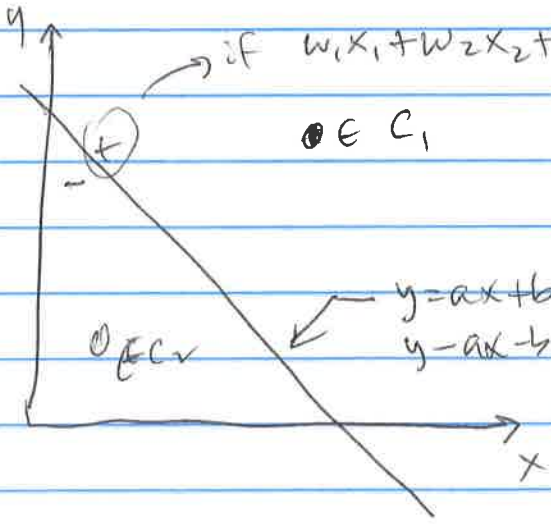
CS6640 Weeks 15-16 26 Nov

Neural Networks: a function
* maps input → output
* given training set, it can learn map

Perceptron: learns a linear decision surface between 2 linearly separable classes



more general
$w_1x_1+w_2x_2+w_3=0$
$\bar{w}^T\bar{x}+w_{n+1}=0$
w_{n+1} ← bias
$\bar{x}$ ← feature vector
$\bar{w}$ ← weight vector

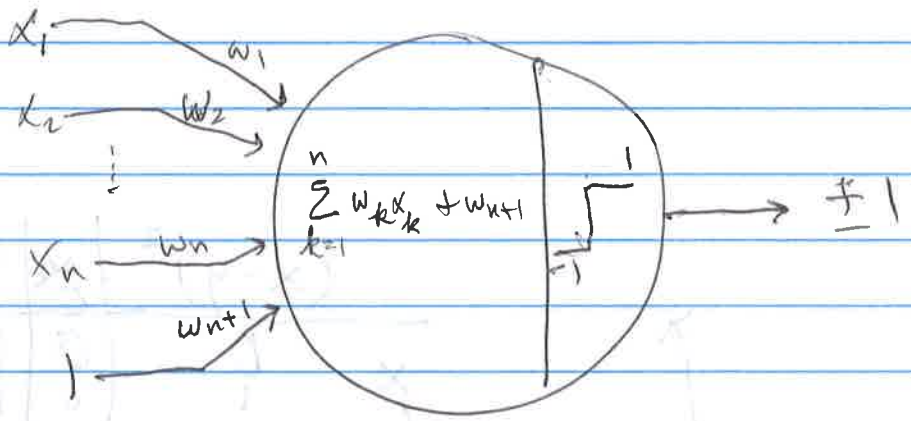
Perceptron algorithm Let $\alpha>0$ be correction increment learning rate
$\bar{w}(1)$: arbitrary $w_{n+1}(1)$ arbitrary
change = 1;
while change == 1
  change = 0;
  if $\bar{x}(k) \in C_1$ and $\bar{w}^T(k)\bar{x}(k)+w_{n+1}(k) \le 0$
    then $\bar{w}(k+1) = \bar{w}(k) + \alpha\bar{x}(k)$
      $w_{n+1}(k+1) = w_{n+1}(k) + \alpha$ change = 1
  elseif $\bar{x}(k) \in C_2$ and $\bar{w}^T(k)\bar{x}(k)+w_{n+1}(k) \ge 0$
    then $\bar{w}(k+1) = \bar{w}(k) - \alpha\bar{x}(k)$
      $w_{n+1}(k+1) = w_{n+1}(k) - \alpha$ change = 1
  else $\bar{w}(k+1) = \bar{w}(k)$
    $w_{n+1}(k+1) = w_{n+1}(k)$

Can augment weight and feature vectors:

$$\bar{w} = [w_1, w_2, \ldots, w_n, w_{n+1}]^T$$
$$\bar{x} = (x_1, x_2, \ldots, x_n, 1]$$

Then use $\bar{w}^T \bar{x} = \begin{cases} > 0 & \bar{x} \in C_1 \\ < 0 & \bar{x} \in C_2 \end{cases}$

Schematic:



Alternative learning algorithm: (LMSE)

$r \equiv$ desired response

Find $\bar{w}$ that minimizes MSE (mean square error) between desired and actual

$$E(\bar{w}) = \frac{1}{2}(r - \bar{w}^T \bar{x})^2$$

error ← weight vector ← pattern ← desired response

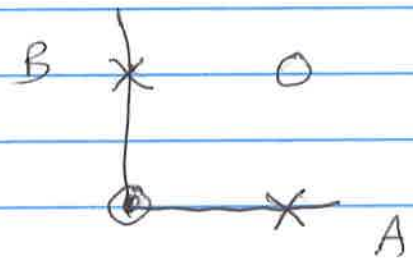
$$= -(r - \bar{w}^T \bar{x})\bar{x}$$

Use: $\bar{w}(k+1) = \bar{w}(k) - \alpha \left[\frac{\partial E(\bar{w})}{\partial \bar{w}}\right]_{\bar{w} = \bar{w}(k)}$

$0 < \alpha < 2$

$$\bar{w}(k+1) = \bar{w}(k) + \alpha [r(k) - \bar{w}^T(k)\bar{x}(k)]\bar{x}(k)$$

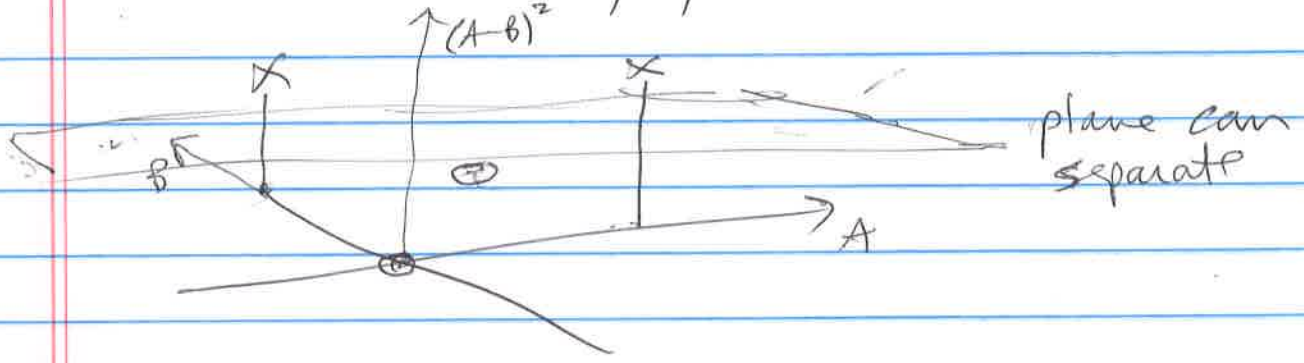
Perceptron's limits: XOR



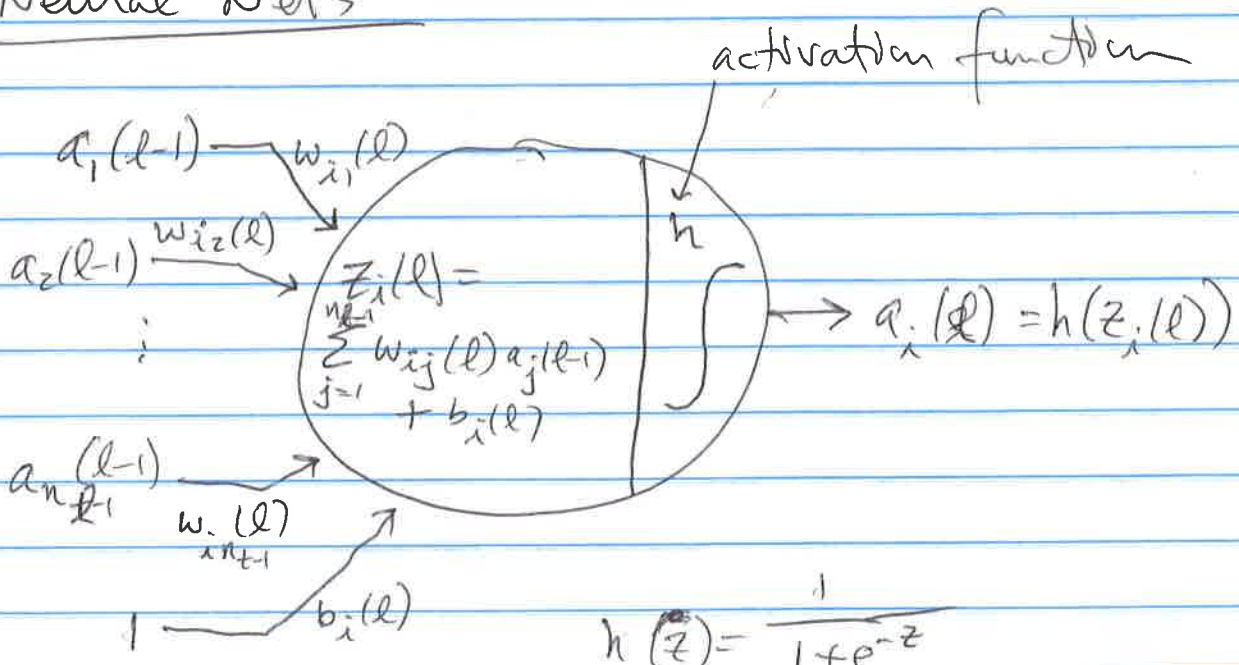
o class 1 $\{(0,0),(1,1)\}$
x class 2 $\{(0,1),(1,0)\}$

No line separates

But consider: $A, B, (A-B)^2$

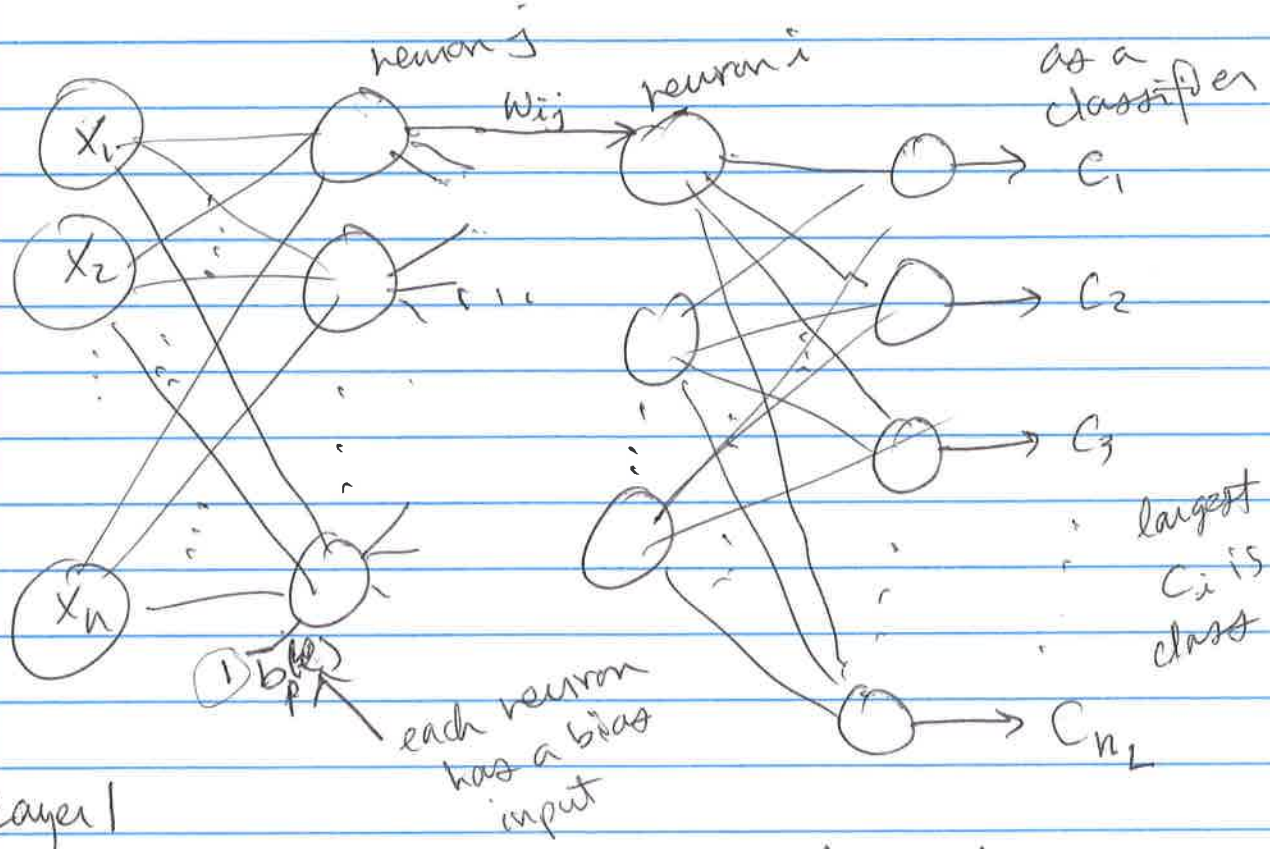


Neural Nets



$$h(z) = \frac{1}{1+e^{-z}}$$

$$h'(z) = h(z)\,(1-h(z))$$



$l = 1, 2, \ldots, L$ layers

n_l : # neurons in layer l

Feed forward

comes from input vector $\bar{x}$

$$a_j(1) = x_j \qquad j = 1, 2, \ldots, n_1$$

$$z_i(l) = \sum_{j=1}^{n_{l-1}} w_{ij}(l)\, a_j(l-1) + b_i(l)$$

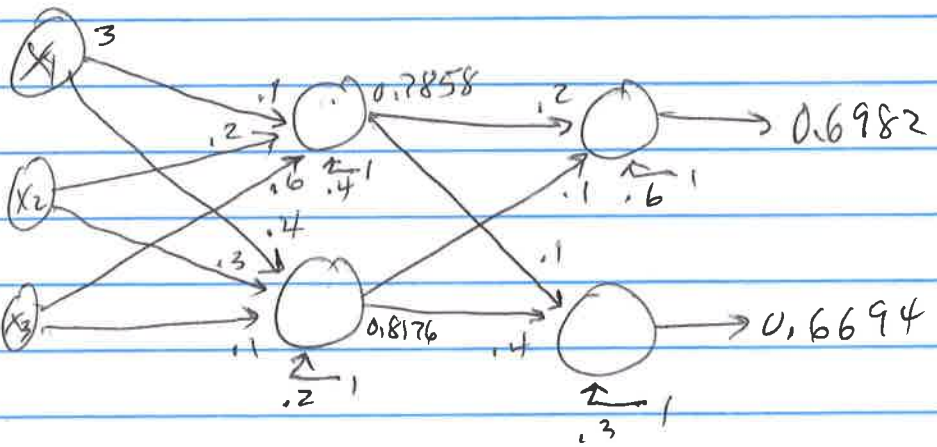
net input $= net_i$

$$a_i(l) = h(z_i(l)) \qquad i = 1 : n_l$$

$$a_i(L) = h(z_i(L)) \qquad i = 1 : n_L$$

Example feed forward

$$\bar{x} = \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix}$$



Matrix formulation

$$W(\ell) = \begin{bmatrix} w_{11}(\ell) & w_{12}(\ell) & \cdots & w_{1 n_{\ell-1}}(\ell) \\ & \vdots & & \\ w_{n_\ell 1}(\ell) & w_{n_\ell 2}(\ell) & \cdots & w_{n_\ell n_{\ell-1}}(\ell) \end{bmatrix}$$

→ a row for each node in layer ℓ

$$\bar{z}(\ell) = W(\ell)\,\bar{a}(\ell-1) + \bar{b}(\ell) \qquad \ell = 2, \ldots, L$$

$$\bar{a}(\ell) = h\left(\bar{z}(\ell)\right)$$

Alg.

$A(1) = \bar{x}$ (we just do 1 feature vector)

for $\ell = 2, L$ compute $\bar{z}(\ell) = W(\ell)\bar{A}(\ell-1) + \bar{B}(\ell)$

end $\bar{A}(\ell) = h(\bar{z}(\ell))$

$A(L)$ is output

NN defined by:
* weights
* bias values
* activation function

← produce these ← training from examples

use backpropagation:

* input pattern vector
* forward pass → classification error
* backward pass → feed error back
* update weights + biases → assign responsibility

Minimize error function

$\bar{r}$: desired response to feature vector $\bar{x}$
$\bar{a}(L)$: actual response

$$E_j = \frac{1}{2}\left[r_j - a_j(L)\right]^2 \qquad j = 1 : n_L$$

$$E = \sum_{j=1}^{n_L} E_j = \frac{1}{2}\sum_{j=1}^{n_L}\left(r_j - a_j(L)\right)^2$$

$$= \frac{1}{2}\left\|\bar{r} - \bar{a}(L)\right\|^2$$

Total network output error: sum of errors of all patterns

Basically, use $\frac{\partial E}{\partial z_i(l)}$ and get $\frac{\partial E}{\partial w_{ij}(l)}$ & $\frac{\partial E}{\partial b_i(l)}$

$z_j(\ell)$: net input to node j in layer ℓ

Let $\delta_j(\ell) \equiv \frac{\partial E}{\partial z_j(\ell)}$

Start at output:

$$\delta_j(L) = \frac{\partial E}{\partial z_j(L)} = \frac{\partial E}{\partial a_j(L)} h'(z_j(L))$$

e.g., $\delta_j(L) = \underbrace{h(z_j(L))}_{\text{from forward pass}} \underbrace{[1-h(z_j(L))]}_{\text{output}} \underbrace{[a_j(L) - r_j]}_{\text{target}}$

relate $\delta_j(\ell)$ in terms of $\delta_j(\ell+1)$

$$\delta_j(\ell) = h'(z_j(\ell)) \sum_i w_{ij}(\ell+1)\, \delta_i(\ell+1)$$

for $\ell = L-1, \ldots, 2$

but $\frac{\partial E}{\partial w_{ij}(\ell)} = a_j(\ell-1)\, \delta_i(\ell)$

$$\frac{\partial E}{\partial b_i(\ell)} = \delta_i(\ell)$$

$\Rightarrow$

$$w_{ij}(\ell) = w_{ij}(\ell) - \alpha\, \delta_i(\ell)\, a_j(\ell-1)$$

$$b_i(\ell) = b_i(\ell) - \alpha\, \delta_i(\ell)$$