**Randomized Optimization**

**Introduction**

This assignment is about implementing four random search algorithms:

* Randomized Hill climbing (RHC)
* Simulated Annealing (SA)
* Genetic Algorithm (GA)
* Mutual-Information-Maximizing Input Clustering (MIMIC)

In the first part of the assignment I had to apply the first tree algorithms to find optimal weights for a neural network where I have used the data from my previous assignment:

* Adult Data Set from UCI

In the second part of the assignment I have implemented tree optimization problem domains including the following optimization problems:

* Continuous Peaks
* 8 Queens
* Travelling Salesperson

The algorithms were implemented using the following Machine Learning libraries for Python:

* Scikit-Learn
* Mlrose

**Randomized Hill climbing (RHC)**

The random search algorithms are useful for global optimization problems where variables are continuous or discrete.

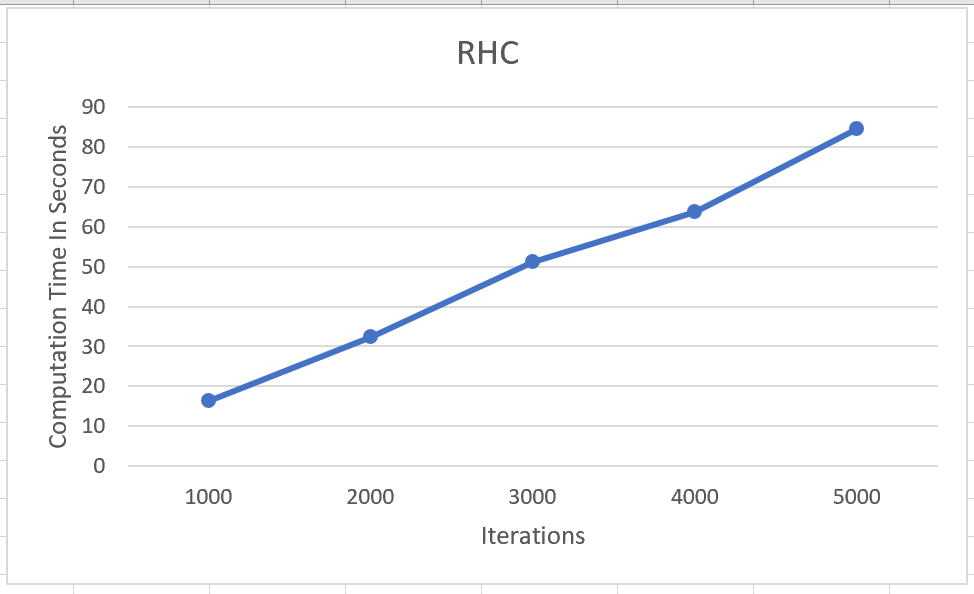
In the case of Randomized Hill climbing algorithm, the algorithm randomly samples points in the neighborhood for the best solution, looking for the global optimum. If the point is better than the current solution then it it makes it the new current solution and continue the search which is achieved with a random restart as the name suggests, otherwise the current solution is the best solution.

The Randomized Hill Climbing for neural network was run in Python using Scikit-Learn and Mlrose.

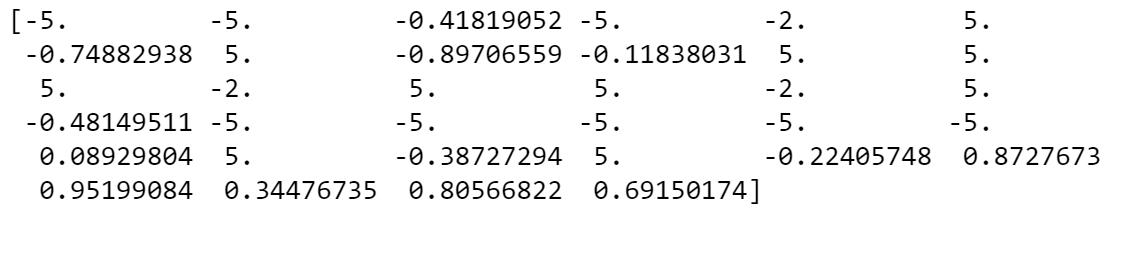
Running the RHC I have split the data for test size 20% and train size 80%. The hidden nodes was set to 2 with the application of relu for activation. For the algorithm I have also used the learning rate as 0.0001, maximum attempts as 100 and clip\_max as 5 .

In the below graph we can see the algorithm run time in seconds which for Random Hill Climbing is monotonically increasing as the number of iterations increasing, but does not exceed the 2 minute runtime.

The accuracy score for RHC is 75.32



The optimal weights for the Neural Network with the RHC:



**Simulated Annealing (SA):**

Simulated Annealing algorithm was also applied to the data I have previously used which is the Adult Data set form UCI.

The Simulated annealing algorithm is a probabilistic technique for getting an approximate for the global optimum of a function. Mainly used when search space is desecrate.

The algorithm is like a hill climbing algorithm except instead of picking the best move it picks a random point. If the randomly chosen point is better than the previous one then it is accepted.

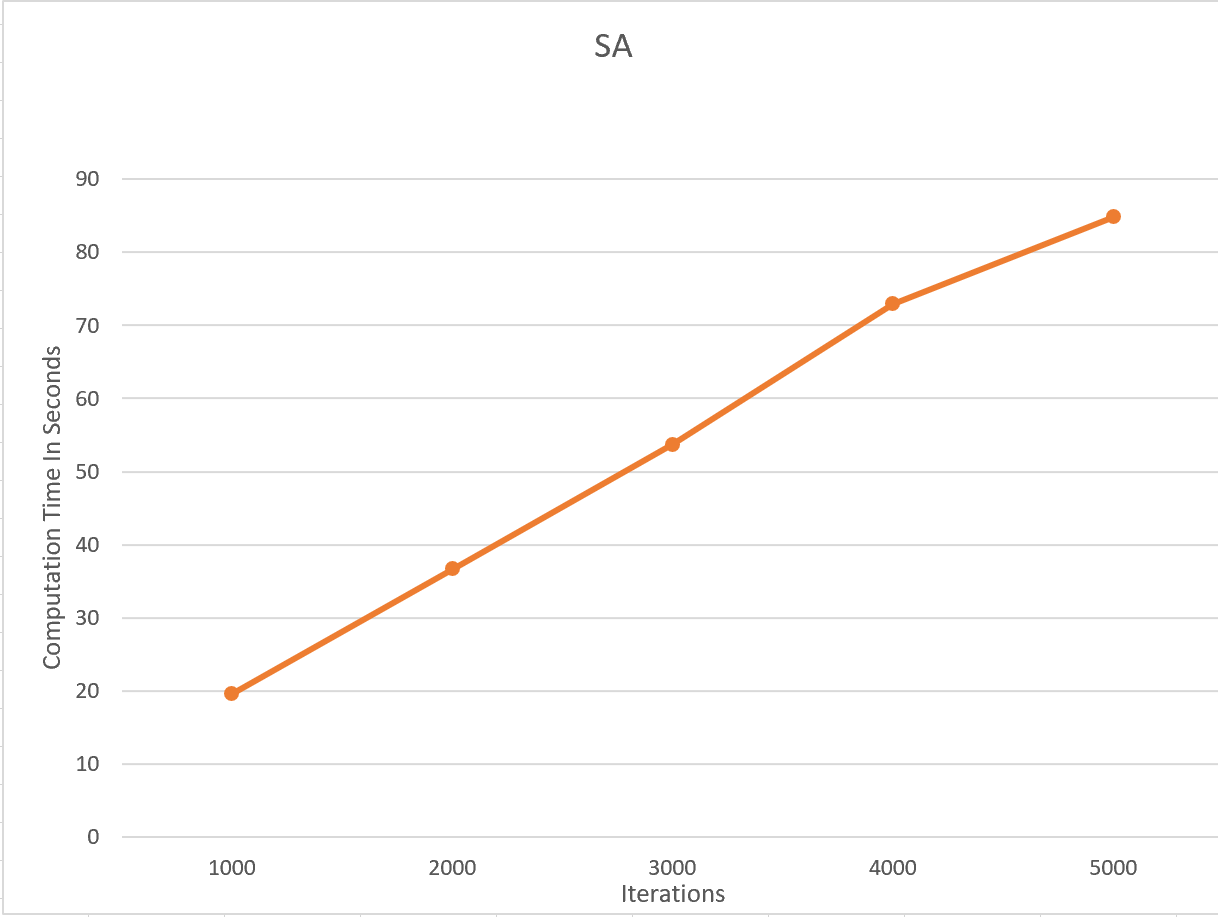
Otherwise the algorithm picks another move.

The Simulated Annealing for neural network was run in Python using Scikit-Learn and Mlrose.

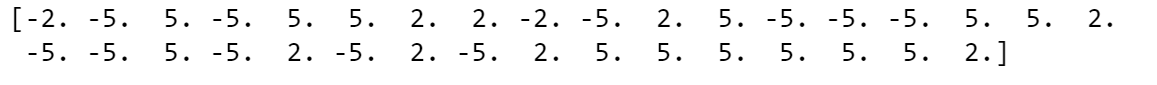
Running the SA, I have split the data for test size 20% and train size 80%. The hidden nodes was set to 2 with the application of relu for activation. For the algorithm I have also used the learning rate as 7, maximum attempts as 100 and clip\_max as 5 .

In the below graph we can see the algorithm run time in seconds which for Simulated Annealing is increasing as the number of iterations increasing.

The accuracy score for SAis 75.33



The optimal weights for the Neural Network via the RHC:



**Genetic Algorithm (GA):**

Genetic Algorithm was also applied to the data I have previously used which is the Adult Data set form UCI.

The GA regularly modifies the solutions. At every step, the genetic algorithm selects individuals at random from the given population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution.

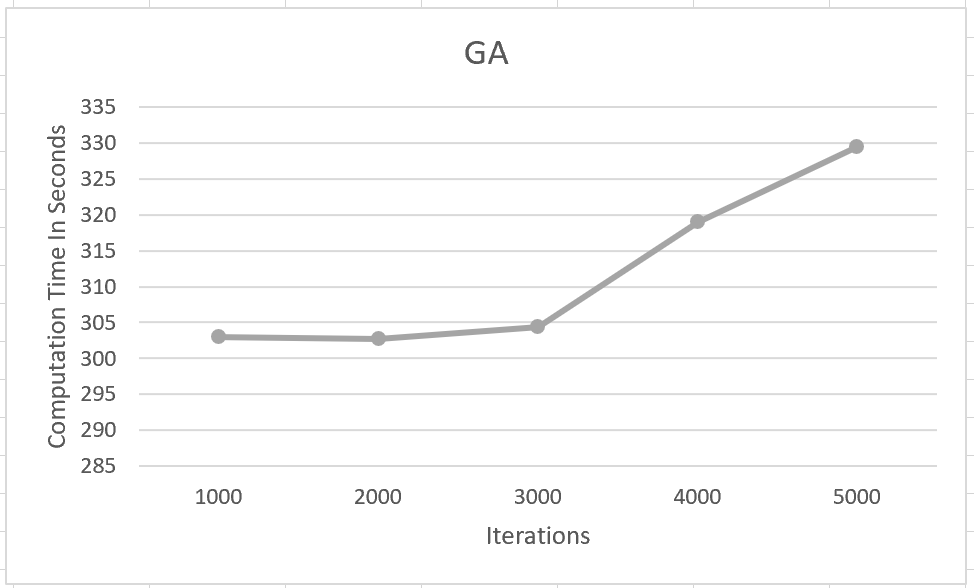
The Genetic Algorithm begins with a random initial population then the algorithm creates new populations in a parallel manner. The algorithm works similar to Random Restart Hill climbing algorithm, but instead of random restart it runs parallel.

The Genetic Algorithm for neural network was run in Python using Scikit-Learn and Mlrose.

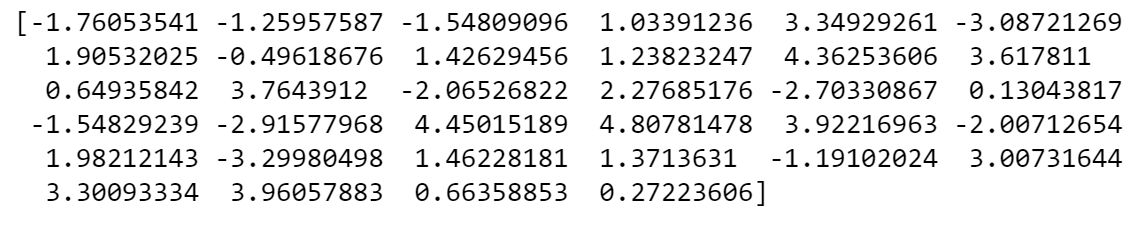
Running the GA, I have split the data for test size 20% and train size 80%. The hidden nodes was set to 2 with the application of relu for activation. For the algorithm I have also used the learning rate as 7, maximum attempts as 100 and clip\_max as 5 .

In the below graph we can see the algorithm run time in seconds which for Genetic Algorithm is increasing as the number of iterations increasing.

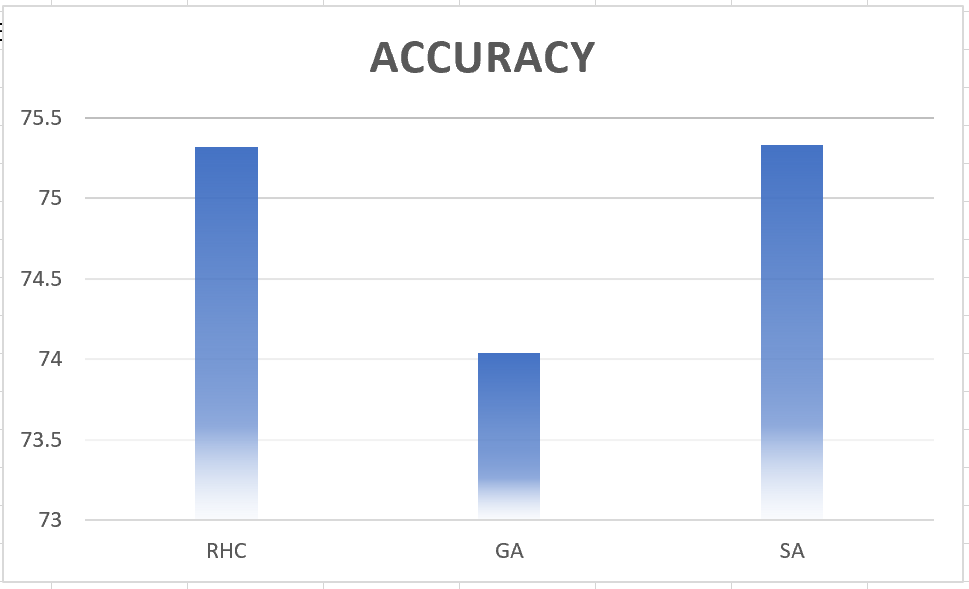
The accuracy score for SAis 74.04



The optimal activation weights for Neural networks with Genetic Algorithms are the following:



Comparing the accuracy of the above mentioned algorithms :



Optimization Problems:

* **Travelling Salesperson(TSP):**

The travelling salesperson problem answers the following question “Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?”

The Travelling Salesperson problem was implemented in python using mlrose.

In this example of the optimization problem I wanted to highlight the performance of Genetic Algorithm as opposed to the other algorithms.

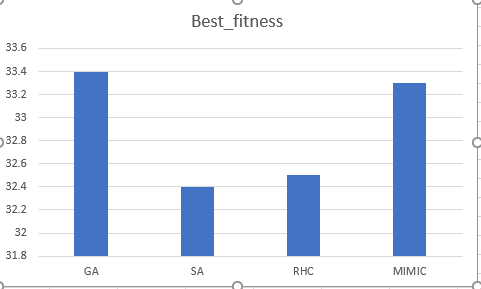
While applying the Genetic Algorithm to the TSP problem with max\_attempts of 100 and mutation\_prob 0.2. The best fitness score yields 33.4

In Case of the Simulated annealing where I have applied the max\_attempts of 100 and max\_iters of 100 gave a fitness of 32.45.

With Random Hill climbing the fitness is 32.5 with the max\_attempts 10 and max\_iters 100.

In the graph below we can see but not very distinguishingly that the GA was giving the lowest fitness value for this problem therefore highlighting the advantages of Genetic algorithm.

MIMIC giving a best fitness value of 33.3



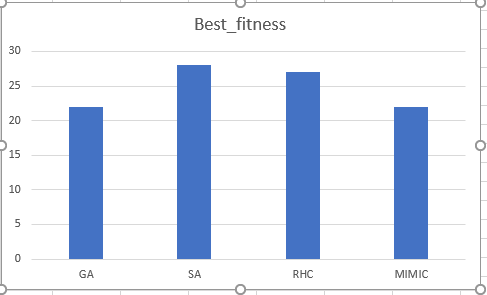
* **8 Queens:**

The eight queens puzzle is the problem of placing eight chess queens on an 8×8 chessboard so that no two queens threaten each other. Hence the solution should show no two queens shareing the same row, column or diagonal.

The 8 Queens problem was implemented in python using mlrose.

In this example of the optimization problem I wanted to highlight the performance of Simulated Annealing as opposed to the other algorithms.

In the graph below we can see the Simulated Annealing is giving the highest score in the case of the 8 Queens problem, thus highlighting the advantages of SA.



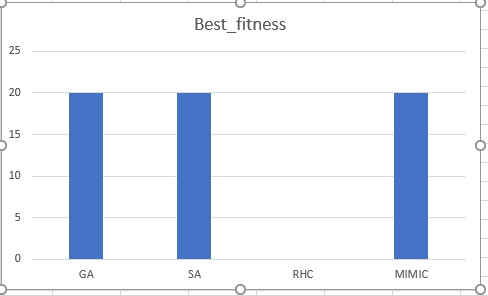
* **Four Peaks:**

The Four Peaks problem was implemented in python using mlrose.

In this example of the optimization problem I wanted to highlight the performance of MIMIC as opposed to the other algorithms.

Here we can see that the different algorithms are not giving distinguishing results for best fitness but the MMIC performs just as good as the Simulated annealing and Genetic Algorithm.

Given that the fitness value is 20 for GA, SA and MIMIC.



**Conclusion:**

The assignment included finding the optimal weights for neural networks. Also Creating three optimization problems where I have seen that the Simulated Annealing performs pretty good in all of the optimization problems.

**Citations :**

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

<https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html>

<https://en.wikipedia.org/wiki/Eight_queens_puzzle>

<https://en.wikipedia.org/wiki/Travelling_salesman_problem>