Nirajan Koirala
CSC 9010_001

1. Summary

The paper describes a new method of repeatedly adjusting the weights of the connections in a neuron-like network in order to minimize the cost function which is the difference between the actual output vector and the desired output vector. When hidden layers are incorporated in a neural network, manually setting the activation energies and weights for the hidden layers is not feasible. Using this method, neural networks can be trained with input-output vector pairs and an internal structure of a network is developed automatically as appropriate weights for the units in the hidden layer is set to decrease the error rate. In the first pass, each of the units in the layers have their states determined using the input they received from the units in the previous layers. This value is the sum of each of the weight times the linear function of the outputs plus some bias unit which always has a value of 1. A sigmoid function is used to compute the output, y using the input, x. Using this process, we end up with some random states for output vector and in the next step we try to find a set of weights such that it'll ensure for each input vector, the error which is the sum of the squares of the difference between the actual and desired output is reduced. A method called gradient descent is used to minimize the error by computing the partial derivative of error with respect to each of the weights in the network. We start at the last layer of the network and propagate backwards using the derivatives. First, the partial derivative of error with respect to x which is the total input is computed and chain rule can be used repeatedly to find the partial derivative of the error with respect to each of the weights of the units. Finally, we can compute the required partial derivative of error with respect to y. This value is the sum of all the partial derivatives of error with respect to the input x times the individual weights. This method can be used recursively to compute the partial derivative of error with respect to y for all the units in the previous layers using the partial derivative of error with respect to y in the current layer. As we move across the middle layers of the network, we have two options: either change the weights after every input-output case or accumulate the partial derivatives of E with respect to w over all the input-output cases before changing the weights. The second approach is used in the paper and each weight is changed by an amount proportional to the accumulated partial derivative. This method can be improved further using an accelerated method in which the current gradient is used to modify the velocity of the point in the weight space instead of its position. Intermediate layers of a network can be used to detect symmetries in a network. One can start with small random weights to break these symmetries. One of the major drawbacks of propagating error backwards to find the minimum is that the function may get stuck in a local minima while trying to find the minimum cost function. But the author claims that getting stuck in a local minima is a rare phenomenon and this problem can be avoided by adding more connections. Extra dimensions are created in the weight space as we add more connections and these dimensions provide paths to avoid the barriers creating local minima in the lower dimensional subspaces. Hence, the current learning procedure is not similar to the learning that occurs in brains and we should be looking for more biologically similar learning methods.

The author assumes that the reader is already familiar with the basic principles and jargons of deep learning and mathematics. The intended audience of this paper are researchers, Computer Scientists, Mathematicians and specialists in the field of Machine Learning.

The main point made in this paper is that the weights of the units in a neural network can be adjusted using back-propagation method and this method is superior to other methods like perceptron-convergence procedure.

2. Strengths

This paper explains a new and simpler method of adjusting weights in the units of the hidden layer of a neural network using gradient descent and basic calculus techniques like partial derivatives. The mathematical formulas are properly described, and it helps the reader in understanding the major technique described in the paper, backpropagation algorithm. Very few references have been mentioned which further strengthens the novelty of the method described in the paper.

3. Weaknesses

The figures provided in the paper are not sufficient for a reader to visually understand the backpropagation algorithm in a neural network. Figure 3 in particular is hard to comprehend as the paper mentions subtle things like small dots in the square to find the correct answers.
Sufficient examples and description are not provided for one of the major drawbacks of this learning method, network getting stuck in the local minima. Author has mentioned that it is a rare phenomenon for a network to get stuck in the local minima but providing some statistics in this regard would have been more substantial.
Also, the reason behind introducing bias units in the networks is not mentioned in the paper.

4. Points of confusion

One of the confusions is regarding the symmetry of the network. The author mentions a method of breaking symmetry using small random weights but the real reason behind such practice is difficult to grasp only information provided in the paper.
Author has mentioned briefly about storing the information in two family trees using a figure but both the figure and the reason behind using such tree like structure is still confusing.
Also, there is a very brief mention of the equivalence between the layered and recurrent networks using a figure which is hard to grasp given the background information about recurrent network provided in the paper.

5. Discussion questions

a) How can one avoid local minima in a network and what is the probability of a network with sufficient connections getting stuck in a local minima?
b) What is a symmetry in the network and why is it important to detect/break it?
c) How backpropagation method compares with other methods like perceptron-convergence procedure in terms of accuracy levels, performance and other pertinent criterions?