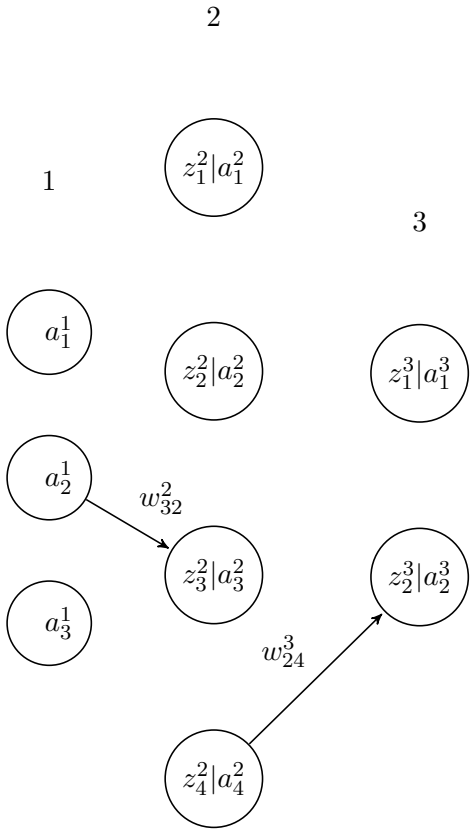
BACKPROPAGATION

Backpropagation is a computer algorithm used for supervised training of Artificial Neural Networks. It is based on two ideas - the Chain Rule of derivatives in Calculus and Dynamic Programming.

1 Notation



The layers are of the Neural Network are denoted by positive integers with input layer denoted by 1, the first hidden layer by 2 and so on till the output layer. The inputs in the input layer are denoted by a_i^1 as shown above. For all the other neurons in other layers, z_i^l denotes the weighted sum of the neuron activations in the previous layer and a_i^l denotes that neuron's

activation. The weights are superscripted by the index of the layer of the neuron at the end, and subscripted first by the layer index of the end neuron and then by that of the start neuron.

2 Forward Pass

For any layer l which is apart from the input layer, it can be written,

$$z_i^l = \left(\sum_j w_{ij}^l a_j^{l-1} \right) + b_i^l$$

Taking examples from the figure,

$$\begin{aligned}
z_1^2 &= (w_{11}^2 a_1^1 + w_{12}^2 a_2^1 + w_{13}^2 a_3^1) + b_1^2 \\
z_2^2 &= (w_{21}^2 a_1^1 + w_{22}^2 a_2^1 + w_{23}^2 a_3^1) + b_2^2 \\
z_3^2 &= (w_{31}^2 a_1^1 + w_{32}^2 a_2^1 + w_{33}^2 a_3^1) + b_3^2 \\
z_4^2 &= (w_{41}^2 a_1^1 + w_{42}^2 a_2^1 + w_{43}^2 a_3^1) + b_4^2
\end{aligned}$$

These equations can be rewritten in matrix form as follows,

$$\begin{pmatrix} z_1^2 \\ z_2^2 \\ z_3^2 \\ z_4^2 \end{pmatrix} = \begin{pmatrix} w_{11}^2 & w_{12}^2 & w_{13}^2 \\ w_{21}^2 & w_{22}^2 & w_{23}^2 \\ w_{31}^2 & w_{32}^2 & w_{33}^2 \\ w_{41}^2 & w_{42}^2 & w_{43}^2 \end{pmatrix} \begin{pmatrix} a_1^1 \\ a_2^1 \\ a_3^1 \end{pmatrix} + \begin{pmatrix} b_1^2 \\ b_2^2 \\ b_3^2 \\ b_4^2 \end{pmatrix}$$

or

$$\boldsymbol{Z}^2 = \boldsymbol{W}^2 \boldsymbol{A}^1 + \boldsymbol{B}^2$$

Generalizing for all layers,

$$\boldsymbol{Z}^l = \boldsymbol{W}^l \boldsymbol{A}^{l-1} + \boldsymbol{B}^l \tag{1}$$

Activation function σ is applied to z_i^l to yield a_i^l.

$$\begin{aligned}
a_1^2 &= \sigma(z_1^2) \\
a_2^2 &= \sigma(z_2^2) \\
a_3^2 &= \sigma(z_3^2) \\
a_4^2 &= \sigma(z_4^2)
\end{aligned}$$

which can be written in matrix form as,

$$\boldsymbol{Z}^2 = \sigma(\boldsymbol{A}^2)$$

or more generally as,

$$\boldsymbol{Z}^l = \sigma(\boldsymbol{A}^l) \tag{2}$$

Given $\boldsymbol{A}^1$, any $\boldsymbol{Z}^l$ or $\boldsymbol{A}^l$ can be calculated using the two equations (1) and (2). This completes the analysis of the forward pass of backpropagation algorithm.

3 Backward Pass

Imagine a point in time where the weights and biases of a neural network are fixed. Now, one training example is taken and its inputs are fed into the neural network. Using the the forward pass equations, the neural network's output is computed. Usually, this output is different from the training example's expected output and thus the goal of backpropagation is to calucate the partial derivates of the cost function w.r.t. each weight and bias for them to be tweaked optimally.

3.1 Cost Function

The notion of the difference between the expected output of a training example and the actual output from neural network is formalized as follows,

$$\begin{aligned} C &= \frac{1}{2}||\boldsymbol{Y} - \boldsymbol{A}^L||^2 \\ &= \frac{1}{2}\sum_j (y_j - a_j^L)^2 \end{aligned}$$

3.2 Backpropagation

$$\begin{aligned} \frac{\partial C}{\partial z_j^L} &= (a_j^L - y_j)\frac{\partial a_j^L}{\partial z_j^L} \\ &= (a_j^L - y_j) \times \sigma^{'}(z_j^L) \end{aligned}$$

Denoting $\frac{\partial C}{\partial z_j^L}$ by δ_j^L and $(\delta_1^L, \delta_2^L, ..)^T$ by $\boldsymbol{\Delta^L}$, it can be written

$$\boldsymbol{\Delta^L} = (\boldsymbol{A^L} - \boldsymbol{Y}) \odot \sigma^{'}(\boldsymbol{Z^L}) \tag{3}$$

where $\odot$ denotes element-wise matrix product.
Can $\boldsymbol{\Delta^L}$ somehow be related to $\boldsymbol{\Delta^{L-1}}$?

$$\begin{aligned}
\delta_j^{L-1} &= \frac{\partial C}{\partial z_j^{L-1}} \\
&= \frac{\partial C}{\partial a_j^{L-1}} \frac{\partial a_j^{L-1}}{\partial z_j^{L-1}} \\
&= \frac{\partial C}{\partial a_j^{L-1}} \times \sigma^{'}(z_j^{L-1}) \\
&= \left(\sum_i \frac{\partial z_i^L}{\partial a_j^{L-1}} \frac{\partial C}{\partial z_i^L} \right) \times \sigma^{'}(z_j^{L-1}) \\
&= \left(\sum_i w_{ij}^L \delta_i^L \right) \times \sigma^{'}(z_j^{L-1})
\end{aligned}$$

For the above example, the equations become

$$\begin{pmatrix} \delta_1^{L-1} \\ \delta_2^{L-1} \\ \delta_3^{L-1} \\ \delta_4^{L-1} \end{pmatrix} = \begin{pmatrix} w_{11}^L & w_{21}^L \\ w_{12}^L & w_{22}^L \\ w_{13}^L & w_{23}^L \\ w_{14}^L & w_{24}^L \end{pmatrix} \begin{pmatrix} \delta_1^L \\ \delta_2^L \end{pmatrix} \odot \begin{pmatrix} \sigma^{'}(z_1^{L-1}) \\ \sigma^{'}(z_2^{L-1}) \\ \sigma^{'}(z_3^{L-1}) \\ \sigma^{'}(z_4^{L-1}) \end{pmatrix}$$

$$\boldsymbol{\Delta^{L-1}} = \boldsymbol{W^L}^T \boldsymbol{\Delta^L} \odot \sigma^{'}(\boldsymbol{Z^{L-1}})$$

The equation can be generalized to,

$$\boldsymbol{\Delta^l} = \boldsymbol{W^{l+1}}^T \boldsymbol{\Delta^{l+1}} \odot \sigma^{'}(\boldsymbol{Z^l}) \tag{4}$$

Using equation (3) and (4) $\boldsymbol{\Delta^l}$ can be calculated for any l.

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial z_j^l}{\partial w_{jk}^l} \frac{\partial C}{\partial z_j^l} = a_k^{l-1} \delta_j^l \tag{5}$$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial z_j^l}{\partial b_j^l} \frac{\partial C}{\partial z_j^l} = \delta_j^l \tag{6}$$

Hence, equations (3) and (4) "propagate" the error backwards and then with help of equations (5) and (6), partial derivative of the error w.r.t. any neural network weight or bias can be calculated.

4 Conclusion

Backpropagation is famous because it can quickly calculate partial derivatives to optimally adjust the parameters of a neural network. In the description above, backpropagation calculates the derivatives w.r.t only one training example. In practice training examples are bunched into mini-batches and partials calculated for each example in a mini batch are averaged for network parameter adjustment.