Assignment 2: Randomized Optimization

Shaikh Shamid

March 15, 2015

Abstract

Randomized Optimization is a popular method of obtaining the global maximum of the fitness function for a given problem. The current analysis will focus on four common algorithms for optimization: Randomized Hill Climbing (RHC), Simulated Annealing (SA) [6], Generic Genetic Algorithm (GA) [5] ,MIMIC [3], I will use these methods on a multi-class classification problem which was analyzed previously in Assignment 1, where the classification task was to predict the age of abalone from physical measurements. A description of the dataset can be found in the previous assignment, see [2]. I will also consider three interesting optimization problems which will highlight the advantages of GA, SA and MIMIC. This analysis will provide comparison among those three algorithms including some built-in drawbacks that the algorithms may have.

1 Introduction

Randomized Optimization algorithms are used to obtain the global maximum of a non-differentiable fitness function for a problem using random search. It worth mentioning that these types of problem cannot be solved using derivatives. In this analysis, I will apply these three algorithms to generate the weights of a neural network and compare results with the previous assignment, see [2]. Furthermore, I will choose three interesting optimization problems that demonstrate the advantages of SA, GA, and MIMIC.

2 Tools Used and Methodology

I will use the ABAGAIL implementation of RHC, SA, GA, and MIMIC. For the optimization problems I an going to choose three problems: the Traveling Salesman Problem (TSP), the Knapsack Problem, and the Continuous Peaks Problem. These implementations are also available in ABAGAIL.

2.1 Abalone Dataset Overview

The Abalone Dataset has 10 attributes. The only nominal attribute is the sex attribute. As a preprocessing step in my previous assignment, I normalized this feature as: (1,0,0) for male, (0,1,0) for female and (0,0,1) for infants. The class variable is how many rings the abalones have, which is classified as three different age groups depending on the number of rings: 0-8 as young, 9-12 as adult and greater than 12 as old. So this problem becomes a three class classification problem.

2.2 Randomized Optimization Algorithms

In this section I will briefly introduce those four optimization algorithms. I will discuss more it later in the context of optimization problems that I have chosen.

2.2.1 Randomized Hill Climbing (RHC)

Randomized Hill Climbing is a searching algorithm that searches for the maximum of the fitness function by randomly picking a point and moving toward the nearest neighbor, until it reaches a maximum. Due to the random nature of the algorithm, this maximum can only be guaranteed to be local, and therefore the algorithm continues to randomize its original location in order to increase the probability that the selected local maximum is actually closer to the global maximum.

2.2.2 Simulated Annealing (SA)

Simulated Annealing is an algorithm adopted from the physical fact of the slow cooling process of metals. The actual algorithm uses boltzmann classical probability distribution to decide in what direction it will search towards in order to find the global maximum. It is important to note that the algorithm

often chooses paths that are initially worse solutions than the current point, in order to increase the search space which the algorithm covers. By doing that it increases the probability of finding the global maximum. As its name suggest, the probability of selecting the wrong path decreases slowly over time.

2.2.3 Genetic Algorithms (GA)

Genetic Algorithm is an evolutionary algorithm. These algorithm selects population and iterates over the population for a certain number of generations. For every generation, parts of the population are eliminated if they are not important, which is determined by a fitness function. This function selects the best population set as it continues to evolve. Furthermore, attributes from different parts of the population can mutate to continue to create better solutions until the algorithm converges. The main drawback with this algorithm is that it assumes that the global maximum is contained within the original population, which must not be the case always.

2.2.4 MIMIC

MIMIC learns from prior iterations when performing a search throughout the space. MIMIC first searches the space randomly, using a distribution that is uniform in relation to the search space. By learning from the search and uses it knowledge from distribution function, MIMIC often performs significantly better that other optimization algorithms.

2.3 Implementation

In this section I will implement three local random search algorithms discussed in the previous section, and perform improvements on some of the algorithm parameters in order to examine the overall understanding of the dataset and the limitations / benefits of each algorithm. Every test is run over 5,500 iterations, and the sum of squared errors are plotted. The effect of random nature of the algorithms to optimize their performance can be easily seen from the noise of the error plots given in Figure 1.

The result given in Table 1 shows that the complexity of Backpropagation algorithm outperforms randomized optimization algorithms for this dataset. Backpropagation is the method I used to obtain the weights for my

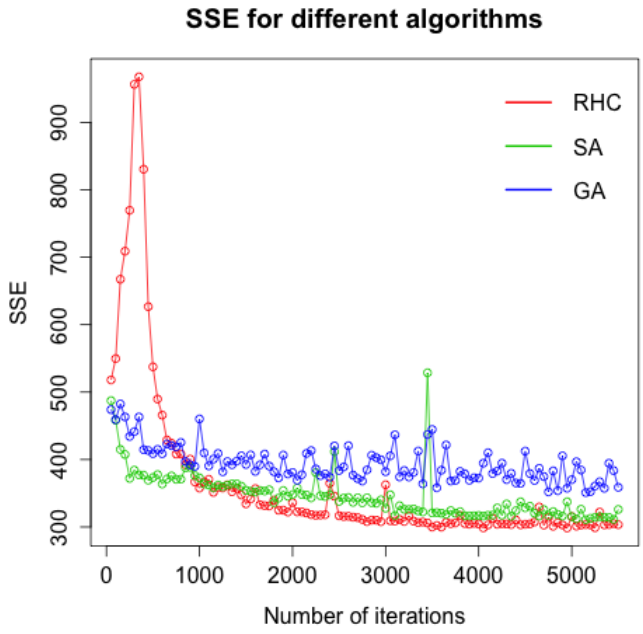

Figure 1: This error plot shows the result gets the plateau, but suffer from fluctuations over some iterations due to their random nature.

neural network in the first assignment where it also outperformed all other supervised machine learning algorithms I compared with.

2.3.1 Improvements

- Simulated Annealing

  I performed an experiment by altering the cooling rate of the algorithm in order to see what effect the parameter had on its performance. The previous experiments had all been conducted over a constant cooling rate of 0.9. In this experiment we iterated in intervals of 0.05 from 0.05 to 0.95. The results in Figure 2 shows that drops in accuracy as we decrease the temperature. SA may exhibit sheer disparity in this low temperature.

- Genetic Algorithms

  For GA, I performed an experiment by changing the population size of the evolutionary algorithm in order to see what effect the parameter had

Weight Algorithm	Accuracy(train)	Accuracy(test)	Avg. Train Time
Backpropagation	66.6%	75.23%	4.3s
RHC	66.67%	68.47%	9.82s
SA	67.09%	71.62%	9.97s
GA	63.5%	62.16%	387s

Table 1: Comparison of Algorithm's Performance - The average test time for all algorithms was nearly identical, averaging 0.001s.

on its performance. I can imagine that increasing the population size would increase the accuracy even before reaching the global maximum of the problem. Indeed, this is the case. As the results in Figure 3 show that if the population size increases accuracy is increasing slowly. There is a big jump in accuracy at population size 400. However, if the population size is too small, it will increase the possibility for error because the ideal solution may no longer be part of the population.

3 Optimization Problems Overview

In this section I will briefly discuss the three optimization problems I have chosen to illustrate the advantages of SA, GA, and MIMIC.

3.1 Traveling Salesman Problem

The Traveling Salesman is a classic problem where one has to find the minimum path to visit every city and return to the starting city given a number of cities and their connections. The metric we use to evaluate this problem is 1/distance, so an algorithm with a larger average 1/distance has a smaller distance, and therefore a better solution to the problem.

3.2 Continuous Peaks Problem

The Continuous Peaks Problem was introduced by Baluja et al. [1]. This problem is in continuation of the Four Peaks and Six Peaks problem, which are used as metrics in mimic [3]. This problem was introduced to find the flaws of hill climbing by presenting continuous local maximum values with

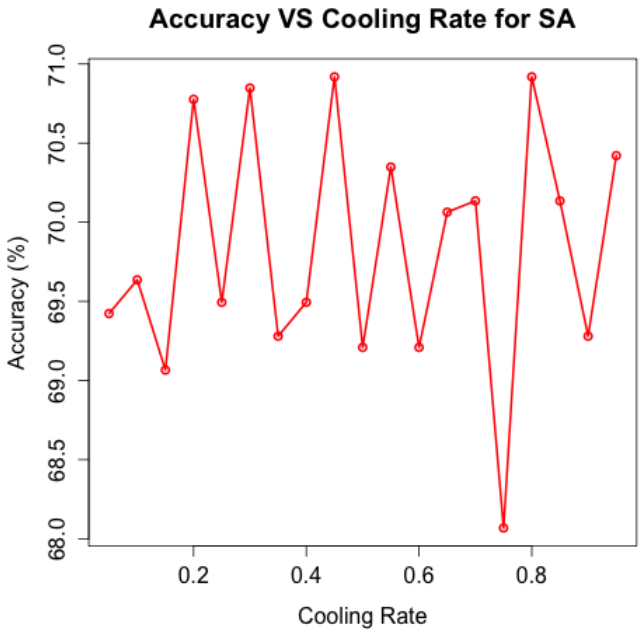

Figure 2: Effect of cooling rate for simulated annealing. Note that I only ran one trial per cooling value, so we see a lot of noise in accuracy due to the random nature of the algorithms.

fairly large plateau. some algorithms may easily get stuck in the area, failing to converge to a better solution.

3.3 Knapsack Problem

The Knapsack Problem is a dilemma in which out of k different objects with different weights and values one has to carry the maximum value in their sack. In my problem, the maximum possible volume of the sack is 3200. The maximum weight and volume are 50 units per item, with 40 items each of 4-copies.

4 Optimization Problems Analysis

The final part of this analysis focuses on three optimization problems that each individually demonstrate the benefits of SA,GA, and MIMIC. As I men-

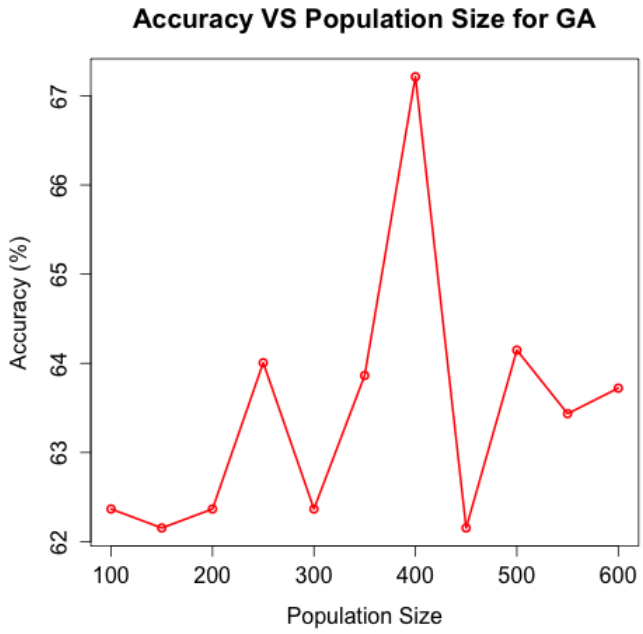

Figure 3: Effect of population size for GA. Note that I only ran one trial per population size, so we see noise in accuracy due to the random nature of the algorithms.

tioned before the three algorithms are the Traveling Salesman Problem, the Knapsack Problem, and the Continuous Peaks Problem.

4.1 Traveling Salesman Problem (TSP)

In order to obtain the minimum distance among all the cities in TSP the algorithm that learns from every iterations best would perform best. In this sense I thought MIMIC would be at the advantage because it learns from previous searches. However, given the complexity of these problems I was proven to be wrong. Instead, GA performs best for this problem. The reason I believe GA outperformed both MIMIC and SA, because the set of populations it considered all were highly probable to contain a small path that was part of the solution. GA was able to combine these small paths into a optimal solution over several generations and with mutations. The results can be seen in Figure 4, which show that GA had the best performance for

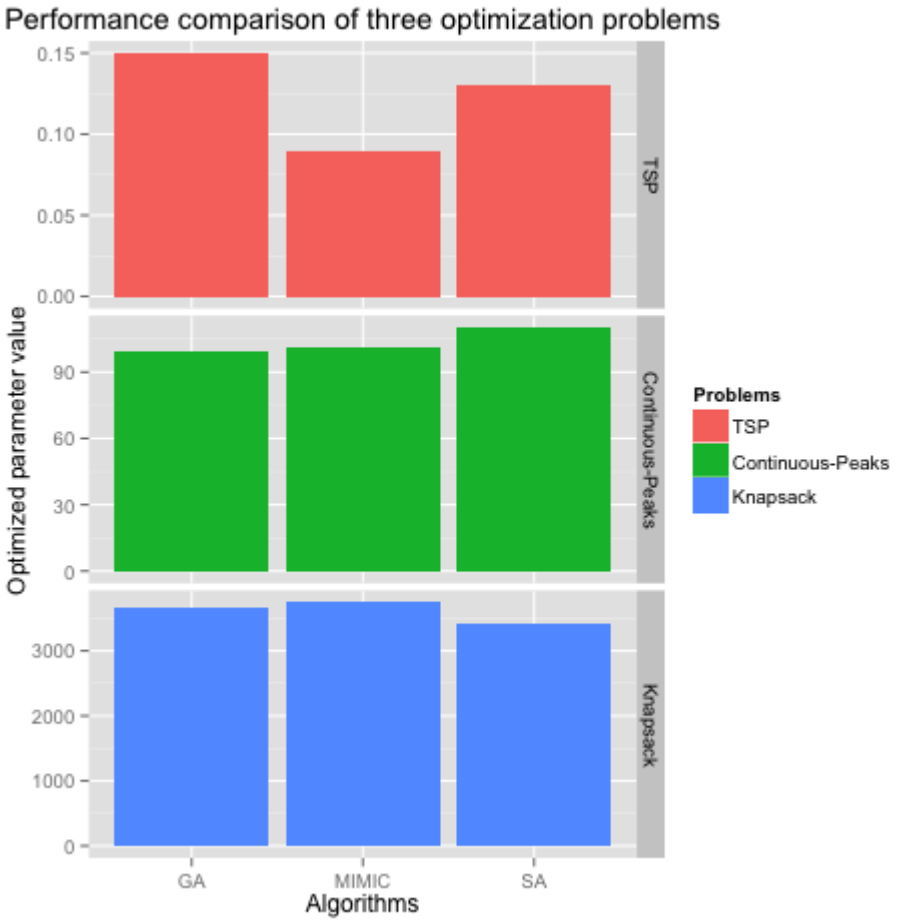

Figure 4: Performance metric comparison among algorithms. The optimized parameters are different for different problems.

this problem.

4.2 Continuous Peaks Problem

As can be seen in Figure 4, SA outperforms both MIMIC and GA. MIMIC was also able to search the space almost correctly for an ideal maximum, close to SA's performance. The reason I believe SA did best for this problems is that the controlled search due to slowly decrease in temperature forbids the probability of selecting the wrong paths and hence moves toward the right direction in the search space in order to find the global maximum of the problem. Note that the nature of the problem plays a big role for SA.

4.3 Knapsack Problem

In this well known Knapsack problem my hypothesis was that this problem would demonstrate the advantages of MIMIC due to its capabilities to learn from searching the space and assumed underlying distribution. This hypothesis was in fact proven true, as can be seen in Figure 4. This problem is vital in demonstrating the weaknesses of SA which perform decently when the outcome need not be close to perfect. In this case, MIMIC outperforms both of them because GA and SA are unable to converge to a better solution, which can be defined as a larger total value in the sack. The standard genetic algorithm used in ABAGAIL actually does a decent job at defining a population which allows it to obtain a decent result.

5 Conclusion

In this project I analyzed the use of three randomized optimization algorithms in order to obtain weights for a neural network, and then compared them to a previous result from [2] on the Abalone dataset which used backpropagation to obtain its weights. In this scenario, backpropagation still outperformed RHC, SA, and GA, who all obtained very similar performance. I then further examined SA, GA and MIMIC with respect to optimization problems that demonstrates the advantages of each approach. This whole analysis enhanced my understanding of optimization algorithms, and their use in real world machine learning problem solving.

Acknowledgements

First, I would like to acknowledge the UCI machine learning repository [4] for providing the dataset. Further, I would like to acknowledge everyone involved in the construction of the ABAGAIL Java library, which was essential for running the algorithms for this project.

References

[1] BALUJA, S., AND DAVIES, S. , Using optimal dependency- trees for combinatorial optimization: Learning the structure of the search space. Tech. rep., DTIC Document, (1997).

[2] `https://github.com/sshamid/ML_projects`

[3] DE BONET, J. S., ISBELL, C. L., VIOLA, P., ET AL, Mimic: Finding optima by estimating probability densities, Advances in neural information processing systems, 424-410, (1997).

[4] FRANK, A., AND ASUNCION, A., UCI machine learning repository (2010).

[5] FRASER, A. S, Simulation of genetic systems by automatic digital computers, Australian Journal of Biological Sciences 13, 2, 150-162.(1957).

[6] KIRKPATRICK, S., Optimization by simulated annealing: Quantitative studies. Journal of statistical physics 34, 5, 975-986. (1984).