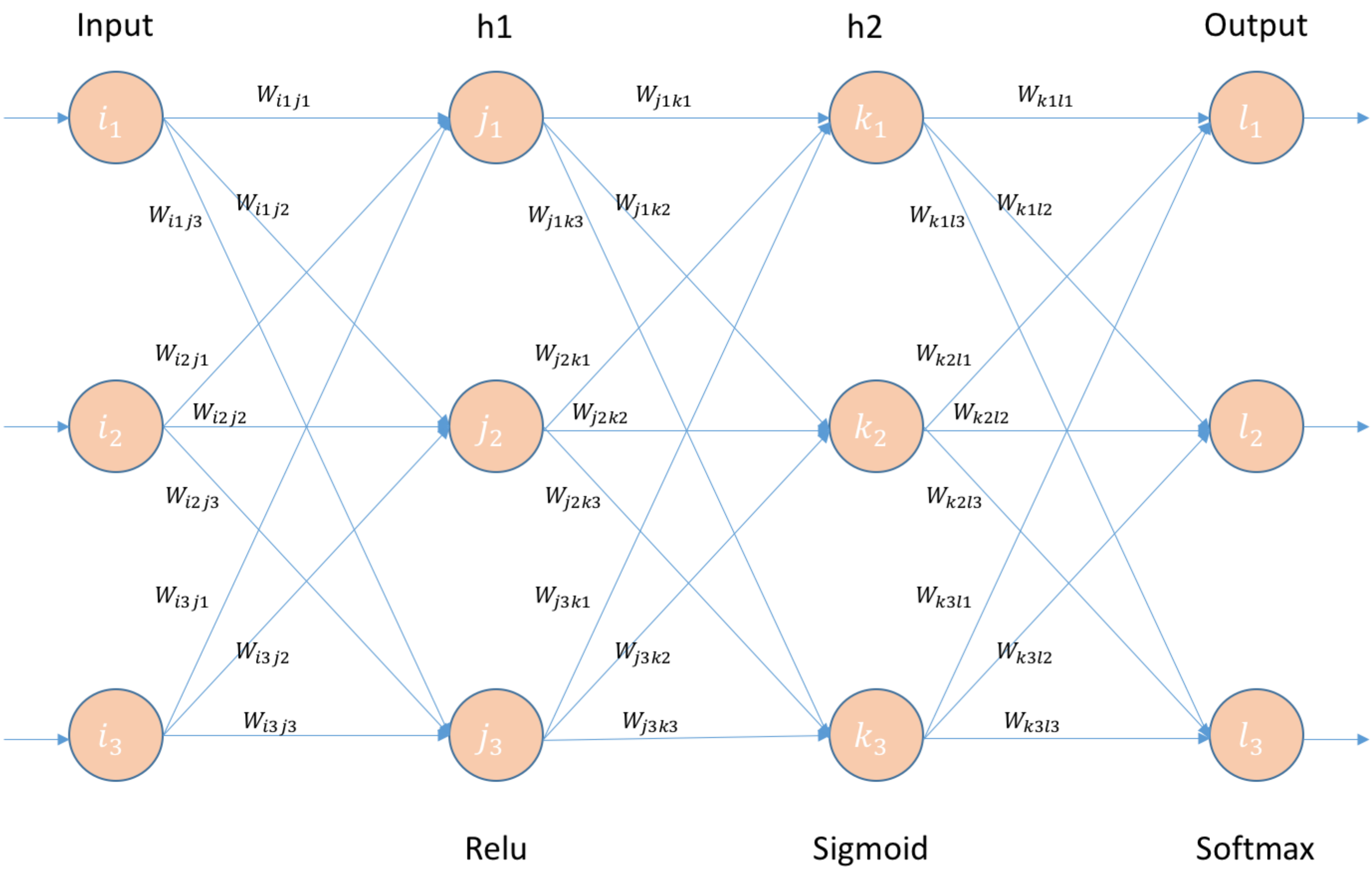
Back-Propagation is very simple. Who made it Complicated ?

Learning Outcome: You will be able to build your own Neural Network on a Paper.

Prakash Jay

Apr 20, 2017 · 9 min read



Feed Forward Neural Network

Almost 6 months back when I first wanted to try my hands on Neural network, I scratched my head for a long time on how Back-Propagation works. When I talk to peers around my circle, I see a lot of people facing this problem. Most people consider it as a black-box and use libraries like Keras, TensorFlow and PyTorch which provide automatic differentiation. Though it is not necessary to write your own code on how to compute gradients and backprop errors, having knowledge on it helps you in understanding a few concepts like Vanishing Gradients, Saturation of Neurons and reasons for random initialization of weights

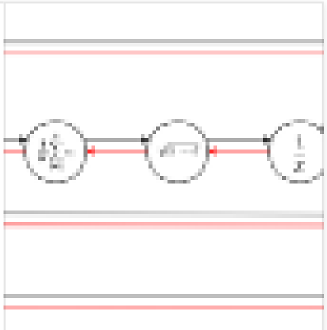
More about why is it important to Understand?

Andrej Karapathy wrote a blog-post on it and I found it useful.

Yes you should understand backprop

When we offered CS231n (Deep Learning class) at Stanford, we intentionally designed the programming assignments to…

medium.com



Approach

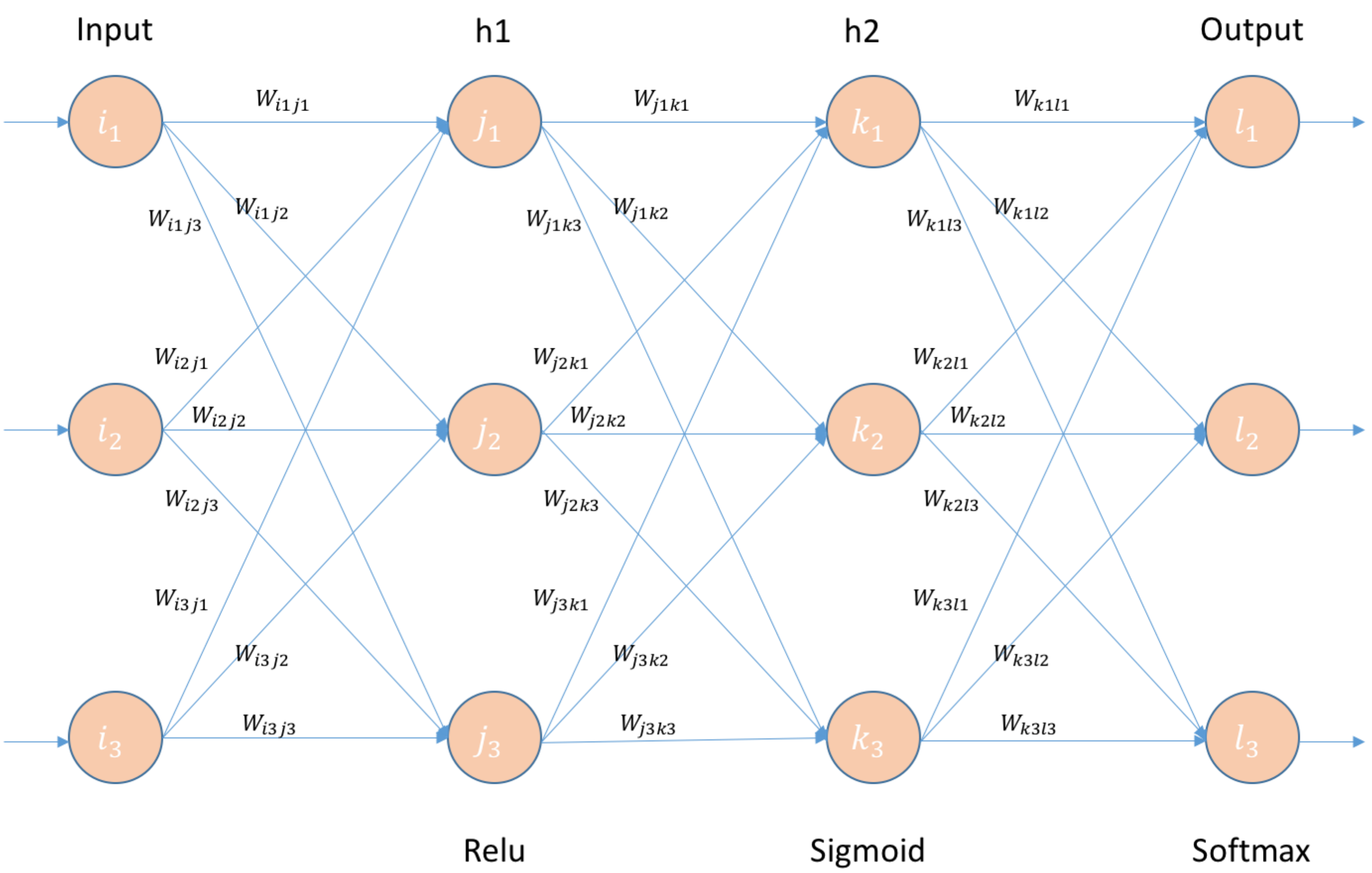
- Build a small neural network as defined in the architecture below.
- Initialize the weights and bias randomly.
- Fix the input and output.
- Forward pass the inputs. calculate the cost.
- compute the gradients and errors.
- Backprop and adjust the weights and bias accordingly

Architecture:

- Build a Feed Forward neural network with 2 hidden layers. All the layers will have 3 Neurons each.
- 1st and 2nd hidden layer will have Relu and sigmoid respectively as activation functions. Final layer will have Softmax.
- Error is calculated using cross-entropy.

Initializing the network

I have taken inputs, weights and bias randomly



$$Input = \begin{bmatrix} 0.1 & 0.2 & 0.7 \end{bmatrix}$$

$$W_{ij} = \begin{bmatrix} W_{i1j1} & W_{i1j2} & W_{i1j3} \\ W_{i2j1} & W_{i2j2} & W_{i2j3} \\ W_{i3j1} & W_{i3j2} & W_{i3j3} \end{bmatrix} = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.3 & 0.2 & 0.7 \\ 0.4 & 0.3 & 0.9 \end{bmatrix}$$

$$W_{jk} = \begin{bmatrix} W_{j1k1} & W_{j1k2} & W_{j1k3} \\ W_{j2k1} & W_{j2k2} & W_{j2k3} \\ W_{j3k1} & W_{j3k2} & W_{j3k3} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.3 & 0.5 \\ 0.3 & 0.5 & 0.7 \\ 0.6 & 0.4 & 0.8 \end{bmatrix}$$

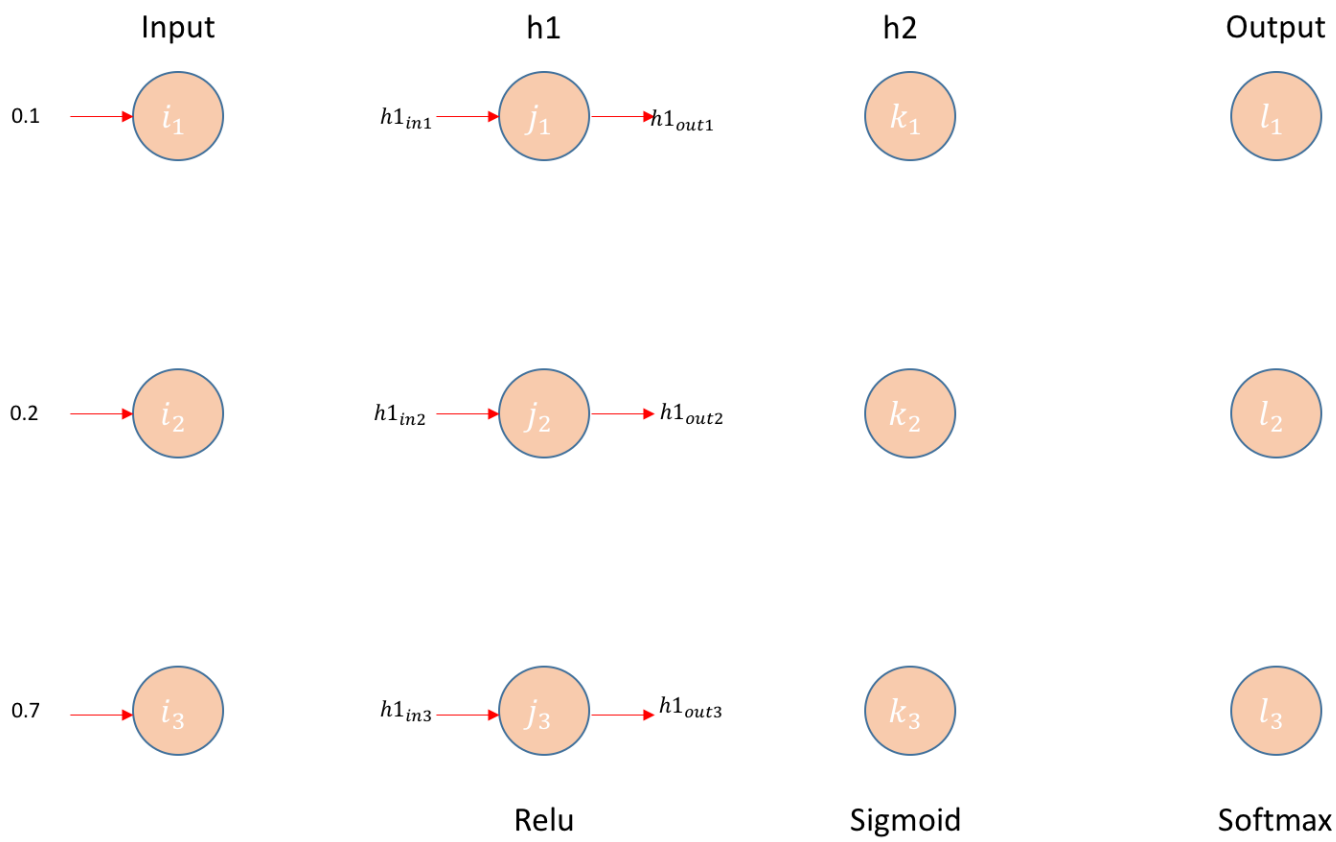
$$\begin{bmatrix} W_{k1l1} & W_{k1l2} & W_{k1l3} \end{bmatrix} \quad \begin{bmatrix} 0.1 & 0.4 & 0.8 \end{bmatrix}$$

$$W_{kl} = \begin{bmatrix} W_{k2l1} & W_{k2l2} & W_{k2l3} \\ W_{k3l1} & W_{k3l2} & W_{k3l3} \end{bmatrix} = \begin{bmatrix} 0.3 & 0.7 & 0.2 \\ 0.5 & 0.2 & 0.9 \end{bmatrix}$$

$$Output = \begin{bmatrix} 1.0 & 0.0 & 0.0 \end{bmatrix}$$

Initializing the network

Layer-1



Neural Network Layer-1

Matrix Operation:

$$\begin{bmatrix} i_1 & i_2 & i_3 \end{bmatrix} \times \begin{bmatrix} W_{i1j1} & W_{i1j2} & W_{i1j3} \\ W_{i2j1} & W_{i2j2} & W_{i2j3} \\ W_{i3j1} & W_{i3j2} & W_{i3j3} \end{bmatrix} + \begin{bmatrix} b_{j1} & b_{j2} & b_{j3} \end{bmatrix} = \begin{bmatrix} h1_{in1} & h1_{in2} & h1_{in3} \end{bmatrix}$$

Layer-1 Matrix Operation

Relu operation:

$$relu = max(0, x)$$

$$\begin{bmatrix} h1_{out1} & h1_{out2} & h1_{out3} \end{bmatrix} = \begin{bmatrix} max(0, h1_{in1}) & max(0, h1_{in2}) & max(0, h1_{in3}) \end{bmatrix}$$

Layer-1 Relu Operation

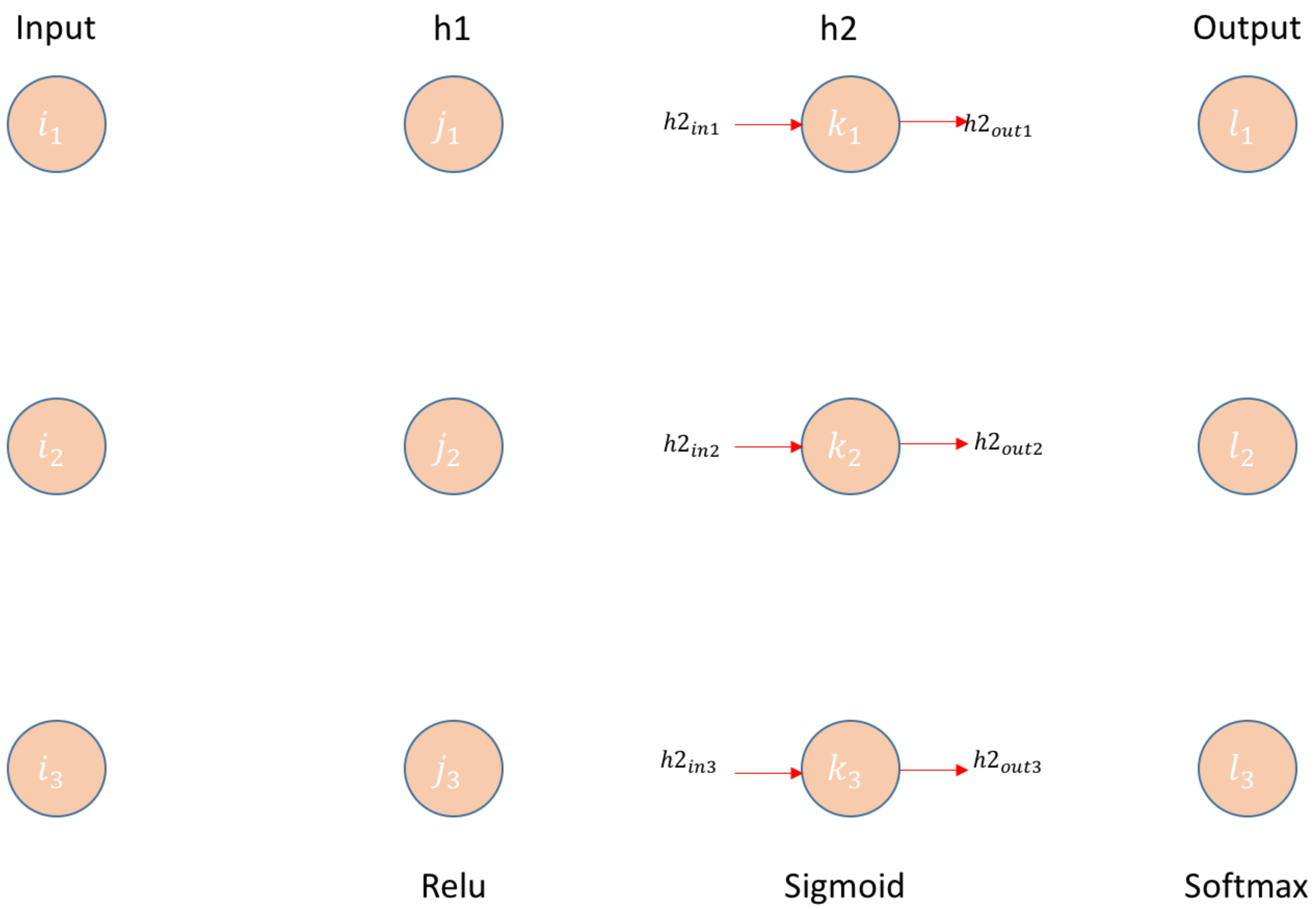
Example:

$$\begin{bmatrix} 0.1 & 0.2 & 0.7 \end{bmatrix} \times \begin{bmatrix} 0.1 & 0.4 & 0.3 \\ 0.3 & 0.7 & 0.7 \\ 0.5 & 0.2 & 0.9 \end{bmatrix} + \begin{bmatrix} 1.0 & 1.0 & 1.0 \end{bmatrix} = \begin{bmatrix} 1.35 & 1.27 & 1.8 \end{bmatrix}$$

$$\begin{bmatrix} h1_{out1} & h1_{out2} & h1_{out3} \end{bmatrix} = \begin{bmatrix} 1.35 & 1.27 & 1.8 \end{bmatrix}$$

Layer-1 Example

Layer-2



Layer-2 Neural Network

Matrix operation:

$$\begin{bmatrix} h1_{out1} & h1_{out2} & h1_{out3} \end{bmatrix} \times \begin{bmatrix} W_{j1k1} & W_{j1k2} & W_{j1k3} \\ W_{j2k1} & W_{j2k2} & W_{j2k3} \\ W_{j3k1} & W_{j3k2} & W_{j3k3} \end{bmatrix} + \begin{bmatrix} b_{k1} & b_{k2} & b_{k3} \end{bmatrix} = \begin{bmatrix} h2_{in1} & h2_{in1} & h2_{in3} \end{bmatrix}$$

Layer-2 Matrix Operation

Sigmoid operation:

$$Sigmoid = 1/(1 + e^{-x})$$

$$\begin{bmatrix} h2_{out1} & h2_{out2} & h2_{out3} \end{bmatrix} = \begin{bmatrix} 1/(1 + e^{-h2_{in1}}) & 1/(1 + e^{-h2_{in2}}) & 1/(1 + e^{-h2_{in3}}) \end{bmatrix}$$

Sigmoid Operation

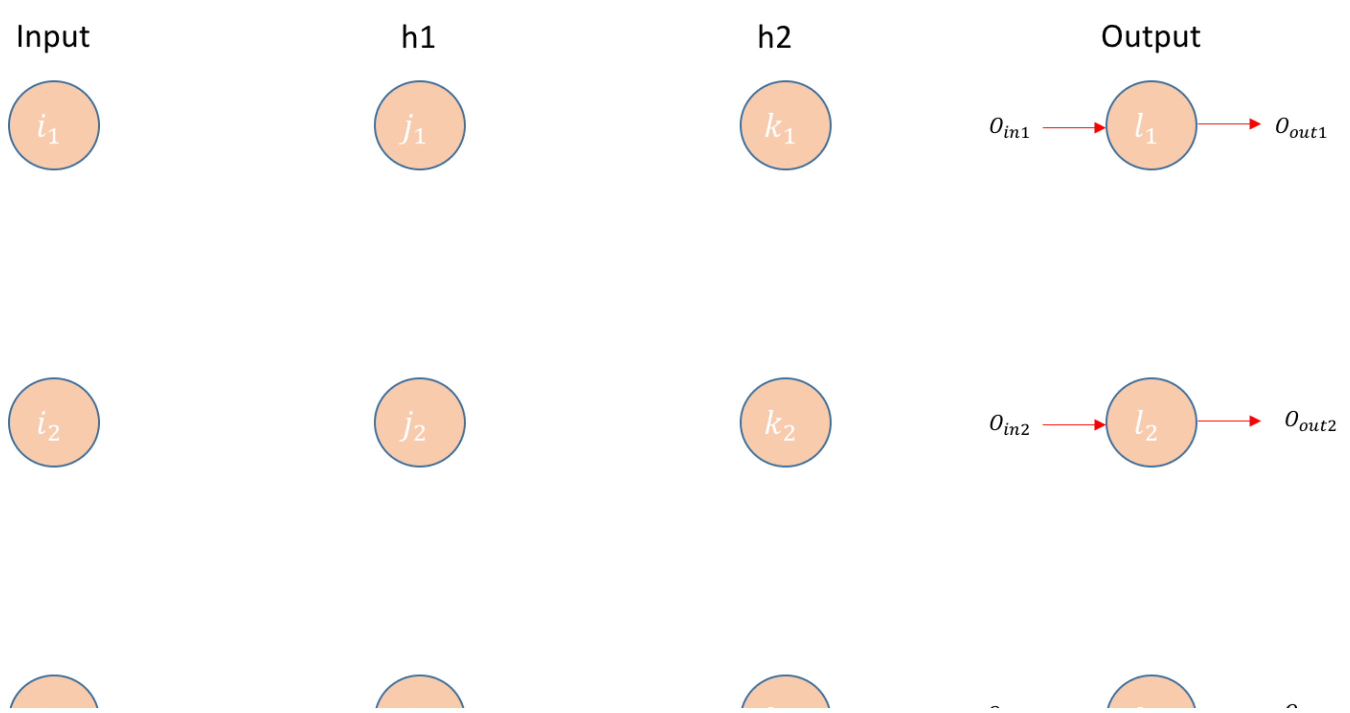
Example:

$$\begin{bmatrix} 1.35 & 1.27 & 1.8 \end{bmatrix} \times \begin{bmatrix} 0.2 & 0.3 & 0.5 \\ 0.3 & 0.5 & 0.7 \\ 0.6 & 0.4 & 0.8 \end{bmatrix} + \begin{bmatrix} 1.0 & 1.0 & 1.0 \end{bmatrix} = \begin{bmatrix} 2.73 & 2.76 & 4.001 \end{bmatrix}$$

$$\begin{bmatrix} h2_{out1} & h2_{out2} & h2_{out3} \end{bmatrix} = \begin{bmatrix} 0.938 & 0.94 & 0.98 \end{bmatrix}$$

Layer-2 Example

Layer-3



Layer-3 Neural Network

Matrix operation:

$$\begin{bmatrix} h2_{out1} & h2_{out2} & h2_{out3} \end{bmatrix} \times \begin{bmatrix} W_{k1l1} & W_{k1l2} & W_{k1l3} \\ W_{k2l1} & W_{k2l2} & W_{k2l3} \\ W_{k3l1} & W_{k3l2} & W_{k3l3} \end{bmatrix} + \begin{bmatrix} b_{l1} & b_{l2} & b_{l3} \end{bmatrix} = \begin{bmatrix} O_{in1} & O_{in2} & O_{in3} \end{bmatrix}$$

Layer-3 Matrix Operation

Softmax operation:

$$Softmax = e^{l_{ina}} / (\sum_{a=1}^{3} e^{O_{ina}})$$

$$\begin{bmatrix} O_{out1} & O_{out2} & O_{out3} \end{bmatrix} = \begin{bmatrix} e^{O_{in1}}/(\sum_{a=1}^{3} e^{O_{ina}}) & e^{O_{in2}}/(\sum_{a=1}^{3} e^{O_{ina}}) & e^{O_{in3}}/(\sum_{a=1}^{3} e^{O_{ina}}) \end{bmatrix}$$

Softmax formula

Example:

$$\begin{bmatrix} 0.938 & 0.94 & 0.98 \end{bmatrix} \times \begin{bmatrix} 0.1 & 0.4 & 0.8 \\ 0.3 & 0.7 & 0.2 \\ 0.5 & 0.2 & 0.9 \end{bmatrix} + \begin{bmatrix} 1.0 & 1.0 & 1.0 \end{bmatrix} = \begin{bmatrix} 1.8658 & 2.2292 & 2.8204 \end{bmatrix}$$

$$\begin{bmatrix} O_{out1} & O_{out2} & O_{out3} \end{bmatrix} = \begin{bmatrix} 0.2698 & 0.3223 & 0.4078 \end{bmatrix}$$

Layer-3 Output Example

Edit1: As Jmuth pointed out in the comments, the output from softmax would be [0.19858, 0.28559, 0.51583] instead of [0.26980, 0.32235, 0.40784]. I have done a/sum(a) while the correct answer would be exp(a)/sum(exp(a)) . Please adjust your calculations from here on using these values. Thank you.

Analysis:

- The Actual Output should be [1.0, 0.0, 0.0] but we got [0.2698, 0.3223, 0.4078].
- To calculate error lets use cross-entropy

Error:

Cross-Entropy:

$$crossentropy = -(1/n)(\sum_{i=1}^{3}(y_i \times \log(O_{outi})) + ((1 - y_i) \times \log((1 - O_{outi}))))$$

Cross-Entropy Formula

Example:

$$Error = -((1 * \log(0.2698) + 0 + 0 * \log(0.3223) + 1 * \log(1 - 0.3223) + 0 * \log(0.4078) + 1 * \log(1 - 0.4078))$$
$$Error = -\log(0.2698) - \log(0.6777) - \log(0.5922)$$
$$Error = +0.569858 + 0.16886 + 0.22753 = 0.985$$

Cross-Entropy calculation

We are done with forward pass. Now let us see backward pass

Important Derivatives:

Sigmoid:

$$Sigmoid = 1/(1 + \mathrm{e}^{-x})$$
$$\frac{\partial(1/(1 + \mathrm{e}^{-x}))}{\partial x} = 1/(1 + \mathrm{e}^{-x}) \times (1 - 1/(1 + \mathrm{e}^{-x}))$$
$$\frac{\partial Sigmoid}{\partial x} = Sigmoid \times (1 - Sigmoid)$$

Derivative of Sigmoid

Relu:

$$relu = max(0, x)$$

$$if x > 0, \frac{\partial(relu)}{\partial x} = 1$$

$$Otherwise, \frac{\partial(relu)}{\partial x} = 0$$

Derivative of Relu

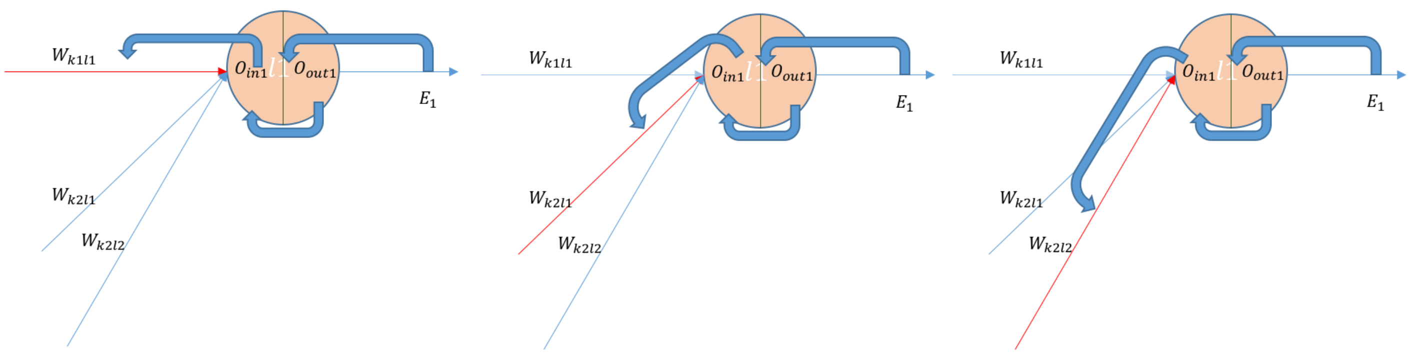
Softmax:

$$Softmax = e^{x_a}/(\sum_{a=1}^{n} e^{x_a}) = e^{x_1}/(e^{x_1} + e^{x_2} + e^{x_3})$$

$$\frac{\partial(Softmax)}{\partial x_1} = (e^{x_1} \times (e^{x_2} + e^{x_3}))/(e^{x_1} + e^{x_2} + e^{x_3})^2$$

Derivative of Softmax

BackPropagating the error — (Hidden Layer2 — Output Layer) Weights



Backpropagating Layer-3 weights

Let us calculate a few derivatives upfront so these become handy and we can reuse them whenever necessary. Here are we are using only one example (batch_size=1), if there are more examples, We just need to average everything.

$$\frac{\partial E_1}{\partial O_{out1}} = \frac{\partial(-1 * ((y_1 * \log(O_{out1}) + (1 - y_1) * \log((1 - O_{out1}))}{\partial O_{out1}}$$

$$\frac{\partial E_1}{\partial O_{out1}} = -1 * ((y_1 * (1/O_{out1}) + (1 - y_1) * (1/(1 - O_{out1}))$$

Example: Derivative of Cross-Entropy

By symmetry we can calculate other derivatives also

$$\begin{bmatrix} \frac{\partial E_1}{\partial O_{out1}} \\ \frac{\partial E_2}{\partial O_{out2}} \\ \frac{\partial E_3}{\partial O_{out3}} \end{bmatrix} = \begin{bmatrix} -1 * ((y_1 * (1/O_{out1}) + (1 - y_1) * (1/(1 - O_{out1})) \\ -1 * ((y_2 * (1/O_{out2}) + (1 - y_2) * (1/(1 - O_{out2})) \\ -1 * ((y_3 * (1/O_{out3}) + (1 - y_3) * (1/(1 - O_{out3})) \end{bmatrix}$$

Matrix of cross-entropy derivatives wrt output

In our example,

$$\begin{bmatrix} \frac{\partial E_1}{\partial O_{out1}} \\ \frac{\partial E_2}{\partial O_{out2}} \\ \frac{\partial E_3}{\partial O_{out3}} \end{bmatrix} = \begin{bmatrix} -3.70644 \\ -1.4755 \\ -1.6886 \end{bmatrix}$$

values of derivative of cross-entropy wrt output.

Next let us calculate the derivative of each output with respect to their input.

$$\frac{\partial O_{out1}}{\partial O_{in1}} = \frac{\partial(e^{O_{in1}}/(e^{O_{in1}} + e^{O_{in2}} + e^{O_{in3}}))}{\partial O_{in1}}$$

$$\frac{\partial O_{out1}}{\partial O_{in1}} = (e^{O_{in1}} \times (e^{O_{in2}} + e^{O_{in3}}))/(e^{O_{in1}} + e^{O_{in2}} + e^{O_{in3}})^2$$

Example: Derivative of softmax wrt output layer input

By symmetry we can calculate other derivatives also

$$\begin{bmatrix} \frac{\partial O_{out1}}{\partial O_{in1}} \\ \frac{\partial O_{out2}}{\partial O_{in2}} \\ \frac{\partial O_{out3}}{\partial O_{in3}} \end{bmatrix} = \begin{bmatrix} (e^{O_{in1}} \times (e^{O_{in2}} + e^{O_{in3}}))/(e^{O_{in1}} + e^{O_{in2}} + e^{O_{in3}})^2 \\ (e^{O_{in2}} \times (e^{O_{in1}} + e^{O_{in3}}))/(e^{O_{in1}} + e^{O_{in2}} + e^{O_{in3}})^2 \\ (e^{O_{in3}} \times (e^{O_{in1}} + e^{O_{in2}}))/(e^{O_{in1}} + e^{O_{in2}} + e^{O_{in3}})^2 \end{bmatrix}$$

Matrix of Derivative of softmax wrt output layer input.

In our example,

$$\begin{bmatrix} \frac{\partial O_{out1}}{\partial O_{in1}} \\ \frac{\partial O_{out2}}{\partial O_{in2}} \\ \frac{\partial O_{out3}}{\partial O_{in3}} \end{bmatrix} = \begin{bmatrix} 0.15911 \\ 0.2040 \\ 0.3685 \end{bmatrix}$$

values of derivative of softmax wrt output layer input .

For each input to neuron let us calculate the derivative with respect to each weight. Now let us look at the final derivative

$$\frac{\partial O_{in1}}{\partial W_{k1l1}} = \frac{\partial((h2_{out1} * W_{j1k1}) + (h2_{out2} * Wj2k1) + (h2_{out3} * Wj3k1) + b_{l1})}{\partial W_{k1l1}}$$

$$\frac{\partial O_{in1}}{\partial W_{k1l1}} = h2_{out1}$$

Example: Derivative of input to output layer wrt weight

By symmetry we can calculate other derivatives also

$$\begin{bmatrix} \frac{\partial O_{in1}}{\partial W_{k1l1}} \end{bmatrix} \quad \begin{bmatrix} h2_{out1} \end{bmatrix} \quad \begin{bmatrix} 0.938 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial O_{in1}}{\partial W_{k2l1}} \\ \frac{\partial O_{in1}}{\partial W_{k3l1}} \end{bmatrix} = \begin{bmatrix} h2_{out2} \\ h2_{out3} \end{bmatrix} = \begin{bmatrix} 0.94 \\ 0.98 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial O_{in2}}{\partial W_{k1l2}} \\ \frac{\partial O_{in2}}{\partial W_{k2l2}} \\ \frac{\partial O_{in2}}{\partial W_{k3l2}} \end{bmatrix} = \begin{bmatrix} h2_{out1} \\ h2_{out2} \\ h2_{out3} \end{bmatrix} = \begin{bmatrix} 0.938 \\ 0.94 \\ 0.98 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial O_{in3}}{\partial W_{k1l3}} \\ \frac{\partial O_{in3}}{\partial W_{k2l3}} \\ \frac{\partial O_{in3}}{\partial W_{k3l3}} \end{bmatrix} = \begin{bmatrix} h2_{out1} \\ h2_{out2} \\ h2_{out3} \end{bmatrix} = \begin{bmatrix} 0.938 \\ 0.94 \\ 0.98 \end{bmatrix}$$

values of derivative of input to output layer wrt weights.

Finally Let us calculate the change in

$$W_{k1l1}$$

Weight from k1 to l1 neuron

Which will be simply

$$\frac{\partial E_1}{\partial W_{k1l1}}$$

Derivative of error wrt weight

Using Chain Rule:

$$\frac{\partial E_1}{\partial W_{k1l1}} = \frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial W_{k1l1}}$$

Chain rule breakdown of Error derivative

By symmetry:

$$\delta W_{kl} = \begin{bmatrix} \frac{\partial E_1}{\partial W_{k1l1}} & \frac{\partial E_2}{\partial W_{k1l2}} & \frac{\partial E_3}{\partial W_{k1l3}} \\ \frac{\partial E_1}{\partial W_{k2l1}} & \frac{\partial E_2}{\partial W_{k2l2}} & \frac{\partial E_3}{\partial W_{k2l3}} \\ \frac{\partial E_1}{\partial W_{k3l1}} & \frac{\partial E_2}{\partial W_{k3l2}} & \frac{\partial E_3}{\partial W_{k3l3}} \end{bmatrix} = \begin{bmatrix} \frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial W_{k1l1}} & \frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial W_{k1l2}} & \frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial W_{k1l3}} \\ \frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial W_{k2l1}} & \frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial W_{k2l2}} & \frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial W_{k2l3}} \\ \frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial W_{k3l1}} & \frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial W_{k3l2}} & \frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial W_{k3l3}} \end{bmatrix}$$

Matrix Form of all derivatives in layer-3

All of the above values are calculated before. We just need to substitute the results.

$$\delta W_{kl} = \begin{bmatrix} \delta W_{k1l1} & \delta W_{k1l2} & \delta W_{k1l3} \\ \delta W_{k2l1} & \delta W_{k2l2} & \delta W_{k2l3} \\ \delta W_{k3l1} & \delta W_{k3l2} & \delta W_{k3l3} \end{bmatrix} = \begin{bmatrix} -3.7064 * 0.1591 * 0.938 & -0.301 * 0.204 * 0.938 & -1.6886 * 0.3685 * 0.938 \\ -3.7064 * 0.1591 * 0.94 & -0.301 * 0.204 * 0.94 & -1.6886 * 0.3685 * 0.94 \\ -3.7064 * 0.1591 * 0.98 & -0.301 * 0.204 * 0.98 & -1.6886 * 0.3685 * 0.98 \end{bmatrix}$$
$$= \begin{bmatrix} -0.5531 & -0.0576 & -0.5836 \\ -0.554347 & -0.0577 & -0.5849 \\ -0.577937 & -0.06017 & -0.6098 \end{bmatrix}$$

Example: Calculation of all the values

Considering a learning rate of 0.01 we get our final weight matrix as

$$\acute{W_{kl}} = \begin{bmatrix} W_{k1l1} - (lr * \delta W_{k1l1}) & W_{k1l2} - (lr * \delta W_{k1l2}) & W_{k1l3} - (lr * \delta W_{k1l3}) \\ W_{k2l1} - (lr * \delta W_{k2l1}) & W_{k2l2} - (lr * \delta W_{k2l2}) & W_{k2l3} - (lr * \delta W_{k2l3}) \\ W_{k3l1} - (lr * \delta W_{k3l1}) & W_{k3l2} - (lr * \delta W_{k3l2}) & W_{k3l3} - (lr * \delta W_{k3l3}) \end{bmatrix}$$

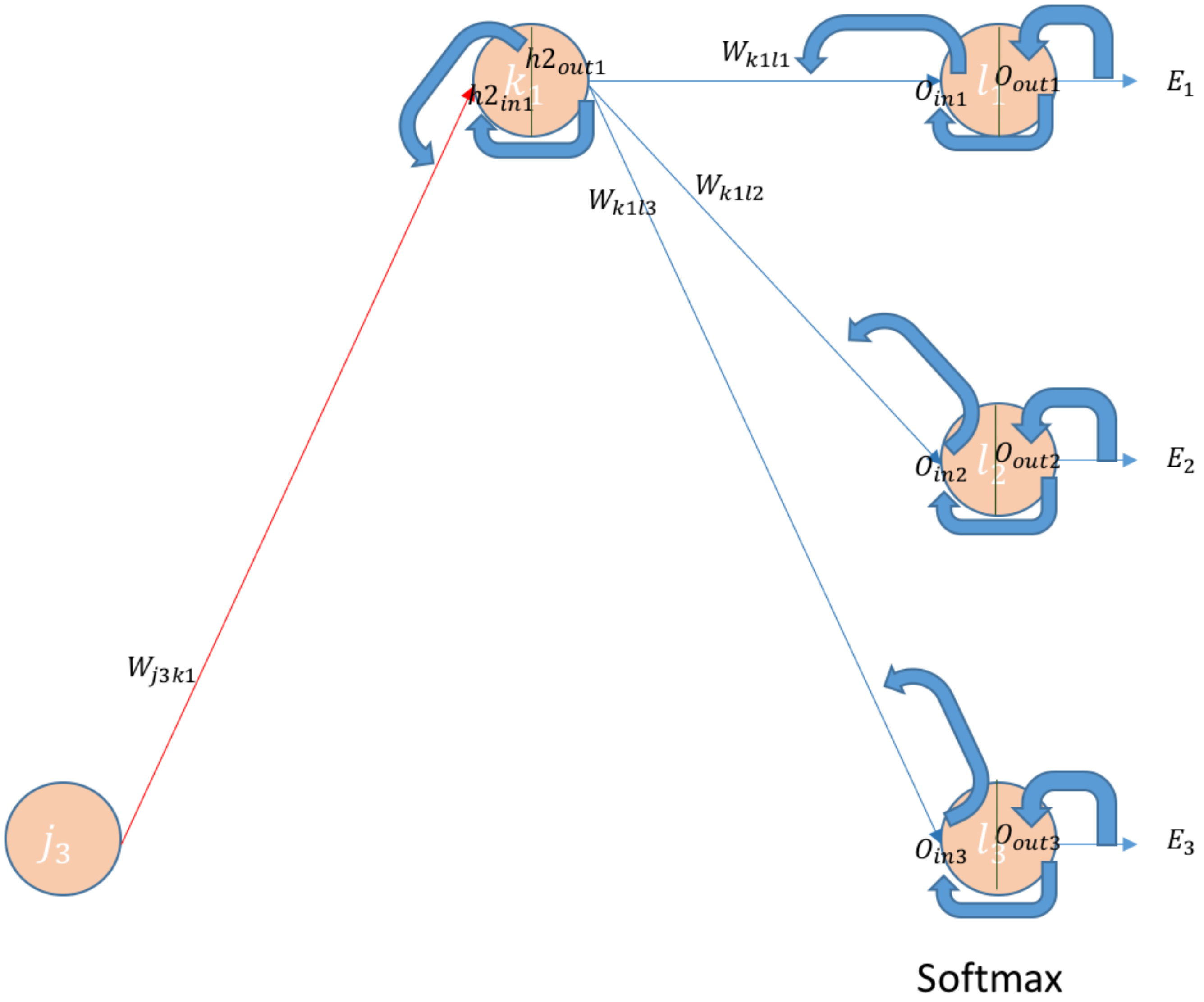
$$\acute{W_{kl}} = \begin{bmatrix} 0.1 - (0.01 * -0.5531) & 0.4 - (0.01 * -0.0576) & 0.8 - (0.01 * -0.5836) \\ 0.3 - (0.01 * -0.554347) & 0.7 - (0.01 * -0.0577) & 0.2 - (0.01 * -0.5849) \\ 0.5 - (0.01 * -0.577937) & 0.2 - (0.01 - 0.06017) & 0.9 - (0.01 * -0.6098) \end{bmatrix}$$

$$\acute{W_{kl}} = \begin{bmatrix} 0.105531 & 0.400576 & 0.805836 \\ 0.30055 & 0.700577 & 0.2005849 \\ 0.5005779 & 0.2006017 & 0.9006098 \end{bmatrix}$$

Modified weights of kl neurons after backprop

So, We have calculated new weight matrix for W_{kl}. Now let us move to the next layer:

BackPropagating the error — (Hidden Layer1 — Hidden Layer 2) Weights



Backpropagating errors to 2nd layer

Let us calculate a few handy derivatives before we actually calculate the error derivatives wrt weights in this layer.

$$\frac{\partial h2_{out1}}{\partial h2_{in1}} = \frac{\partial Sigmoid(h2_{in1})}{\partial h2_{in1}}$$

$$\frac{\partial h2_{out1}}{\partial h2_{in1}} = Sigmoid(h2_{in1}) * (1 - Sigmoid(h2_{in1}))$$

$$\begin{bmatrix} \frac{\partial h2_{out1}}{\partial h2_{in1}} \\ \frac{\partial h2_{out2}}{\partial h2_{in2}} \end{bmatrix} = \begin{bmatrix} Sigmoid(h2_{in1}) * (1 - Sigmoid(h2_{in1})) \\ Sigmoid(h2_{in2}) * (1 - Sigmoid(h2_{in2})) \end{bmatrix}$$

$$\begin{bmatrix} \partial h2_{in2} \\ \frac{\partial h2_{out3}}{\partial h2_{in3}} \end{bmatrix} \quad \begin{bmatrix} Sigmoid(h2_{in1}) * (1 - Sigmoid(h2_{in1})) \end{bmatrix}$$

Example: Derivative of sigmoid output wrt layer 2 input

In our example:

$$\begin{bmatrix} \frac{\partial h2_{out1}}{\partial h2_{in1}} \\ \frac{\partial h2_{out2}}{\partial h2_{in2}} \\ \frac{\partial h2_{out3}}{\partial h2_{in3}} \end{bmatrix} = \begin{bmatrix} Sigmoid(2.73) * (1 - Sigmoid(2.73)) \\ Sigmoid(2.76) * (1 - Sigmoid(2.76)) \\ Sigmoid(4.001) * (1 - Sigmoid(4.001)) \end{bmatrix} = \begin{bmatrix} 0.058156 \\ 0.0564 \\ 0.0196 \end{bmatrix}$$

Values of derivative of output of layer-2 wrt input of layer1

For each input to neuron let us calculate the derivative with respect to each weight. Now let us look at the final derivative

$$\frac{\partial h2_{in1}}{\partial W_{j1k1}} = \frac{\partial((h1_{out1} * W_{j1k1}) + (h1_{out2} * Wj2k1) + (h1_{out3} * Wj3k1) + b_{k1})}{\partial W_{j1k1}}$$

$$\frac{\partial h2_{in1}}{\partial W_{j1k1}} = h1_{out1}$$

Derivative of layer 2 input wrt weight

By symmetry we can calculate:

$$\begin{bmatrix} \frac{\partial h2_{in1}}{\partial W_{j1k1}} \\ \frac{\partial h2_{in1}}{\partial W_{j2k1}} \\ \frac{\partial h2_{in1}}{\partial W_{j3k1}} \end{bmatrix} = \begin{bmatrix} h1_{out1} \\ h1_{out2} \\ h1_{out3} \end{bmatrix} = \begin{bmatrix} 1.35 \\ 1.27 \\ 1.8 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial h2_{in2}}{\partial W_{j1k2}} \\ \partial h2_{in2} \end{bmatrix} = \begin{bmatrix} h1_{out1} \\ \end{bmatrix} = \begin{bmatrix} 1.35 \\ \end{bmatrix}$$

$$\begin{bmatrix} \frac{}{\partial W_{j2k2}} \\ \frac{\partial h2_{in2}}{\partial W_{j3k2}} \end{bmatrix} = \begin{bmatrix} h1_{out2} \\ h1_{out3} \end{bmatrix} = \begin{bmatrix} 1.27 \\ 1.8 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial h2_{in3}}{\partial W_{j1k3}} \\ \frac{\partial h2_{in3}}{\partial W_{j2k3}} \\ \frac{\partial h2_{in3}}{\partial W_{j3k3}} \end{bmatrix} = \begin{bmatrix} h1_{out1} \\ h1_{out2} \\ h1_{out3} \end{bmatrix} = \begin{bmatrix} 1.35 \\ 1.27 \\ 1.8 \end{bmatrix}$$

Values of derivative layer 2 input wrt of weight

Now we will calculate the derivative of

$$W_{j3k1}$$

weight from j3 to k1

which will be simply.

$$\frac{\partial E_{total}}{\partial W_{j3k1}}$$

Derivative of Error wrt weight j3-k1

Using chain rule,

$$\frac{\partial E_{total}}{\partial W_{j3k1}} = \frac{\partial E_{total}}{\partial h2_{out1}} * \frac{\partial h2_{out1}}{\partial h2_{in1}} * \frac{\partial h2_{in1}}{\partial W_{j3k1}}$$

chain rule of derivative of error wrt weight

By symmetry we get the final matrix as,

$$\delta W_{jk} = \begin{bmatrix} \frac{\partial E_{total}}{\partial W_{j1k1}} & \frac{\partial E_{total}}{\partial W_{j1k2}} & \frac{\partial E_{total}}{\partial W_{j1k3}} \\ \frac{\partial E_{total}}{} & \frac{\partial E_{total}}{} & \frac{\partial E_{total}}{} \end{bmatrix} = \begin{bmatrix} \frac{\partial E_{total}}{\partial h2_{out1}} * \frac{\partial h2_{out1}}{\partial h2_{in1}} * \frac{\partial h2_{in1}}{\partial W_{j1k1}} & \frac{\partial E_{total}}{\partial h2_{out2}} * \frac{\partial h2_{out2}}{\partial h2_{in2}} * \frac{\partial h2_{in2}}{\partial W_{j1k2}} & \frac{\partial E_{total}}{\partial h2_{out3}} * \frac{\partial h2_{out3}}{\partial h2_{in3}} * \frac{\partial h2_{in3}}{\partial W_{j1k3}} \\ \frac{\partial E_{total}}{} * \frac{\partial h2_{out1}}{} * \frac{\partial h2_{in1}}{} & \frac{\partial E_{total}}{} * \frac{\partial h2_{out2}}{} * \frac{\partial h2_{in2}}{} & \frac{\partial E_{total}}{} * \frac{\partial h2_{out3}}{} * \frac{\partial h2_{in3}}{} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial E_{total}}{\partial W_{j3k1}} & \frac{\partial E_{total}}{\partial W_{j3k2}} & \frac{\partial E_{total}}{\partial W_{j3k3}} \end{bmatrix} \begin{bmatrix} \frac{\partial E_{total}}{\partial h2_{out1}} * \frac{\partial h2_{out1}}{\partial h2_{in1}} * \frac{\partial h2_{in1}}{\partial W_{j3k1}} & \frac{\partial E_{total}}{\partial h2_{out2}} * \frac{\partial h2_{out2}}{\partial h2_{in2}} * \frac{\partial h2_{in2}}{\partial W_{j3k2}} & \frac{\partial E_{total}}{\partial h2_{out3}} * \frac{\partial h2_{out3}}{\partial h2_{in3}} * \frac{\partial h2_{in3}}{\partial W_{j3k3}} \end{bmatrix}$$

final matrix of derivatives of weights_{jk}

We have already calculated the 2nd and 3rd term in each matrix. We need to check on the 1st term. If we see the matrix, the first term is common in all the columns. So there are only three values. Let us look into one value

$$\frac{\partial E_{total}}{\partial h2_{out1}} = \frac{\partial E_1}{\partial h2_{out1}} + \frac{\partial E_2}{\partial h2_{out1}} + \frac{\partial E_3}{\partial h2_{out1}}$$

Breakdown of error.

Lets see what each individual term boils down too.

$$\frac{\partial E_1}{\partial h2_{out1}} = \frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial h2_{out1}}$$

$$\frac{\partial E_2}{\partial h2_{out1}} = \frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial h2_{out1}}$$

$$\frac{\partial E_3}{\partial h2_{out1}} = \frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial h2_{out1}}$$

breakdown of each error derivative

by symmetry we get the final matrix as,

$$\begin{bmatrix} \frac{\partial E_{total}}{\partial h2_{out1}} \\ \frac{\partial E_{total}}{\partial h2_{out2}} \\ \frac{\partial E_{total}}{\partial h2_{out3}} \end{bmatrix} = \begin{bmatrix} (\frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial h2_{out1}}) + (\frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial h2_{out1}}) + (\frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial h2_{out1}}) \\ (\frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial h2_{out2}}) + (\frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial h2_{out2}}) + (\frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial h2_{out2}}) \\ (\frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial h2_{out3}}) + (\frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial h2_{out3}}) + (\frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial h2_{out3}}) \end{bmatrix}$$

Derivative of error wrt output of hidden layer 2

Again the first two values are already calculated by us when dealing with derivatives of W_{kl}. We just need to calculate the third one, Which is the derivative of input to each output layer wrt output of hidden layer-2. It is nothing but the corresponding weight which connects both the layers.

$$\begin{bmatrix} \frac{\partial O_{inl}}{\partial h2_{out1}} & \frac{\partial O_{in2}}{\partial h2_{out1}} & \frac{\partial O_{in3}}{\partial h2_{out1}} \\ \frac{\partial O_{inl}}{\partial h2_{out2}} & \frac{\partial O_{in2}}{\partial h2_{out2}} & \frac{\partial O_{in3}}{\partial h2_{out2}} \\ \frac{\partial O_{inl}}{\partial h2_{out3}} & \frac{\partial O_{in2}}{\partial h2_{out3}} & \frac{\partial O_{in3}}{\partial h2_{out3}} \end{bmatrix} = \begin{bmatrix} W_{k1l1} & W_{k1l2} & W_{k1l3} \\ W_{k2l1} & W_{k2l2} & W_{k2l3} \\ W_{k3l1} & W_{k3l2} & W_{k3l3} \end{bmatrix}$$

Derivative of input of output layer wrt hidden layer -2

$$\begin{bmatrix} \frac{\partial E_{total}}{\partial h2_{out1}} \\ \frac{\partial E_{total}}{\partial h2_{out2}} \\ \frac{\partial E_{total}}{\partial h2_{out3}} \end{bmatrix} = \begin{bmatrix} (\frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * W_{k1l1}) + (\frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * W_{k1l2}) + (\frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * W_{k1l3}) \\ (\frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * W_{k2l1}) + (\frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * W_{k2l2}) + (\frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * W_{k2l3}) \\ (\frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * W_{k3l1}) + (\frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * W_{k3l2}) + (\frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * W_{k2l3}) \end{bmatrix}$$

Final Matrix of derivative of total error wrt output of hidden layer-2

All Values are calculated before we just need to impute the corresponding values for our example.

$$\begin{bmatrix} \frac{\partial E_{total}}{\partial h2_{out1}} \\ \frac{\partial E_{total}}{\partial h2_{out2}} \\ \frac{\partial E_{total}}{\partial h2_{out3}} \end{bmatrix} = \begin{bmatrix} (-3.70644 * 0.15911 * 0.1) + (-1.4755 * 0.2040 * 0.4) + (-1.6886 * 0.3685 * 0.8) \\ (-3.70644 * 0.15911 * 0.3) + (-1.4755 * 0.2040 * 0.7) + (-1.6886 * 0.3685 * 0.2) \\ (-3.70644 * 0.15911 * 0.5) + (-1.4755 * 0.2040 * 0.2) + (-1.6886 * 0.3685 * 0.9) \end{bmatrix}$$

$$= \begin{bmatrix} (-0.0589) + (-0.2383) + (-0.5931) \\ (-0.1769) + (-0.417) + (-0.14828) \\ (-0.2948) + (-0.119) + (-0.667) \end{bmatrix} = \begin{bmatrix} -0.8903 \\ -0.74218 \\ -1.0810 \end{bmatrix}$$

calculations using an example

Let us look at the final matrix

$$\delta W_{jk} = \begin{bmatrix} \frac{\partial E_{total}}{\partial W_{j1k1}} & \frac{\partial E_{total}}{\partial W_{j1k2}} & \frac{\partial E_{total}}{\partial W_{j1k3}} \\ \frac{\partial E_{total}}{\partial W_{j2k1}} & \frac{\partial E_{total}}{\partial W_{j2k2}} & \frac{\partial E_{total}}{\partial W_{j2k3}} \\ \frac{\partial E_{total}}{} & \frac{\partial E_{total}}{} & \frac{\partial E_{total}}{} \end{bmatrix} = \begin{bmatrix} \frac{\partial E_{total}}{\partial h2_{out1}} * \frac{\partial h2_{out1}}{\partial h2_{in1}} * \frac{\partial h2_{in1}}{\partial W_{j1k1}} & \frac{\partial E_{total}}{\partial h2_{out2}} * \frac{\partial h2_{out2}}{\partial h2_{in2}} * \frac{\partial h2_{in2}}{\partial W_{j1k2}} & \frac{\partial E_{total}}{\partial h2_{out3}} * \frac{\partial h2_{out3}}{\partial h2_{in3}} * \frac{\partial h2_{in3}}{\partial W_{j1k3}} \\ \frac{\partial E_{total}}{\partial h2_{out1}} * \frac{\partial h2_{out1}}{\partial h2_{in1}} * \frac{\partial h2_{in1}}{\partial W_{j2k1}} & \frac{\partial E_{total}}{\partial h2_{out2}} * \frac{\partial h2_{out2}}{\partial h2_{in2}} * \frac{\partial h2_{in2}}{\partial W_{j2k2}} & \frac{\partial E_{total}}{\partial h2_{out3}} * \frac{\partial h2_{out3}}{\partial h2_{in3}} * \frac{\partial h2_{in3}}{\partial W_{j2k3}} \\ \frac{\partial E_{total}}{} * \frac{\partial h2_{out1}}{} * \frac{\partial h2_{in1}}{} & \frac{\partial E_{total}}{} * \frac{\partial h2_{out2}}{} * \frac{\partial h2_{in2}}{} & \frac{\partial E_{total}}{} * \frac{\partial h2_{out3}}{} * \frac{\partial h2_{in3}}{} \end{bmatrix}$$

$$\partial W_{j3k1} \quad \partial W_{j3k2} \quad \partial W_{j3k3} \quad \partial h2_{out1} \quad \partial h2_{in1} \quad \partial W_{j3k1} \quad \partial h2_{out2} \quad \partial h2_{in2} \quad \partial W_{j3k2} \quad \partial h2_{out3} \quad \partial h2_{in3} \quad \partial W_{j3k3}$$

Our final matrix of derivatives of Weights connecting hidden layer-1 and hidden layer-2

In our example,

$$\delta W_{jk} = \begin{bmatrix} -0.8903 * 0.058156 * 1.35 & -0.74218 * 0.0564 * 1.35 & -1.0810 * 0.0196 * 1.35 \\ -0.8903 * 0.058156 * 1.27 & -0.74218 * 0.0564 * 1.27 & -1.0810 * 0.0196 * 1.27 \\ -0.8903 * 0.058156 * 1.8 & -0.74218 * 0.0564 * 1.8 & -1.0810 * 0.0196 * 1.8 \end{bmatrix}$$

$$\delta W_{jk} = \begin{bmatrix} -0.06989 & -0.0565 & -0.0286 \\ -0.06575 & -0.05316 & -0.0269 \\ -0.0932 & -0.0753 & -0.03813 \end{bmatrix}$$

Calculations from our examples

Consider a learning rate (lr) of 0.01 We get our final Weight matrix as

$$\acute{W_{jk}} = \begin{bmatrix} W_{j1k1} - (lr * \delta W_{j1k1}) & W_{j1k2} - (lr * \delta W_{j1k2}) & W_{j1k3} - (lr * \delta W_{j1k3}) \\ W_{j2k1} - (lr * \delta W_{j2k1}) & W_{j2k2} - (lr * \delta W_{j2k2}) & W_{j2k3} - (lr * \delta W_{j2k3}) \\ W_{j3k1} - (lr * \delta W_{j3k1}) & W_{j3k2} - (lr * \delta W_{j3k2}) & W_{j3k3} - (lr * \delta W_{j3k3}) \end{bmatrix}$$

$$\acute{W_{jk}} = \begin{bmatrix} 0.2 - (0.01 * -0.06989) & 0.3 - (0.01 * -0.0565) & 0.5 - (0.01 * -0.0286) \\ 0.3 - (0.01 * -0.06575) & 0.5 - (0.01 * -0.05316) & 0.7 - (0.01 * -0.0269) \\ 0.6 - (0.01 * -0.0932) & 0.4 - (0.01 - 0.0753) & 0.8 - (0.01 * -0.03813) \end{bmatrix}$$

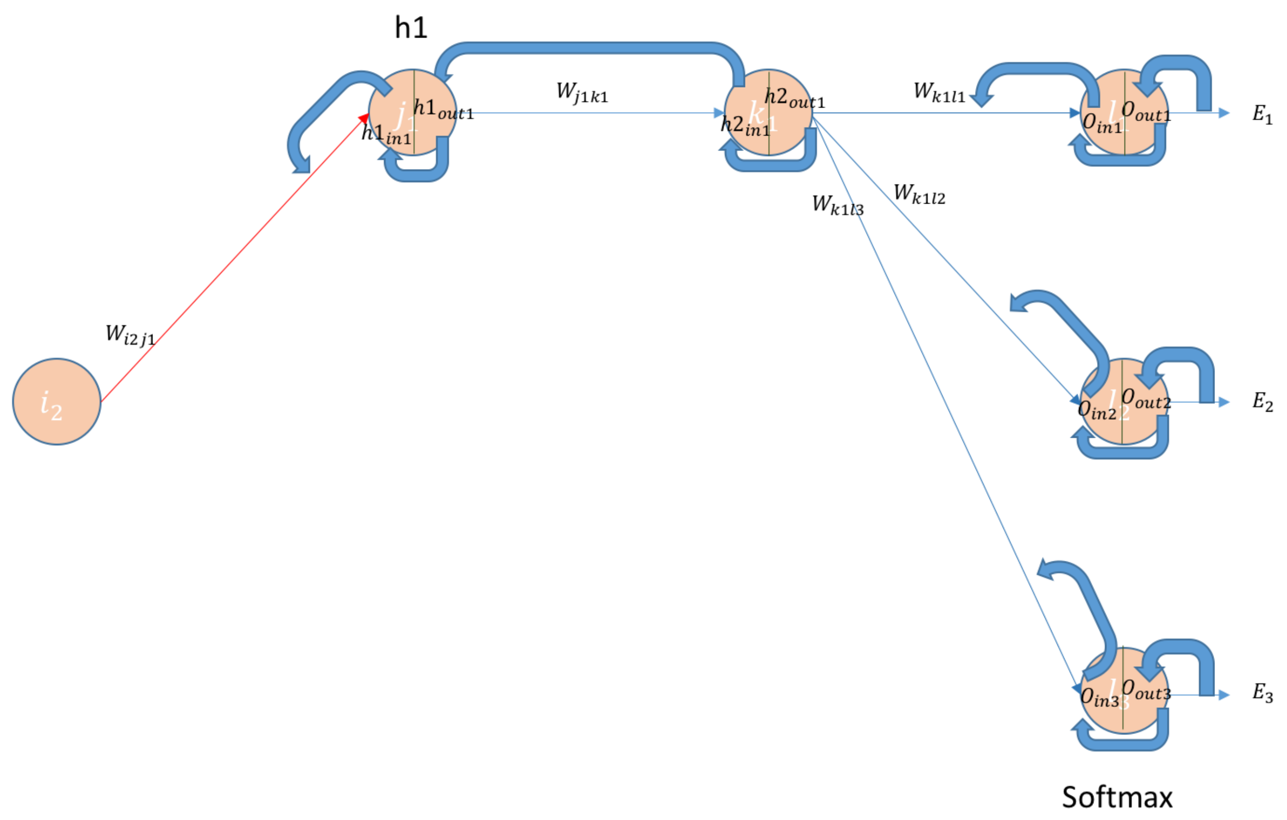
$$\acute{W_{jk}} = \begin{bmatrix} 0.2006989 & 0.300565 & 0.500286 \\ 0.3006575 & 0.5005316 & 0.700269 \\ 0.600932 & 0.400753 & 0.803813 \end{bmatrix}$$

Final modified matrix of W_{jk}

So, We have calculated new weight matrix for W_{jk}. Now let us move to the next layer:

BackPropagating the error — (Input Layer — Hidden Layer 1) Weights.

Edit:1 the following calculations from here are wrong. I took only wj1k1 and ignored wj1k2 and wj1k3. This was pointed by an user in comments. I would like someone to edit the jupyter notebook attached at the end. Please refer to some other implementations if u still didn't understand back-prop here.



Backpropagating errors to 1st layer

Let us calculate a few handy derivatives before we actually calculate the error derivatives wrt weights in this layer.

$$\frac{\partial h1_{out1}}{\partial h1_{in1}} = \frac{\partial Relu(h1_{in1})}{\partial h1_{in1}}$$

Derivative of hidden layer 1 output wrt to its input

We already know the derivative of relu (We have seen it at the beginning of the post). Since all inputs are positive, We will get output as 1

$$\frac{\partial h1_{out1}}{\partial h1_{in1}} = 1.0$$

$$\begin{bmatrix} \frac{\partial h1_{out1}}{\partial h1_{in1}} \\ \frac{\partial h1_{out2}}{\partial h1_{in2}} \\ \frac{\partial h1_{out3}}{\partial h1_{in3}} \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}$$

$$\left[\frac{\partial h1_{out3}}{\partial h1_{in3}}\right] \quad [1.0]$$

Calculations

For each input to neuron let us calculate the derivative with respect to each weight. Now let us look at the final derivative.

$$\frac{\partial h1_{in1}}{\partial W_{i1j1}} = \frac{\partial((I_{out1} * W_{j1k1}) + (I_{out2} * W_{i2j1}) + (I_{out3} * W_{i3j1}) + b_{j1})}{\partial W_{i1j1}}$$

$$\frac{\partial h1_{in1}}{\partial W_{i1j1}} = I_{out1}$$

Derivative of input to hidden layer wrt to weights

By symmetry we can write,

$$\begin{bmatrix} \frac{\partial h1_{in1}}{\partial W_{i1j1}} \\ \frac{\partial h1_{in1}}{\partial W_{i2j1}} \\ \frac{\partial h1_{in1}}{\partial W_{i3j1}} \end{bmatrix} = \begin{bmatrix} I_{out1} \\ I_{out2} \\ I_{out3} \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.7 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial h1_{in2}}{\partial W_{i1j2}} \\ \frac{\partial h1_{in2}}{\partial W_{i2j2}} \\ \frac{\partial h1_{in2}}{\partial W_{i3j2}} \end{bmatrix} = \begin{bmatrix} I_{out1} \\ I_{out2} \\ I_{out3} \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.7 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial h1_{in3}}{\partial W_{i1j3}} \\ \frac{\partial h1_{in3}}{\partial W_{i2j3}} \\ \frac{\partial h1_{in3}}{\partial W_{i3j3}} \end{bmatrix} = \begin{bmatrix} I_{out1} \\ I_{out2} \\ I_{out3} \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.7 \end{bmatrix}$$

Final derivative calculations

Now we will calculate the change in

$$W_{i2j1}$$

Weight connecting i2 neuron to j1

and generalize it to all variables. This will be simply

$$\frac{\partial E_{total}}{\partial W_{i2j1}}$$

derivative of error wrt to weight

Using chain rule,

$$\frac{\partial E_{total}}{\partial W_{i2j1}} = \frac{\partial E_{total}}{\partial h1_{out1}} * \frac{\partial h1_{out1}}{\partial h1_{in1}} * \frac{\partial h1_{in1}}{\partial W_{i2j1}}$$

chain rule for calculating error

By symmetry,

$$\delta W_{ij} = \begin{bmatrix} \frac{\partial E_{total}}{\partial W_{i1j1}} & \frac{\partial E_{total}}{\partial W_{i1j2}} & \frac{\partial E_{total}}{\partial W_{i1j3}} \\ \frac{\partial E_{total}}{\partial W_{i2j1}} & \frac{\partial E_{total}}{\partial W_{i2j2}} & \frac{\partial E_{total}}{\partial W_{i2j3}} \\ \frac{\partial E_{total}}{\partial W_{i3j1}} & \frac{\partial E_{total}}{\partial W_{i3j2}} & \frac{\partial E_{total}}{\partial W_{i3j3}} \end{bmatrix} = \begin{bmatrix} \frac{\partial E_{total}}{\partial h1_{out1}} * \frac{\partial h1_{out1}}{\partial h1_{in1}} * \frac{\partial h1_{in1}}{\partial W_{i1j1}} & \frac{\partial E_{total}}{\partial h1_{out2}} * \frac{\partial h1_{out2}}{\partial h1_{in2}} * \frac{\partial h1_{in2}}{\partial W_{i1j2}} & \frac{\partial E_{total}}{\partial h1_{out3}} * \frac{\partial h1_{out3}}{\partial h1_{in3}} * \frac{\partial h1_{in3}}{\partial W_{i1j3}} \\ \frac{\partial E_{total}}{\partial h1_{out1}} * \frac{\partial h1_{out1}}{\partial h1_{in1}} * \frac{\partial h1_{in1}}{\partial W_{i2j1}} & \frac{\partial E_{total}}{\partial h1_{out2}} * \frac{\partial h2_{out2}}{\partial h1_{in2}} * \frac{\partial h1_{in2}}{\partial W_{i2j2}} & \frac{\partial E_{total}}{\partial h1_{out3}} * \frac{\partial h2_{out3}}{\partial h1_{in3}} * \frac{\partial h1_{in3}}{\partial W_{i2j3}} \\ \frac{\partial E_{total}}{\partial h1_{out1}} * \frac{\partial h1_{out1}}{\partial h1_{in1}} * \frac{\partial h1_{in1}}{\partial W_{i3j1}} & \frac{\partial E_{total}}{\partial h1_{out2}} * \frac{\partial h2_{out2}}{\partial h1_{in2}} * \frac{\partial h1_{in2}}{\partial W_{i3j2}} & \frac{\partial E_{total}}{\partial h1_{out3}} * \frac{\partial h2_{out3}}{\partial h1_{in3}} * \frac{\partial h1_{in3}}{\partial W_{i3j3}} \end{bmatrix}$$

Final matrix using symmertry

We know the 2nd and 3rd derivatives in each cell in the above matrix. Let us look at how to get to derivative of 1st term in each cell.

$$\begin{bmatrix} \frac{\partial E_{total}}{\partial h1_{out1}} \\ \frac{\partial E_{total}}{\partial h1_{out2}} \end{bmatrix} = \begin{bmatrix} (\frac{\partial E_{total}}{\partial h2_{out1}} * \frac{\partial h2_{out1}}{\partial h2_{in1}} * \frac{\partial h2_{in1}}{\partial h1_{out1}}) \\ (\frac{\partial E_{total}}{\partial h2_{out2}} * \frac{\partial h2_{out2}}{\partial h2_{in2}} * \frac{\partial h2_{in2}}{\partial h1_{out2}}) \end{bmatrix}$$

$$\left[\frac{\partial E_{total}}{\partial h1_{out3}} \right] \quad \left[\left(\frac{\partial E_{total}}{\partial h2_{out3}} * \frac{\partial h2_{out3}}{\partial h2_{in3}} * \frac{\partial h2_{in3}}{\partial h1_{out3}} \right) \right]$$

Watching for the first term

We have calculated all the values previously except the last one in each cell, which is a simple derivative of linear terms.

$$\begin{bmatrix} \frac{\partial E_{total}}{\partial h1_{out1}} \\ \frac{\partial E_{total}}{\partial h1_{out2}} \\ \frac{\partial E_{total}}{\partial h1_{out3}} \end{bmatrix} = \begin{bmatrix} \left(\frac{\partial E_{total}}{\partial h2_{out1}} * \frac{\partial h2_{out1}}{\partial h2_{in1}} * W_{j1k1} \right. \\ \left(\frac{\partial E_{total}}{\partial h2_{out2}} * \frac{\partial h2_{out2}}{\partial h2_{in2}} * W_{j2k2} \right. \\ \left(\frac{\partial E_{total}}{\partial h2_{out3}} * \frac{\partial h2_{out3}}{\partial h2_{in3}} * W_{j3k3} \right. \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial E_{total}}{\partial h1_{out1}} \\ \frac{\partial E_{total}}{\partial h1_{out2}} \\ \frac{\partial E_{total}}{\partial h1_{out3}} \end{bmatrix} = \begin{bmatrix} -0.8903 * 0.058156 * 0.2 \\ -0.74218 * 0.0564 * 0.5 \\ -1.0810 * 0.0196 * 0.8 \end{bmatrix} = \begin{bmatrix} -0.01035 \\ -0.0209 \\ -0.0169 \end{bmatrix}$$

calculations in our example

In our example,

$$\delta W_{ij} = \begin{bmatrix} \frac{\partial E_{total}}{\partial W_{i1j1}} & \frac{\partial E_{total}}{\partial W_{i1j2}} & \frac{\partial E_{total}}{\partial W_{i1j3}} \\ \frac{\partial E_{total}}{\partial W_{i2j1}} & \frac{\partial E_{total}}{\partial W_{i2j2}} & \frac{\partial E_{total}}{\partial W_{i2j3}} \\ \frac{\partial E_{total}}{\partial W_{i3j1}} & \frac{\partial E_{total}}{\partial W_{i3j2}} & \frac{\partial E_{total}}{\partial W_{i3j3}} \end{bmatrix} = \begin{bmatrix} \frac{\partial E_{total}}{\partial h1_{out1}} * \frac{\partial h1_{out1}}{\partial h1_{in1}} * \frac{\partial h1_{in1}}{\partial W_{i1j1}} & \frac{\partial E_{total}}{\partial h1_{out2}} * \frac{\partial h1_{out2}}{\partial h1_{in2}} * \frac{\partial h1_{in2}}{\partial W_{i1j2}} & \frac{\partial E_{total}}{\partial h1_{out3}} * \frac{\partial h1_{out3}}{\partial h1_{in3}} * \frac{\partial h1_{in3}}{\partial W_{i1j3}} \\ \frac{\partial E_{total}}{\partial h1_{out1}} * \frac{\partial h1_{out1}}{\partial h1_{in1}} * \frac{\partial h1_{in1}}{\partial W_{i2j1}} & \frac{\partial E_{total}}{\partial h1_{out2}} * \frac{\partial h2_{out2}}{\partial h1_{in2}} * \frac{\partial h1_{in2}}{\partial W_{i2j2}} & \frac{\partial E_{total}}{\partial h1_{out3}} * \frac{\partial h2_{out3}}{\partial h1_{in3}} * \frac{\partial h1_{in3}}{\partial W_{i2j3}} \\ \frac{\partial E_{total}}{\partial h1_{out1}} * \frac{\partial h1_{out1}}{\partial h1_{in1}} * \frac{\partial h1_{in1}}{\partial W_{i3j1}} & \frac{\partial E_{total}}{\partial h1_{out2}} * \frac{\partial h2_{out2}}{\partial h1_{in2}} * \frac{\partial h1_{in2}}{\partial W_{i3j2}} & \frac{\partial E_{total}}{\partial h1_{out3}} * \frac{\partial h2_{out3}}{\partial h1_{in3}} * \frac{\partial h1_{in3}}{\partial W_{i3j3}} \end{bmatrix}$$

Final matrix

$$\delta W_{ij} = \begin{bmatrix} -0.01035 * 1 * 0.1 & -0.0209 * 1 * 0.1 & -0.0169 * 1 * 0.1 \\ -0.01035 * 1 * 0.2 & -0.0209 * 1 * 0.2 & -0.0169 * 1 * 0.2 \\ -0.01035 * 1 * 0.7 & -0.0209 * 1 * 0.7 & -0.0169 * 1 * 0.7 \end{bmatrix}$$

$$\delta W_{ij} = \begin{bmatrix} -0.001035 & -0.00209 & -0.00169 \\ -0.00207 & -0.00418 & -0.00338 \end{bmatrix}$$

$$\begin{bmatrix} -0.007245 & -0.01463 & -0.01183 \end{bmatrix}$$

calculations in our example

Consider a learning rate (lr) of 0.01 We get our final weight matrix as

$$\acute{W_{ij}} = \begin{bmatrix} W_{i1j1} - (lr * \delta W_{i1j1}) & W_{i1j2} - (lr * \delta W_{i1j2}) & W_{i1j3} - (lr * \delta W_{i1j3}) \\ W_{i2j1} - (lr * \delta W_{i2j1}) & W_{i2j2} - (lr * \delta W_{i2j2}) & W_{i2j3} - (lr * \delta W_{i2j3}) \\ W_{i3j1} - (lr * \delta W_{i3j1}) & W_{i3j2} - (lr * \delta W_{i3j2}) & W_{i3j3} - (lr * \delta W_{i3j3}) \end{bmatrix}$$

$$\acute{W_{ij}} = \begin{bmatrix} 0.1 - (0.01 * -0.001035) & 0.2 - (0.01 * -0.00209) & 0.3 - (0.01 * -0.00169) \\ 0.3 - (0.01 * -0.00207) & 0.2 - (0.01 * -0.00418) & 0.7 - (0.01 * -0.00338) \\ 0.4 - (0.01 * -0.0007245) & 0.3 - (0.01 - 0.01463) & 0.9 - (0.01 * -0.01183) \end{bmatrix}$$

$$\acute{W_{ij}} = \begin{bmatrix} 0.10001035 & 0.2000209 & 0.3000169 \\ 0.3000207 & 0.2000418 & 0.7000338 \\ 0.40007245 & 0.3001463 & 0.9001183 \end{bmatrix}$$

using learning rate we get final matrix

The End of Calculations

Our Initial Weights:

$$W_{ij} = \begin{bmatrix} W_{i1j1} & W_{i1j2} & W_{i1j3} \\ W_{i2j1} & W_{i2j2} & W_{i2j3} \\ W_{i3j1} & W_{i3j2} & W_{i3j3} \end{bmatrix} = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.3 & 0.2 & 0.7 \\ 0.4 & 0.3 & 0.9 \end{bmatrix}$$

$$W_{jk} = \begin{bmatrix} W_{j1k1} & W_{j1k2} & W_{j1k3} \\ W_{j2k1} & W_{j2k2} & W_{j2k3} \\ W_{j3k1} & W_{j3k2} & W_{j3k3} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.3 & 0.5 \\ 0.3 & 0.5 & 0.7 \\ 0.6 & 0.4 & 0.8 \end{bmatrix}$$

$$W_{kl} = \begin{bmatrix} W_{k1l1} & W_{k1l2} & W_{k1l3} \\ W_{k2l1} & W_{k2l2} & W_{k2l3} \\ W_{k3l1} & W_{k3l2} & W_{k3l3} \end{bmatrix} = \begin{bmatrix} 0.1 & 0.4 & 0.8 \\ 0.3 & 0.7 & 0.2 \\ 0.5 & 0.2 & 0.9 \end{bmatrix}$$

Our Final Weights:

$$\acute{W}_{ij} = \begin{bmatrix} 0.10001035 & 0.2000209 & 0.3000169 \\ 0.3000207 & 0.2000418 & 0.7000338 \\ 0.40007245 & 0.3001463 & 0.9001183 \end{bmatrix}$$

$$\acute{W}_{jk} = \begin{bmatrix} 0.2006989 & 0.300565 & 0.500286 \\ 0.3006575 & 0.5005316 & 0.700269 \\ 0.600932 & 0.400753 & 0.803813 \end{bmatrix}$$

$$\acute{W}_{kl} = \begin{bmatrix} 0.105531 & 0.400576 & 0.805836 \\ 0.30055 & 0.700577 & 0.2005849 \\ 0.5005779 & 0.2006017 & 0.9006098 \end{bmatrix}$$

Important Notes:

- I have completely eliminated bias when differentiating. Do you know why ?
- Backprop of bias should be straightforward. Try on your own.
- I have taken only one example. What will happen if we take batch of examples?
- Though I have not mentioned directly about vanishing gradients. Do you see why it occurs?
- What would happen if all the weights are the same number instead of random ?

References Used:

- http://csrgxtu.github.io/2015/03/20/Writing-Mathematic-Fomulars-in-Markdown/
- https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/
- http://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/

Code available on Github :

- https://github.com/Prakashvanapalli/TensorFlow/blob/master/Blogposts/Backpropogation_with_Images.ipynb

I have hand calculated everything. Let me know your feedback. If you like it, please recommend and share it. Thank you.

Neural Networks Backpropagation Deep Learning Artificial Intelligence TensorFlow