Training Feedforward Neural Networks Using Genetic Algorithms

Christian Højer Skjellerup, Erik Belhage
rjk679, srz936

March 2018

1 Introduction

We summarise the article "Training Feedforward Neural Networks Using Genetic Algorithms" by David J. Montana and Lawrence Davis of BBN System and Technologies Corp. It lays out a method for training feedforward neural networks using genetic algorithms. Genetic algorithms is a type of optimisation algorithms inspired by evolution.

2 Genetic Algorithms

To understand this better we should consider how evolution functions: natural selection. This means that in a genetic algorithm we would expect to find a means of reproduction as well as some method of selection. We will return to these later, but first we must consider representation.

Representation is the first problem that must be solved when implementing a genetic algorithm. Generally, we want to represent the solution as some variant of a string. This representation means that instead of moving the solution about with, for example, a random walk, the genetic algorithm uses mutation, cloning, crossover, breeding, etc, on the multiple solutions, called chromosomes as within biology.

An obvious advantage of this version of optimisation is that it covers the entirety of the parameter space, and cannot get stuck as mutation and crossover operations can always move a solution to a very different spot in the parameter space. At the same time this means that there are no guarantees that a solution will converge as it moves almost randomly about the parameter space. This means that genetic algorithms require large ensemble sizes to function properly. By continuously selecting and eliminating the worst solutions the ensemble will move towards the best solution.

3 Feedforward Neural Networks

The feedforward neural network is a classification algorithm. It functions by imposing different cuts and then weighing then in relation to each other. In general we have an input layer and output layer.

The input layer takes whatever datapoints we have as well as some function of the datapoints that can be easily computed, for example the sine-function of some variable in the datapoints. The output layer returns whatever classification was assigned to the datapoints. Between the in- and output layers are the hidden layers which are made up of nodes. The nodes are equivalent

to different cuts across the data and the final classification is given by the weighted sum of the classification given by each node.

The weights of the initial neural network is likely wrong and must be optimized to minimize the error function via the genetic algorithm.

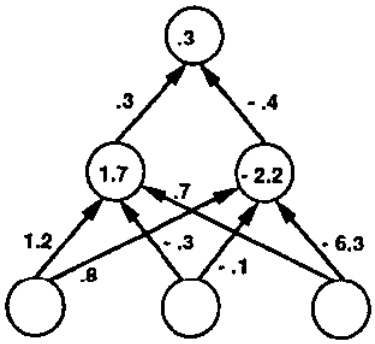

Figure 1: Picture of a chromosome

4 In combination

Normally the neural network is trained using a method known as backpropagation. The values that describe a fixed neural network can be written as a sequence, and the error function makes for an obvious selection function. The paper tests several different variants of the standard genetic algorithm to see if specific functions tailormade for training neural networks offer any special advantages. These algorithms can be divided into two groups: variants on ordinary genetic algorithm methods and backpropagation-inspired methods.

In the article we have specific crossover and mutation functions, which either switches or mutates individual weights, or takes all the weights describing a given node and mutates the lot or uses them for crossover. There is a backpropagation based method, called hillclimb, which calculates the normalized neural network gradient of the parent and child solution, and then increases or decreases the change made between them, in order to move more quickly towards a nearby minimum.

The paper then presents the results of a series of experiments with these different algorithms as follows. The first experiment showed that a function biased by the previous value of a mutated weight performed better than an unbiased function, and that a biased function that mutated a whole node with its weights outperformed both of those. The second experiment showed that there is no real difference between different crossover methods, ie. between nodes, weight or features. The third experiment compared a method that mutated the weakest nodes, with a genetic algorithm without special methods and found no improvement. The fourth experiment tested the hillclimb function, and found that while it can give rapid improvement, it also is susceptible to getting stuck in local minima. Finally, the fifth experiment demonstrated that the genetic algorithm was superior to backpropagation.

While the paper uses a specific dataset consisting of sonar data, there is no reason that the method could not be expanded to other types of neural networks. In the conclusion of the paper suggestions are made for future work to expand into neural networks that do not have a training dataset of fixed size.