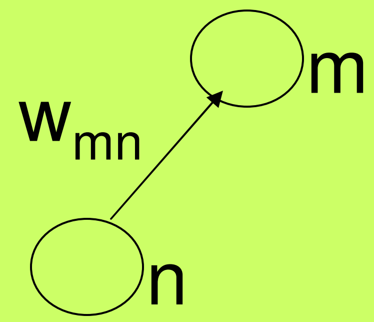
# CS623: Introduction to Computing with Neural Nets *(lecture-4)*

Pushpak Bhattacharyya
Computer Science and Engineering Department
IIT Bombay

# 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

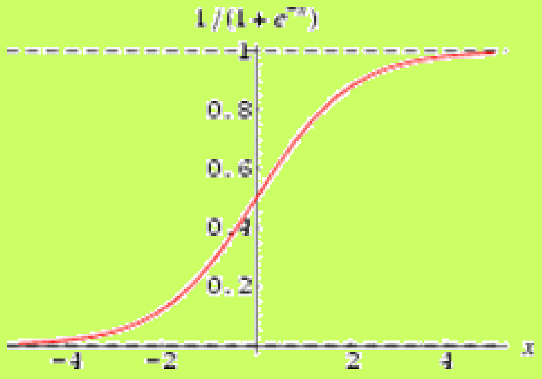



$$\Delta w_{mn} \propto -\frac{\delta E}{\delta w_{mn}}$$

# 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 = <$x_n$,$x_{n-1}$,…,$x_0$>, 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 $\Delta w_i$

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

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

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

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

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

# Observations

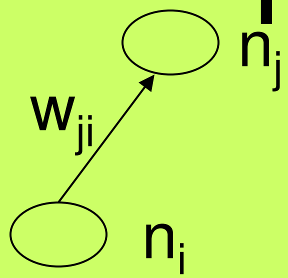
*Does the training technique support our intuition?*

- The larger the $x_i$, larger is $\Delta w_i$
  - Error burden is borne by the weight values corresponding to large input values

# Observations contd.

- $\Delta w_i$ is proportional to the departure from target
- Saturation behaviour when o is 0 or 1
- If $o < t$, $\Delta w_i > 0$ and if $o > t$, $\Delta w_i < 0$ which is consistent with the Hebb's law

# Hebb's law




- If $n_j$ and $n_i$ are both in excitatory state (+1)
  - Then the change in weight must be such that it enhances the excitation
  - The change is proportional to both the levels of excitation

$$\Delta w_{ji} \propto e(n_j)\, e(n_i)$$

- If $n_i$ and $n_j$ are in a mutual state of inhibition ( one is +1 and the other is -1),
  - Then the change in weight is such that the inhibition is enhanced (change in weight is negative)

# Saturation behavior

- The algorithm is iterative and incremental
- If the weight values or number of input values is very large, the output will be large, then the output will be in saturation region.
- The weight values hardly change in the saturation region

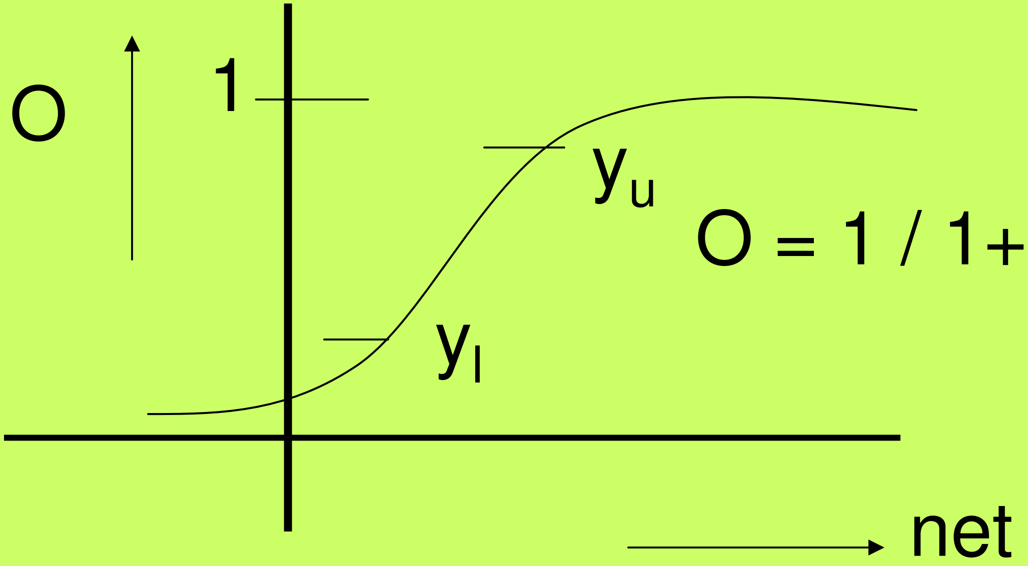
# If Sigmoid Neurons Are Used, Do We Need MLP?

Does sigmoid have the power of separating non-linearly separable data?

Can sigmoid solve the X-OR problem

(X-ority is non-linearly separable data)

link




$O = 1 / 1+ e^{-net}$

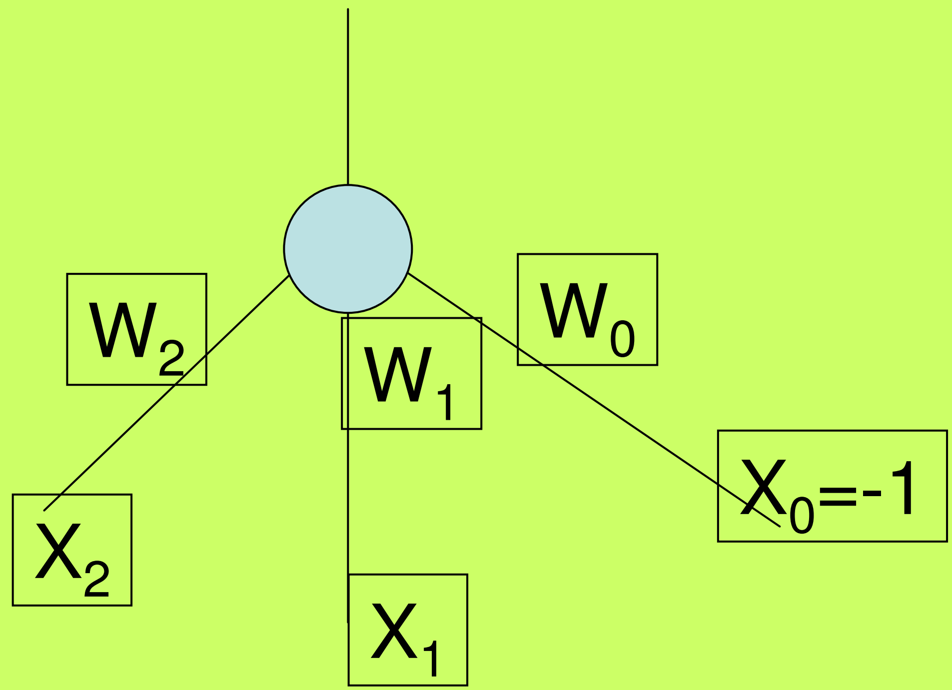
$O = 1$ if $O > y_u$

$O = 0$ if $O < y_l$

Typically $y_l << 0.5$ , $y_u >> 0.5$

# Inequalities

$$O = 1 / (1 + e^{-net})$$



# <0, 0>

$O = 0$

i.e $0 < y_l$

$$1 / 1 + e^{(-w_1 x_1 - w_2 x_2 + w_0)} < y_l$$

i.e. $(1 / (1 + e^{w_o})) < y_l$ (1)

# <0, 1>

$O = 1$

i.e. $0 > y_u$

$$1/(1+ e^{(-w_1 x_1 - w_2 x_2 + w_0)}) > y_u$$

$$(1 / (1+ e^{-w_2+w_0})) > y_u \qquad (2)$$

# <1, 0>

O = 1

i.e. $(1/1+ e^{-w_1+w_0}) > y_u$ (3)

# <1, 1>

O = 0

i.e. $1/(1+ e^{-w_1-w_2+w_0}) < y_l$ (4)

# Rearranging, 1 gives

$$1/(1+ e^{w}{}_{o}) < y_l$$

i.e. $1+ e^{w}{}_{o} > 1 / y_l$

i.e. $W_o > \ln((1- y_l) / y_l)$ (5)

# 2 Gives

$$1/1+ e^{-w_2+w_0} > y_u$$

i.e. $1+ e^{-w_2+w_0} < 1 / y_u$

i.e. $e^{-w_2+w_0} < 1-y_u / y_u$

i.e. $-W_2 + W_o < \ln (1-y_u) / y_u$

i.e. $W_2 - W_o > \ln (y_u / (1-y_u))$ (6)

3 Gives

$$W_1 - W_o > \ln \left( y_u / (1 - y_u) \right) \qquad (7)$$

4 Gives

$$-W_1 - W_2 + W_o > \ln \left( (1 - y_l) / y_l \right) \qquad (8)$$

# 5 + 6 + 7 + 8 Gives

$$0 > 2\ln (1- y_l)/ y_l + 2 \ln y_u / (1 - y_u)$$

i.e. $0 > \ln [ (1- y_l)/ y_l * y_u / (1 - y_u)]$

i.e. $((1- y_l) / y_l) * (y_u / (1 - y_u)) < 1$

i. $[(1- y_l) / (1- y_y)] * [y_u / y_l] < 1$

ii. 2) $Y_u >> 0.5$

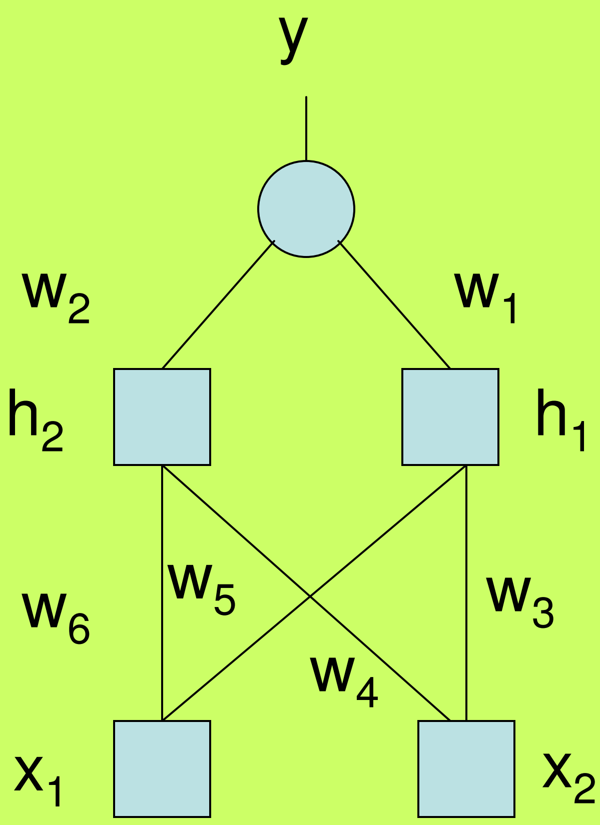
iii. 3) $Y_l << 0.5$

From i, ii and iii; Contradiction, hence sigmoid cannot compute X-OR

# Exercise

Use the fact that any non-linearly separable function has positive linear combination to study if sigmoid can compute any non-linearly separable function.

# Non-linearity is the source of power




$y = m_1(h_1.w_1 + h_2.w_2) + c_1$

$h_1 = m_2(w_3.x_1 + w_4.x_2) + c_2$

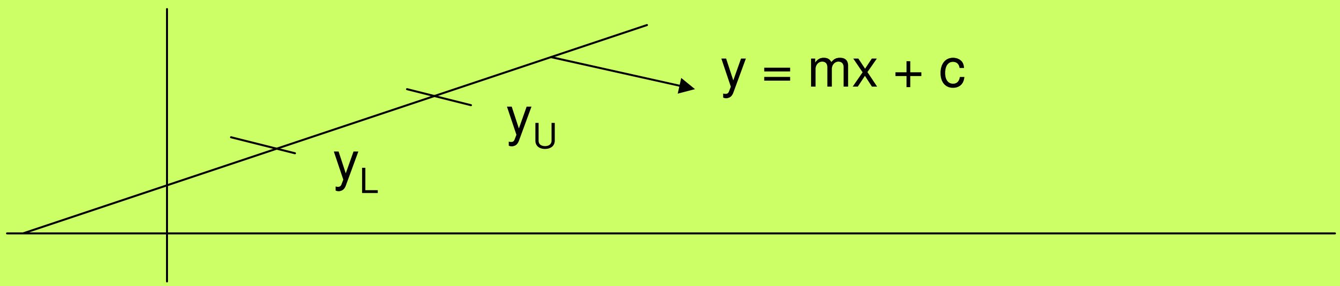
$h_2 = m_3(w_5.x_1 + w_6.x_2) + c_3$

Substituting $h_1$ & $h_2$

$y = k_1x_1 + k_2x_2 + c'$

***Thus a multilayer network can be collapsed into an eqv. 2 layer n/w without the hidden layer***

# Can a linear neuron compute X-OR?

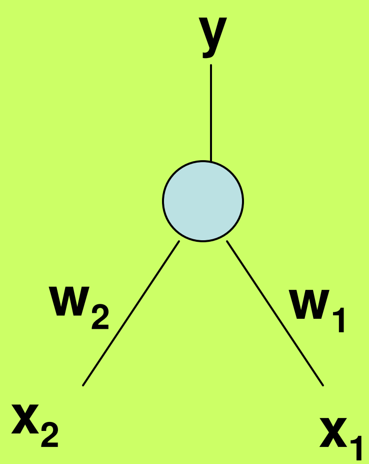



- $y > y_U$ is regarded as $y = 1$
- $y < y_L$ is regarded as $y = 0$

$$y_U > y_L$$

# Linear Neuron & X-OR

We want




$$y = w_1x_1 + w_2x_2 + c$$

# Linear Neuron 1/4

for (1,1), (0,0)

$y < y_L$

For (0,1), (1,0)

$y > y_U$

$y_U > y_L$

Can ($w_1$, $w_2$, c) be found

# Linear Neuron 2/4

<u>(0,0)</u>

$y = w_1.0 + w_2.0 + c$

$\quad = c$

$y < y_L$

$c < y_L \qquad - (1)$

<u>(0,1)</u>

$y = w_1.1 + w_2.0 + c$

$y > y_U$

$w_1 + c > y_U \quad - (2)$

# Linear Neuron 3/4

1,0

$w_2 + c > y_U$ - (3)

1,1

$w_1 + w_2 + c < y_L$ - (4)

$y_U > y_L$ - (5)

# Linear Neuron 4/4

$c < y_L$ - (1)

$w_1 + c > y_U$ - (2)

$w_2 + c > y_U$ - (3)

$w_1 + w_2 + c < y_L$ - (4)

$y_U > y_L$ - (5)

Inconsistent

# Observations

- A linear neuron cannot compute XOR
- A multilayer network with linear characteristic neurons is collapsible to a single linear neuron.
- Therefore addition of layers does not contribute to computing power.
- Neurons in feedforward network must be non-linear
- Threshold elements will do iff we can linearize a non-linearly function.