Report 3 - Backpropagation

Khoa Doan

Before we begin, there are some terminology:

- I assume that we have known about perceptron and its learning model (at least we have known about this in class).
- The delta rule of perceptron is simply as following: the weight change is proportional to the difference of the target value and the predicted (output) value by computing the activation of the node.
- Each neuron (or node) in an artificial neural network has an activation function, whose output is called activation of a node. In a feedforward multilayer network, the layers that are not input and output layers are called hidden layers.

I. History

In class, we have come to know that perceptron and other single-layer networks are seriously limited in their capabilities, e.g. they cannot learn a non-linear decision boundary such as the XOR problem [1]. In 1969, in a controversial book by Minsky and Papert, they took their criticism further by conjecturing that *"if you had hidden units, you could compute any boolean function; however, no learning rule exists for such multi-layer networks, and we don't think one will ever be discovered!"* In other words, the simple learning mechanism of perceptron (such as Delta Rule) cannot be extended easily in networks that have more than 2 layers (input and output). Nevertheless, as the history has shown us, Backpropagation (Backprop) was one such successful mechanism.

The term backpropagation (or its earliest version [*]) was proposed by Frank Rosenblatt in 1961 but his proposal was crippled by the use of the perceptron's step function which is not differential everywhere when differentiable function must be used for successful application of backpropagation [4]. In 1974, Webos proposed a new backpropagation learning algorithm but was largely ignored by the scientific community until 1980's. In 1985, Parker and LeCun simultaneously rediscovered it, but the modern specification of backpropagation was proposed and popularized by Rumelhart, Hinton and Williams in 1986. Since then, backpropagation has had major impacts on the field of neural networks and has been widely applied to many problems in other disciplines, as well as has been used to solve several kinds of applications including classification, function approximation and forecasting.

II. The Learning Algorithm

It is useful to note that Backprop was not the only attempted learning model for multi-layer networks. In particular, several attempts had considered a learning model where we (1) randomly perturbed the hidden activations, (2) assessed if it improved the network's performance and then (3) made the weight changes if the performance increases. However, as one may expect, this way of learning is very inefficient since we need several forward passes (computation of activation in each assessment of the performance) for just some weight changes. It is also unstable in many problems where a large perturbation at the end of the learning would nearly always make things worse. In essence, it is not as simple as the perceptron's learning that, every time we make weight changes, they get closer to a "generally optimal" set of weights with respect to the training data.

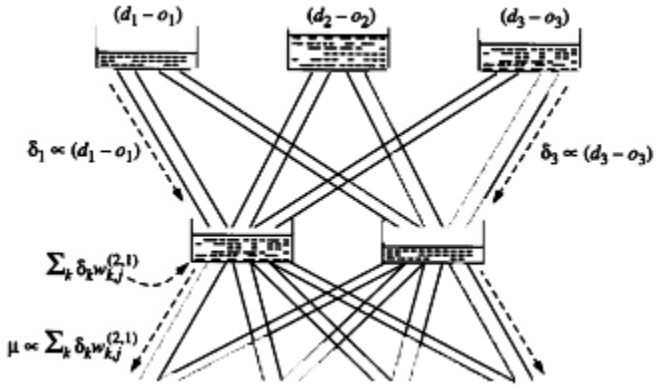
One can ask, can we make weight changes in the hidden layer that almost always guarantee that the network will perform better, as in the case of the perceptron's? If we consider the performance of the network is the error of its prediction (e.g. sum of squared errors over the training data), we can certainly

This is why we need a differentiable function. Probably, since Rosenblattused the step function, he never think about the fact that using thegradient, or approximating it, could give him something as elegant asthe modern backpropagation algorithm.

know how fast this error changes as we change a hidden activity because this is essentially finding the derivative of the error function with respect to the hidden activities. Then a weight change that is relative to this error gradient will ensure the network to get to closer to the optimal error state (although probably a local minima). This is the main idea of Backprop learning. The Backprop learning algorithm for a feedforward multi-layer network can be simply described as following:

i. Forward Pass: Activation of each neuron is computed by applying the activation function to the net input to the neuron, from the input layer to the output layer.

ii. Backward Pass (Weight Correction): Weights are updated in order from the output layer back to the input layer by computing their error gradient, as in Figure 1.



w_{ji} : connection weight from neuron i to j
S(.) : activation function
net_j : net weighted input into neuron j
d_k : target (actual) value of output node k
o_k : output value (predicted or activation) of output node k
ɳ : learning rate (how big the step relative to the gradient we want to make).

Figure1. Flow of Error in Backpropagation. Dashed arrows show the order of weight updates.

Because of limited space, the derivations of the error gradient with respect to the network weights at different layers are not shown here. In general, in a 3-layer feedforward network (layer 1 is input, layer 2 is hidden, layer 3 is output), the weights are changed according to the following rules:

$$\Delta w_{k,j}^{(2,1)} = \eta \times \delta_k \times x_j^{(1)} \qquad \Delta w_{j,i}^{(1,0)} = \eta \times \mu_j \times x_i$$

where $\delta_k = (d_k - o_k)S'\left(net_k^{(2)}\right)$

and

$$\mu_j = \left(\sum_k \delta_k w_{k,j}^{(2,1)}\right) S'\left(net_j^{(1)}\right).$$

($w_{k,j}^{(m,n)}$ denotes weight connection from node j in layer n to node k in layer m)

The propagation of the weight changes, according to the above formulation, expresses a very elegant recursion that is, in fact, a generalization of the delta rule of perceptron. The delta value (δ_k) at an output node k is the difference of its target and output values, multiplied by the derivative of its activation function with respect to the net input to the node. The delta value (μ_j) at a non-output node j is the weighted delta values of the nodes that node j connects to, multiplied by the multiplied by the derivative of its activation function with respect to the net input to the node. This is, indeed, a learning model that is very similar to the perceptron's, but with much better computational power.

There are many interesting theoretical, as well as experimental aspects of backpropagation since its creation. As for its model of learning (or the optimization of the error function), there are various ways, such as online (weight changes are performed at each presentation of a training example; this is essentially a stochastic gradient descent method), batch (weight changes after presentation of the entire training data), or mini-batch (weight changes after some presentation of the training data) [6]. As for its generalization as a learning algorithm (i.e. how do we ensure that the learned weights work well for examples that we have not seen before), there are versions using fixed learning rate, using adaptive learning rate, using separate adaptive learning rate for each connection, or using momentum [6]. Backpropagation certainly has limitations as a learning algorithm. Essentially, learning with backpropagation is very slow, and it could result in a local optimal solution on its error surface. However, backpropagation is still a very useful method because it is very robust to noise in training data, as well as the fact that the investigation of the hidden layers can give us useful representation of the data (e.g. rule extraction of hidden layers).

Biological Plausibility

The creation of Backpropagation did not base its derivation on any biological consideration of an actual biological neuron. Optimistically, recent studies have shown that, much similar to the backpropagation learning, action potential of a neuron creates a "voltage spike both at the end of the axon" [3] (forward pass) "and back through the dendrites" [3] (backward pass). The backpropagation signal is interesting because it increases the concentration of Ca^{2+} in the postsynaptic dendrites, an event which is in certain models of synaptic plasticity (i.e. it can increase or decrease the synaptic strength). This, in my opinion, mean that the neural error backpropagation signal is entirely independent of any activities in the postsynaptic neurons (i.e those that the above neuron is connected to), and therefore is very different from the mechanism behind backpropagation, where error signals are propagated from the postsynaptic neurons. Indeed, the mechanism behind the neural backpropagation is currently not known. One final note about neural backpropagation is that it occurs more actively in areas that synaptic plasticity is more apparent (e.g. actively in hippocampus which controls memory, and passively in cerebellum, which controls vegetable functions).

Another comment on the Backprop is that algorithm is that the weight change, as we see above, although involves the presynaptic activation (x_i or x_j), but is not dependent on the postsynaptic activation. Hebb's Rule suggests otherwise, where the synaptic changes are some function of the activations of both connected neurons. There were, in fact, attempts to model this biological significance in Backpropagation networks. One example is the Generalized Recirculation Algorithm, whose a weight change is an approximation of the Backprop's weight change, but the nice property of the change is that this approximation is exhibiting the involvement of both neurons in the change.

References

[1] Minsky M. L. and Papert S. A. 1969. Perceptrons. Cambridge, MA: MIT Press.
[2] http://en.wikipedia.org/wiki/Delta_rule
[3] http://en.wikipedia.org/wiki/Neural_backpropagation
[4] Mehrotra. Elements of Artificial Neural Networks.
[5] O'Reilly C. R. Computational Explorations in Cognitive Neuroscience
[6] http://en.wikipedia.org/wiki/Backpropagation