Assignment 2: CS7641- Machine Learning Yogesh Edekar (GTUID – yedekar3)

March 13, 2021

1. Introduction

As part of assignment 2 I am trying to apply the Randomized Optimization algorithms to three problem domains in order to maximize the value of fitness function so that we can find out Optimum or near optimum solution. In this process we are also trying to analyze the optimization algorithms to find out the strength and weakness of each algorithm and come up with the problem domains to which these algorithms can be applied effectively. Along with this I have also applied the optimization algorithms to find out optimal weights for a neural network for term deposit data set for segregating the bank clients who open a term deposit account at the bank from the ones who do not open a term deposit account at the bank (sharanmk, 2020) from assignment 1.

2 Randomized Optimization Algorithms

Randomized optimization algorithms are used in order to determine the global optimum for the problems that are otherwise difficult to be derived. We can say that the aim of Random Optimization algorithms is to find out the best state following an objective function. The notion of state usually varies from problem to problem but in our analysis we take state as a one dimensional array or vector of values. What is meant by best us defined by a function known as “fitness function”, “objective function” or “loss function” which we maximize or minimize. However for this assignment we are trying to maximize the function so we will not consider loss function which we minimize instead we will be focusing on the fitness function. The optimization algorithms being analyzed are Randomized hill climbing, simulated annealing, genetic algorithms and MIMIC.

2.1 Randomized hill climbing

Hill climbing belongs to a family of local search algorithms. It starts with a random solution to the problem and then tries to optimize the solution by continuously trying to find a better solution than the current solution. If a better solution is found then the algorithm switches to the better solution and keeps on searching for a better solution until no further solution which is better than the current solution can be found. Random hill climbing works the best for convex problems, however it can not work on concave problems since it usually results in finding the local optima for such problems. In order to avoid getting stuck in the local optima Random restarts can be used to select a random solution in case a local optimum is reached in order to find Global Optimum.

2.2 Simulated Annealing

Simulated annealing is a probabilistic technique for approximating global optimum of a given function. The algorithm picks up it’s logic from the concept of annealing from metallurgy. Here a temperature parameter T is utilized to determine the randomness in selection of the solution thus it brings in deterministic probability in picture with variation in the value of temperature T. The algorithm can be given as follows.

Assumptions –

F -: Fitness function, curr -: current position, negh -: neighbor, T – temperature, CE – Cooling Efficiency

Algorithm –

If F(negh) > F(curr) => curr = negh

If F(negh) < F(curr) => curr = negh with probability EXP(F(negh) – F(curr))/T)

After every iteration T = T*CE.

This indicates that for smaller values of T closer to 0 we will move to negh only if F(negh) is greater than F(curr) indicating we will move only in the best case where as for larger values of T we will move to negh irrespective of value if f(negh). Thus by keeping a high initial value of T and slowly cooling it down we start with random walk and slowly improve the algorithm with best neighbor giving us a better chance of hitting global optimum.

2.3 Genetic Algorithm

Genetic algorithm finds it's root in the mutation process. The algorithm starts with an initial fixed population. We first find out fitness of all the possible vectors in the input space. Upon finding out the fitness we choose the most fit inputs. We simply take the top half of the inputs with greater fit score. Now for the remaining population we replace the not so fir inputs with the cross over with most fit inputs (this process is called as mutation). This is a computationally intensive process. We get a particularly good fit with Genetic Algorithm owing to the fact that we always choose the population with good fitness score rather than a random walk. However Genetic Algorithms take significantly more time to converge.

2.4 MIMIC (Mutual Information Maximizing Input clustering)

MIMIC varies from genetic algorithm in the sense that it conveys the underlying structure of the search space and learns from the structure. We determine fitness score of all the points in the input space and determine a threshold theta and take all the points in sample space whose fitness value is greater than theta. The advantage of mimic is in the sample generation process we pick the theta for next iteration such that the next iteration samples are generated within the current iteration.

3 Neural Network with optimization algorithms

This part of the assignment tries to identify how the three optimization algorithms Randomized Hill Climbing, Simulated Annealing and Genetic Algorithm help in balancing the weights of the neural network model for one of the data set chosen in Assignment 1 .

3.1Dataset

The data set chosen for neural network optimization is the Bank deposit dataset. This dataset tries to identify whether a given customer with number of features will be opening a term deposit account at the bank or not. The data set is cleansed and processed to be used for neural networks and all the non numerical features are converted into numerical values for the purpose of using the dataset in neural network model.

3.2 Implementation

In order to compare the algorithms along with comparison for back propagation I have used mlrose package of python to execute neural network model on the term deposit dataset with the given algorithms and activation function as Sigmoid function in order to give a smooth continuous behavior fitting to the needs of the neural network. I have split the data into 80/20 splits so that 80% of the data is used for training the model and remaining 20% data is used for testing the accuracy of the model

3.3 Parameter search

I have used the grid search technique for identifying the tuning parameters for the different algorithms. The parameters considered for Random Hill Climb and Simulated Annealing are maximum iterations while for Genetic algorithm along with maximum iterations the mutation rate is also used for tuning the neural network to obtain the comparison results.

Upon grid search the values obtained for **Random Hill Climb** are as follows

Accuracy – 0.53

Maximum iterations – 96

The values obtained for Simulated Annealing for accuracy and maximum iterations after grid search are as follows.

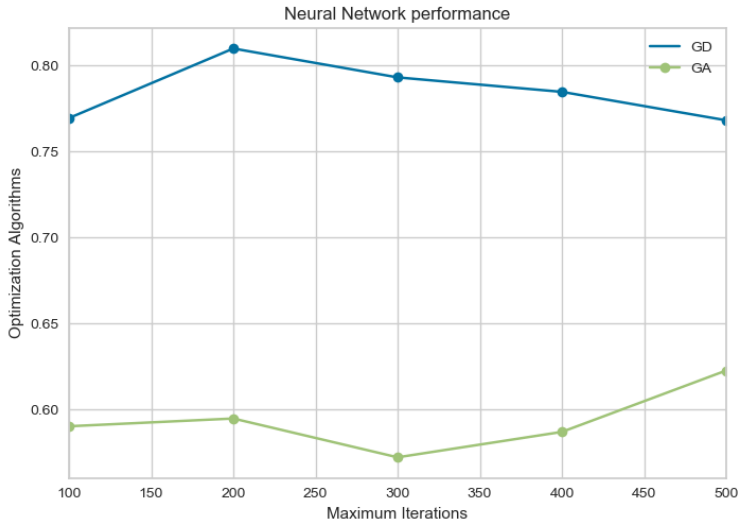
Accuracy – 0.53

Maximum iterations – 20

This indicates the Random hill climb and Simulated Annealing are performing equally well. However, the controlled nature of the Simulated Annealing algorithm to slowly reduce randomness and incorporate the learning done while traversing comes in handy and we can achieve the same accuracy in less iterations. Due to time constraints, I could not delve further into the RHS and SA algorithms.

3.4 Genetic Algorithm and Backpropagation comparison

Since grid search gives better results with Genetic Algorithm I have compared Genetic Algorithm with back propagation by varying the maximum iterations over a range from 100 to 500. I have compared the weighted F1 accuracy score for both Genetic algorithm and back propagation at each maximum iteration value from 100 to 500. The results for those comparisons are as follows.



Looking at the above graph we can clearly see that back-propagation has better accuracy as compared to the Genetic Algorithm. We can see that the backpropagation F1 scores are optimum at n = 200 and as we get more and more iterations Genetic algorithm starts to pick up the F1 scores. This can be attributes to the fact that with more maximum iterations Genetic algorithms get more opportunity for mutation and fitting the data well so that the scores start improving.

Hence we can say here that Genetic algorithms can improve given more data but with limited time frame at hand we can see that for lower number of maximum iterations backpropagation is clearly the winner over Random Hill Climb, Simulated Annealing and Genetic algorithms.

4 Optimization Problems

I have used three optimization problems mentioned below to analyze the Optimization algorithms. For analyzing the algorithms, I have used the package mlrose-hiive that allows us to specify the optimization and the fitness function for the problem in order to analyze the corresponding algorithm. Here the input size of the vector is changed in the order of 20,40,60,80 and 100 to increase the complexity of the problem gradually and the algorithms are analyzed based on the fitness score and wall clock time for comparison

4.1 Four peaks problem

The four peaks problem as defined by Baluja and Karuana is designed specifically for Genetic Algorithms. We have chosen this problem specifically to see where we can validate that the genetic algorithm does better on this problem in the given set of problems to surpass the local optima. The problem for an input vector X can be given as follows

z(x) = number of contiguous zeros in position N for an N bit vector (trailing 0's)

o(x) = number of contiguous ones starting in position 1 (leading 1's)

Reward = N if o(x) > T and z(x) > T

0 else

Where optimization function f(x) = MAX(o(x), z(x)) + Reward

We can see that there is a possibility of two global optimum clearly when the number of leading ones is T + 1 followed by trailing zeros or the number of trailing zeros is T+1 preceded by leading ones. For this assignment t is chosen to be 0.15. The results obtained for four peaks problem are as follows

I/P length	RHC	SA	GA	MIMIC
20	0	0.008	1.04	13.16
40	0.008	0.016	3.18	62.33
60	0.008	0.024	3.53	220.76
80	0	0.032	3.45	239.38
100	0	0.024	1.95	392.26

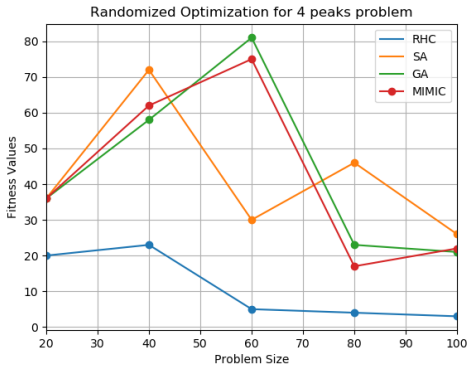


Table 1 – Wall clock times in seconds for all algorithms

The above graph and table make it pretty clear that the 4 peaks problem works very well for the Genetic Algorithm. We can see that the Genetic algorithm as well as MIMIC perform almost equally well with genetic algorithm getting slight edge over MIMIC. However, when we look at the wall clock times for the algorithm, we can see that MIMIC performs the worst as it takes maximum time to converge. From time standpoint Randomized Hill climbing takes the least amount of time and MIMIC takes the most time. Due to the random nature of the algorithm, we can see that the Randomized Hill Climbing algorithm selects the input vectors randomly without focusing on the best fit and hence it hits the local optimum pretty quickly, so that no best neighbor can be found anymore at local optimum and hence the algorithm converges. With Simulated annealing we can see that the introduction of temperature adds a little bit of optimization to randomized hill climbing and it performs well in the beginning stages as the problem is simple to resolve. However at n = 60 when the input vector is sufficiently large and problem becomes complex we see Genetic Algorithm and MIMIC performing well due to their nature of defining thresholds and then selection. However since these operations are computation expensive we can see that this impacts the convergence time for the algorithms and they start taking more and more time as the complexity increases. Overall we can see that for complex problems such as 4 peaks problem we can use Genetic algorithm as it hits the global optimum for given input size for larger inputs.

4.2 Flip flop problem

The flip flop problem consists of counting the total number of flipped bits in a given bit string and the goal is to maximize the flipped bits in the given bit string. In the flip flop problem the input is given as a vector x and we need to evaluate the fitness of the state vector x as the total number of consecutive elements of x , (X_i and X_{i+1}) where $X_{i \text{ and } } X_{i+1}$ are different. Now there is a possibility that different flip flop problems for e.g. 10101010 and 01010101 can be considered same and different by different algorithms. Since flip flop problem is a complex problem which is not easy to learn we expect GA or

MIMIC to perform well on the Flip Flop problem. The fitness values of the flip flop function and the wall clock times for all 4 algorithms can be shown as follows.

I/P length	RHC	SA	GA	MIMIC
20	0.002	0.012	4.395	43.902
40	0.005	0.062	7.003	169.10
60	0.006	0.057	8.250	370.17
80	0.007	0.061	9.254	667.49
100	0.022	0.090	11.883	1084.75

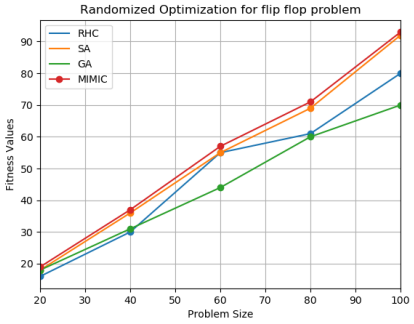


Table 2 – Wall clock times in seconds for flip problem

From the above graph and table we can clearly say that our assumption of MIMIC or Genetic algorithm to be the best algorithms for flip flop problem turns out to be correct. MIMIC specifically works well for the problem due to it's nature of learning the structure of the problem. For e.g. for input 10101010 and 01010101 other algorithms will consider them as different problems but MIMIC will consider them as same problems since both the problems specify exactly same structure of alternating bit patterns. The structure is represented by spanning trees in mimic and since we have a maximization problem we will try to find out the maximum spanning tree. Since MIMIC also uses the top Nth percentile samples every iteration to identify the next population of samples each and every iteration guarantees us to find out the input samples with better fitness score. This results in obtaining a better fit for the problem. Genetic algorithms basically perform poorly in this problem due to the nature of the cross over happening between the parents. Often because of this cross over the local optimum would be hit and it would be difficult to travel towards global optimum. For e.g. for parents 11110000 and 00001111 a single point cross over would lead to children 00000000 and 11111111 which would result in same fitness value or in some cases it might even worsen. Hence GA becomes ineffective in giving the optimal solution for the problem. Randomized Hill Climbing and Simulated Annealing don't do as well as MIMIC in this case due to the fact that they are performing random search over the input space. However random annealing makes use of the temperature parameter with gradual decay of the parameter to slowly decrease the value of T over iterations which indicates that the fitness scores improve as number of iterations are improved.

4.3 One max problem

The one max problem takes an n dimensional state vector x = [x_0, x_1, x_2, …., x_{n-1}] Such that

$$Fitness(x) = \sum_{i=0}^{n-1} x_i$$

The one max function is suitable to take either continuous or discrete optimization state problems however we are considering the discrete state problems to be solved using the one max function. The one max problem is a very simple problem to solve where we have to find out the optimal solution for the input vector in terms of sum of all the dimensions. Since the only possible values for dimensions are 1 and 0 we can clearly see that the global optimum will always match the value of N i.e. the dimensions of the input vector. One max problem is an easy problem to solve where there really is no need to

understand the structure of the problem and ideally random search should give us the answer quickly since there are not a lot of local optimums where the random search algorithm will get stuck. The wall clock time as well as fitness value chart for the one max problem can be shown as below.

I/P length	RHC	SA	GA	MIMIC
20	0.001	0.00758	1.428	12.36
40	0.002	0.01	1.24	50.56
60	0.003	0.007	2.14	118.55
80	0.003	0.009	2.84	219.33
100	0.004	0.009	1.697	364.41

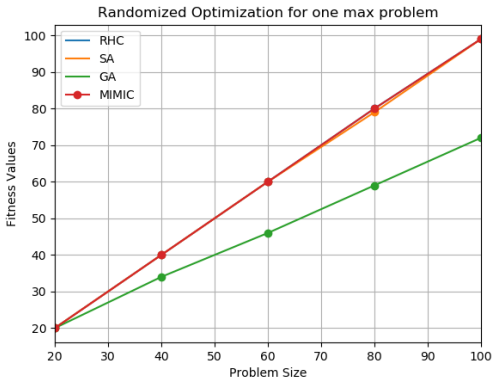


Table 3 – Wall clock times for input length for one max

Problem

The times and fitness function plot clearly show that the one max problem can be solved for optimum solution by Random Hill Climb, Simulated Annealing and MIMIC. Genetic Algorithm does not perform well for the one max problem. The mutation strategy of genetic algorithm does not work well with one max problem and it gets stuck in local optimum and hence can not give us the optimum solution for the one max problem. Since the problem is very simple to solve without a lot of local optimum the Random hill climbing and Simulated annealing algorithms perform really well on one max problem and give us the optimal solution. MIMIC also fairs well on the one max problem and gives us the optimal solution. However MIMIC involves finding out the nth percentile of the sample space and takes the fittest of the sample which makes it very computationally intensive. Hence in case of a simple problem like One max problem based on the time and fitness score tradeoff our choice of algorithm can be Random hill Climb or Simulated Annealing.

5 Conclusion

This assignment helped me understand the Optimization algorithms in much better manner. The Randomized Optimization algorithms help us solve otherwise difficult to solve problems. The algorithms take different approaches towards solving the problems and hence the algorithms can be used to solve different types of optimization problems. The pros and cons of the algorithms studied can be given in concise manner as follows .

Algorithm	Random Hill Climb	Simulated Annealing	Genetic Algorithm	MIMIC
Problem Domain	Solves easy problems	Solves easy problems	Solves complex and lengthy problems	Solves complex and lengthy problems
Core logic	Takes completely random approach	Gradually reduces random approach to maximization	Use mutation and Cross over for finding best fit	Uses a threshold to select top n% of the distribution

Local optimum	Almost every time hits local optimum due to random nature.	Less probability of hitting local optimum due to temperature factor and it's gradual decrease.	High probability of eliminating local optimum and hitting global optimum due to mutation probability.	High probability of eliminating local optimum and hitting global optimum due to keep percentage which allows to choose the samples to be carried over to next iteration.
Computation time	Algorithm tuns fast due to complete randomness	Algorithm runs considerably faster still slower than the RHC due to impact of T and computation needed to reduce T.	Algorithm runs slowly due to the computation complexity of mutation, finding optimum fit for the population in each iteration.	Algorithm runs the slowest due to additional complexity involved in finding the top n% samples, preparing spanning tree to maximize the function.
Sample space	Algorithm does not learn about the structure of the sample space.	Algorithm does not learn about the structure of the sample space.	Algorithm does not learn about the structure of the sample space.	Algorithm learns and remembers the sample space structure of the problem.

When we compared the Random Hill Climbing, Simulated Annealing and Genetic algorithms with the backpropagation for optimizing weights in the neural network. As per my observation back-propagation won against all the algorithms in accuracy score with only Genetic Algorithm being able to marginally match the performance of the back propagation.

3.3 References

https://towardsdatascience.com/getting-started-with-randomized-optimization-in-python-f7df46babff0

https://mlrose.readthedocs.io/en/stable/source/neural.html

https://mlrose.readthedocs.io/en/stable/source/fitness.html

https://mlrose.readthedocs.io/en/stable/source/tutorial2.html

https://en.wikipedia.org/wiki/Random_optimization

https://en.wikipedia.org/wiki/Hill_climbing

https://en.wikipedia.org/wiki/Simulated_annealing