Assignment 2

Gary Saavedra

gsaavedra3@gatech.edu

Introduction

This assignment was done using Pushkar’s ABAGAIL code.

Training a Neural Net

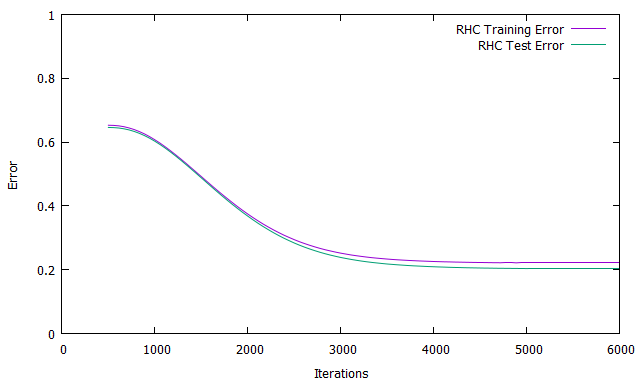
For the neural net 27 input nodes were used at the input layer, 7 nodes at the hidden layer, and 1 node at the output layer. Seven nodes were used at the hidden layer because that is the number at which the original neural net performed best. The neural net was trained using Random Hill Climbing (RHC), Simulated Annealing (SA), and a Genetic Algorithm (GA)(which one, parameters?) instead of BackPropagation. The training error is used as a fitness function (how to word this, since we are trying to optimize fitness not minimize cost). Sum of squared errors is used as the error measurement.

Random Hill Climbing

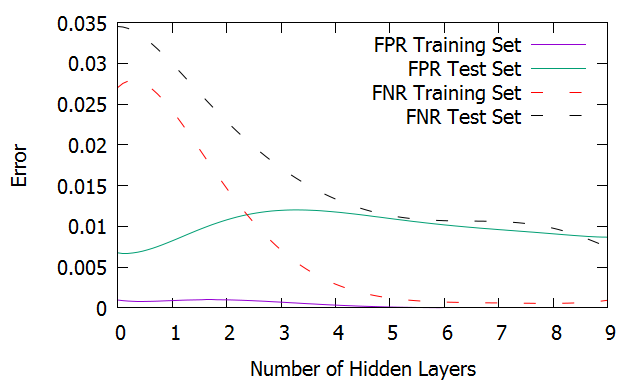
Figure 1 shows a graph of error vs iterations for a RHC algorithm. In the figure we can that both the training error and testing error start at about 65% for 1000 iterations. After about 4000 iterations the error bottoms out at 21%. The RHC algorithm is not able to choose a set of weights that improve the error rate beyond 21%. The RHC is most likely not doing well because it is getting stuck in a local optima. The error stays at .221 even after 1,000,000 iterations. If we look at the error from Assignment 1 using this same network we get a sum of squared errors of 16%.

There are a few reasons we may be getting an error higher than our original. The global optima may be very narrow and hard to reach. This is known as a basin of attraction. Even though RHC has random restarts and therefore many chances to converge to the global optimum, the large basin of attraction makes it unlikely the global optimum will ever be found. The algorithms may also not generalize a continuous space perfectly.

Strange part of the graph. The training error is lower than the testing error. However, this is not a major deal since they are very close together that means they are within the range of variance for each other. The training and testing error closely matches in the graph.



*Figure 1: Error vs Iterations for RHC*



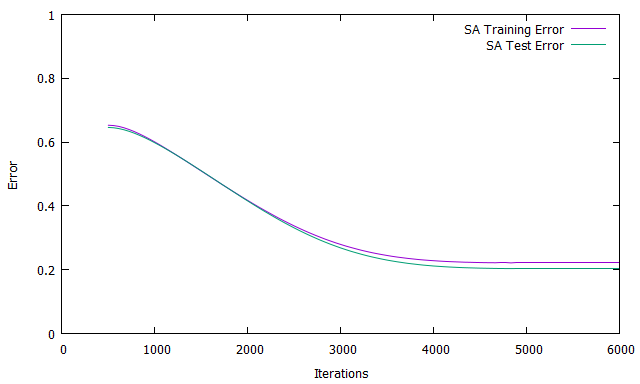
*Figure 2: Error rate for neural net using BackPropogation*

Simulated Annealing

For the SA algorithm the temperature starts at 10^11 and steps down 5% every iteration. The error stays at 21% even after 1,000,000 iterations.

Figure 3 shows the error vs number of iterations for the SA algorithm. This graph shows a similar trend to the RHC graph. At 1000 iterations the error is about 60%. After 6000 iterations the error has dropped to 21% error similar to the RHC algorithm. Why doesn’t SA converge after fewer iterations than RHC. It seems like it would since it’s less likely to get stuck at local optima.

It is possible there are many local optima with a weight configuration that gives approximately 21% error. These many local optima would dominate the probability function that decides which optima is chosen. This is probably not the case though since every algorithm is bottoming out at 21%. It is likely that this is the global optima and there is something else hindering the algorithms from reaching 16%.

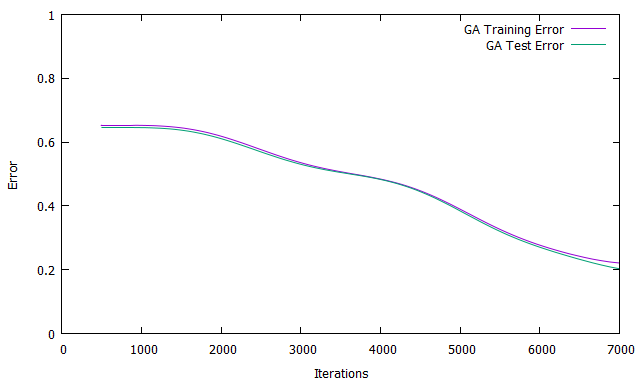


*Figure 3: Error vs Iterations for SA*

Genetic Algorithm

For the GA algorithm the parameters are as follows. The initial population size is 200. 100 members are chosen to mate and 10 members are chosen for random mutation. Is crossover used and how are the local bits chosen?

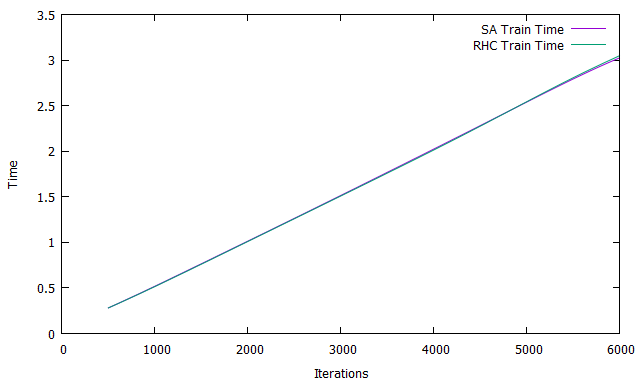
The error vs training iterations plot is shown in Figure 4. This plot differs significantly from the RHC and SA algorithms. The error also starts out at 65% but takes much longer to converge to the optimum of 21% error. It does not reach 21% until about 7000 iterations. We also see a bit of error increase around 4000-5000 iterations. This means even though the GA has had more training opportunities it actually had a larger error. This may be due to the randomness of the algorithm itself. The randomness of mutations and matings may cause this.



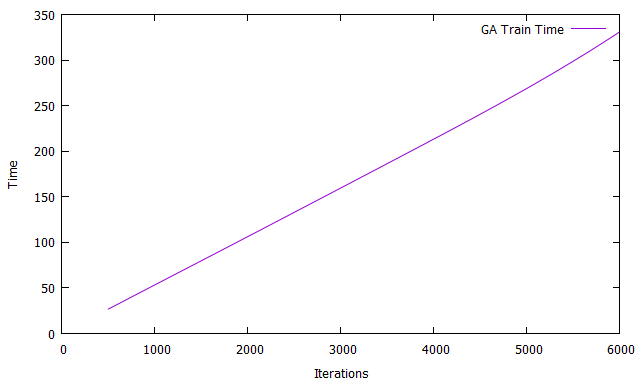
*Figure 4: Error vs Iterations for GA*

**Performance Comparison**

There are two figures showing the training time versus the number of iterations used for each algorithm. Figure 5 shows the training times for SA and RHC. Figure 6 shows the training times for the GA. As the figures show there is significant difference in training times between the GA and the other algorithms. The RHC and SA take only about 3 seconds at 6000 iterations. Their lines are overlapping. The GA takes about 325 seconds at 6000 iterations, an increase by a factor of 10. Both graphs show a linear increase in training times.



*Figure 5: Training time for SA and RHC in seconds*



*Figure 6: Training Time for GA in seconds*