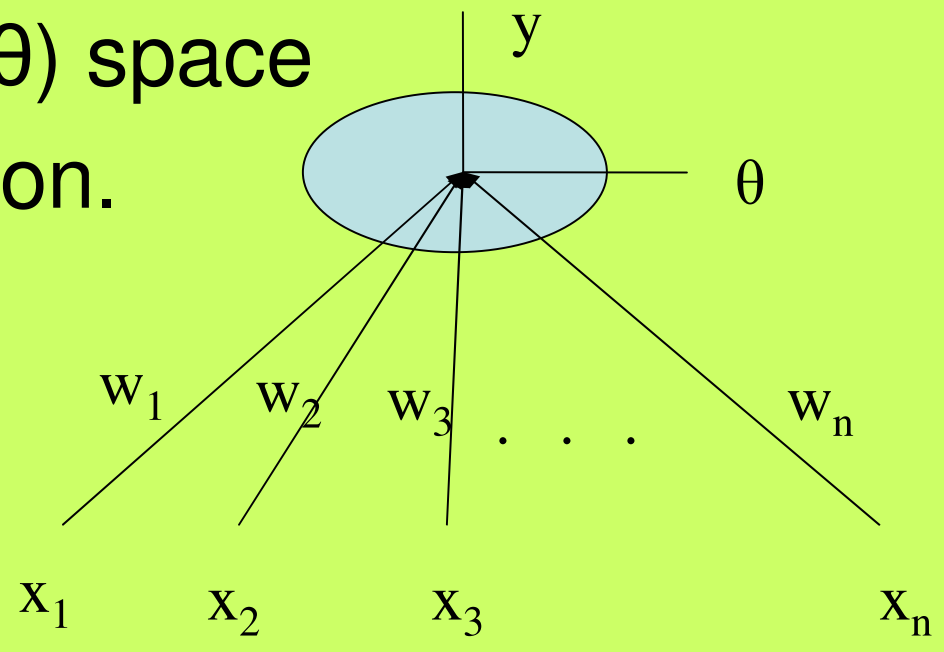
CS623: Introduction to Computing with Neural Nets *(lecture-3)*

Pushpak Bhattacharyya
Computer Science and Engineering Department
IIT Bombay

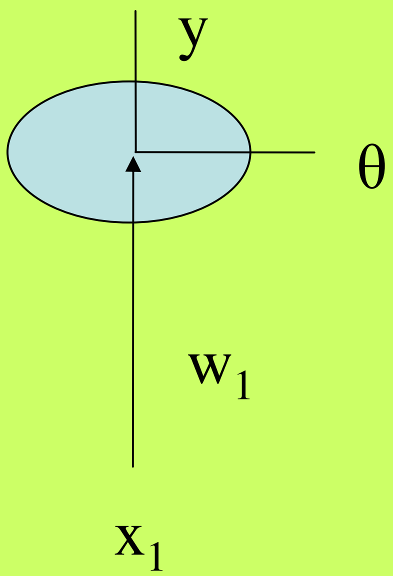
Computational Capacity of Perceptrons

Separating plane

- $\sum w_i x_i = \theta$ defines a linear surface in the (W,θ) space, where $W=\langle w_1,w_2,w_3,\ldots,w_n\rangle$ is an n-dimensional vector.
- A point in this (W,θ) space defines a perceptron.



The Simplest Perceptron



Depending on different values of
w and θ, four different functions are possible

Simplest perceptron contd.

x	f1	f2	f3	f4
0	0	0	1	1
1	0	1	0	1

<u>True-Function</u>

$\theta < 0$

$W < 0$

<u>0-function</u>

$\theta \geq 0$

$w \leq 0$

<u>Identity Function</u>

$\theta \geq 0$

$w > 0$

<u>Complement Function</u>

$\theta < 0$

$w \leq 0$

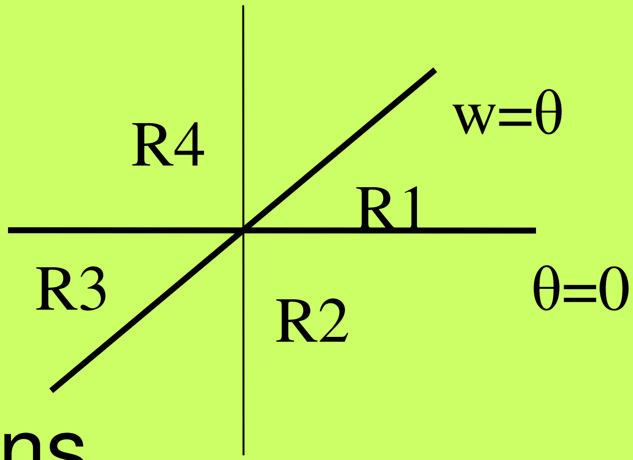
Counting the #functions for the simplest perceptron

- For the simplest perceptron, the equation is $w.x=\theta$.

Substituting $x=0$ and $x=1$,

we get $\theta=0$ and $w=\theta$.

These two lines intersect to form four regions, which correspond to the four functions.



Fundamental Observation

- The number of TFs computable by a perceptron is equal to the number of regions produced by 2^n hyper-planes,obtained by plugging in the values $<x_1,x_2,x_3,\ldots,x_n>$ in the equation

$$\sum_{i=1}^{n} w_i x_i = \theta$$

- **Intuition:** How many lines are produced by the existing planes on the new plane? How many regions are produced on the new plane by these lines?

The geometrical observation

- **Problem:** m linear surfaces called hyper-planes (each hyper-plane is of (d-1)-dim) in d-dim, then what is the max. no. of regions produced by their intersection?

  i.e. $R_{m,d} = ?$

Concept forming examples

- Max #regions formed by m lines in 2-dim is $R_{m,2} = R_{m-1,2} + ?$

  The new line intersects m-1 lines at m-1 points and forms m new regions.

  $$R_{m,2} = R_{m-1,2} + m , \quad R_{1,2} = 2$$

- Max #regions formed by m planes in 3 dimensions is

  $$R_{m,3} = R_{m-1,3} + R_{m-1,2} , \quad R_{1,3} = 2$$

Concept forming examples contd..

- Max #regions formed by m planes in 4 dimensions is

$$R_{m,4} = R_{m-1,4} + R_{m-1,3}\ , \quad R_{1,4} = 2$$

$$\mathbf{R_{m,d} = R_{m-1,d} + R_{m-1,d-1}}$$

Subject to

$R_{1,d} = 2$

$R_{m,1} = 2$

General Equation

$$\mathbf{R_{m,d} = R_{m-1,d} + R_{m-1,d-1}}$$

Subject to

$R_{1,d} = 2$

$R_{m,1} = 2$

All the hyperplanes pass through the origin.

Method of Observation for lines in 2-D

$$R_{m,2} = R_{m-1,2} + m$$
$$R_{m-1,2} = R_{m-2,2} + m-1$$
$$R_{m-2,2} = R_{m-3,2} + m-2$$
$$\vdots$$
$$R_{2,2} = R_{1,2} + 2$$

Therefore, $R_{m,2} = R_{m-1,2} + m$

$= 2 + m + (m-1) + (m-2) + \ldots + 2$

$= 1 + (1 + 2 + 3 + \ldots + m)$

$= 1 + [m(m+1)]/2$

Method of generating function

$R_{m,2} = R_{m-1,2} + m$

$f(x) = R_{1,2}\,x + R_{2,2}\,x^2 + R_{3,2}\,x^3 + \ldots + R_{i,2}\,x^m + \ldots + \alpha$ –>Eq1

$xf(x) = R_{1,2}\,x^2 + R_{2,2}\,x^3 + R_{3,2}\,x^4 + \ldots + R_{i,2}\,x^{m+1} + \ldots + \alpha$ –>Eq2

Observe that $R_{m,2} - R_{m-1,2} = m$

Method of generating functions cont…

Eq1 – Eq2 gives

$$(1-x)f(x) = R_{1,2}\, x + (R_{2,2} - R_{1,2})x^2 + (R_{3,2} - R_{2,2})x^3 + \ldots + (R_{m,2} - R_{m-1,2})x^m + \ldots + \alpha$$

$$(1-x)f(x) = R_{1,2}\, x + (2x^2 + 3x^3 + \ldots + mx^m + ..)$$

$$= 2x^2 + 3x^3 + \ldots + mx^m + ..$$

$$f(x) = (2x^2 + 3x^3 + \ldots + mx^m + ..)(1-x)^{-1}$$

Method of generating functions cont…

$f(x) = (2x^2 + 3x^3 + \ldots + mx^m + ..)(1 + x + x^2 + x^3 \ldots)$

→Eq3

Coeff of x^m is

$R_{m,2} = (2 + 2 + 3 + 4 + \ldots + m)$

$= 1 + [m(m+1)/2]$

The general problem of *m* hyperplanes in *d* dimensional space

$$c(m,d) = c(m-1,d) + c(m-1,d-1)$$

subject to

$$c(m,1) = 2$$

$$c(1,d) = 2$$

Generating function

$$f(x,y) = R_{1,1}xy + R_{1,2}xy^2 + R_{1,3}xy^3 + \ldots$$
$$+ R_{2,1}x^2y + R_{2,2}\, x^2y^2 + R_{2,3}x^2y^3+\ldots$$
$$+ R_{3,1}x^3y + R_{3,2}x^3y^2 + \ldots\ldots\ldots$$

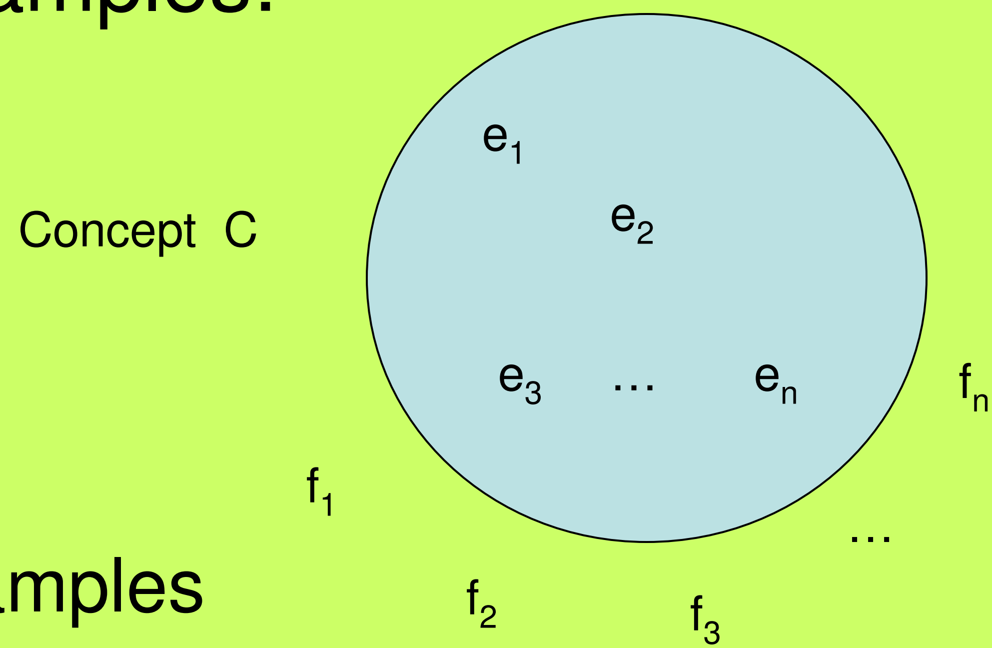
$$f(x,y) = \sum_{m\geq 1}\sum_{n\geq 1} R_{m,d}\, x^m y^d$$

of regions formed by m hyperplanes passing through origin in the d dimensional space

$$c(m,d) = 2.\Sigma^{d-1}{}_{i=0}{}^{m-1}c_i$$

Machine Learning Basics

- Learning from examples:

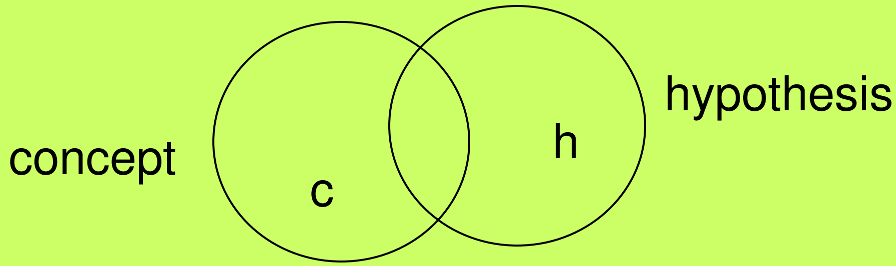


e_1, e_2, e_3… are +ve examples

f_1, f_2, f_3… are –ve examples

Machine Learning Basics cont..

- Training: arrive at hypothesis *h* based on the data seen.
- Testing: present new data to *h* test performance.



Feedforward Network

Limitations of perceptron

- Non-linear separability is all pervading
- Single perceptron does not have enough computing power
- Eg: XOR cannot be computed by perceptron

Solutions

- Tolerate error (Ex: *pocket algorithm* used by connectionist expert systems).
  - Try to get the best possible hyperplane using only perceptrons
- Use higher dimension surfaces

  Ex: Degree - 2 surfaces like parabola
- Use layered network

Pocket Algorithm

- Algorithm evolved in 1985 – essentially uses PTA
- Basic Idea:
  - Always preserve the best weight obtained so far in the "pocket"
  - Change weights, if found better (i.e. changed weights result in reduced error).

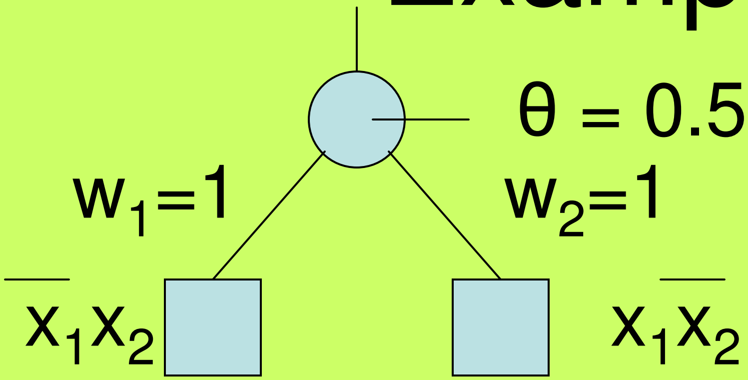
XOR using 2 layers

$$x_1 \oplus x_2 = \left(x_1 \overline{x_2}\right)\left(\overline{x_1} x_2\right)$$

$$= OR(AND(x_1, NOT(x_2)), AND(NOT(x_1), x_2)))$$

- Non-LS function expressed as a linearly separable function of individual linearly separable functions.

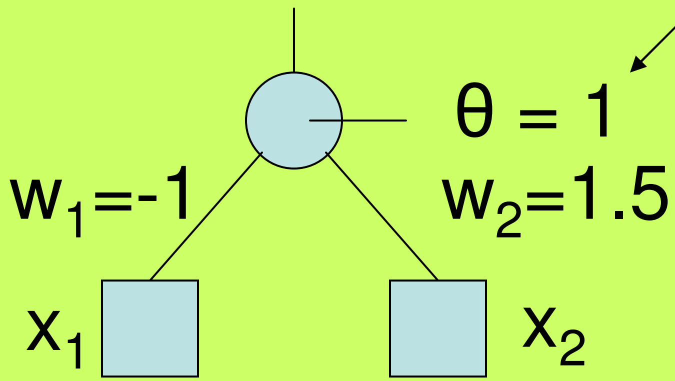
Example - XOR



← Calculation of XOR

x_1	x_2	$x_1 \bar{x}_2$
0	0	0
0	1	1
1	0	0
1	1	0

Calculation of $x_1 \bar{x}_2$



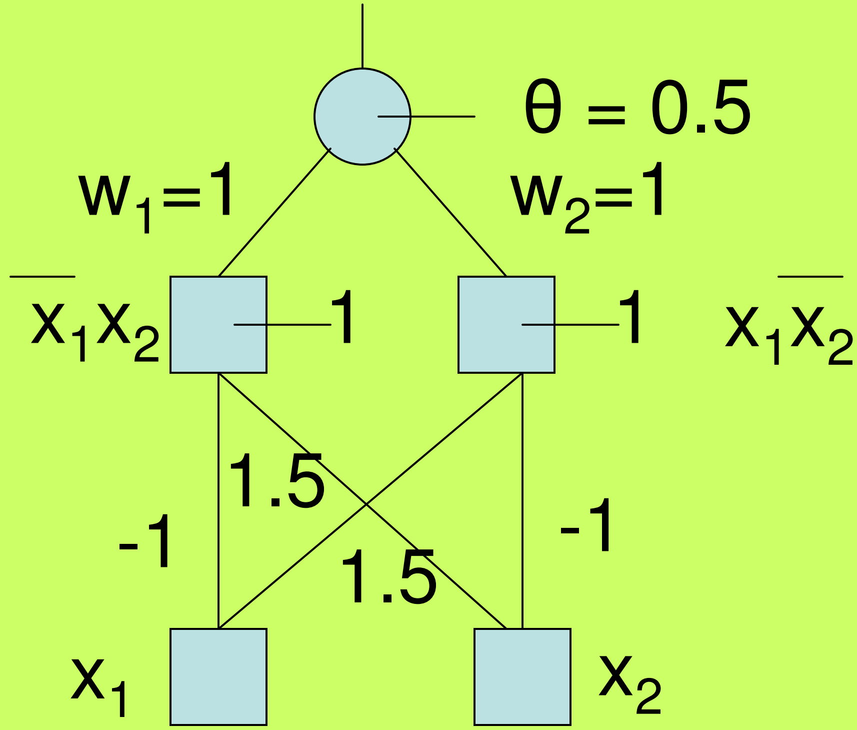
$0 < \Theta$

$w2 \geq \Theta$

$w1 < \Theta$

$w1 + w2 < \Theta$

Example - XOR



Some Terminology

- A multilayer feedforward neural network has
  - Input layer
  - Output layer
  - Hidden layer (asserts computation)

Output units and hidden units are called computation units.

Training of the MLP

- Multilayer Perceptron (MLP)
- Question:- How to find weights for the hidden layers when no target output is available?
- Credit assignment problem – to be solved by "*Gradient Descent*"

Gradient Descent Technique

- Let E be the error at the output layer

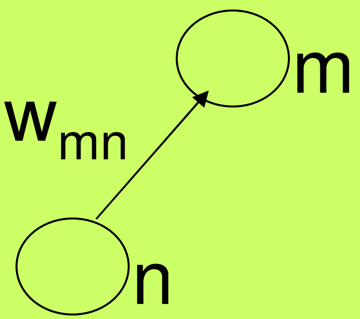
$$E = \frac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{n}(t_i - o_i)_j^2$$

- t_i = target output; o_i = observed output

- i is the index going over n neurons in the outermost layer
- j is the index going over the p patterns (1 to p)
- Ex: XOR:– p=4 and n=1

Weights in a ff NN

- w_{mn} is the weight of the connection from the n^{th} neuron to the m^{th} neuron
- E *vs* $\overline{W}$ surface is a complex surface in the space defined by the weights w_{ij}
- $-\frac{\delta E}{\delta w_{mn}}$ gives the direction in which a movement of the operating point in the w_{mn} co-ordinate space will result in maximum decrease in error

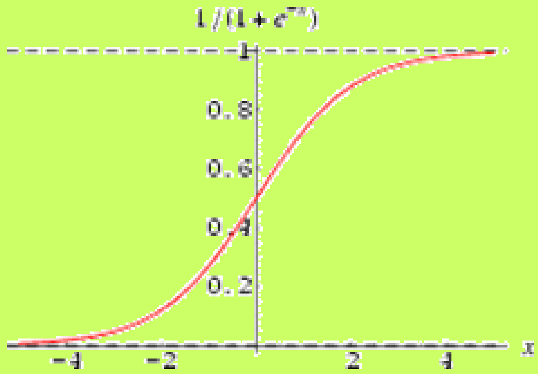


$$\Delta w_{mn} \propto -\frac{\delta E}{\delta w_{mn}}$$

Sigmoid neurons

- Gradient Descent needs a derivative computation
  - not possible in perceptron due to the discontinuous step function used!

  → Sigmoid neurons with easy-to-compute derivatives used!



$$y \rightarrow 1 \text{ as } x \rightarrow \infty$$
$$y \rightarrow 0 \text{ as } x \rightarrow -\infty$$

- Computing power comes from non-linearity of sigmoid function.

Derivative of Sigmoid function

$$y = \frac{1}{1+e^{-x}}$$

$$\frac{dy}{dx} = -\frac{1}{(1+e^{-x})^2}(-e^{-x}) = \frac{e^{-x}}{(1+e^{-x})^2}$$

$$= \frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right) = y(1-y)$$

Training algorithm

- Initialize weights to random values.
- For input $x = <x_n, x_{n-1}, \ldots, x_0>$, modify weights as follows

  Target output = t, Observed output = o

$$\Delta w_i \propto -\frac{\delta E}{\delta w_i}$$

$$E = \frac{1}{2}(t - o)_2$$

- Iterate until $E < \delta$ (threshold)

Calculation of Δw_i

$$\frac{\delta E}{\delta W_i} = \frac{\delta E}{\delta net} \times \frac{\delta net}{\delta W_i} \left(where \; : net = \sum_{i=0}^{n-1} w_i x_i \right)$$

$$= \frac{\delta E}{\delta o} \times \frac{\delta o}{\delta net} \times \frac{\delta net}{\delta W_i}$$

$$= -(t-o)o(1-o)x_i$$

$$\Delta w_i = -\eta \frac{\delta E}{\delta w_i} (\eta = \text{learning constant}, \; 0 \le \eta \le 1)$$

$$\Delta w_i = \eta (t-o)o(1-o)x_i$$

Observations

Does the training technique support our intuition?

- The larger the x_i, larger is Δw_i
  - Error burden is borne by the weight values corresponding to large input values