Gradient backpropagation

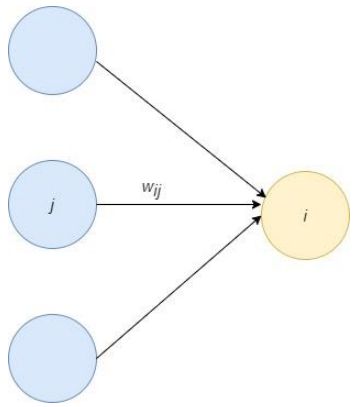
Detailed explanation

Abdelwahid Benslimane
wahid.benslimane@gmail.com

In this document, I will explain in detail the principle of gradient backpropagation which is a method that optimises the calculation of the gradient of the loss function involved in the training of a neural network. This gradient is then used in gradient descent which is a technique for converging the weights of the arcs to the values that will minimise the loss function during the training of the neural network.

As a reminder, training a neural network involves two phases: the forward pass and the backward pass. The forward pass phase corresponds to the passage of training examples through the layers of the network and the calculation of the loss function. The backward pass consists of updating the network weights to minimise this loss function during the subsequent forward passes.

Below I have represented the output layer of a multilayer perceptron as well as the hidden layer that precedes it. In this example, the output layer contains only one neuron, the neuron i. The neuron j is a neuron of the preceding layer and w_{ij} is the weight of the arc which connects the neuron j to the neuron i.



- f_o: activation function associated with the output layer
- θ_i is the output signal of the neuron i: $\theta_i = f_o(s_i)$
- s_i is the weighted sum of the signals of the neurons of the preceding layer weighted by the weights of the arcs (one specific weight per arc) that connect these neurons to neuron i:

$$s_i = \sum_{j=1}^{n_j} w_{ij}\,\theta_j$$

We want to be able to determine the partial derivative of the loss function to be minimised, which I call R, with respect to the weight of each arc. To do so, all we

need is to use the chain rule, which allows us to find the derivative of a composite function. As a reminder, $(g \circ h)' = h'.g'(h)$, therefore:

$$\frac{\partial R}{\partial w_{ij}} = \frac{\partial s_i}{\partial w_{ij}} \frac{\partial R}{\partial s_i}$$

In the general case, for a given arc between a neuron i of the output layer and one of its predecessors j:

$$\frac{\partial s_i}{\partial w_{ij}} = \theta_j$$

We will consider the loss function below as an example. Note that the number of input data items used in the training phase is n_a, the number of output neurons is q, the superscript a represents a particular input data item and the subscript i represents a particular neuron of the output layer:

$$R = \sum_{a=1}^{n_a} \sum_{i=1}^{q} (\theta_i^a - d_i^a)^2 = \sum_{a=1}^{n_a} \sum_{i=1}^{q} (f_o(s_i^a) - d_i^a)^2$$

θ_i^a: value of the signal of the output neuron i obtained with the training data item a

d_i^a: value to be approximated by θ_i^a/ real output value of the neuron i associated with the training data item a

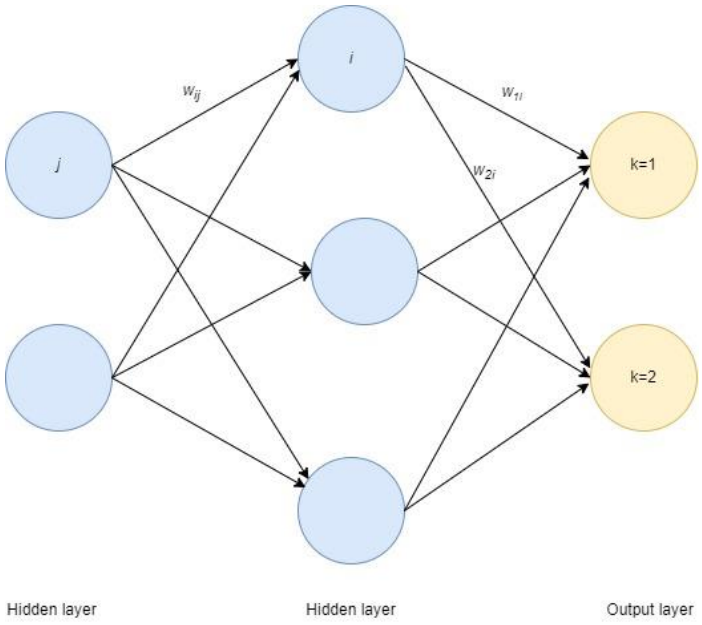
Let us place ourselves in the context of a specific input data item, the one designated by the superscript a, and of a specific output neuron, the one designated by the subscript i. We can write:

$\frac{\partial R^a}{\partial s_i^a} = f_o'(s_i^a) \frac{\partial R^a}{\partial f_o(s_i^a)} = f_o'(s_i^a) \frac{\partial R^a}{\partial \theta_i^a} = \delta_i^a$ (this value is easy to compute)

Therefore, the contribution of the input data a to the partial derivative of the error function with respect to the weight w_{ij} is:

$$\frac{\partial R^a}{\partial w_{ij}} = \theta_j^a \delta_i^a$$

In the following, we still place ourselves in the context of the particular input data item a. Also, we will now take into consideration the below neural network arhitecture:



We want now to compute the partial derivative of the error function with respect to the weight of a specific arc which connects a neuron of the first hidden layer starting from the right, the neuron i, with a neuron of the preceding hidden layer, the neuron j. We already know that: $\frac{\partial R^a}{\partial w_{ij}} = \frac{\partial s_i^a}{\partial w_{ij}} \frac{\partial R^a}{\partial s_i^a}$ and $\frac{\partial s_i^a}{\partial w_{ij}} = \theta_j^a$.

We can then write:

$$\theta_k^a = f_o(s_k^a)$$

$$s_k^a = \sum_{i=1}^{n_i} w_{ki}\theta_i^a$$

$$\frac{\partial R^a}{\partial s_k^a} = f_o'(s_k^a)\frac{\partial R^a}{\partial f_o(s_k^a)} = f_o'(s_k^a)\frac{\partial R^a}{\partial \theta_k^a} = \delta_k^a$$

Neurons within the same layer share the same activation function, and f_o represents the activation function of neurons of the output layer.

If n_k is the number of neurons of the output layer that the neuron i is connected to, it comes that:

$$\frac{\partial R^a}{\partial s_i^a} = \sum_{k=1}^{n_k} \left(\frac{\partial s_k^a}{\partial s_i^a} \frac{\partial R^a}{\partial s_k^a} \right) = \sum_{k=1}^{n_k} (w_{ki} f_h'(s_i^a) \delta_k^a) = \delta_i^a$$

The activation function f_h is common to all neurons of the layer where the neuron i finds itself.

The value δ_i^a is easy to compute as we already know the values of all the parameters involved at this stage. It follows that the contribution of the input data item a to the partial derivative of the error function with respect to w_{ij} is:

$$\frac{\partial R^a}{\partial w_{ij}} = \theta_j^a \delta_i^a$$

We have seen how the computation of the gradient of the loss function is performed and propagated from the last layer to all the previous ones, this is the gradient backpropagation.