Vinodh Kumar Sunkara, Microsoft Redmond WA

# Assignment 2: Randomized Optimization

There are two parts of this assignment. First part is to find the weights for neural networks using Randomized hill climbing, Simulated Annealing and Genetic Algorithms compared to Back propogation. Second part is to come up with three optimization problem domains (fitness functions) on discrete valued parameter spaces. I've selected KnapSack, PairedFlipFlop and Travelling Salesman Problem as fitness functions to optimize. I've used ABAGAIL for all the algorithm implementations. ABAGAIL contains a number of interconnected java packages that implement machine learning and artificial intelligence algorithms.

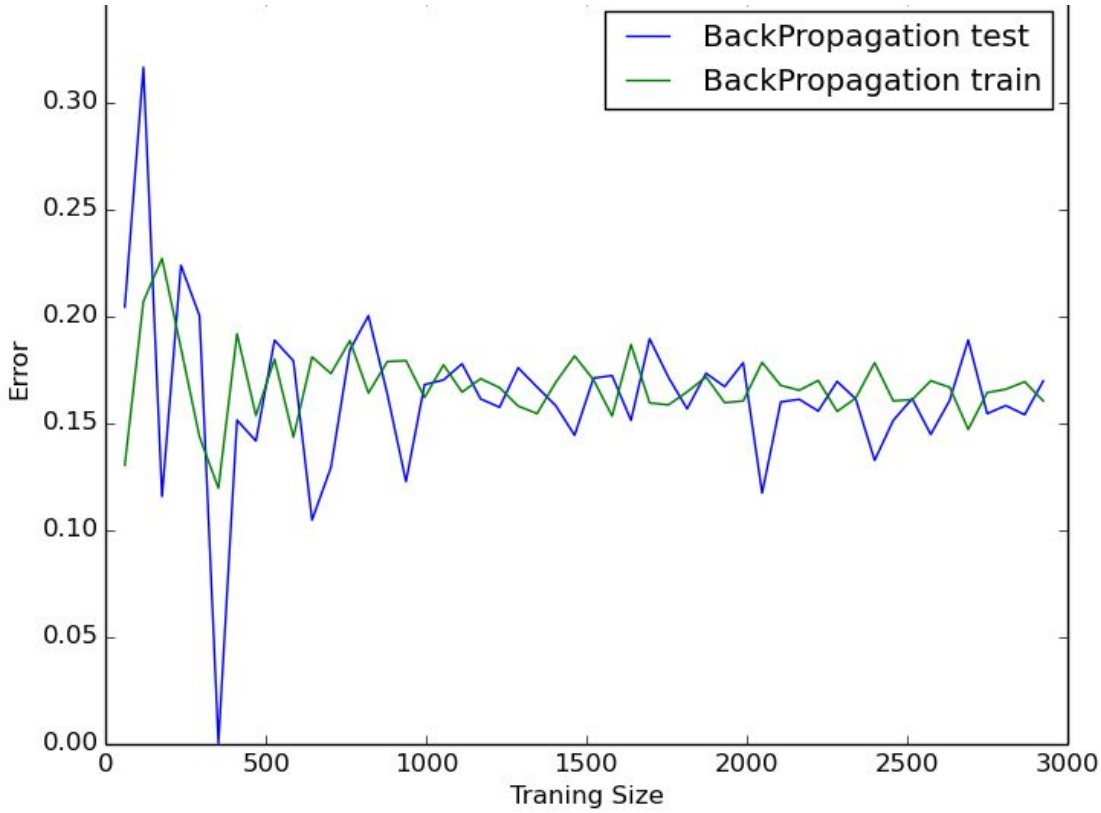
Abalone: It's a dataset taken from a field survey of abalone, a shelled sea creature. The task is to predict the age of abalone given various physical statistics. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope -- a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem. Abalone have gone through a period of significant decline in recent years due to both natural and unnatural environmental pressures (reintroduction of sea otters, loss of habitat). If techniques to determine age could be used that did not require killing the abalone in order to sample the population, this would be beneficial. The abalone dataset is of interest in many ways because it is so resistant to good performance under machine learning techniques. None of the supervised learning techniques that are presented here achieve particularly good results.

Problem: To determine whether Whether an abalone is small, good and old. The problem is a slight modified version for predicting age of abalone from physical measurements. Given the different attributes of abalone, the objective is to label if abalone is good (age >=5 years but <20 yrs), small (age<5 years) or old (age>=20 years). Relation between age and the abalone rings is given by: age = (rings + 1.5) years.

BackPropagation:

We mention here the results of backpropagation for comparison. Earlier implementation was done using Weka tool but for an apples to apples comparison of different methods, we implement again using ABAGAIL library. This neural network is trained using ConvergenceTrainer with 500 training iterations and error threshold of 1E-5. Blue curve below shows the backpropagation test and green curve shows the backpropagation train error rate plotted over training size - learning curve.

Results: Train Error 0.17 i.e. 17% Inaccurate classifications
Test Error 0.18 i.e. 18% Inaccurate classifications



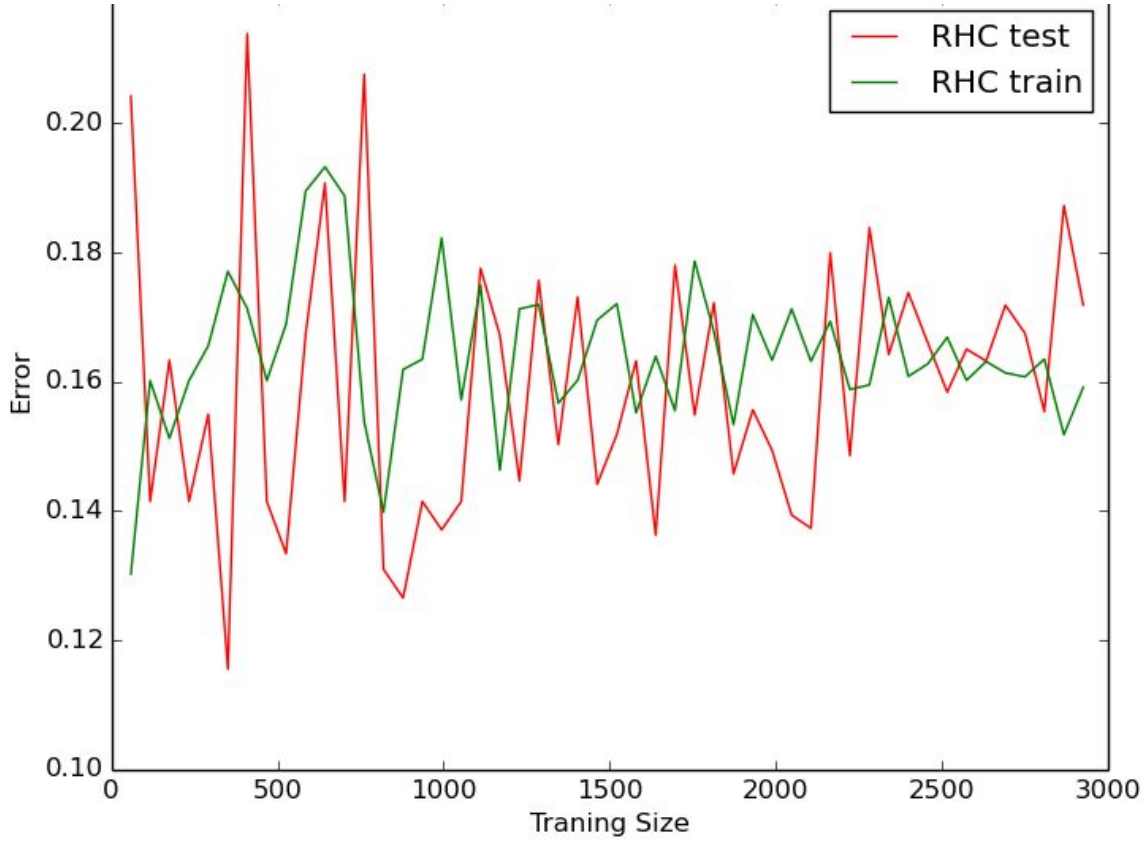

Randomized Hill Climbing

We sample p points randomly in the neighborhood of the currently best solution and determine the best solution of n samples points. If it's better than the current solution, we make it the new current solution and continue with the search, otherwise terminate search returning current solution. Using Randomized Hill learning, neural network is trained using 500 training iterations with sum of squared error measure. Algorithm was run over varying size of samples from full dataset. The test and train errors are recorded. When data is small, the difference between test and train errors is high. Both seem to converge when we increase the size of dataset.

Results:

Train Error: 0.16 (16% Inaccurate classifications)

Test Error: 0.17 (17% Inaccurate classifications)



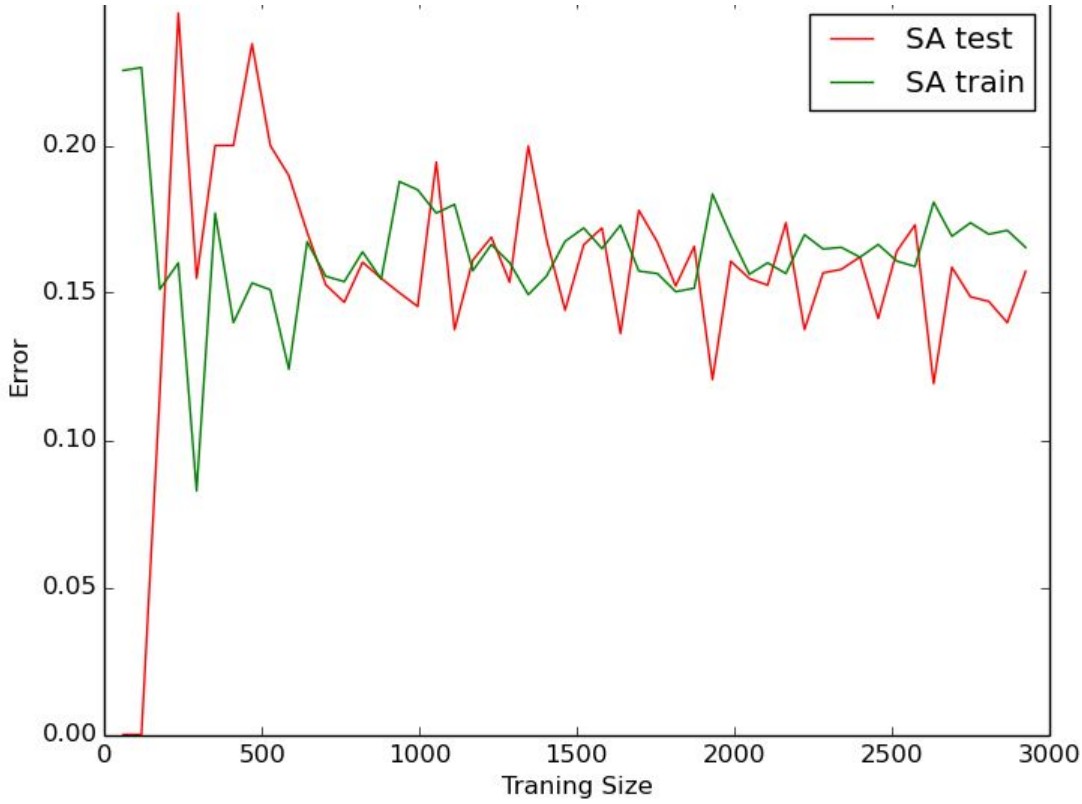

Simulated Annealing:

SA models the physical process of heating a material and then slowly lowering the temperature to decrease defects, thus minimizing the system's energy. At each iteration of the simulated annealing algorithm, a new point is randomly generated. The distance of new point from current point or the extent of search depends on probability distribution with a scale proportional to temperature. The algorithm accepts all points that lower the objective, but also with certain probability, the points that raise the objective. By accepting the points that raise the objective, the algorithm avoids getting trapped in local minima in early iterations and able to explore globally for better solutions. Simulated Annealing is initialized on abalone data sets with a starting temperature of 1E11 and a cooling exponent 0.7. Using SA, the neural network is trained using 500 training iterations with sum of squared error measure.

Results:

Train error is 0.16 (16% Inaccurate classifications)

Test error is 0.17 (17% Inaccurate classifications)



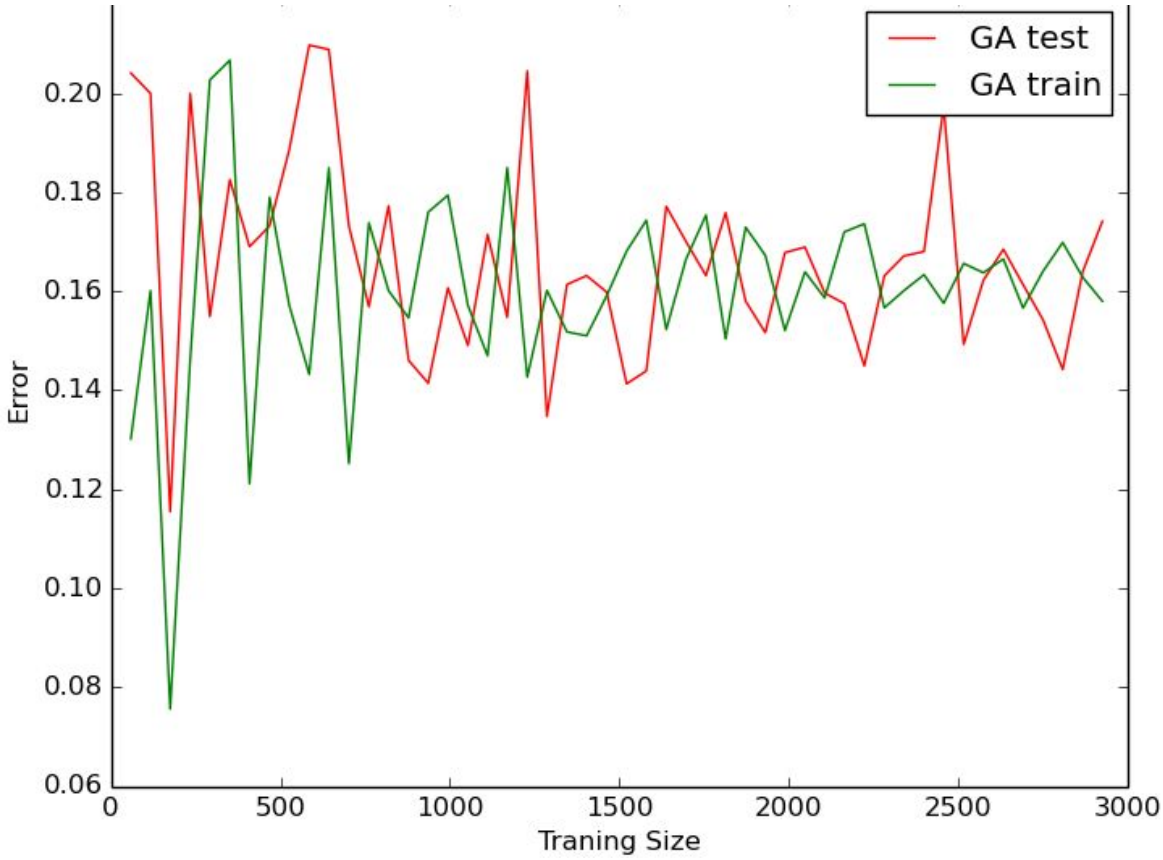

Genetic Algorithm:

It's based on a natural selection process that mimics biological evolution. The algorithms repeatedly modifies a population of individual solutions. At each step, the genetic algorithm randomly selects individuals from current population and uses them as parents to produce the children for the next generation. Over successive generations, the population "evolves" towards an optimal solution. Using genetic algorithms, I trained the neural network using 500 iterations with sum of squared error measure. Following other parameters like mutation size 10, mating size 100 and population size 200 are also provided.

Results:

Test error is 0.17 (17% Inaccurate classifications)

Train error is 0.16 (16% Inaccurate classifications)




Overall Analysis:

All algorithms converged almost to same (In) accuracy 17-18% on test data. All algorithms converged faster than backpropagation. Randomized hill climbing and simulated annealing took almost the same time to converge, but Genetic algorithm is the worst on time taken to converge with an increase in data size.

**Optimization domains using GA, SA, MIMIC:**

*PairedFlipFlop:*

We're trying to maximize number of bit pairs that are same and neighbor of these pairs are complement. Eg: "001100", "100110", "11001100110011", etc.

```
N -> 10
       RHC: 7.0000 in        201 function evaulations in     2 ms
        SA: 9.0000 in        201 function evaulations in     2 ms
        GA: 8.0000 in       4020 function evaulations in    50 ms
     MIMIC: 9.0000 in      10000 function evaulations in   180 ms
N -> 50
       RHC: 34.0000 in        201 function evaulations in     0 ms
        SA: 42.0000 in        201 function evaulations in     0 ms
        GA: 35.0000 in       4020 function evaulations in     5 ms
     MIMIC: 42.0000 in      10000 function evaulations in   249 ms
N -> 200
```

```
        RHC: 130.0000 in         201 function evaulations in      0
ms
         SA: 143.0000 in         201 function evaulations in      0
ms
         GA: 111.0000 in        4020 function evaulations in     19
ms
      MIMIC: 169.0000 in       10000 function evaulations in   2268
ms
```

Analysis: Winners are MIMIC and SA. When the size of data increase, MIMIC benefits are more apparent while the fitness function complexity increases.

*Travelling Salesman Problem:*

It's an NP complete problem aiming to find a hamiltonian cycle of shortest length on a graph. So, the fitness function should be distance minimization. We can modify the definition to adapt in maximization strategy. Fitness = maximize (1/distance).

```
N -> 10
       RHC: 0.3156 in     200001 function evaulations in    55 ms
        SA: 0.3131 in     200001 function evaulations in   220 ms
        GA: 0.3235 in     150200 function evaulations in   208 ms
     MIMIC: 0.3037 in     200000 function evaulations in   762 ms
N -> 50
       RHC: 0.1195 in     200001 function evaulations in   116 ms
        SA: 0.1158 in     200001 function evaulations in   307 ms
        GA: 0.1341 in     150200 function evaulations in   486 ms
     MIMIC: 0.0775 in     200000 function evaulations in 17979 ms
N -> 100
       RHC: 0.0718 in     200001 function evaulations in   225 ms
        SA: 0.0812 in     200001 function evaulations in   427 ms
        GA: 0.1121 in     150200 function evaulations in   956 ms
     MIMIC: 0.0447 in     200000 function evaulations in 197245 ms
N -> 200
       RHC: 0.0429 in     200001 function evaulations in   771 ms
        SA: 0.0465 in     200001 function evaulations in  1230 ms
        GA: 0.0639 in     150200 function evaulations in  2829 ms
      MIMIC: 0.0160 in      200000 function evaulations in 2691301
ms
```

Analysis: GA is the winner since it has maximum in all cases, and therefore the reciprocal of it would give the shortest path.

*KnapSack problem:*

Using dynamic programming, we can solve it in polynomial time for decent size but still an NP problem if input is large. Fitness function is to maximize total value of items

constrained by the sack size. Fitness function returns maximum value items if the total volume is less than or equal to sack capacity. Otherwise, it returns a very small value.

Results:

```
Knapsack Volume: 700.0
RHC: 863.3166570946979
SA: 887.1647284717823
GA: 966.452114497052
MIMIC: 904.4153918829657
```

Analysis: Winner is GA here and it's kind of apparent since knapsack is a greedy problem to maximum profit and GA fits well. MIMIC is second best here.

Conclusion:

Simulated Annealing and Randomized hill climbing work to a moderate degree. MIMIC works best most of the time but it takes too much time that it's not preferred for many problems. Although little inaccurate, GA gives similar results as MIMIC at faster rate.

References:

Genetic Algorithm: https://www.mathworks.com/discovery/genetic-algorithm.html
Simulated Annealing: https://www.mathworks.com/discovery/simulated-annealing.html

# THANK YOU