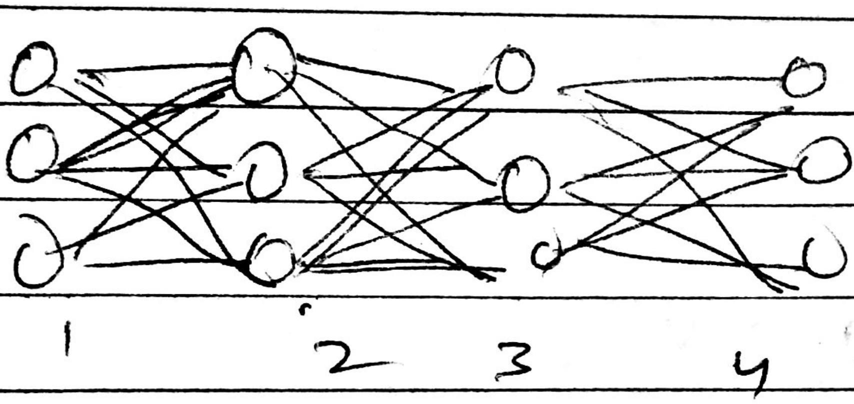
Neural Network (classification)



$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots (x^{(m)}, y^{(m)})\}$

L = total no. of layers in network ($=4$)

S_l = no. of units (excluding bias) in layer l.

$S_1 = 3$, $S_2 = 5$, $S_4 = 4$

Binary classification	Multi classes (K classes)
y = 0 or 1	
1 output units	K output units

* Dont regularize bias term

Logistic regression:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1- h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m} \underbrace{\sum_{j=1}^{n} \theta_j^2}_{\text{Regularization term}}$$

Neural Network:

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)}) \log(1-h_\Theta(x^{(i)}))_k\right] + \underbrace{\frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_l+1} (\Theta_{ji}^{(l)})^2}_{\text{Regularization for } \Theta_{ij}^{(K)}}$$

Cost function

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

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

→ ~~we~~ are finding theta to minimize $J(\theta)$

compute

$$\left[J(\theta) \quad \& \quad \frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) \right]$$

Remember [forward propagation.]

$a^{(1)} = x$

$z^{(2)} = \theta^{(1)} a^{(1)}$

$a^{(2)} = g(z^{(2)})$ .... [add $a_0^{(2)}$)

$z^{(3)} = \theta^{(2)} a^{(2)}$

$a^{(3)} = g(z^{(3)})$ ... [add $a_0^{(2)}$]

$z^{(4)} = \theta^{(3)} a^{(3)}$

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

↑ 4 layers only

for derivatives, we use back propagation

we calculate

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

$\therefore \delta_j^{(4)} = \boxed{a_j^{(4)}} - y_j$ actual value

$(h_\theta(x))_j$

$\delta^{(3)} = (\theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)})$

$\delta^{(2)} = (\theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)})$

g' = derivative

No $\delta^{(1)}$ term → it has input so you don't need to calculate (δ)

So now,

$$\boxed{\frac{\partial}{\partial \Theta^{(l)}} J(\Theta) = a_j^l \delta_j^{(l+1)}}$$

→ if we ignore the λ terms
i.e., if we ignore
the regularization term

What we do,

set $\Delta_{ij}^{(l)} = 0$ ... (not δ)
↑ (used to compute $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$)

for $i = 1$ to m
set $a^{(1)} = x^{(i)}$
Now, compute forward propagation
to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$
using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$

Back prop

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

vectorized $\boxed{\Delta^{l} = \Delta^{l} + \delta^{(l+1)} (a^{(l)})^T}$

end for

After for loop

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

or

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

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

So now, you can use these in some gradient descent or advanced optimization algorithm