Critique of backprop by Hinton et. Al

Based on the work on Artificial Neural Networks that had made some progress, Hinton et. al. have constructed an algorithm for adjusting the errors/difference as compared to the actual outputs and thereby reflect them to the weights of the hidden units.

The network can be as below. There are a number of neurons in each layer. While the neurons between two consecutive layers are connected, the neurons run parallel (i.e., not connected) in the same layer. The last layer is supposed to produce the desired output. But, how can that occur? This is easily possible by taking the squared error of the actual and predicted output.

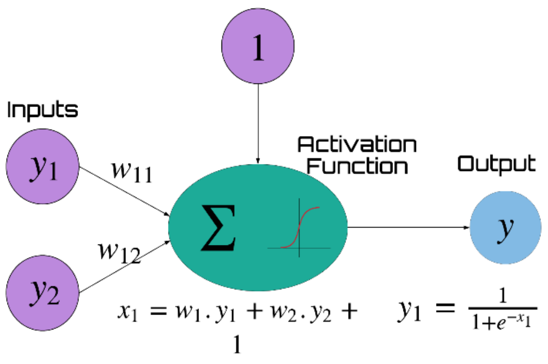
Before we dive into the back-propagation algorithm, let's understand the forward propagation in terms of equations. Let's assume the input to a layer is given $\boldsymbol{x_j}$ is the total sum of the outputs of the previous layer $\boldsymbol{y_i}$ multiplied by their respective weights $\boldsymbol{w_{ji}}$. They can be expressed as below:

$$\boldsymbol{x_j = \sum_i y_i . w_{ji}}$$

An additional bias of $\mathbf{1}$ can be added to this. Lastly, the neuron gets activated by a non-linear function as below:

$$\boldsymbol{y_j = \frac{1}{1 + e^{x_j}}}$$

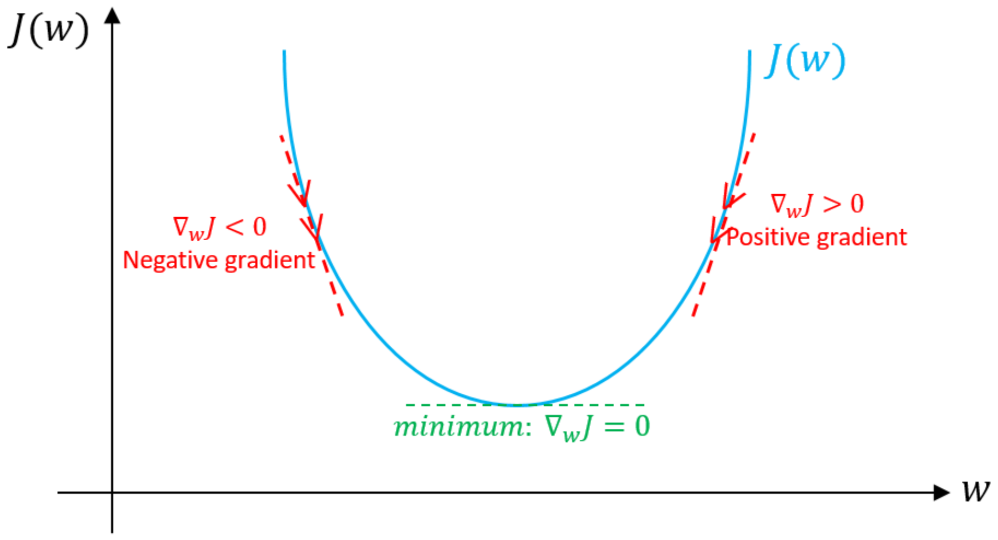
The pictorial representation can be as below:



This operation recursively occurs to all the neurons till the output unit. The output unit is compared with the actual output using squared error as below:

$$\boldsymbol{E = \frac{1}{2} \sum_c \sum_j (y_{j,c} - d_{j,c})^2}$$

Now let's move to the next phase, i.e., back pass. We can accomplish the reduction of error by reflecting a reduction in the error to all the weights of the network. Hence, we perform gradient descent where you find you effectively find the slope of the $Error\ Function$.



To get an intuition od the gradient descent, let's consider the picture above (Please ignore the notations as they are different). Suppose the slope(gradient) is negative and we subtract the gradient from w, we are actually adding some value to $\boldsymbol{w}$ ($\because -(-) = +$)and hence, $\boldsymbol{w}$ moves towards the minima, while if the gradient is positive, we still subtract from the gradient but this time, we are effectively subtracting as the gradient is positive. Hence, this methodology of reducing the error so that we reach the minima is called gradient descent. This above procedure is performed for all the variables as follows:

1st the output:

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{y_j}} = \boldsymbol{y_j} - \boldsymbol{d_j}$$

and since $\boldsymbol{y_j}$ is dependent of $\boldsymbol{x_j}$ as:

$$\boldsymbol{y_j} = \frac{\boldsymbol{1}}{\boldsymbol{1} + \boldsymbol{e^{x_j}}}$$

We reflect the same changes to $\boldsymbol{x_j}$ as:

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{x_j}} = \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{y_j}} . \frac{\partial \boldsymbol{y_j}}{\partial \boldsymbol{x_j}}$$

$$= \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{y_j}} * \boldsymbol{y_j}(\boldsymbol{1} - \boldsymbol{y_j})$$

Now the last set of variables left are $\boldsymbol{w_{ji}}$ which are in turn dependent on $\boldsymbol{x_j}$ in the back-tracking order as:

$$\boldsymbol{x_j} = \sum_{\boldsymbol{i}} \boldsymbol{y_i} . \boldsymbol{w_{ji}}$$

$$\Rightarrow \frac{\partial x_j}{\partial w_{ji}} = y_i$$

And

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial x_j} . \frac{\partial x_j}{\partial w_{ji}}$$

$$= \frac{\partial E}{\partial x_j} . y_i$$

Now moving a layer backward, we calculate y_i as:

$$\frac{\partial x_j}{\partial y_i} = w_{ji}$$

$$\& \therefore \frac{\partial E}{\partial y_i} = \frac{\partial E}{\partial x_j} . \frac{\partial x_j}{\partial y_i}$$

$$\Rightarrow \frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} . w_{ji}$$

Now that we have calculated all the gradients, we must now reduce the errors of the respective variables by subtracting the gradients from them. Additionally, we could use a dampening factor incase the gradient values are too high. Let the factor be ϵ lying between $\mathbf{0}$ and $\mathbf{1}$. This procedure is apparently quiet slow and hence a momentum factor α can be utilized along with the weights. These two combined are shown below:

$$\Delta w(t) = \alpha . \Delta w(t-1) - \epsilon . \frac{\partial E}{\partial w(t)}$$

Where t is the current sweep/iteration and w can be any of the weights.

The other aspects mentioned in the paper include:

1. Random initialization of the weights. If the weights were to be initialized with the same value, then all would get updated with the same values, leading to all the weights learning the same values. Hence, random initialization ensures that the symmetry is broken.
2. It must also be noted that when only sufficient number of neurons are used, the network might get caught up in a local minima.

This paper has presented a breakthrough in terms of the learning that an artificial network has achieved although this does not exemplify the mechanism or learning in a biological neural network!