Day-1

M T W T F S S
Page No.:
Date:
YOUVA

XOR problem (internal working)

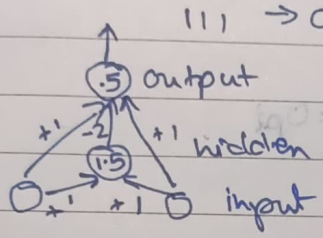
- single layer perceptron can only solve linearly separable

$(0,0), (1,1) \rightarrow (0)$
$(1,0), (0,1) \rightarrow (1)$
} straight line can't solve this

- solvable when mapped to high-D (add one more input)

000 → 0
010 → 1
100 → 1
111 → 0

- hidden layer introduced
  (feat 1 A || B
  feat 2 A && B)



XOR = (A or B) And (Not(A And B))
= $(A \vee B) \wedge \neg(A \wedge B)$

Delta Rule - only for single layer (linear) → act.

$$w_i = w_i + \eta(d - \hat{y})x_i$$

Generalised Δ rule:

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2$$

$E = \sum E_p$, overall error
$\delta_{pj} = t_{pj} - o_{pj}$
o_{pj} = j^{th} element of actual outp.
$i^{th} \rightarrow j^{th}$ unit

$$-\frac{\partial E_p}{\partial w_{ij}} = \delta_{pj} i_{pi}$$

$$= \frac{\partial E_p}{\partial o_{pj}} \times \frac{\partial o_{pj}}{\partial w_{ij}}$$

$-(t_{pj} - o_{pj}) = -\delta_{pj}$

$o_{pj} = \sum w_{ji} i_{pi}$ (for linear unit)

$\frac{\partial \theta_p}{\partial w_j} = i_{pi}$

$\therefore \Rightarrow -\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} i_i$

finally $\frac{\partial E}{\partial w_{ji}} = \sum_p \frac{\partial E_p}{\partial w_{ji}}$

- addition of hidden unit → graph no longer concave.

$$net_{pj} = \sum_i w_{ji} o_{pi} \quad -(7)$$

$$o_{pj} = \underbrace{f_j}_{\text{differentiable}}(net_{pj})$$

Semilinear func^n → whose output func^n is diff func^n of input unit

Correct generalisation if,

$$\Delta_p w_{ji} \alpha - \frac{\partial E_p}{\partial w_{ji}} \quad -(8)$$

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\underbrace{\partial net_{pj}}_{\delta_{pj}}} \frac{\partial net_{pj}}{\partial w_{ji}}$$

(7) ↳ $\frac{\partial net_{pj}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum w_{jk} o_{pk} = o_{pi}$

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} o_{pi}$$

(8) ↳ $\Delta_p w_{ji} = \eta \delta_{pj} o_{pi}$

↳ what should for each unit u_j

for $\delta_{pj} = \frac{-\partial E}{\partial net_{pj}}$, apply partial $= \frac{-\partial E_p}{\partial net_{pj}} = \frac{-\partial E_p}{\partial o_{pj}} \cdot \underbrace{\frac{\partial o_{pj}}{\partial net_{pj}}}_{= f'(net_{pj})} \quad -(12)$

- Two cases

(1) u_j is the output of network

$$\frac{\partial E_p}{\partial o_{pj}} = -(t_{pj} - o_{pj}) \xrightarrow{(12)} \delta_{pj} = (t_{pj} - o_{pj}) f'_j(net_{pj})$$

(2) Not

$$\sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial o_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial}{\partial o_{pj}} \sum_k w_{kl} o_{pi} = \sum_k \frac{\partial E_p}{\partial net_{pk}} w_{kj} = \sum_k \delta_{pk} w_{kj}$$

$$\delta_{pj} = f'_j(net_{pj}) \sum_k \delta_{pk} w_{kj}$$

$\Delta w_{ji} = \eta \delta_{pj} o_{pj}$

$\delta_{pj} = (t_{pj} - o_{pj}) f'_j(net_{pj})$ (u output unit)

$\delta_{pj} = f'(net_{pj}) \sum_k \delta_{pk} w_{kj}$ (u not output)

* 2 phases → 2nd phase involve ~~comparing~~ backward pass.
  - input propagate to find output o_{pj} for each unit
  - compared with target ⇝ error signal δ_{pj} for each output unit.
- Compute δ for penultimate layer. This propagate the error back one layer & same process for every layer.

Logistic

$$o_{pj} = \frac{1}{1 + e^{-(\sum_i w_{ji} o_{pi} + \theta_j)}} \qquad net_{pj} = \sum w_{ji} o_{pi} + \theta_j.$$

$$\frac{do_{pj}}{dnet_{pj}} = o_{pj}(1 - o_{pj})$$

$$\Rightarrow \delta_{pj} = (t_{pj} - o_{pj})\, o_{pj}(1 - o_{pj}) \quad [\text{output unit } u]$$

$$= o_{pj}(1 - o_{pj}) \sum \delta_{pk} w_{kj} \quad [\text{hidden unit } w_{ij}]$$

max for $0.5 = o_{pj}$

min for $o_{pj} = 0, 1$

Learning rate

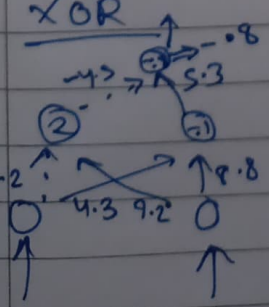
- keep large without oscillation

Momentum → helps

$$\Delta w_{ji}(n+1) = \eta(\delta_{pj} o_{pi}) + \alpha \Delta w_{ji}(n)$$

- filter high freq. variations in weight surface.
- sharp curvature tends to cause oscillation
- filter out high curvatures & allows effective weight to be larger.

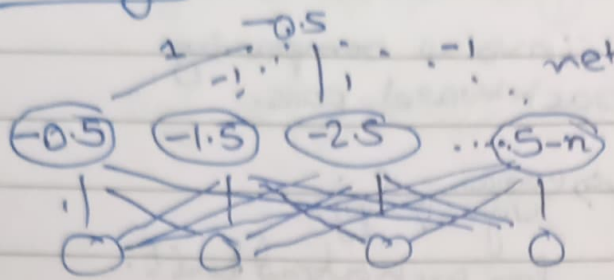
XOR



Symmetry breaking

- start at equal value
- equal error signal
- stuck at local maximum

Parity



net input from hidden → 0 (even)
→ 1 (odd)

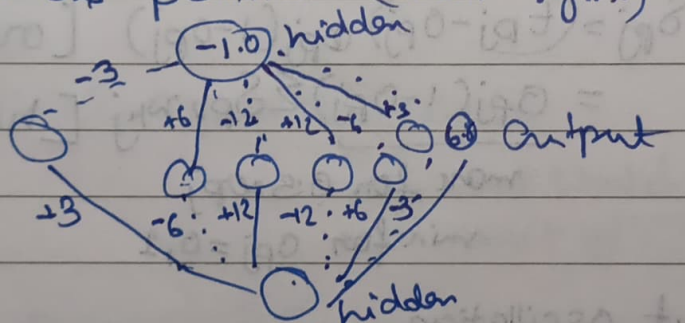
- not readily discovered by unsupervised learning scheme

Encoding problem

- $\log_2 N$ hidden unit force to learn in binary.
- tendency of hidden unit to take extreme value.
  But when req. to learn → can learn (logistic activation?)

Symmetric problem

- hidden unit receive zero act. from input (symm.)
- atleast one positive (non-sym.)



Eqⁿ of backprop

$$\delta^L = \nabla_a C \odot \sigma'(z^L) = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L}$$

$$\delta^L = ((w^{L+1})^T \delta^{L+1}) \odot \sigma'(z^L)$$

$$\frac{\partial C}{\partial b_j^L} = \delta_j^L$$

$$\frac{\partial C}{\partial w_{jk}^L} = a_k^{L-1}\delta_j^L$$

Proof for BP2 → $\delta_j^L = \frac{\partial C}{\partial z_j^L} = \sum_k \frac{\partial C}{\partial z_k^{L+1}} \frac{\partial z_k^{L+1}}{\partial z_j^L}$

$$= \sum_k \frac{\partial z_k^{L+1}}{\partial z_j^L}\delta_k^{L+1}$$

$$z_k^{L+1} = \sum_j w_{kj}^{L+1} a_j^L + b_k^{L+1} = \sum_j w_{kj}^{L+1}\sigma(z_j^L) + b_k^{L+1}$$

$$\frac{\partial z_k^{L+1}}{\partial z_j^L} = w_{kj}^{L+1}\sigma'(z_j^L)$$

finally, $\delta_j^L = \sum w_{kj}^{L+1}\delta_k^{L+1}\sigma'(z_j^L)$