A training algorithm for feedforward NN with biological plausibility

Bernadette Garner

CSSE

Monash University

Clayton Vic. 3168

Australia

Bmg@csse.monash.edu.au

Abstract

The dominant training algorithm for training artificial feed forward neural networks (ANN) is backpropagation. This algorithm has no biological plausibility incorporated into the training of the ANN. The new training algorithm proposed in this paper does not feed back error during the training process, but instead finds regions in its weight-space that satisfies the learning conditions that exists within the input data. This algorithm is based on the McCulloch-Pitt neuron model. This paper formalizes this idea and incorporates a strategy for the network to determine its own architecture, which imitates the behavior of biological neurons in the learning process.

Keywords: biological feedforward neural network training algorithm, constraints, and quantization.

Summary

The McCulloch-Pitt neuron model uses the step function as its transfer function. In other words, neurons either fire or not. Experiments have shown that neurons use the

step function as the transfer function (Beale and Jackson, 1992, Neural Computing, an introduction, IOP). Despite recent speculation that the biological neuron does not use the step function, no experimentation has demonstrated that this is not the case (http://news.bmn.com/news/story?day=011012&story=1, Carandini and Ferster, 2000, Membrane Potential and Firing Rate in Cat Vi, J. Neurosci, 20(1):470-484). The algorithm being discussed in this paper uses the step function.

This method learns by finding relationships between the neuron's incoming connection weights and the neuron's threshold to produce the required output for the input the neuron is to classify. The algorithm learns by modifying the connection weights by ranges of response, which can be likened to quantization.

This algorithm will add neurons into the network as required thus building its own architecture. When a single neuron cannot learn a particular input, one or more additional neurons are added to the ANN to enable the network to learn the input. This is similar to the biological brain, which can add connections between neurons in a number of ways (http://ucsdnews.ucsd.edu/newsrel/science/mccell.htm).

This neuron being used by this algorithm is defined as (Beale and Jackson, 1992, Neural Computing, an introduction, IOP), where i is the input vector, n the number of inputs into the neuron, and W is the weight vector associated with the incoming weights of the neuron. T is the threshold of the neuron and O is the neuron's output.

$$\text{net}_n = \mathbf{i}_n \,.\, \mathbf{W}_n = \sum_{j=1}^{n} i_j \; W_j$$

where $\mathbf{i} \in \{0, 1\}^n$ and $\mathbf{W} \in \mathbf{R}^n$.

$$O = \begin{cases} 1, & if \quad net_n \geq \quad T \\ 0, & otherwise \end{cases}$$

The relationships are constructed as constraints given by these definitions. The constraints define relationships between the weights associated with each neuron's input connection and the neuron's threshold. If the weighted summed input is < than the neurons threshold, then the neuron will not fire, and it will be ≥ T is the neuron is supposed to fire.

$$i_p.W \geq T \rightarrow 1$$

or

$$i_p.W < T \rightarrow 0$$

where p is the pattern that the neuron is currently learning.

In an example of a 2 input neuron, if the input is [1 1] and this causes the neuron to fire, then the constraint produced is $W_1 + W_2 \geq T$. During the neurons lifetime it is likely to have been subjected to many other patterns. If the neuron has also learnt that the input pattern [0 0] inhibits the firing the neuron then the neuron will have learnt $0 < T$.

A diagram of the weight space of this is seen in Figure 1 of what the neuron has learnt.

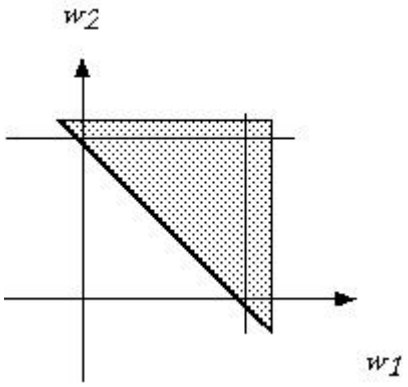

Figure 1. The weight space for a two-input neuron that has learnt $\{W_1 + W_2 \geq T, 0 < T\}$.

The neuron may then learn to classify more patterns. If the neuron has to learn [1 0] produces an output of 0, then the weight space in Figure 2 is produced.

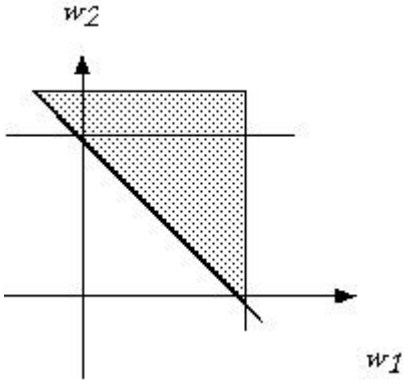

Figure 2. The weight-space for a two-input neuron that has learnt $\{W_1 + W_2 \geq T, 0 < T, W_1 < T\}$

The neuron can still learn either [0 1] fires or not. If it fires then the weight-space will remain unchanged but if it learns that the neuron doesn't fire then the weight space will be reduced to $\{W_1 + W_2 \geq T, 0 < T, W_1 < T, W_2 < T\}$. In this case the neuron has learnt to fire if both input 1 AND input 2 are present. If the neuron learns pattern [0 1] causes the neuron to fire, then the neuron has learnt that only input 2 is required to

make it fire. Freeman and Skapura (1992, Neural Networks, algorithms, applications, and programming techniques, Addison-Wesley) say that neurons conform to Boolean logic.

This algorithm learns connections extremely quickly, as it does not use iteration. It does not train the weights and thresholds as single numbers, if finds ranges of values that satisfy the training conditions. It is often desirable to know what the relationships are being formed in artificial neural networks. However there may be ways to determine what features individual neurons are forming by examining the connections between neurons, for instance, connections the different types of inhibitory and excitatory neurons (Szentagothi, 1983, http://ucsdnews.ucsd.edu/newsrel/science/mccell.htm).

The idea of strengthening active connections is called Hebbian learning (Beale et al, 1992). One adaptation of Hebbian learning could cause neurons to forget what they have already learnt. If a neuron has learnt something in the past, it is possible that this may be forgotten if the pattern is not refreshed. This will allow the neuron to be retrained to respond to other stimuli. This is called plasticity. While there is some correspondence between Hebbian learning and this algorithm, strengthening active connections too much could stop the neuron acting as it is supposed to. For instance in the above example if the neuron has been taught to distinguish input 1 AND input 2, then if the connections of W_1 and W_2 become strengthened to the point that either $W_1 \geq T$ or $W_2 \geq T$ then the neuron may fire if either input is present. However, this algorithm does not attempt to explain other features such as plasticity.

Also outside the scope of this discussion is the nature of the spiking frequencies of neurons. There are many phenomena occurring in the biological brain. Together all the phenomena combines in some way to produce all the features they exhibit. Nor does this algorithm readily explain the 'sorting' of the neuron classifiers which is seen in brain map response patterns. Kohonen developed his famous algorithm to be able to simulate the sorting of neurons into response patterns. However Kohonen's algorithm can be very slow. It is likely that sorting is achieved biologically by a combination of Hebbian learning and matching response properties. This algorithm is an attempt to demonstrate only a small part of their behavior.

With respect to ANN, this algorithm is not subjected to the local minima problem that ANN exhibit since it does not feed back error or rely on gradient descent. It also produces relationships between the weights and thresholds that can be easily interpreted and adds neurons as required. It is not iterative so it learns fast.