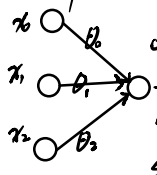
Neuron Network

input (dentrites) 树突



output (axons)

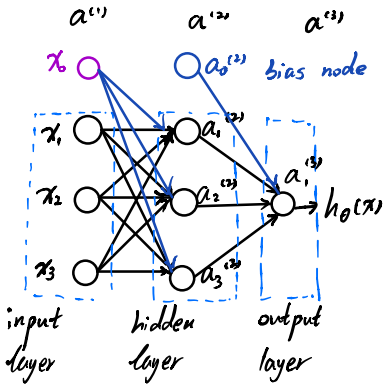
$h_\theta(x) = \text{sigmod}(\theta^T x)$

sigmod: activation function $g(z) = \frac{1}{1+e^{-z}}$

θ : the weight

$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow [\ ] \rightarrow h_\theta(x)$

$L = 3$, 3 layers



$x_0 (a_0^{(1)}), a_0^{(2)}$: the "bias node", always 1

$a_0^{(2)}, a_1^{(2)}, a_2^{(2)}$: the "activation" unit

$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \rightarrow h_\theta(x)$

$a_i^{(j)}$ = activation unit i in layer j

$\theta^{(j)}$ = matrix weights controlling function mapping from layer j to layer $j+1$

each layer get its own matrix of weight $\theta^{(j)}$

If network has s_j units in layer j and s_{j+1} units in layer $j+1$, then $\theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$, the "+1" comes from "bias nodes"

e.g. layer 1 has 2 input and layer 2 has 4 activation nodes, dimension of $\theta^{(1)}$ is 4×3 ($s_j = 2$, $s_{j+1} = 4$, $s_{j+1} \times (s_j + 1) = 4 \times 3$)

$$a_1^{(2)} = g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3) = g(z_1^{(2)})$$
$$a_2^{(2)} = g(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3) = g(z_2^{(2)})$$
$$a_3^{(2)} = g(\theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3) = g(z_3^{(2)})$$
$$h_\theta(x) = g(\theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)}) = g(z_1^{(3)})$$

$$z_k^{(2)} = \theta_{k,0}^{(1)} x_0 + \theta_{k,1}^{(1)} x_1 + \cdots + \theta_{k,n}^{(1)} x_n$$

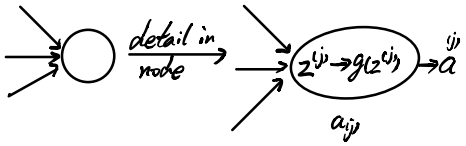
vectorize x and z^{j} :

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \vdots \\ z_n^{(j)} \end{bmatrix}$$

Setting $x = a^{(1)}$

$$z^{(j)} = \theta^{(j-1)} a^{(j-1)}$$

$$a^{(j)} = g(z^{(j)}) = g(\theta^{(j-1)} a^{(j-1)})$$



where $g()$ applied element-wise to vector $z^{(j)}$,

when computed $a^{(j)}$, we add a bias node $a_0^{(j)} = 1$, compute another z vector

$$z^{(j+1)} = \theta^{(j)} a^{(j)}$$

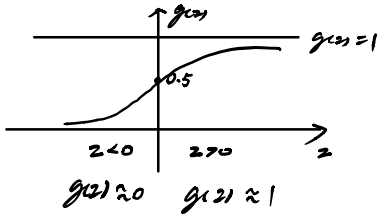
$$a^{(j+1)} = g(z^{(j+1)})$$

when comes to the last layer, $j = L-1$, $\Theta^{(j)} = \Theta^{(L-1)}$ has only one row, the multiplition result is a single number; the final hypothesis:

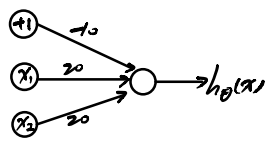
$$h_\theta(x) = a^{(j+1)} = g(z^{(j+1)})$$

Examples:

newral network used to simulate logicod gates:



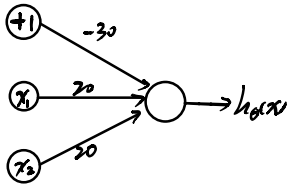
$h_\theta(x) = g(z)$



x_1	x_2	$h_\theta(x)$
0	0	$g(-10) \approx 0$
1	0	$g(10) \approx 1$
0	1	$g(10) \approx 1$
1	1	$g(30) \approx 1$

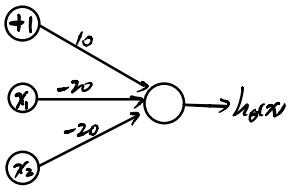
"OR" function

$\Theta^{(1)} = [-10 \ 20 \ 20]$



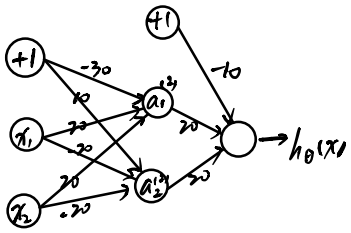
"AND" function

$\theta^{(1)} = [-30 \quad 20 \quad 20]$



"NOR" function

$\theta^{(1)} = [10 \quad -20 \quad -20]$



$\theta^{(1)} = \begin{bmatrix} -30 & 20 & 20 \\ 10 & -20 & -20 \end{bmatrix}$

$\theta^{(2)} = [-10 \quad 20 \quad 20]$

"XNOR" function (combine AND and OR)

Multiclass Classification:

to classify class into multiple classes, we let hypothesis return a vector.

$$y^{(i)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{(4 classes)}$$

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ \vdots \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(3)} \\ a_1^{(3)} \\ \vdots \end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix} h_\theta(x)_1 \\ h_\theta(x)_2 \\ \vdots \end{bmatrix}$$

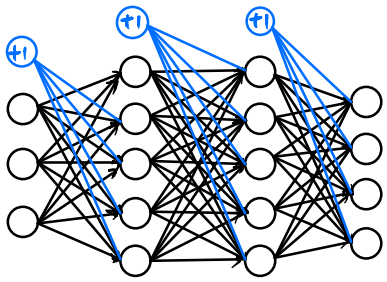
Backpropagation:

Cost function:

e.g. $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(m)}, y^{(m)})\}$, k = total number of output,

L = total number of layers in the network

S_l = number of units (not counting bias unit) in layer l



layer 1 layer 2 layer 3 layer 4

+1: bias nodes

$L = 4$, $k = 4$

$S_1 = 3$ $S_2 = 5$ $S_3 = 5$ $S_4 = S_L = 4$

Binary classification	Multi-class classification (k classes)
$y = 0, 1$	$y \in \mathbb{R}^k$, k output units
$S_L = 1$	$S_L = k$ $\quad k \geq 3$

Cost function:

Logistic Regression: $J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{1}{2m}\sum_{j=1}^{n} \theta_j^2$

Neural Networks: $J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{k} y_k^{(i)} \log (h_\theta(x^{(i)}))_k + (1-y_k^{(i)}) \log (1-h_\theta(x^{(i)}))_k\right]$

$$+ \frac{1}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_{l+1}} (\theta_{ji}^{(l)})^2$$

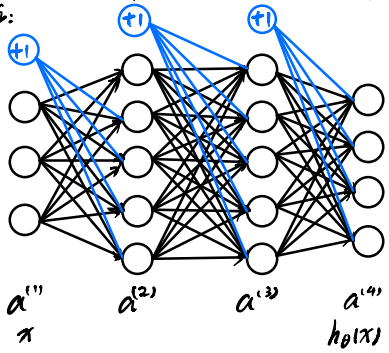
$h_\theta(x) \in \mathbb{R}^k$ $\quad (h_\theta(x))_i = i^{th}$ output

i, j start from 1: ignore the bias value

to compute $\min_\theta J(\theta)$, we need to compute $J(\theta)$ and $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta)$.

Gradient computation: Back propagation algorithm

e.g.:



$a^{(1)} = x$
$z^{(2)} = \theta^{(1)} a^{(1)}$
$a^{(2)} = g(z^{(2)})$
$z^{(3)} = \theta^{(2)} a^{(2)}$
$a^{(3)} = g(z^{(3)})$
$z^{(4)} = \theta^{(3)} a^{(3)}$
$a^{(4)} = g(z^{(4)}) = h_\theta(x)$

intuition: $\delta_j^{(l)}$ = "error" of node j in layer l

for the neural network above: $L = 4$

vectorize ↓

$\delta_j^{(4)} = a_j^{(4)} - y_j = (h_\theta(x))_j - y_i$

$\delta^{(4)} = a^{(4)} - y$

$\delta^{(3)} = (\theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)})$ $\quad g'(z^{(l)}) = a^{(l)} .* (1 - a^{(l)})$

$\delta^{(2)} = (\theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)})$ $\quad \delta^{(l)} = ((\theta^{(l)})^T \delta^{(l+1)}) .* a^{(l)} .* (1 - a^{(l)})$

no $\delta^{(1)}$

$\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) = a_j^{(l)} \delta_i^{(l+1)}$ (ignoring λ, $\lambda = 0$)

Give the training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$,

set $\Delta_{ij} = 0$ (for all i, j, l) accumulator for computing $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta)$

For $i = 1$ to m ← $(x^{(i)}, y^{(i)})$

set $a^{(1)} = x^{(i)}$

perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \cdots, L$

Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$

compute $\delta^{(L-1)}, \delta^{(L-2)}, \cdots, \delta^{(2)}$ no $\delta^{(1)}$

$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$?

vectorize → $\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T$

$$D_{ij}^{(l)} := \frac{1}{m}\Delta_{ij}^{(l)} + \lambda\theta_{ij}^{(l)} \quad \text{if } j \neq 0$$

$$D_{ij}^{(l)} := \frac{1}{m}\Delta_{ij}^{(l)} \quad \text{if } j = 0 \quad \text{(bias node)}$$

$$\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) = D_{ij}^{(l)}$$

backpropagation intuition:

recall the cost function for a neural network is:

$$J(\theta) = -\frac{1}{m}\sum_{t=1}^{m}\sum_{k=1}^{k}\left[y_k^{(t)}\log(h_\theta(x^{(t)}))_k + (1-y_k^{(t)})\log(1-h_\theta(x^{(t)})_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_{l+1}}(\theta_{ij}^{(l)})^2$$

if non-multiclass classification ($k=1$) and disregard regularization ($\lambda=0$)

$$cost(t) = y^{(t)}\log(h_\theta(x^{(t)})) + (1-y^{(t)})\log(1-h_\theta(x^{(t)}))$$

$$\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} cost(t)$$