Back Propagation

Training a neural network by adjusting connection weights is very computationally intensive. Consequently, though they had been studied for decades prior, Artificial Neural Networks(ANN) were rarely applied to real-world learning tasks until the mid-to-late 1980s, when an efficient method of training an ANN was discovered. The algorithm, which used a strategy of back-propagating errors, is now known simply as backpropagation.

In its most general form, the backpropagation algorithm iterates through many cycles of two processes. Each cycle is known as an epoch. Because the network contains no a priori (existing) knowledge, the starting weights are typically set at random. Then, the algorithm iterates through the processes, until a stopping criterion is reached. Each epoch in the backpropagation algorithm includes:

- A forward phase in which the neurons are activated in sequence from the input layer to the output layer, applying each neuron's weights and activation function along the way. Upon reaching the final layer, an output signal is produced.
- A backward phase in which the network's output signal resulting from the forward phase is compared to the true target value in the training data. The difference between the network's output signal and the true value results in an error that is propagated backwards in the network to modify the connection weights between neurons and reduce future errors.

Over time, the network uses the information sent backward to reduce the total error of the network. Yet one question remains: because the relationship between each neuron's inputs and outputs is complex, how does the algorithm determine how much a weight should be changed? The answer to this question involves a technique called gradient descent(which we have already seen).

In a similar process, the backpropagation algorithm uses the derivative of each neuron's activation function to identify the gradient in the direction of each of the incoming weights—hence the importance of having a differentiable activation function. The gradient suggests how steeply the error will be reduced or increased for a change in the weight. The algorithm will attempt to change the weights that result in the greatest reduction in error by an amount known as the learning rate. The greater the learning rate, the faster the algorithm will attempt to descend down the gradients.

References:

For more details on Backpropogation,
`https://en.wikipedia.org/wiki/Backpropagation`

`https://brilliant.org/wiki/backpropagation/`