**Randomized Optimization Assignment**

Anurag Sahoo  
[asahoo41@gatech.edu](mailto:asahoo41@gatech.edu)

GT ID: 90368005

***Abstract—*** This report deals with the usage of four different random optimization algorithms (Random Hill Climbing, Simulated Annealing, Genetic Algorithm and MIMIC) on three different discrete optimization problems: Flip Flop, Four Peaks and N Queens. In Part 2 of the report, Random Hill Climbing, Simulated Annealing and Genetic Algorithm are implemented to determine the Neural Network classifier weights on the Wisconsin Breast Cancer dataset problem.

**Part 1: Analysis of Optimization Problems**

In this part, we will be looking at the different optimization algorithms and assess which performs better in different optimization problems. The four optimization algorithms are:

1. Random Hill Climbing: Random Hill Climbing looks at the neighbors and if the fitness score is higher than that of current position, it makes a move. Once the optima is reached, it makes a random restart to avoid ending up at local optima. Random Hill Climbing is extremely inexpensive, but it may end up in local optima if the attraction basin is smaller.
2. Simulated Annealing: Simulated Annealing works like Random Hill Climbing but with an idea that it is sometimes better to explore the space than going uphill. The decision to move is based on the acceptance probability which is a function of difference in fitness score of two points. Simulated Annealing has a better chance of not landing up at local optima and is inexpensive.
3. Genetic Algorithm: It employs randomized, parallel hill climbing algorithm to search a hypothesis that optimizes a fitness function. It operates by iteratively updating a pool of hypothesis, termed as population. A new population is generated by selecting the most fit individuals, and then an offspring is produced via crossover. A small percentage of the population is altered via mutation. As the movement of hypothesis happens abruptly, Genetic Algorithm is less likely to get tapped in local optima. It is computationally expensive though.
4. MIMIC: It stands for Mutual Information Maximizing Input Clustering. Unlike other algorithms, MIMIC uses the structure to help a randomized search through the space. MIMIC tries to find optima by estimating probability densities. MIMIC is also computationally expensive.

For this assignment, I will be using three different discrete optimization problems: Flip Flop, N Queens and Four Peaks.

**Flip Flop**: Flip Flop is an optimization problem which counts the number of times of bits alternation in a bit string. As the strings are randomized, there is a possibility of local maxima/minima.

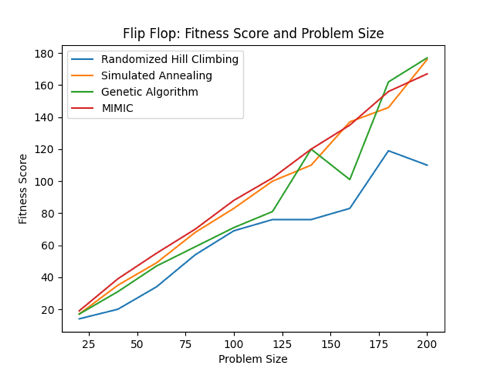
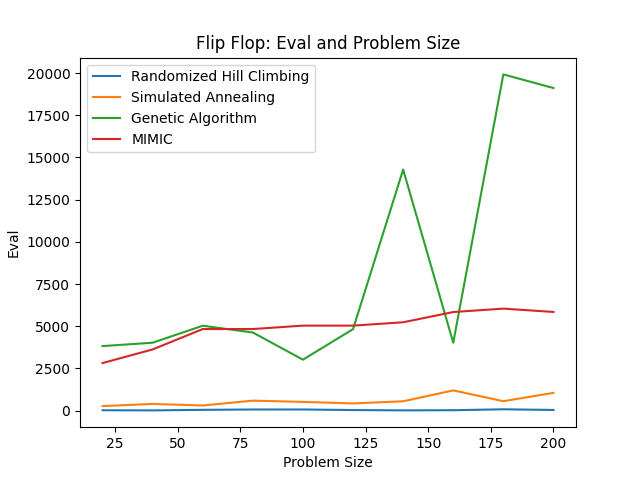
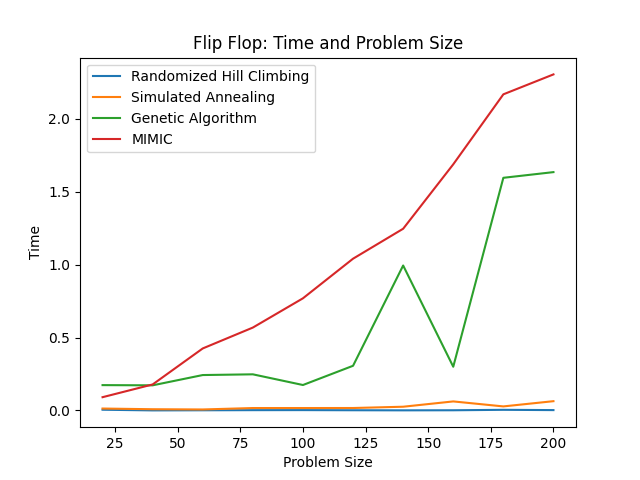
 

Fig 1: Flip flop optimization problem: fitness score, time and function eval vs problem size

I ran the experiment of assessing the fitness score, time taken, and function evaluation for the Flip Flop optimization problem with varying problem sizes. MIMIC seems to be beating others frequently at different problem sizes. However, it does take more time and function evaluations comparatively. Simulated Annealing is performing decently at different problem sizes. Genetic Algorithm is more volatile and Random Hill Climbing gives poor results. So, MIMIC is preferred for this problem.

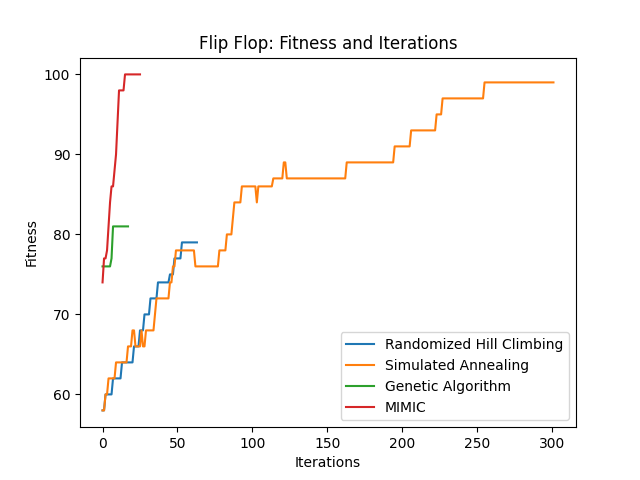
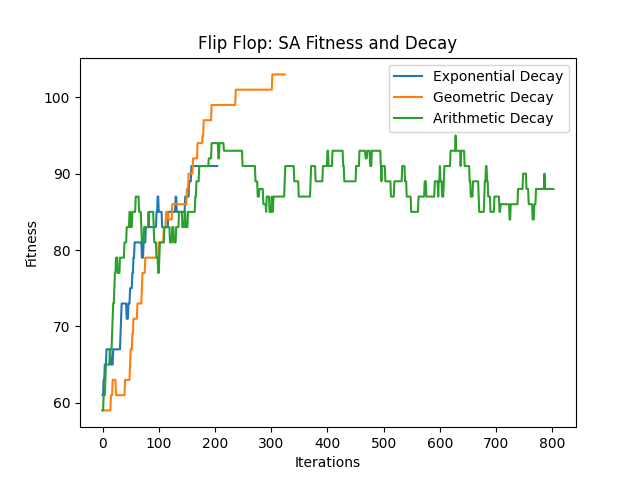


Fig 2: Flip Flop: Fitness Score vs Iterations

From the above chart, it is evident that both MIMIC and Genetic Algorithm takes very less iterations to converge but Genetic Algorithm is converging at a very low score. Simulated Annealing reaches a good fitness score steadily with increasing iterations. Random Hill Climbing performs poorly.

Chart, line chart

Description automatically generated

Chart, line chart

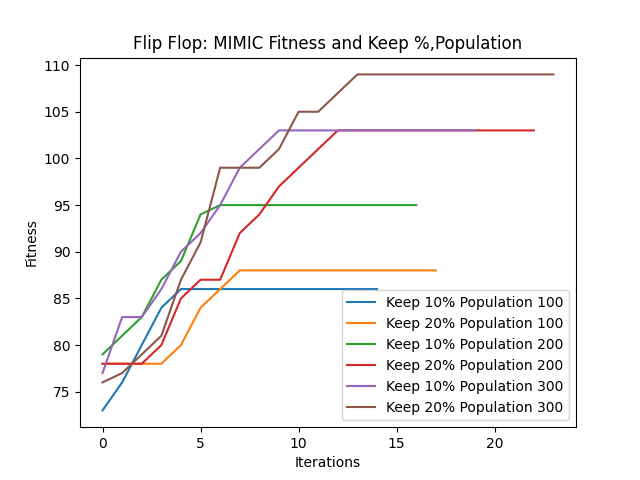
Description automatically generated

Fig 3: Flip Flop: Varying hyper-parameters

For each of the algorithms, I tried varying the hyper-parameters and the charts were generated. For Simulated Annealing, I varied the schedule with different types of decay: Geometric, Exponential and Arithmetic with the default value of decay. Geometric Decay of Temperature reaches the global optima, whereas the Exponential Decay and Arithmetic Decay gets stuck in local optima. The Arithmetic Decay reduces the temperature extremely slow and Exponential Decay reduces the temperature vigorously. So, both might miss or may not achieve the global optima. The Geometric Decay reduces the Temperature in a decent speed and might achieve optima. For Random Hill Climbing, I varied it with number of restarts. The restarts help in evading the local optima. Low restarts value might get tapped in local optima, whereas extremely high value of restarts may miss the attraction basin. For the Flip Flop optimization problem, restarts of 8 achieves higher fitness score. The population (hypothesis) size and mutation probability were varied for Genetic Algorithm. Higher the population size and mutation probability, more the variability within the hypothesis. Low value of population and mutation probability may not reach the attraction basin because of its slow change. Mutation probability of 20% and population size of 200 gives better fitness for this problem. For MIMIC algorithm, I varied the population (hypothesis) size and keep percentage (what proportion of samples to retain at each iteration of algorithm). MIMIC gives the best fitness with population size of 300 and a keep percent of 20%.

**N Queens problem**: N Queens is an optimization problem of placing N Queens in a NxN chess board such that no two queens threaten each other. No two queens share the same row, column or diagonal. The number of arrangements at which this can happen is going to be minimal. Hence, we would have global optima with a very narrow basin of attraction.

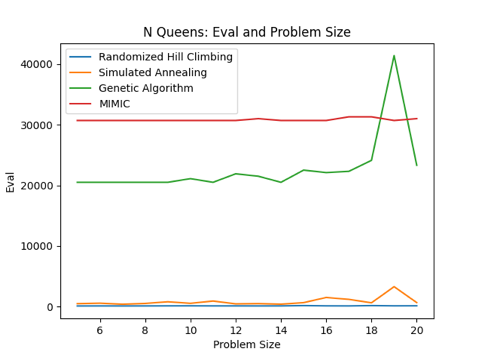
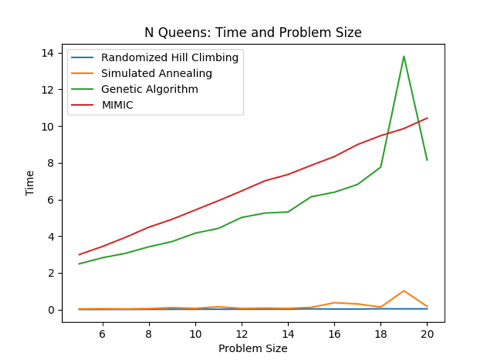
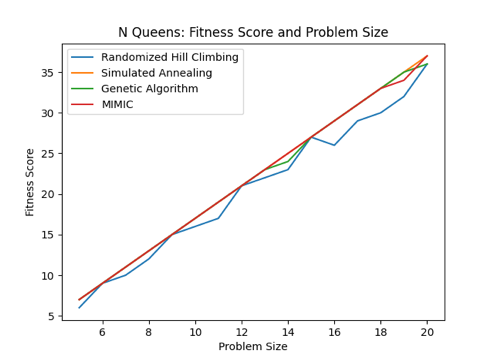


Fig 4: N Queens: Fitness score, Time, Function Eval vs Problem Size

Fitness score, Time and Function eval curve was plotted for varying problem sizes of N Queens optimization problem. From the fitness score curve, Simulated Annealing and MIMIC seems to be doing well at different problem sizes. However, looking at the time and function evaluation, it is evident that Simulated Annealing beats MIMIC as the former takes less time and less function evaluations. Simulated Annealing, because of its ability to explore the space than just moving uphill, can evade the local optima. Hence, it can perform as good as MIMIC. However, MIMIC is computationally expensive. Hence, Simulated Annealing would be preferred for this optimization problem.

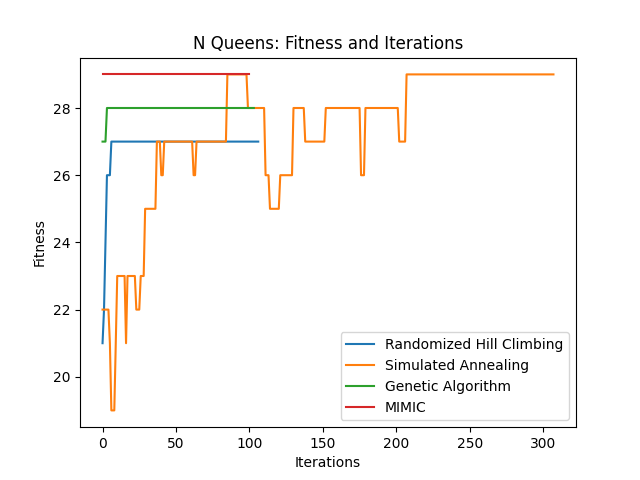


Fig 5: N Queens: Fitness Score vs Iterations

The above chart was generated for 16 Queens problem. Clearly, MIMIC and Simulated Annealing are achieving a good fitness score and possibly getting the global optima. Random Hill Climbing and Genetic Algorithm might have been stuck in local optima, hence having lower fitness scores.

Chart, line chart

Description automatically generatedChart

Description automatically generated

Chart

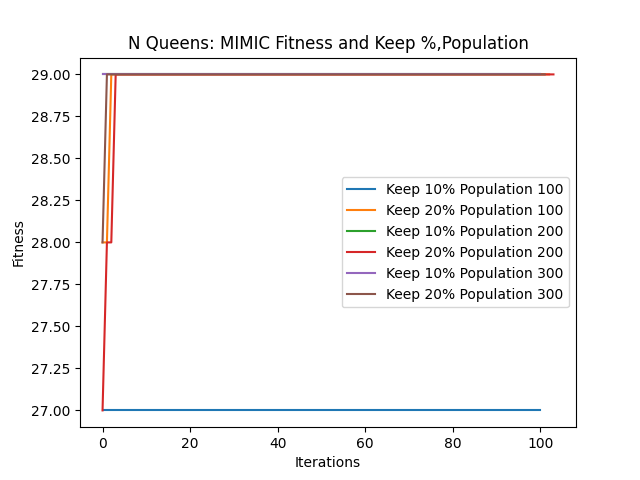
Description automatically generated with medium confidence

Fig 6: N Queens: Varying hyper-parameters

I tried varying different hyper-parameters for each of the algorithm. For Simulated Annealing, Arithmetic Decay might be getting tapped in local optima regions because of the slow decay in Temperature, hence we see lot of variability even after many iterations. On the other hand, Exponential and Geometric Decay is taking fewer iterations to reach the optima. Geometric Decay achieves optima tad bit faster than Exponential Decay. For Random Hill Climbing, restart of 10 achieves the optima fastest indicating that the algorithm needs higher number of restarts to reach the attraction basin. I varied the mutation probability and population size for Genetic Algorithm. Higher the population size and mutation probability, more the variability within the hypothesis, but extremely high value might make the movement more abrupt and might miss the attraction basin. A mutation probability of 10% and population size of 200 works well for this problem. For MIMIC, I varied both keep % and population size. It turns out that it achieves global optima for almost all configurations. However, a low value of keep % (10%) and population size of 100 might be tapped in local optima.

**Four Peaks**: Four Peaks is a simple toy problem and is good for testing algorithms. The fitness consists of counting the number of zeros at the start and ones at the end and returning the maximum. It is a problem with two local optima with wide basins of attraction, and two sharp global optima at the edges. So, achieving global optima would be tough for this problem.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Fig 7: Four Peaks: Fitness Score, Time, Function Eval vs Problem Size

For the Four Peaks optimization problem, I implemented different algorithms with varying problem sizes. Genetic Algorithm seems to be the clear winner at different problem sizes. Although, the Genetic Algorithm takes more time and function evaluations to reach optima.

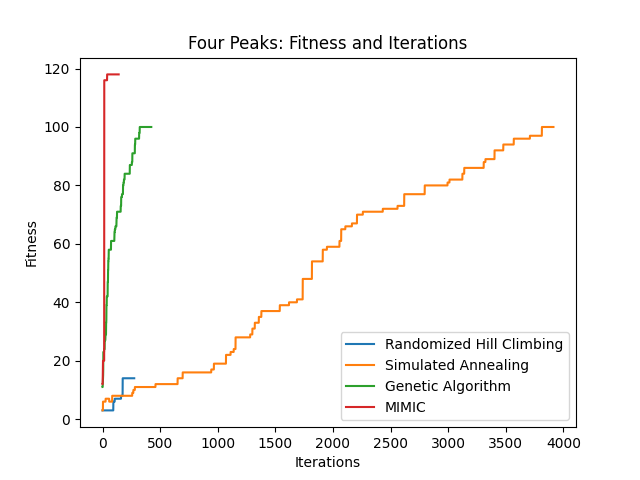


Fig 8: Four Peaks: Fitness Score vs Iterations

Genetic Algorithm and MIMIC reaches the optima quickly in fewer iterations. MIMIC’s fitness score is observed to be higher. Simulated Annealing also achieves a good fitness score, but it takes more iterations.

Chart

Description automatically generatedChart, histogram, box and whisker chart

Description automatically generated

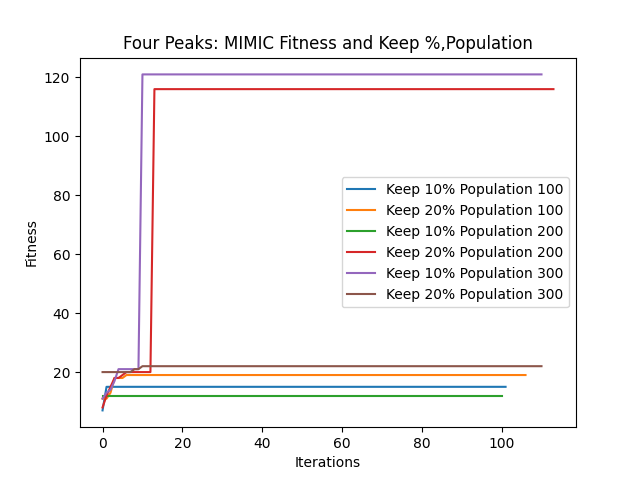
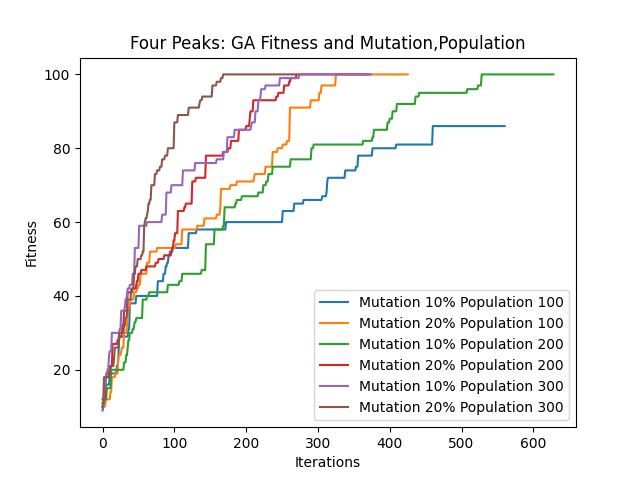


Fig 9: Four Peaks: Varying hyper-parameters

Different hyper-parameters for each of the algorithm were varied and analyzed. For Simulated Annealing, all the decays are behaving in tandem. However, Geometric Decay reaches optima in slightly lower iterations. For Random Hill Climbing, restart of 8 and 10 achieves the optima faster signifying that the algorithm needs higher number of restarts to reach the attraction basin. Mutation probability and Population size are varied to assess the fitness score. Higher population size and mutation probability increases the variability, and hence might achieve global optima. Mutation probability of 20% and population size of 300 achieves higher fitness score with lower iterations. For MIMIC, higher population size of 300 and a keep % of 10% gives the best fitness score indicating higher variability with a good population size may achieve global optima.

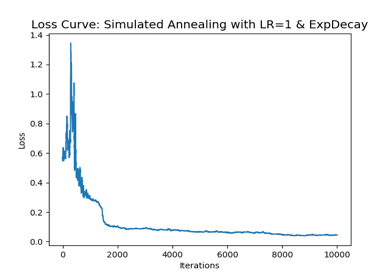
**Part 2: Neural Network using Randomized Optimization Algorithms**

We use the Random Hill Climbing, Simulated Annealing and Genetic Algorithm to generate weights for the Neural Network Classifier to be trained on Wisconsin Breast Cancer data. We are using mlrose to get the job done. The below description of data is borrowed from my report on Supervised Learning for CS 7641.

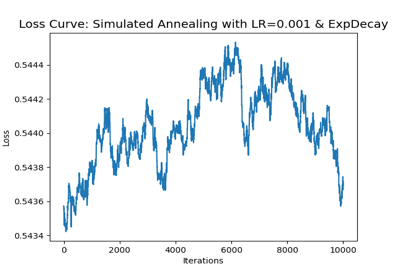
Breast Cancer Wisconsin (Diagnostic) Dataset: The dataset is sourced from UCI Machine Learning Repository. The examples are collected from the University of Wisconsin Hospitals, dealing in with the results from the diagnosis which are to be used to indicate if the person is diagnosed with Breast Cancer or not. The original dataset has 699 examples and 10 attributes excluding the predicted column (class). Class 2 indicates benign and 4 indicates malignant. I re-labelled the class 2 as 0 indicating absence of cancer cells and class 4 as 1 indicating presence of cancer cells. The class distribution is as follows: 65.52% absence of cancer cells (0) and 34.48% presence of cancer cells (1). The column “Bare Nuclei” had 16 missing instances. Since, this is breast cancer problem and we do not want our model to wrongly classify, I didn’t impute the missing values of “Bare Nuclei”. Instead, I dropped those 16 examples. All these features are numerical and are on different scale. Hence, normalized the features to bring it to a similar scale. I used the Standard Scaler in this data. I would be using the Recall metric to select the best classifier.

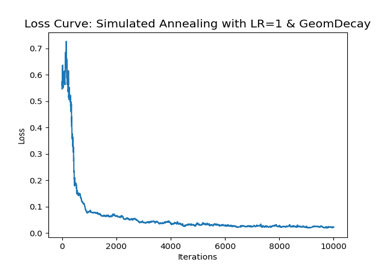
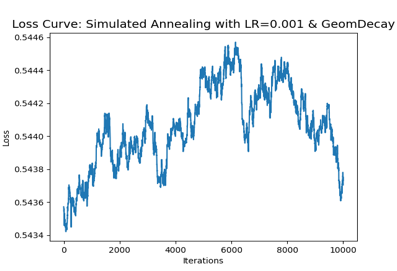
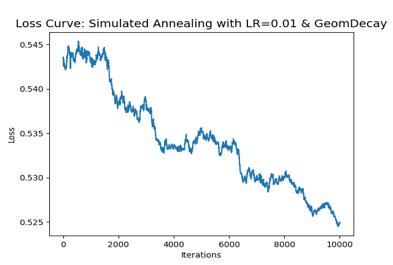
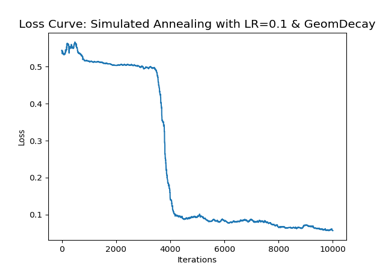
**Simulated Annealing**:

For determining weights using Simulated Annealing, I ran the experiment by varying the learning rate and decay type (with the default decay value). The loss curves and classification reports for each of the runs were generated. Since this is a small dataset with a relatively simpler decision boundary, just relying on the Recall metric won’t be sufficient, so we will consider loss curves as well.

Chart, line chart, histogram

Description automatically generatedChart, scatter chart

Description automatically generated

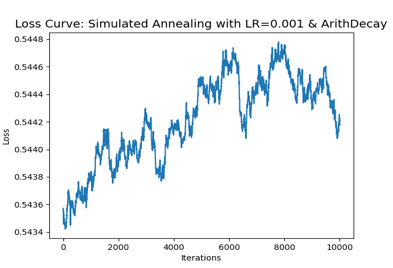
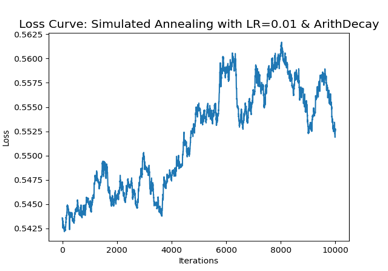
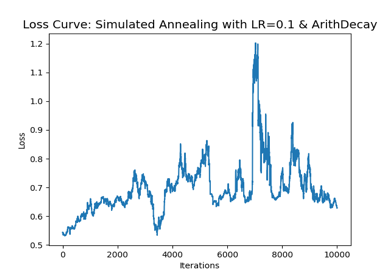
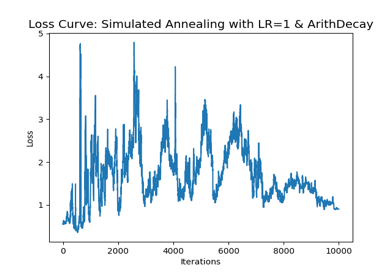


Fig 10: Loss Curves for each of the iteration for Simulated Annealing

Table

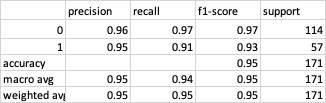
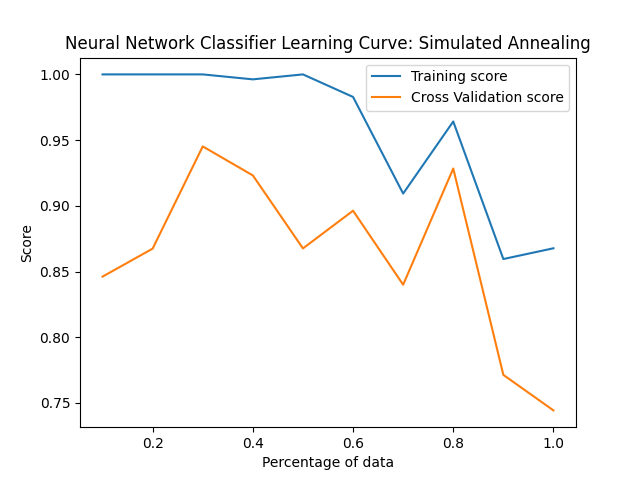
Description automatically generated

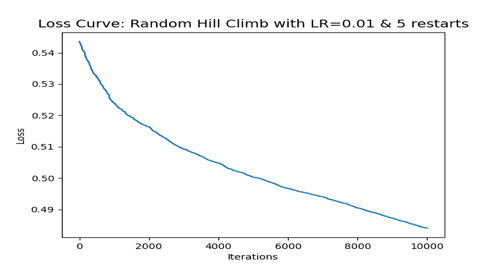
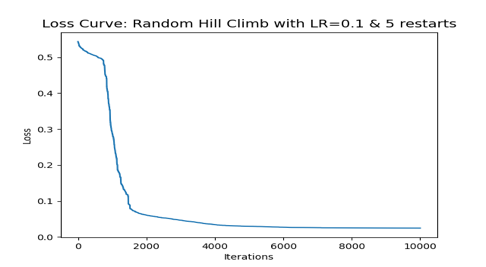
Fig 11: Learning Curve for Simulated Annealing

From the loss curves chart, it is evident that Arithmetic Decay doesn’t work well for this problem. One possible reason being the decay of Temperature such slow that it is not getting the optima. For both Exponential and Geometric decay, learning rate of 0.01 and 0.001 gives relatively high error as the learning becomes too slow (low change in weights) to get to the optima before the maximum iterations. Geometric Decay and a learning rate of 1 generates a better loss curve where the loss is reducing with iterations and nearing zero. We could have used the training recall and training time (recorded in the above table) to select the best model but this problem being a simple, relying on these metrics won’t be enough. With the Geometric Decay and Learning rate of 1, we generate a learning curve and the model’s classification report. Because of the size of data, we see variability in the cross-validation score. The training and cross-validation score were nearing convergence at 80% of data but diverged later indicating the need of more data to build a model with solid decision boundary.

**Random Hill Climbing**:

For Random Hill Climbing, I ran experiments by varying the learning rate and the number of restarts. Similarly, we will rely on the loss curves as well for selecting the best model.

Chart, line chart

Description automatically generated

Chart

Description automatically generatedChart

Description automatically generatedChart, line chart

Description automatically generated

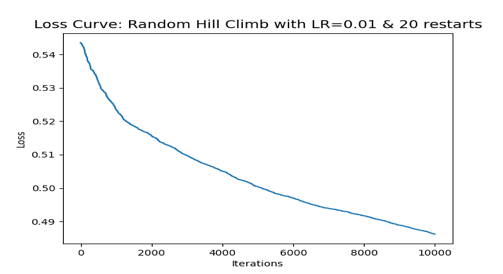
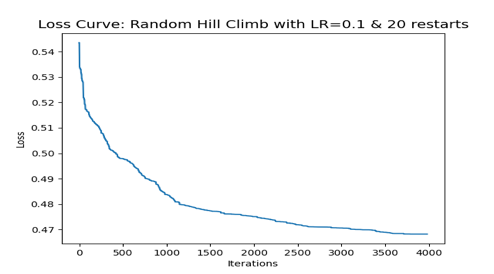
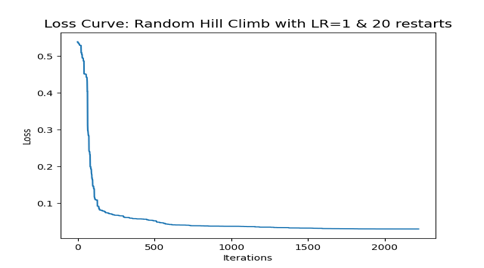
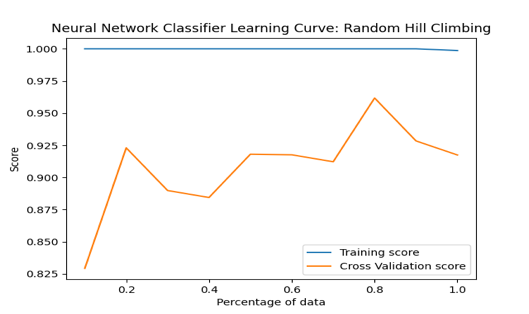


Fig 12: Loss curves for different configurations of Random Hill Climb

From the loss curves, it seems that we have huge error for learning rate of 0.1 and 0.01 (except the one with 0.1 rate and 5 restarts). Low number of restarts would converge in lower iterations, but it might get stuck in local optima. More the number of restarts, higher the training time. The training recall and training time for each run are recorded in the below table.

Table

Description automatically generatedTable

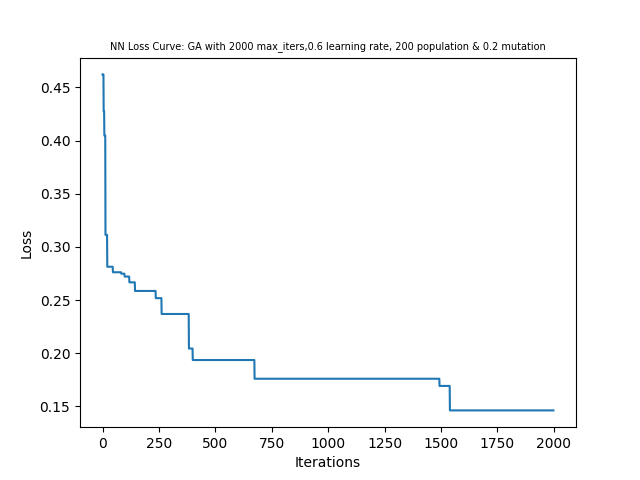
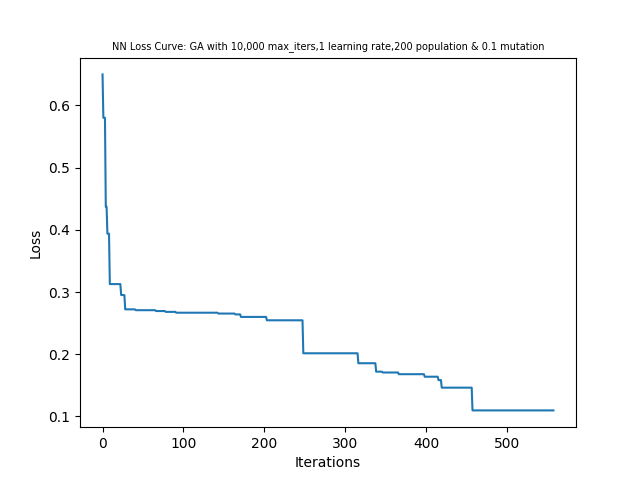
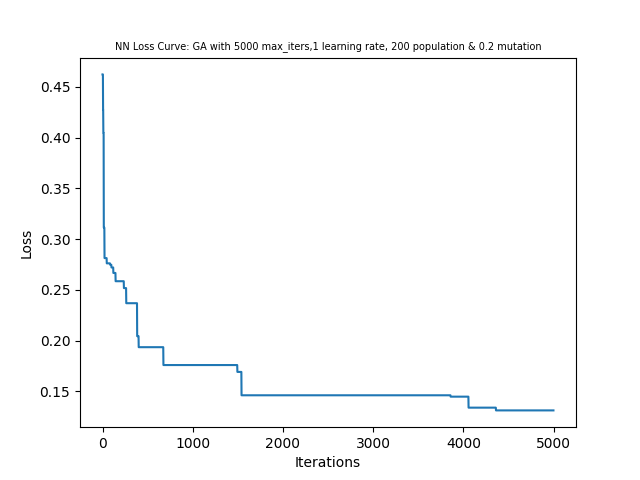
Description automatically generated

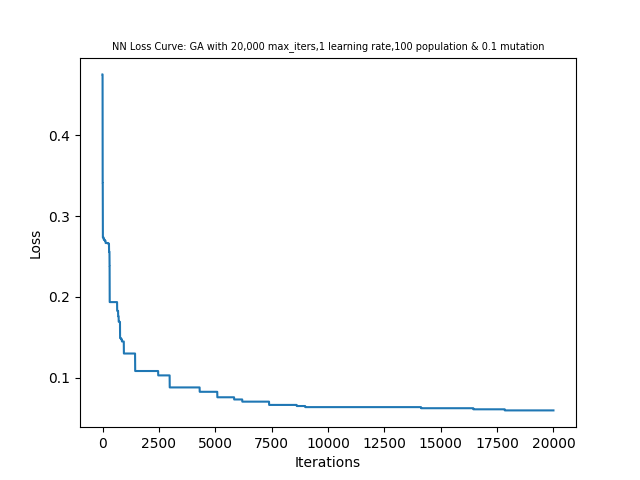
Fig 13: Random Hill Climb Learning Curve

As higher number of restarts avoids local optima, the one with 20 restarts and learning rate of 1 is selected. The learning curve and the classification report is generated for the model. From the learning curve, it looks like the curves are trying to converge, but haven’t achieved that. So, more data would be helpful.

**Genetic Algorithm**:

For Genetic Algorithms, I ran the experiment by varying the population size, mutation probability, learning rate and maximum iterations in a directional way.

Chart

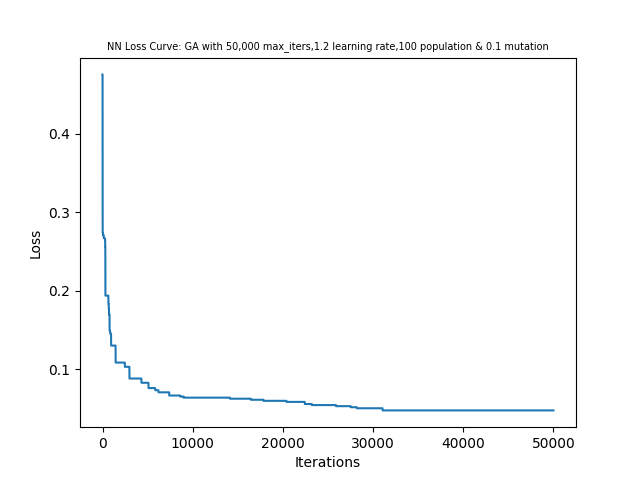
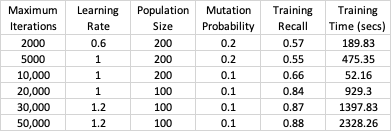
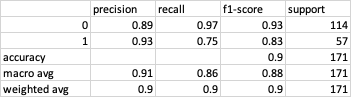
Description automatically generated

Fig 14: Loss curves for different configurations of Genetic Algorithm

Genetic Algorithm usually achieves global optima in lesser iterations, but the above charts tell a different story because of the type and size of data that is used. For this Breast Cancer data, it looks like higher population size and mutation probability do not yield good results. Possibly the higher variability because of higher population size and mutation probability leads to dramatic movement of hypothesis, and it is unable to reach the global basin of attraction which might be narrower. The movement is so abrupt that the maximum iterations had to be increased in order to achieve the global optima. Clearly, Genetic Algorithm is not doing well for this problem. Enhanced maximum iterations increases the training time. Also, the loss curves are smooth for higher iterations. The training recall value and training time are recorded for each of the run in the below table. Based on the loss curve, training time and training recall value, I selected the model with 20,000 maximum iterations, learning rate of 1, population size of 100 and 0.1 mutation probability. Learning curve and classification report was generated for this model.

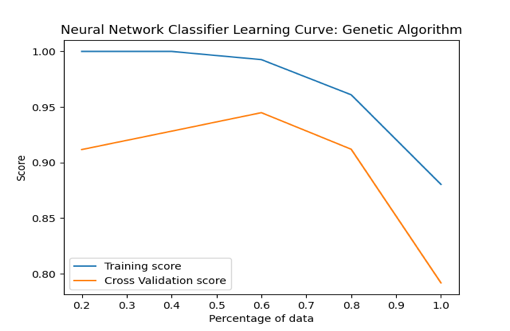


Fig 15: Learning Curve for Genetic Algorithm

The learning curve for the selected model was generated. The training and cross-validation curves are diverging after 80% of training data, indicating that model is not well fit.

**Comparison of Algorithms**:

The best selected model for each of the optimization algorithms are compared in the below table. Both Simulated Annealing and Random Hill Climbing gives good test recall value. However, Simulated Annealing is relatively faster. Random Hill Climbing, because of its high restarts (20 for our problem), takes more time to train. Because of high restarts, Random Hill Climbing is achieving the global optima and hence better results. Simulated Annealing, on the other hand, takes about one-third (for our Breast Cancer problem) of training time of Random Hill Climbing and gets nearly similar results. So, Simulated Annealing is the clear winner. Genetic Algorithm performs poorly for our problem because of vigorous change in hypotheses. Genetic Algorithm is extremely computationally expensive.

|  |  |  |
| --- | --- | --- |
| Optimization Algorithm | Test Recall | Training Time (secs) |
| Simulated Annealing | 0.91 | 7.35 |
| Random Hill Climbing | 0.93 | 22.5 |
| Genetic Algorithm | 0.75 | 929.3 |
| Gradient Descent | 0.98 | 0.16 |

Gradient Descent performed exceptionally well compared to the randomized optimization algorithms. Gradient Descent got trained in 0.16 seconds and got a test recall of 0.98 (sourced from Supervised Learning Assignment of CS-7641). Although Gradient Descent performed the best in our Breast Cancer dataset, it might get tapped in local minima. Backpropagation relies on its learning occurred during training, which might not be appropriate for new data. Simulated Annealing, on the other hand, would initiate a new search because of its randomized approach, and might converge in global minima.

**Conclusion**:

As we observed in this report, that different optimization problems suit different optimization algorithms. Before using a certain optimization algorithm for an optimization problem, it is recommended to understand the structure of the solution space and domain knowledge of the problem.

**References**:

1. UCI Machine Learning Repository – Breast Cancer Wisconsin (Diagnostic) Data Set : <https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic)>
2. Learning Curve: <https://www.scikit-yb.org/en/latest/api/model_selection/learning_curve.html>
3. <https://mlrose.readthedocs.io/en/stable/>