Abstract

Random Hill Climbing (RHC), Simulated Annealing (SA), Genetic Algorithms (GA), and MIMIC are Optimization algorithms that are used to compute the optimal solution of a problem. Each has its advantages and disadvantages. This paper attempts to highlight the characteristics of each of these algorithms through implementing them on various types of problems and comparing their results. The first part of the paper utilizes RHC, SA and GA to compute the weights of a neural network for predicting the classification of breast cancer being benign or malignant. The second part identifies three optimization problems, namely, four peaks, knapsack and k-colors to compare and contrast the performance of RHC, SA, GA and MIMIC algorithm.

Part 1: Optimizing the Weights of a Neural Network

Neural Network Dataset

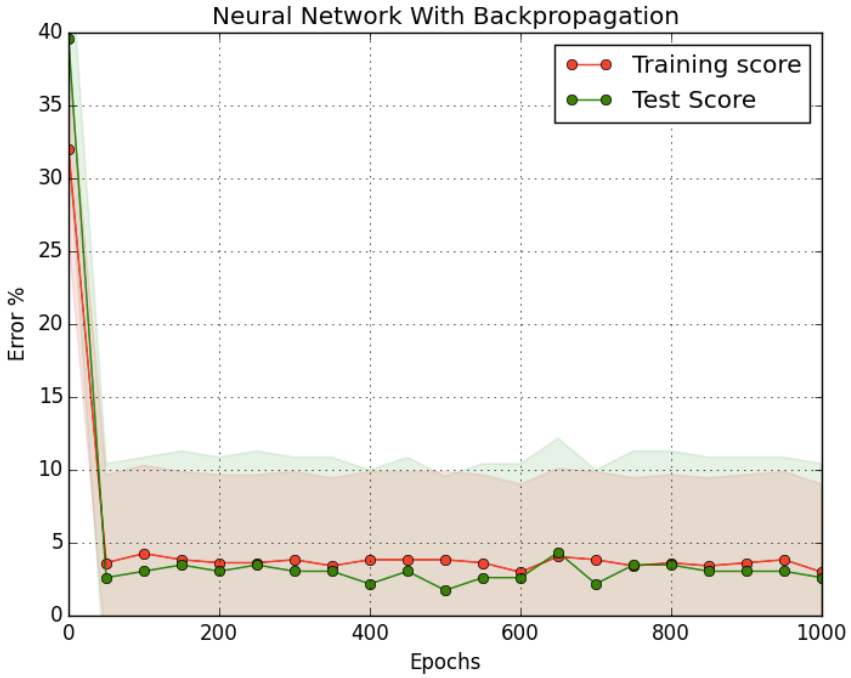
The Wisconsin breast cancer data obtained from UCI Machine Learning Repository contains 699 samples with 9 input attributes. The samples are classified as either benign or malignant. In addition there were 16 missing values within the Bare Nuclei attribute.

In preparing the data the sample code number was removed, as it is just an ID number that doesn't have any relationship with the cancer data. Next the 16 missing values of Bare Nuclei were populated using a mean strategy, where the missing values were filled with the mean of all the other data in that column, which turned out to be 4.

Breast Cancer Learning Algorithm & Data Analysis

Neural Network Using Backpropagation

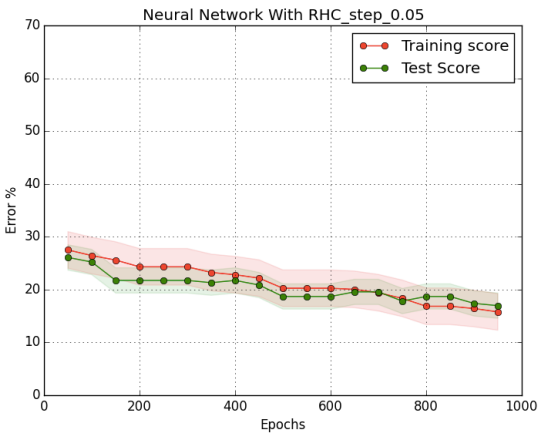
The neural network was constructed using 9 input nodes, 1 hidden layer with 2 nodes, and 2 output nodes. The reason for creating 2 output nodes is to accommodate for the two classes (benign and malignant). After training the neural net with backpropagation I was achieve the best result at 600 epochs. At 600 epochs the training error was 2.99% and the test data error was 2.61%. As you can see in the graph below this wasn't the best result for the test error. A lower test error of 1.74% occurred at 500 epochs with a training error of 3.84%. This is where we have some trade of to make between the training and testing error in deciding which weights to use.

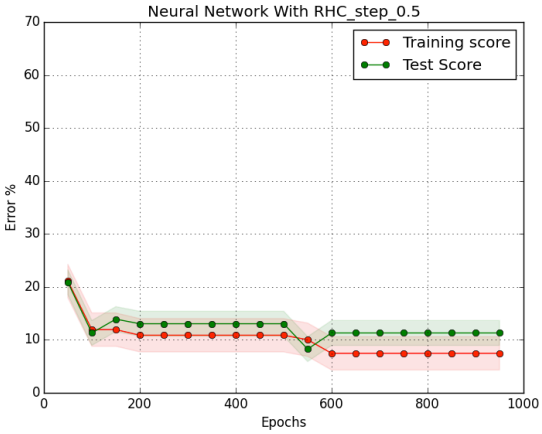


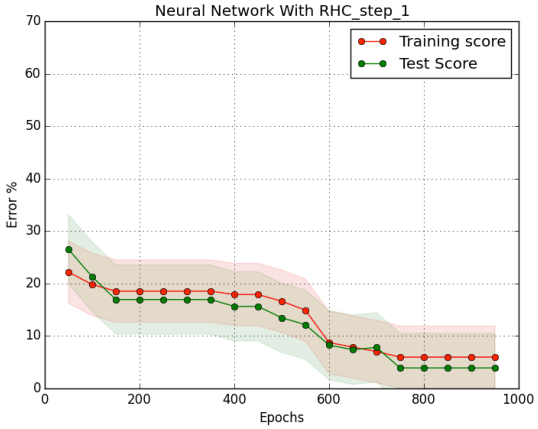
Neural Network Using Random Hill Climbing

Random hill climbing works by searching through the fitness function for the best-fit solution. The algorithm attempts to find the best fit by starting at a random location and evaluates its neighbors to find the best next move. As the name suggests it moves in a hill climbing approach to find the local optima in search for the global optima. For our neural network the fit function was the error percentage as a function of the neural net weights. The algorithm that I used randomly steps the weight to a neighbor and then computes the error percentage on the training error for the neighbors. It then moves to the neighbor that produces the lowest error.

The below three graphs shows the performance of the training and test error across different step size for the hill climbing whiling keeping epoch constant to 1000 for all three. As expected the training error only climbs downwards or stays flat because the algorithm only moves to a neighbor only if the neighbor presents a lower error percentage. Additionally you can see that the graph for step size 0.5 and 1 plateaus at the end. This is potentially for two reasons, either the program got stock at a local minimum or it hit a very long plateau. The former is more reasonable given this algorithm can potentially get stuck at a local minimum. The best error percentage found was 5.97% at around 750 epoch with step size 1. This not as good as backpropagation which did must better and converged to a better error with lower iteration.







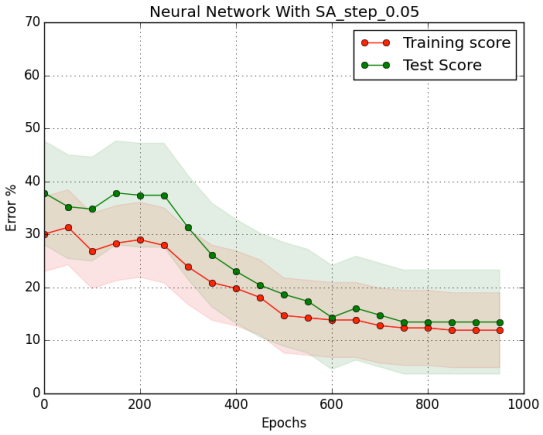
Neural Network Using Simulated Annealing

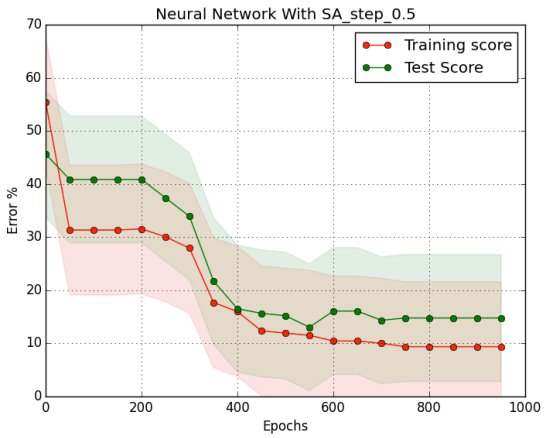
Simulated Annealing is a very similar algorithm as Random Hill Climbing with one difference. Where as RHC only moved to the best neighbor SA sometimes moved to a worst performing neighbor in the hopes that it would step through any local minimum and find the basin that would lead to the global minimum.

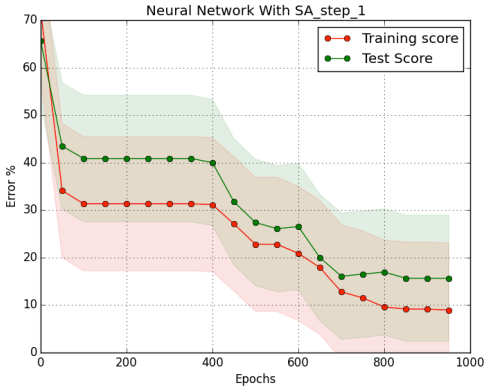
In my implementation the algorithm always moved to the better neighbor. However, it moved to a worst neighbor with a probability of 1/(1+exp(neigh_error – current_error)/T). The T variable in this formula represented the temperature. With high T the algorithm more freely moved to its neighbors even if it was worse. This allowed it to overcome local minimum. As iteration progressed the temperature was slowly cooled at a rate of .95*T. As temperature reached closed to zero the algorithm behaved more like a RHC.

The below three graphs shows the performance of Simulated Annealing for varying step size while keeping epoch constant to 1000. As expected you can see that the training error doesn't always steer downwards. Occasionally the error gets higher when the algorithm takes a step toward a worse performing neighbor when the temperature is high. However around epoch 400 the temperature reaches closure to zero as such the training error only descends. The best performance again was

achieved with step size 1 toward the end of the iteration. The lowest error was 8.96% for training data and 15.65% for test data. Though we expected SA to overcome local minimum the execution of the algorithm probably got unlucky where the temperature cooled to near zero when we weren't the basin of global minimum. As such it was not able to descend to the best minimum or do any better than RHC.



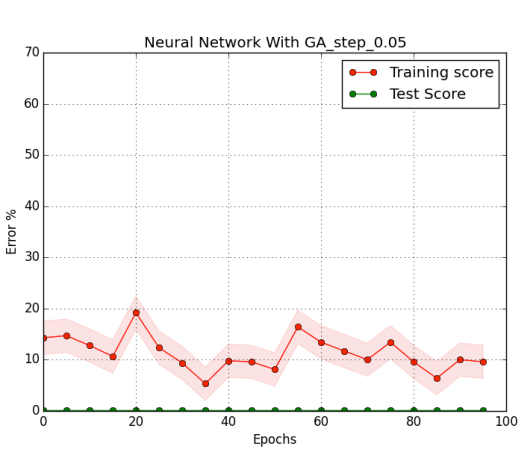


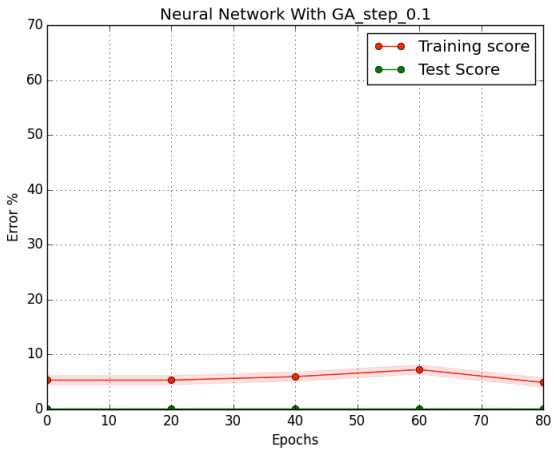


Neural Network Using Genetic Algorithm

Genetic Algorithm (GA) is an alternative approach to optimization that takes the approach of natures selection of the best fit. GA starts with a population size and then creates new children population for the original population. The idea behind this approach is that the combination of multiple solutions may provide the better solutions. As in nature two parents with high intellect are likely to produce offspring that have high intellect as well, the GA algorithm attempts to produce children from two parents by taking part of the parameters from one solution and the other part of a different parameter. This approach is known as cross over. The cross over can happen in one or multiple location. Alternative method of generating children is taking one parent and mutating (altering) some of its parameter values to produce a new solution of parameters.

In my algorithm the initial population size is 100 that is randomly generated. The algorithm then keeps 20% of the top performing population in each of its iteration and then either mutates or does cross over with the remaining population. The mutation is done 20% of the time and cross over is done remainder of the time. The below graph shows the performance of the neural network using GA to train the data. The graphs show how training error varies as the algorithm tries to evaluate the population that it creates through mutation or cross over. Compared to RHC and SA the Genetic Algorithm reaches low error points very quickly. The graphs show only a 100 epoch where we already achieve error rates of approximately 5% within a epoch of 35. However, I was surprised to find that the error rate for test data is steady at .21%.





Part 2: Comparing Optimization Algorithm Across Multiple Problems

For the second part of the paper we analyze the performance of RHC, SA, GA, and MIMIC on three optimization problems, namely, knapsack, four peaks, and k-colors problem. Each problem will utilize to highlight the strength of SA, GA, and MIMIC with RHC as the benchmark.

These problems are interesting problems as they have many real world applications. Knapsack for example can be used in investment portfolio problems to analyze the stocks/other investments that will maximize the value of the portfolio with the total funds you have. K-Colors problem can be applied to pattern matching, graph coloring, or scheduling of jobs in a large infrastructure to avoid similar jobs being assigned to the same slot.

Aminul Khan
Machine Learning

Assignment # 2
March 15, 2015

Knapsack

The knapsack problem analyzed takes five items with a specific weight and value and tries to identify the maximum value that can fit into a knapsack without overcoming a the weight limit of the knapsack. The items and their value are described below. The maximum weight the knapsack can hold is 15 pounds. Additional constrain was put as being able to only take 1 item of each. This will allow us to have a bit string for the parameters.

Item	Weight (pounds)	Value ($)
1	12	4
2	1	2
3	2	2
4	4	10
5	1	1

We expect the Genetic Algorithm to do better than Simulated Annealing and MIMIC in the knapsack problem. The knapsack problem presents several local maxima, which can be overcome by GA through the mutation and crossovers of various solutions. Since GA tries to mix best parts of various parents to create the best children I believe it will reach the global maximum much faster.

Due to lack of time was not able to implement the actual algorithm.

Four Peaks

The four peaks problem has four peaks as the name suggest, two of which are local maxima and the other two global maxima. The problem can be setup as described by the below functions from the paper "MIMIC: Finding Optima by Estimating Probability densities". The two local maxima are achieved with all 1's or all 0's. The two maxima are achieved when the number of 1s is T+1 followed by all 1s or when there are T+1 0's preceded by all 1's.

$$f(\vec{X}, T) = \max[tail(0, \vec{X}), head(1, \vec{X})] + R(\vec{X}, T)$$

$$tail(b, \vec{X}) = \text{number of trailing } b\text{'s in } \vec{X}$$
$$head(b, \vec{X}) = \text{number of leading } b\text{'s in } \vec{X}$$
$$R(\vec{X}, T) = \begin{cases} N & \text{if } tail(0, \vec{X}) > T \text{ and } head(1, \vec{X}) > T \\ 0 & \text{otherwise} \end{cases}$$

In the problem I have setup T to be N/5 where N is the size of the population. I expect Simulated Annealing to do better with the four peaks problem than GA and MIMIC. SA should be able to overcome the two local maxima when temperature is high. GA on the other hand will not do well because the order of the bit string

matters and crossover or mutation can create solution that may not necessarily perform very well.

Due to lack of time was not able to implement the actual algorithm.

Four Colors

The K-Colors problem attempts to assigns K colors to N nodes where no two nodes having the same color are connected by the edges of the node. Because there is a structural element to this problem MIMIC will tend to do very well in this problem.

Due to lack of time was not able to implement the actual algorithm.

Bibliography

1. O. L. Mangasarian and W. H. Wolberg: "Cancer diagnosis via linear programming", SIAM News, Volume 23, Number 5, September 1990, pp 1 & 18.
2. Stephan Chalup and Frederic Maire: “Design Issues In Hill Climbing For Neural Network Training”.
3. Jeremy S. De Bonet, Charles L. Isbell, Jr. and Paul Viola: “MIMIC: Finding Optima by Estimating Probability Densities”.