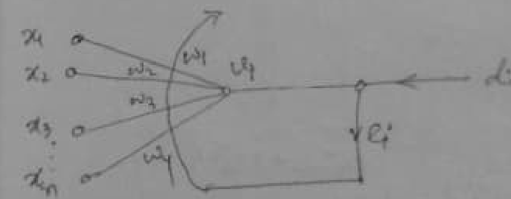
Derivative w.r.t $\Delta w(n)$ & setting it it zero

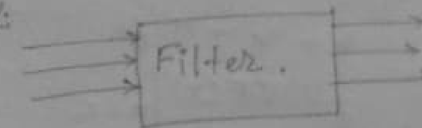
$$J^T(n)\, e(n) + \vec{J}^T(n)\vec{J}(n)\left(w(n+1) - w(n)\right)$$

$$\boxed{w(n+1) = w(n) - \left(\vec{J}^T(n)\vec{J}(n)\right)^{-1}\vec{J}^T(n)\, e(n).}$$

$$w(n+1) = w(n) - \left(\vec{J}^T(n)\vec{J}(n) + \delta I\right)^{-1}\vec{J}^T(n)\, e(n)$$

Linear Least Square Filter:





$\xi(\vec{w}) = f(e(i))$ $\qquad e(n) = d(n) - \vec{x}(n)\vec{w}(n)$.

Differentiating w.r.t $\vec{w}(n)$.

$$\nabla \vec{e}(n) = -x^T(n)$$

$$J(n) = -x(n).$$

$$w(n+1) = w(n) + \left(x^T(n)\, x(n)\right)^{-1} x^T(n)\left(d(n) - \vec{x}(n)\vec{w}(n)\right)$$

$$= \vec{w}(n) + \left(x^T(n)x(n)\right)^{-1} x^T(n)\, d(n) - \left(x^T(n)x(n)\right)^{-1}\left(x^T(n)\right) x(n)\, W(n)$$

$$= \vec{w}(n) + \left(x^T(n)\, x(n)\right)^{-1} x^T(n)\, d(n) - w(n)$$

$$= \left(x^T(n)\, x(n)\right)^{-1} x^T(n)\, d(n)$$

$w(n+1) =$

$X^+(n) = (X^T(n)\,X(n))^{-1}\,X^T(n)$

↳ Pseudo-inverse of data members.

$$\boxed{W(n+1) = X^+(n)\,d(n)}$$

Tutorial - 5

Wed 19.02.14

[Q] Apply Newton's method upto 1 iteration.

$$F(x) = e^{(x_1^2 - x_1 + 2x_2^2 + 4)}$$

$x_0 = [1 \ \ -2]^T$

$x_1 = x_0 - H^{-1} g_0$

$g = \nabla F(x)\big|_{x=x_0}$

i.e.,
$$g = \begin{bmatrix} \frac{\partial}{\partial x_1}F(x) \\ \frac{\partial}{\partial x_2}F(x) \end{bmatrix} = \begin{bmatrix} (2x_1-1)e^{(x_1^2-x_1+2x_2^2+4)} \\ 2x_2\, e^{(\;\;"\;\;)} \end{bmatrix}$$

$$g_0 = \begin{bmatrix} (2-1)e^{(1-1+8+4)} \\ 8\,e^{(1-1+8+4)} \end{bmatrix} = \begin{bmatrix} e^{12} \\ -8e^{12} \end{bmatrix} = \begin{bmatrix} 1.63\times10^{6} \\ \; \end{bmatrix}$$

$$H = \nabla^2 F(x) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2}F(x) & \frac{\partial^2}{\partial x_1 \partial x_2}F(x) \\ \frac{\partial^2}{\partial x_2 \partial x_1}F(x) & \frac{\partial^2}{\partial x_2^2}F(x) \end{bmatrix}$$

$\frac{\partial^2}{\partial x_1^2}F(x) = 2e^{12} + e^{12} = 3e^{12}$

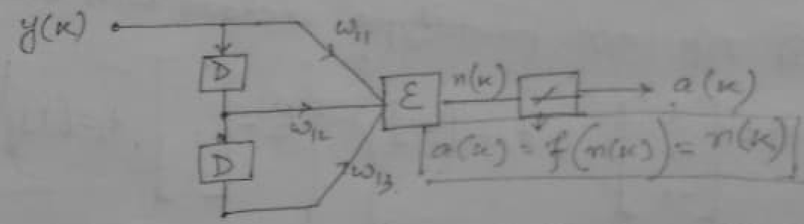
$\frac{\partial^2}{\partial x_1 \partial x_2}F(x) = 4x_2(2x_1-1)\,e^{12} = -8e^{12}$

$\frac{\partial^2}{\partial x_2^2}F(x) = 4e^{12} + 64e^{12} = 68e^{12}$.

so $H_0 = \begin{bmatrix} 0.049\times10^7 & -0.13\times10^7 \\ -0.13\times10^7 & 1.107\times10^7 \end{bmatrix}\Bigg|_{x=x_0}$

$$\boxed{x_1 = x_0 - H^{-1} g_0} = \begin{bmatrix} 0.971 \\ -1.886 \end{bmatrix}$$

[Q] Consider an Adaline filter, figure given below:



The weights are w_{11}, w_{12}, w_{13} & i/p sequence are
$\;\downarrow \;\;\; \downarrow \;\;\; \downarrow$
$\;2 \;\;\; -1 \;\;\; +3$

$y(k) = \{ \ldots 0, 0, 0, 5, -4, 0, 0, 0 \ldots \}$

where $y(0) = 5$, $y(1) = -4$

a) what is filter o/p just ~~prop~~ prior to $k=0$?

b) " " " " from $k=0$ to $k=5$?

c) How long the $y(0)$ compute the o/p?

Ans (i) $n(k) = wp^T = [w_{11} \ w_{12} \ w_{13}] \begin{bmatrix} y(k) \\ y(k-1) \\ y(k-2) \end{bmatrix}$

$a(k) = f(n(k)) = n(k)$

$a(-1) = [2 \ \ -1 \ \ 3] \begin{bmatrix} y(-1) \\ y(-2) \\ y(-3) \end{bmatrix} = 0 \quad \therefore$ o/p $= 0$

b) $a(0) = 10$.

$$a(1) = \begin{bmatrix} 2 & -1 & 3 \end{bmatrix} \begin{bmatrix} -4 \\ 5 \\ 0 \end{bmatrix} = -13$$

$a(2) = 19$

$a(3) = -12$

$a(4) = 0$

* Perceptron Learning

Q. I/p o/p prototype vectors will be

$$\left\{ p_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \; t_1 = [0] \right\} \left\{ p_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \; t_2 = [1] \right\}$$

Initial weight $w = [0.5 \; -1 \; -0.5]$

$b = [0.5]$

Find bias & weight is updated using perceptron learning rule.

Solⁿ: here

$$wp + b = \left[\begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0.5 \right]$$

$$= 0.5 + 1 + 0.5 + 0.5 = 2.5$$

$a = f(2.5) = 1$; hard limit func.

but $t_1 = 0$

$\therefore e = t_1 - a = 0 - 1 = -1$

The weight is updated.

$$W_{new} = W_{old} + e p^T$$

$$= \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} + (-1) \begin{bmatrix} 1 & -1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} \quad \boxed{(-1+0.5) \;\; (-1+1) \;\; (1-0.5)}$$

The bias is updated also,

$$b^{new} = b^{old} + e = 0.5 - 1 = -0.5$$

This completes the 1st iteration.

2nd iteration:

$$a = \text{hardlimit}\left(\begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + (-0.5) \right)$$

$$= \text{"} \; (-1.5) = 0.$$

but $t_2 = 1$.

$e = 1 - 0 = 1$.

$$\therefore W_{new} = \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} + (1) \begin{bmatrix} 1 & 1 & -1 \end{bmatrix}$$

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

$$\therefore b_{new} = -0.5 + 1 = 0.5$$

3rd iterations:

$$a = \text{hardlimit}\left(\begin{bmatrix} 0.5 & 1 & -0.5 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0.5\right)$$

$$= u(0.5) = 1.$$

$$e = 0 - 1 = -1.$$

$$\therefore W_{new} = \begin{bmatrix} 0.5 & 1 & -0.5 \end{bmatrix} + (-1)\begin{bmatrix} 1 & -1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} -0.5 & 2 & 0.5 \end{bmatrix}.$$

$$\therefore b_{new} = 0.5 - 1 = -0.5.$$