CS-7641 Machine Learning: Randomized Optimization

Junle Lu
junle.lu@gatech.edu
Georgia Institute of Technology
October 14, 2018

Abstract – In this paper, we investigated and analyzed four random search algorithms: Randomized Hill Climbing (RHB), Simulated Annealing (SA), Genetic Algorithm (GA), and MIMIC. The experiments and analysis consist two parts: neural network optimizer and optimization problems. The first three algorithms are utilized as weight optimizer for neural network to find the optimal weight for a given dataset from the previous assignment. The performance is evaluated and analyzed. To better understand the random search algorithms, three optimization problems are chosen to demonstrate the advantages and disadvantages of each algorithm: Max K-coloring, Travel Salesman Problem, Flip Flop.

1. Randomized Weight Optimizer for Neural Network

1.1 Introduction

In the last paper, the performance of a neural network was analyzed with two datasets, but only one is chosen to be used for the experiments: gender recognition by voice.

Gender Recognition by Voice: this database was created to identify a voice as male or female, based upon acoustic properties of the voice and speech. The dataset consists of 3,168 recorded voice samples, collected from male and female speakers. The voice samples are pre-processed by acoustic analysis in R using the seewave and tuneR packages, with an analyzed frequency range of 0hz-280hz. It has 3168 instance and 20 features.

Weight optimizer	Hidden layers	Neuron	Train score	Test score	CV score	Iteration	Runtime
lbfgs	2	17	0.934	0.934	0.931	2286	Slow
sgd	4	17	0.625	0.614	0.613	20	Fast
adam	11	17	0.865	0.866	0.843	46	Fast

Table 1: Performance Summary for MLPClassifier (sk-learn)

Although the dataset was analyzed with Multi-layer Perceptron classifier (scikit-learn library), but the multi-layer neural network from ABAGAIL library is used for analysis with existing randomized search algorithms. From table 1 above, each existing weight optimizer behaves dramatically different from each other with respect to hidden layer and accuracy etc.

Therefore, it is not significant to compare the accuracies for the randomized search algorithms. Moreover, the mean square value and training score provide enough insights to compare the three randomized search algorithms. In terms of hidden layer and neurons, it is been observed that building complex neural network does not help the analysis, but it is a waste run time to show case the same observations with simpler neural network. For this particular dataset, it only requires 2 hidden layers to achieve 93% score for 'lbfgs' optimizer. Therefore, the neural network is built with 2 hidden layers and 3 neurons to effectively test the randomized search algorithms. This is simple and yet sensitive to hyperparameter on each algorithm.

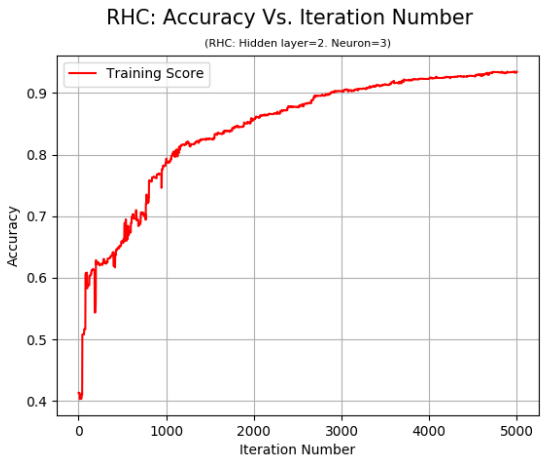
1.2 Experiment Setup

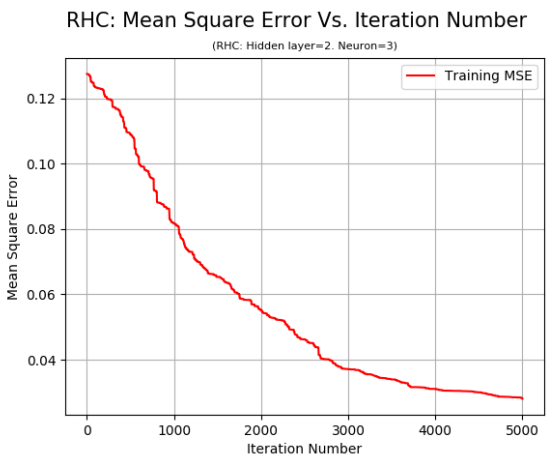
Each optimization algorithm (RHC, GA, and SA) utilized to find the optimal weights for the neural network in multiple experiments. All algorithms are implemented in ABAGAIL package. In addition to each algorithm's hyperparameters, the following variables are investigated and analyzed for all the algorithms against number of iterations: training score, mean square error and run-time. Although cross-validation is excellent to determine if the training model is overfitting or underfitting, but it is not available in ABAGIAL package. However, the performance of the algorithms can still be compared and analyzed.

Each algorithm is trained for 5000 iterations, and the data is collected at every iteration. The neural network is trained on each iteration. The optimal weights are applied to the neural network to collect the mentioned data. Then, the neural network is restored with its sub-optimal weights to continue its training.

1.3 Randomized Hill Climbing

Randomized Hill Climbing (RHC) is a simple and yet an effective randomized optimization algorithm in cases. It starts at a random point. For every iteration, it gauges its neighbors' fitness. It will move to the point with the best fitness. It repeats this search until none of the neighbors have better fitness evaluation. At this point, algorithm either have reach the search goal or it is stuck at a local optimum location.

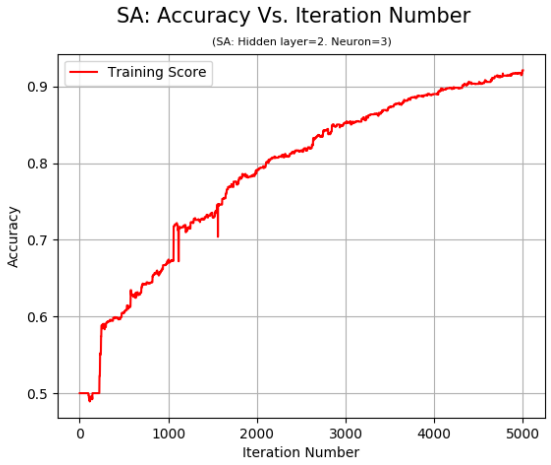


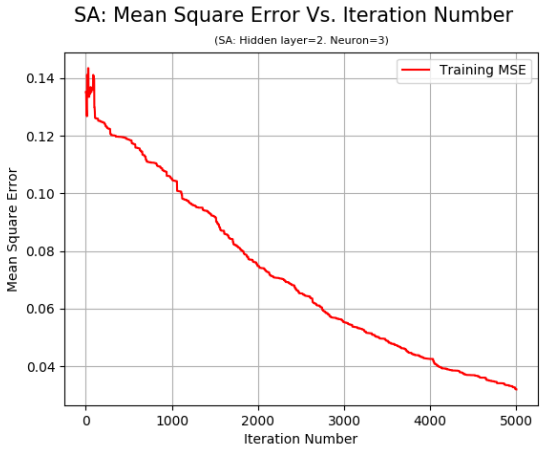


From the figure above, the algorithm performs well as fast with 61 seconds wall time. The training score increases with iteration number as the mean square error decreases. The dataset is still converging beyond the maximum 5000 iterations. It has achieved 95% training score at the 5000 iteration, but it has already achieved 80% training score at the 1000th iteration. The number of restarts is the only hyperparameter for this algorithm, but the experiment did not show that its beneficial for this data set.

1.4 Simulated Annealing

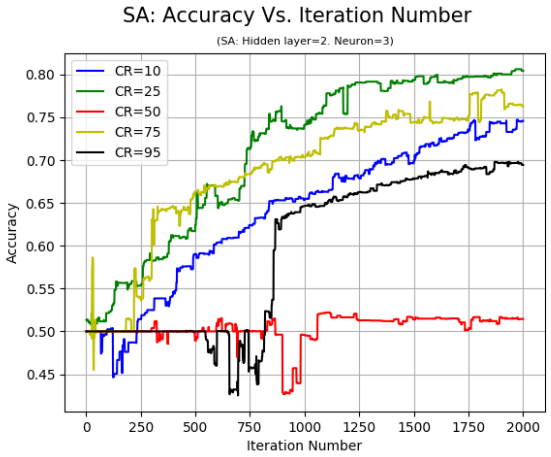
Simulated Annealing algorithms is based off the idea of annealing in metals, which the metal heats up and cools down. In general, the algorithm starts off from a random point, then it samples another random point in the next iteration. The two data points are evaluated by a fitness function. As a result, the algorithm decides if it wants to move to that point based off the fitness function's return value. It does not always move toward the target or goal, which may cause it to be stuck at the local optimal. This key feature differentiates it or advantage from the randomized hill climbing algorithm that often gets stuck at the local optimal.

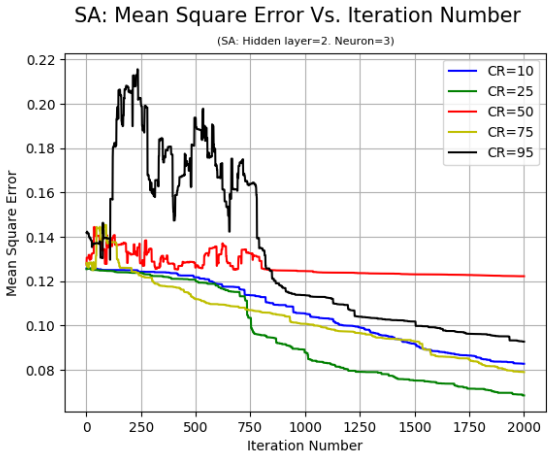




As expected, the overall performance is very similar to randomized hill climbing algorithm as both seek for random points toward the target. But the initial peaks are different as simulated annealing has evaluation for fitness. There are two hyperparameters to be analyzed for simulated annealing: cooling rate and starting temperature. The starting temperature was studied but it did not result in a significant difference in this case, so it is ruled out in this analysis. On the other hand, cooling rate is a interesting parameter. Cooling rate is a constant defined from 0 to 1. It represents the temperature decreasing rate in each iteration and faster with lower cooling rate value. This rate determines the eagerness to move toward the target of the iteration run. In each iteration, it is more likely to move to the data points that are closer to

the target with lower temperature. But this may increase the chances of getting stuck in a local optimal location. Five cooling rate values are investigated as shown in figures below.

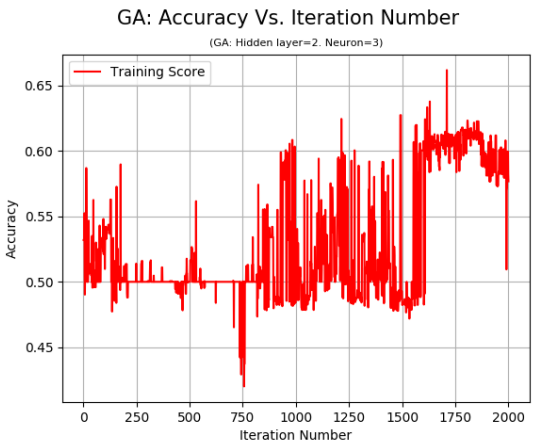


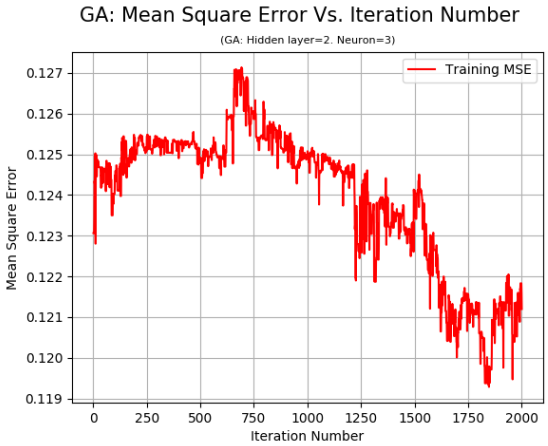


First of all, accuracy and mean square error curves are almost identify to the randomized hill climbing algorithm when the cooling rate is set to 0.10 (blue lines). This is expected from as randomized hill climbing is intended to move closer to the target as possible. The cooling rate here is to mitigate the issue that randomized hill climbing may be getting stuck at the local optimal location. With higher cooling rates, the randomness is reflected in the first 750 iterations where mean square error is unstable. After 750, all of curves are converging just as in randomized hill climbing except when the cooling rate is equal to 0.5. it seems to be stuck after 750 iterations. This is interesting and yet to be further investigated. One assumption is that it won't be effective to solve a problem with 50% chance of pass or failure, with another function with 50% chances of true or false. In term of performance, simulated annealing is just as effective as the randomized hill climbing algorithm but with the stuck at local optima mitigated with an evaluation function and hyperparameters.

1.5 Genetic Algorithm

For genetic algorithm, the neural network's weights will be represented in a population. The error function is evaluated based evolution fitness and traits of the weight. Each iteration is seen as a generation evolution. The new weights are born in a new generation, and they are evaluation for their survival fitness. The good fit weights stay and evolve new generation the next generation. Therefore, there is a better fitness as the iteration progress. It almost entails a simulation game, and it can be seen as computation costly as our neural network weight optimizer.





1.6 Conclusion

From the experiments, randomized hill climbing and simulated annealing shares common characteristics. They both attempts to move to closer to target in the next iteration, but simulated annealing has evaluation function that may decide not to move to the next location which could be a local optimal location. This can be very problematic for randomized hill climbing algorithm to be stuck at a local optimal location. In addition, they are both very fast and effective. It's as if the simulated annealing is a better version of randomized hill climbing algorithm with some fine tune functionalities: cooling rate and starting temperature. Genetic algorithm has the worst run-time which is about 10 times worse than randomized hill climbing or simulated annealing algorithms. Its performance is not that impressive either.

Algorithm	Wall clock time (sec)	Mean Square Error
RHC	64	0.03
SA	63	0.035
GA	650	0.124

Table 2: Performance Summary for Three Optimization Algorithms

2 Three Optimization Problems

Three optimization problem all test with all four algorithms for investigation and analysis.

2.2 Traveling Salesman Problem

This problem is popular and yet simple questions. You know some cities and the distance in between all the cities. It is tasked to find the shortest path a traveling sales man can visit all the cities exactly once, but it will also take the man to the origin city. I have heard that Genetic algorithm may do a good job on this problem than the others.

2.3 Flip Flop

A problem to find the number of times the bits change in a bitstring. This problem may show that Genetic algorithm isn't the best on this.

2.4 Max K-Coloring

2.5 Conclusion

The problems are interresting and yet to be investigated.