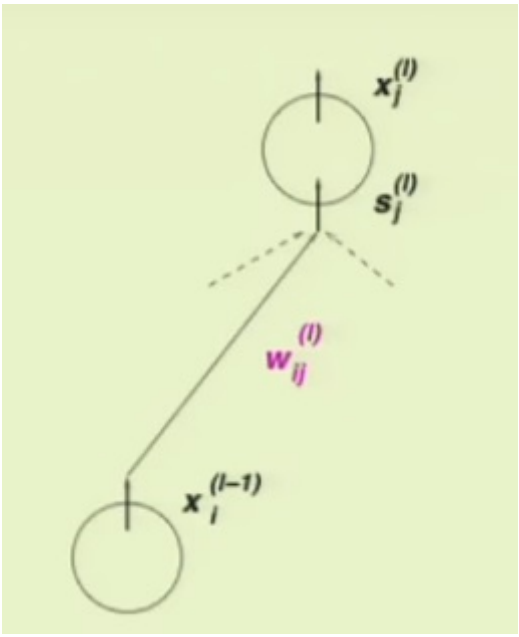

Figure 1:

Neural Network

Neural network is a combination of several nodes layer by layer such that we trigger a layer of nodes as input and those nodes trigger the nodes they are connected to (nodes in next layer). It has input layer where we give input signals and an output layer and multiple layers in between which are known as hidden layers.

For every signal there is a weight assigned to it which allows the nodes to decide which signal is to be taken from the previous node.

example: if a node is simulated by 2 different nodes, it will take the signal which has higher associated weight.

The question is, how are the weights determined? This what the neural network learn through a process called back propagation.So how does it go, we first give the input signal by assigning random weights and we obtain output, this output we obtain is the calculated one which should be equal to desired output but it is not, so we take the difference between the two to get the error and now that we have error we need to optimize the weights so as to obtain minimum error.

Lets see how this is done mathematically:

As you know neural network has different layer, we have non linearily associated with each layer. *Why non linearity?*, so as to make the function soft threshold i.e. function is diffrentiable infinitely. First layer is always the input layer and then we follow a rule of derivation from one layer to another till we reach the output h(x).

The non linearity used here is **tanh** given as:

$\tanh(x)=\theta(x)=\frac{e^s-e^-s}{e^s+e^-s}$ value of which lies in the range of [-1,1]

Now, I introduce to you the Notation of Neural Network, understand this in order to implement neural network.

amsmath

$$w_{ij}^{(l)} \begin{cases} 1 \le l \le L & \text{layers} \\ 0 \le i \le d^{(l-1)} & \text{inputs} \\ 1 \le j \le d^{(l)} & \text{outputs} \end{cases}$$

The parameters of neural network is called w (weights) which has 3 indices, layers are denoted by small l,input we feed is denoted by i and output we get is denoted by j. l represented as superscript in paranthesis of w and i and j are represented as subscript of w. The layer goes from 1 to L which is from first layer to the last layer.The output goes from 1 to d(l) as in we have output from node 1, node 2 and so on in a particular layer l. The input goes from 0 to d(l-1) i.e. the inputs are the previous layer output and 0 because we have constant associated with each layer.

Let's go to the function

we want to get output of layer **l** in terms of $x_j^{(l)}$

Therefore,$x_j^{(l)}$ can be written as:

$x_j^{(l)}$=$\theta(s_j^{(l)})$

where, $s_j^{(l)}$ is the signal of the output which passed through a soft threshold θ. This can also be written in terms of weights of the current layer and the inputs coming from the previous layer and then we sum up all the signals of the previous layers.

$x_j^{(l)}= \theta \sum_{i=0}^{i=d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)}$

For each layer we get an output which becomes the input of next layer and finally we arrive at a single output $x_j^{(L)}$ we term as h(x).

Optimization of weights:

In order to obtain optimised weight we will be doing gradient descent. We consider all the weights w=$w_{ij}^{(l)}$ to determine h(x) and also we get an error measure as e(h(x_n),y_n). Where, h(x_n) is the output we calculated and y_n is the desired output.

Now error measure can be written as:

e(h(x_n),y_n)=e(w)

where, e(w) is the function of all weights

To implement gradient descent, what we need to do is to take the gradient of e(w) which is nothing but partial the error term by partial the weights, given as:

$\nabla e(w) = \frac{\partial e(w)}{\partial w_{ij}^{(l)}}$ for all i,j,l

Computing $\frac{\partial e(w)}{\partial w_{ij}^{(l)}}$ as:

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

So, here we are applying nothing but the chain rule taking signal as intermediate as $w_{ij}^{(l)}$ only affects the signal $s_j^{(l)}$ therefore, we need not sum up anything here and we can proceed with the chain rule.

$\frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}}$ this quantiy is a simple one and we easily find out how the signal is changing with the weight but this quantity $\frac{\partial e(w)}{\partial s_j^{(l)}}$ is computed reccursively.

so what we have now is

$\frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}} = x_i^{(l-1)}$

And we need

$\frac{\partial e(w)}{\partial s_j^{(l)}}$ which we denote as $\delta_j^{(l)}$

Therefore, change in weight is proportional to product of these two guy $x_i^{(l-1)}$ and $\delta_j^{(l)}$

Now we calculate delta for the output layer as we are doing back propagation here. Putting it this way, if we know the δ of last , we can calculate the delta of previous layer.

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

This is the delta calculated for any layer, now for final layer l=L and j=1, substituting this we get $\delta_1^{(L)} = \frac{\partial e(w)}{\partial s_1^{(L)}}$

Lets go to the equation, e(w)=e(h(x_n),y_n)

where, e(w) is the error measure between y_n and h(x_n), h(x_n) here is nothing but the output of the last layer i.e. $x_1^{(L)}$

The above equation can be written as

e(w)=e($x_1^{(L)}$,y_n)

$x_1^{(L)}$ is nothing but $\theta(s_1^{(L)})$ and we are almost done as the signal is what we are differentiating with respect to.

To proceed further we need to know the differentiation of θ i.e. tanh which is given by

$\theta'(x)$=(1- $\theta^2(x)$) for tanh

Using this we calculate δ for the final layer

The next thing is to back propagate δ to get the previous layer δ i.e. we want to compute,

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

again we apply the chain rule as:

$\delta_j^{(l-1)} = \sum_{j=1}^{i=d^{(l)}} \frac{\partial e(w)}{\partial s_j^{(l)}} \times \frac{\partial s_j^{(l)}}{\partial x_i^{(l-1)}} \times \frac{\partial x_i^{(l-1)}}{\partial s_i^{(l-1)}}$

Here we need to sum these as $s_i^{(l-1)}$ is affected by all weights calculated while going backward.

The above equation can be written as: $\delta_j^{(l-1)} = \sum_{j=1}^{i=d^{(l)}} \delta_j^{(l)} \times w_{ij}^{(l)} \times \theta' s_i^{(l-1)}$

As $\theta' s_i^{(l-1)}$ does not have j, we bring that out and our final equation becomes,

$\delta_j^{(l-1)} = \theta' s_i^{(l-1)} \sum_{j=1}^{i=d^{(l)}} w_{ij}^{(l)} \times \delta_j^{(l)}$

Thats interesting,we can calculate all deltas using the above equation

To brief:

- initialize the weight $w_{ij}^{(l)}$ at random
- compute all $x_j^{(l)}$ as we move forward
- compute $\delta_1^{(L)}$
- compute all $\delta_j^{(l)}$ as we move backward
- Update the weights $w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta x_i^{(l-1)} \delta_j^{(l)}$
- Iterate the next step and return the final weight $w_{ij}^{(l)}$