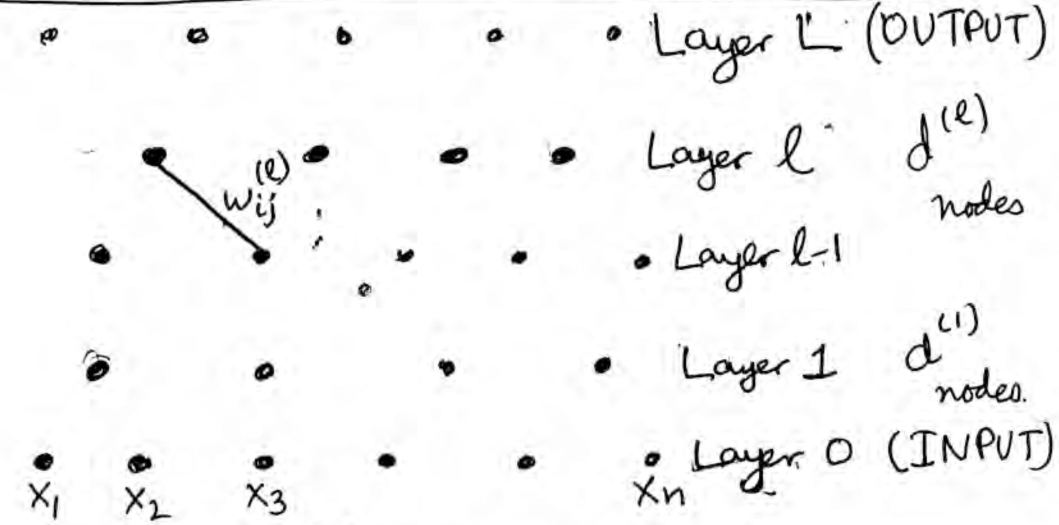


Output of neuron j in layer l

$$x_j^{(l)} = g\left(\sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)}\right) = g\left(s_j^{(l)}\right)$$

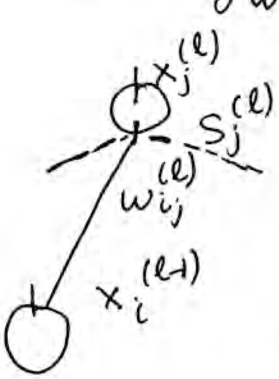
$\boxed{s_j^{(l)} \text{ is the weighted input to node } j \text{ in layer } l}$

$w_{ij}^{(l)}$ is weight from node i (in layer $l-1$) to node j (in layer l)

Error on example $(\underset{\sim}{x}_n, y_n)$ is $e(h(x_n), y_n) = e(w)$

We use h to denote the function computed by the neural network.

$$\nabla e(w) = \frac{\partial e(w)}{\partial w_{ij}^{(l)}}$$



$$\frac{\partial e(w)}{\partial w_{ij}^{(l)}} = \frac{\partial e(w)}{\partial s_j^{(l)}} \times \frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} \; x_i^{(l-1)}$$

$$\boxed{\text{Define } \delta_j^{(l)} = \frac{\partial e(w)}{\partial s_j^{(l)}}}$$

For the final layer $l = L$ and $j = 1$. For $e(h(x_n), y_n)$ squared error,

$$\frac{\partial e(w)}{\partial s_1^{(L)}} = \frac{\partial}{\partial s_1^{(L)}} \left\{ \frac{1}{2}\left(g(s_1^{(L)}) - y_n\right)^2 \right\}$$

BASE STEP

$$= \left(g(s_1^{(L)}) - y_n\right) \cdot g'(s_1^{(L)}) \quad \text{--- } (1)$$

This will change if we use a different loss function.

INDUCTION STEP

$$\delta_i^{(l-1)} = \frac{\partial e(w)}{\partial s_i^{(l-1)}}$$

$$= \sum_j \frac{\partial e(w)}{\partial s_j^{(l)}} \times \frac{\partial s_j}{\partial x_i^{(l-1)}} \times \frac{\partial x_i^{(l-1)}}{\partial s_i^{(l-1)}}$$

$$\delta_i^{(l-1)} = \sum_j \delta_j^{(l)} \, w_{ij}^{(l)} \, g'(s_i^{(l-1)})$$

$$= g'(s_i^{(l-1)}) \sum_j w_{ij}^{(l)} \delta_j^{(l)} \quad \text{--- (2)}$$

We have thus defined an iterative algorithm for computing δ for every node

- Compute it for nodes on output layer L using (1)
- Compute it for a layer $(l-1)$ using δ values at a layer (l) using (2)

Once this process has concluded, we can use

$$\frac{\partial e(w)}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} \, x_i^{(l-1)}$$

to compute the derivative with respect to any weight $w_{ij}^{(l)}$

These notes are based on Prof. Yaser Abu-Mostafa (Caltech)'s lectures on machine learning which are available on Youtube. The backpropagation algorithm is described in Lecture 10.