Randomized Optimization - Github Link

Joseph Waugh
Georgia Tech ID: 903563084

CS 7641: Machine Learning
Spring 2022

March 6, 2022

1 Description of Optimization Problem

This research consists of utilizing four local random search optimization algorithms: randomized hill climbing, simulated annealing, genetic algorithm, and MIMIC. Using these optimization algorithms, the following three optimization problems were tested:

- **Four Peaks:** An optimization problem where two local optima and two global optima at the edges of the range exist, and the goal is to identify the maxima via an optimization function.
- **Knapsack Problem:** With a given set of items (featuring a mass and a value), determine the selection of items so that the total value of the items is maximized while the weight is less than or equal to the capacity of the limit defined by the knapsack.
- **Flip Flop:** With a given bit string, the number of times of alterations in the given bits is counted from a given number to any number. A maximized bit string will consist of all alternating digits.

This paper will consist of an overview of each algorithm, performance in relation to the three optimization problems, and an application with a neural network infrastructure used for the previous assignment (Supervised Learning).

1.1 Algorithm Overview

1.1.1 Randomized Hill Climb

The randomized hill climb (RHC) algorithm is a search algorithm that aims to identify an optimal solution by continuously moving towards a maximization point until the best state is achieved based on the given range of data. The algorithm starts at a given non-optimized space, where the function then identifies a direction that moves towards the maximum value, thus optimizing the maximum value until an optimal value is achieved. The function will iterate until a given peak is identified; however, this function can potentially identify either a global or local maxima, given that finding any maximum value results in the function stopping since going in the reverse direction is also not allowed per the algorithm. While RHC can work well in continuous and discrete ranges of data, in addition to being able to identify optimal values per an objective function, there are distinct disadvantages of using this algorithm. As mentioned before, obtaining a local maxima instead of the true maximum value is shortfall with using this algorithm. In addition, if the data is a plateau (the search space is flat relative to the target), then the algorithm is not able to identify the optimal direction in which to move, given that the values in either direction are the same.

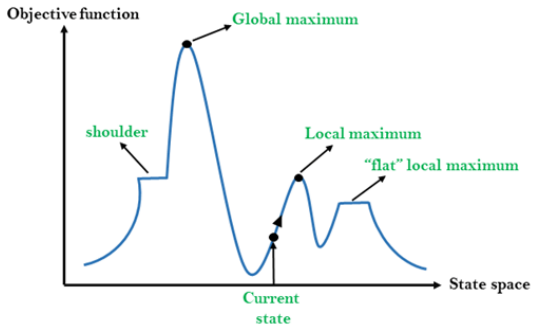

Figure 1: Randomized Hill Climb (RHC) Algorithm

1.1.2 Simulated Annealing

The simulated annealing (SA) algorithm is another search algorithm that operates in a similar fashion to RHC. SA also operates in a single-state format, where modifications to the output are based on a search of the nearby values, with the model moving towards the maxima; however, SA differs by allowing the stepwise function to traverse towards a worse solution (a future value with a lower value than the current state value) in the early stages of the search based on a given probability prior to performing a hill climb towards the maxima. The ability to traverse in a direction different than RHC allows the model to potentially avoid the shortfall of being limited to finding the local maxima instead of the true maxima. This method essentially guarantees an optimal solution, given that the function will first traverse the entire range prior to performing the hill climb portion of the function; however, this can be very computationally expensive and also the SA algorithm fails to identify whether an optimal solution has been identified. In addition, if the data is a plateau, SA also faces similar issues as RHC and thus has

to traverse in a potentially incorrect direction due to not understanding the immediate optimal path.

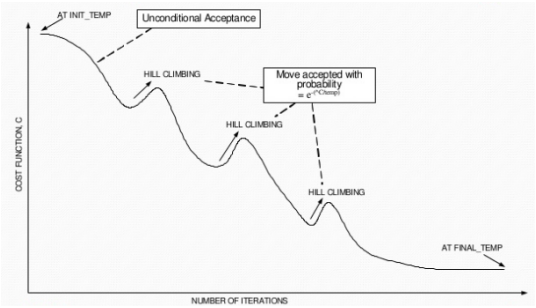

Figure 2: Simulated Annealing (SA) Algorithm

1.1.3 Genetic Algorithm

The genetic algorithm (GA) is another search algorithm that has strong similarities to the SA algorithm. GA utilizes random search to determine "parent nodes" as the origin for the search, and then uses these parents to identify "children nodes" for the next iteration of the search. These unique combinations of parent and child nodes then are measured in terms of the proximity towards the true maxima, with the population of parent and child nodes eventually moving towards the optimal maxima value. This algorithm has gained popularity due to the speed in comparison to RHC and SA; however, similar to the prior algorithms, there are still issues with the model identifying local maxima or failing to identify a maxima at all (i.e., plateaued data). To resolve this issue, GA utilizes crossover to take the optimal results from multiple parent nodes to shape the direction of a new child node.

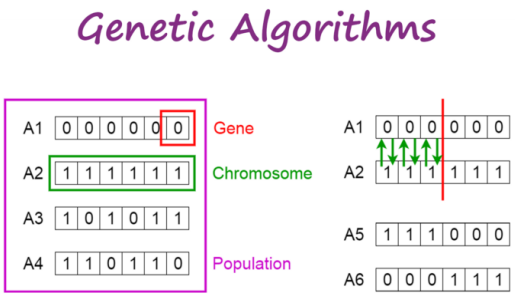

Figure 3: Genetic Algorithm (GA)

1.1.4 Mutual-Information-Maximizing Input Clustering (MIMIC)

The MIMIC algorithm utilizes prior iterations and uses probability densities to work towards a solution and identify the local maxima. This algorithm utilizes information regarding the associated optimization function search results to future search iterations, thus creating the situation where a gradually weighted threshold forces a solution towards the local maxima. This function is computationally expensive, given the need for calculating conditional probabilities and mutual information; however, the obtained results have been proven to converge quickly using high cost functions.

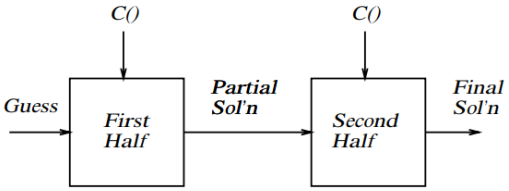

Figure 4: MIMIC Algorithm

2 Results

2.1 Algorithm Results

2.1.1 Randomized Hill Climb

Overview:
The randomized hill climb algorithm works well at improving the fitness of a given function based on testing neighbors and moving towards the optimal region; however, this algorithm struggles in situations where a local optima is reached instead of the maxima. To alleviate this issue, a randomized version of the algorithm is used, where the origin is randomly selected.

Optimization Problem - Flip Flop:
In the Flip Flop optimization problem, 50 random restarts were used with 100 max attempts in order to avoid the issue of finding only local maxima values. The algorithm took a total of 1,376 seconds (13.63 seconds avg.), and there were 255,000 iterations in total (5,000 per step). For each iteration, the algorithm appears to be stuck in terms of the fitness value. This is likely due to the local minima being found despite the random origins. The function settles at an average fitness value of 318.9.

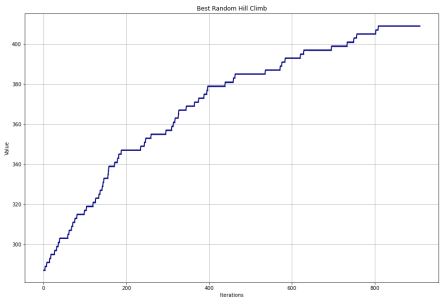

Figure 5: Flip Flop - RHC Fitness

Optimization Problem - Four Peaks:
In the Four Peaks optimization problem, 50 random restarts were also used with 100 max attempts to avoid finding local maxima. The algorithm took a total of 13.63 seconds (0.48 seconds avg.), and there were 70,000 iterations in total (5,000 per step). This algorithm performed poorly in terms of not being able to escape local minima, which is reflective in the avg. fitness score of 4.5 after 5,000 iterations.

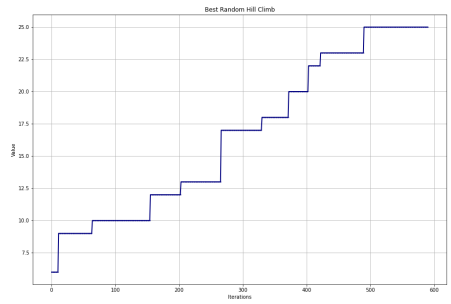

Figure 6: Four Peaks - RHC Fitness

Optimization Problem - Knapsack:
In the Knapsack optimization problem, 50 random restarts were used with 100 max attempts to avoid finding local maxima. The algorithm took a total of 20.25 seconds (0.20 seconds avg.), and there were 255,000 iterations in total (5,000 per step). This algorithm performed the best among the RHC fitness functions in terms of escaping local minima, which is reflective in the avg. fitness score of 1,473 after 5,000 iterations. This application appears to pleateau, therefore suggesting that the performance won't increase due to the presence of local optima that don't allow the randomized optimization function to proceed any further.

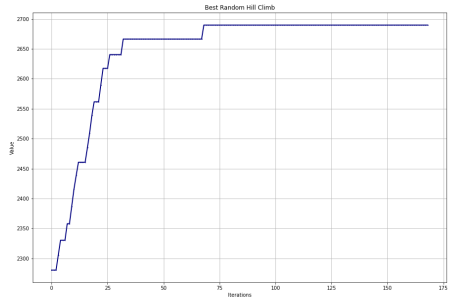

Figure 7: Knapsack - RHC Fitness

2.1.2 Simulated Annealing:

The simulated annealing algorithm aims to reduce the issues viewed with the random hill climb algorithm, by allowing the model to step backwards in order to escape the local maxima and find the true optima values. The results shown in each of the three algorithms show that the SA model performs well in reducing this issue. For the Simulated Annealing algorithm, both the Geometric and Exponential temperature decay functions were used via a GridSearch method, along with 5 different parameters for initial temperature (1, 10, 50, 500, 5000). These resulted in 10 different SA algorithms to be tested for each of the three optimization problems.

Optimization Problem - Flip Flop:
For the Flip Flop optimization problem, the avg. fitness value that was achieved was 360.8. The algorithms took 35.6 seconds to train (1.18 seconds avg.), thus showing an increase in runtime compared to RHC. For this problem, the geometric temperature decay, with a schedule initial temperature value of 1 (optimized to 0.6) resulted in the highest fitness value at 464.

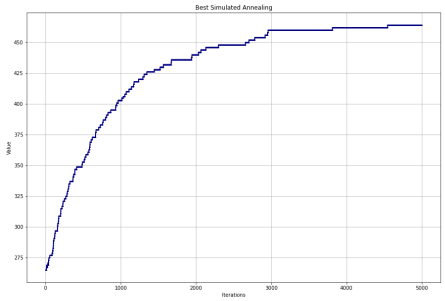

Figure 8: Flip Flop - SA Fitness

Optimization Problem - Four Peaks:
For the Four Peak optimization problem, the simulated annealing algorithm struggled to optimize the fitness score, with a maximum score of 100.0 achieved. The algorithm took 276.2 seconds to train (27.6 seconds avg.), still showing an increase over RHC. The optimal parameters for this function included a geometric temperature decay and a scheduled initial temperature value of 1 (optimized to 0.77). The function parameters are almost identical to those for the Flip Flop optimization problem; however, the presence of local minima appears to still be playing a major role in the poor performance of this model. Despite having a maximum of 5,000 iterations, the model struggles to identify a maximum value even with the ability to move in the opposite direction based on the temperature decay. To potentially increase the performance, altering the temperature decay values to increase the range may potentially support the model's ability to escape the local minima; however, this is not a guarantee, given the poor performance across a wide range of values (1 - 5000) for temperature decay.

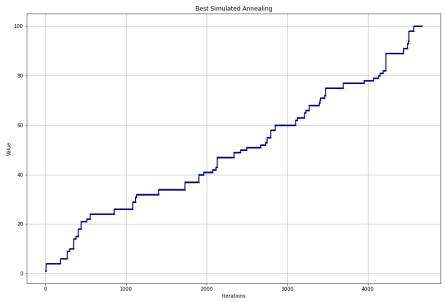

Figure 9: Four Peaks - SA Fitness

Optimization Problem - Knapsack:
For the Knapsack optimization problem, the simulated annealing algorithm appeared to perform well in terms of achieving an optimal fitness score. The maximum fitness score achieved in 20 iterations (maximum of 5,000 iterations using 5 different values for scheduled temperature decay (1, 10, 50, 500, 5000)). The optimal parameters included utilizing all 5,000 iterations with an exponential temperature decay initialized at 5,000 (optimized to 4999.98). The performance appears to be slightly higher than all other trained models; therefore, the tuning of these parameters may not have a huge weight, unless a significant change to the input data was made.

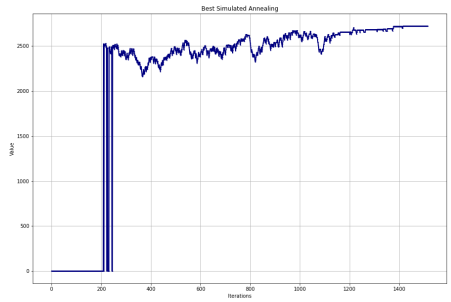

Figure 10: Knapsack - SA Fitness

2.1.3 Genetic Algorithm

Overview:
For the genetic algorithm, I utilized 2 different variations of population size (500, 1000) and three different variations of mutation rates (0.1, 0.25, 0.5) for a total of 6 trained models per each optimization problem.

Optimization Problem - Flip Flop:
For the Flip Flop problem, a maximum fitness score of 454 was achieved, which again shows an increase in performance compared to random hill climb and simulated annealing. The optimal parameters that were utilized were a population size of 500 and mutation rate of 0.5; however, the results obtained were very similar in terms of the proximity to the maximum fitness score. These results are shown below:

Flip Flop Parameters			
Population Size	Mutation Rate	Fitness	Time
500	0.5	454	43.98
1000	0.25	452	62.34
1000	0.5	452	66.63
1000	0.1	451	87.47

Figure 11: Flip Flop - GA Fitness Scores

The primary driver in higher run times appears to be population size, with minimal change to the overall fitness score. In order to improve results shown below, a likely next step could be to include a wider range of population sizes, as this could either continue to improve run times (and thus use this computational power to increase the number of iterations), or increase the population size to allow for a more generalized estimate of how the model will react to additional data.

Optimization Problem - Four Peaks:
For the Four Peaks problem, the Genetic Algorithm performed poorly across all iterations, regardless of tuning parameters. The maximum value was achieved at 5,000 iterations for all combinations of tuning parameters at a fitness score value of 189. The key driver here appears to be the number of iterations, in terms of pushing this number higher; however, the presence of local minima in addition to relatively low mutation score variants (tested 0.1, 0.25, and 0.5) could lead to the model not being able to escape the local optima values, similar to many of the shortfalls in RHC and SA that have been observed thus far. These results are shown below:

Four Peaks Parameters			
Population Size	Mutation Rate	Fitness	Time
1000	0.1	189	0.04
500	0.5	189	29.39
500	0.25	189	31.55
1000	0.5	189	86.25

Figure 12: Flip Flop - GA Fitness Scores

Similar to before, there's a wide range in terms of runtimes for the different parameters with the same fitness score. To move beyond this, an increase in the number of iterations would support the chances of the model escaping a local minima (though it's not guaranteed); however, increasing the mutation rate would also potentially allow for a greater chance at escaping local minima, but would still introduce variance between runs and potentially would result in either escaping a maxima range unintentionally, or moving towards other local minima within the region.

Optimization Problem - Knapsack:
The Genetic Algorithm performed moderately well compared to random hill search and simulated annealing, with an average fitness score achieved of 3,224. This model also contains a very tight window of values based on the tuning parameters included, with a significant change showing in terms of runtimes. These values are shown below:

Knapsack Parameters			
Population Size	Mutation Rate	Fitness	Time
1000	0.1	3224.5	33.06
1000	0.25	3224.5	36.04
500	0.5	3224.4	23.50

Figure 13: Knapsack - GA Fitness Scores

Again, modifications to parameters can be attempted in order to reduce computational complexity (i.e., reducing population size and achieving similar fitness score,

using additional computational complexity to perform additional iterations); however, it appears this model is limited in terms of functionality regarding the current dataset provided.

2.1.4 MIMIC

Overview:
For the MIMIC algorithm, I utilized two variants for population size (500, 1000) and three parameters for percentage retained (0.1, 0.25, 0.5). Therefore, 6 models were created to train on each of the three optimization problems.

Optimization Problem - Flip Flop:
The MIMIC algorithm performed well in the Flip Flop problem, achieving a maximum fitness score of 444 (avg. score of 433) over. The outputs appear to show that there isn't a clear linear trend with population size and retained percentage, as the top two values show alternate trends: higher population size & lower kept percentage, and vice-versa; however, by utilizing a high population size with a low kept population size, the model appears to a regression in terms of fitness score. These values are shown below:

Flip Flop Parameters			
Population Size	Retained Percentage	Fitness	Time
1000	0.1	444	1267.09
500	0.5	438	897.99
500	0.1	265	0.02
1000	0.5	265	0.14

Figure 14: Flip Flop - MIMIC Fitness Scores

Optimization Problem - Four Peaks:
The MIMIC algorithm performed moderately well again for FourPeaks optimization problem. In tuning the model results, the trend that was discovered was a higher population size led to a better overall fitness score, whereas a clear trend couldn't be established for retained percentage values. Therefore, this suggests that the greater amount of data used to generate population distributions for this algorithm leads to higher fitness results. The top fitness value obtained was 155 with a runtime of 271.5 seconds. Similar results are shown below:

Four Peaks Parameters			
Population Size	Retained Percentage	Fitness	Time
1000	0.25	155	271.50
1000	0.1	146	138.02
500	0.1	132	66.18

Figure 15: Four Peaks - MIMIC Fitness Scores

The potential improvement that can be applied here is using an increased population size in order to maximize the fitness score; however, this would be more computationally expensive as well, based on the trends with the runtimes in the grid search.

Optimization Problem - Knapsack:
THe MIMIC algorithm performed very well for the Knapsack optimization problem. The maximum fitness score earned was 3237.75 with a runtime of 326.37 seconds. The ideal parameters to maximize the fitness score included a population of 1000 and a kept percentage value of 0.1. Interestingly, the lower retained population values appeared to have a better fitness curve. Therefore, in order to maximize the fitness, a couple of options could be exercised. Given that a lower retained percentage value results in a higher fitness score, one could test smaller increments of this parameter; however, it would be important to ensure that additional data is provided to the model in order to accurately generalize the dataset for proper fitting. The results of MIMIC for the Knapsack problem are shown below:

Knapsack Parameters			
Population Size	Retained Percentage	Fitness	Time
1000	0.1	3237.75	326.36
1000	0.25	3235.33	251.24
1000	0.5	3217.53	266.15

Figure 16: Knapsack - MIMIC Fitness Scores

3 Analysis of Results

3.1 Flip Flop:

In the Flip Flop optimization problem the following comparative results were achieved:

Flip Flop Problem				
Metric	RHC	SA	GA	MIMIC
Avg. Fitness	318	361	356	339
Max Fitness	409	464	454	444
Avg. Time(s)	7.91	18.1	27.6	1267.9

Figure 17: Flip Flop - Algorithm Metric Comparison

Based on these results, the **Simulated Annealing** algorithm is the most successful based on avg. fitness and max fitness, in addition to have the second quickest runtime. This advantage for Simulated Annealing is the opportunity to step backwards in terms of the fitness function, which allows the algorithm to escape local maxima. Conversely, the algorithm that can't perform this "backward step", randomized hill climbing, performs the worst with an average fitness score of 4.55. This is also reflective with a low average run time, given that the algorithm doesn't have much room for improvement by only being able to travel in a single direction based on a random origin. Genetic Annealing and MIMIC perform similarly to Simulated Annealing, with the primary difference being larger run times with slightly smaller fitness scores. MIMIC likely performs very well due to utilizing the density estimator from successful iterations of the algorithm,

whereas the Genetic Algorithm is similar in terms of combining successful iterations based on a given population size. With the optimal parameters of population size being higher, this will lead to higher run times of the algorithm as well.

3.2 Four Peaks:

Four Peaks Problem				
Metric	RHC	SA	GA	MIMIC
Avg. Fitness	4.55	44.7	95	50.8
Max Fitness	25	100	189	155
Avg. Time(s)	3.74	13.8	26.5	65.9

Figure 18: Four Peaks - Algorithm Metric Comparison

Based on these results, the **Genetic Algorithm** performed the best based on the best avg. fitness and max fitness. This algorithm appeared to perform well based on the mutation rates allowing consecutive iterations with good performance to continue improving the model. Nonetheless, these consecutive iterations are computationally expensive, which is reflected in the run time being higher compared to two of the three remaining models. In terms of similar performance, the MIMIC algorithm appeared to work well but at roughly three times as many seconds due to the need to determine multiple densities for all iterations. Simulated annealing and Random Hill Climb appear to perform worse, likely due to an inability in escaping local minima.

3.3 Knapsack:

Knapsack Problem				
Metric	RHC	SA	GA	MIMIC
Avg. Fitness	1473.9	1290.8	1611.9	2817.2
Max Fitness	2689.6	2611.9	3224.5	3237.7
Avg. Time(s)	6.9	0.18	16.6	109.9

Figure 19: Knapsack - Algorithm Metric Comparison

Based on these results, the **MIMIC algorithm** performed the best in terms of average fitness and maximum fitness. This is likely a result of the multiple iterations of samples that are generated based on the density estimator, which we can also see its' impact in terms of a significantly longer run time compared to the other algorithms. Interestingly, the Genetic Algorithm performs similarly in terms of the max fitness score, whereas the average fitness score is significantly less compared to the MIMIC algorithm. This can likely be attributed to the similar function of combining multiple states with a wide degree of variance to reach a successful result. This would likely be the case in terms of bringing the average fitness score down, but still allowing for a successful maximum fitness score to be determined.

4 Optimized Neural Network

4.1 Overview

After determining the optimal parameters across the three different optimization problems (Flip Flop, Four Peaks, Knapsack), these parameters have been fed into a Neural Network utilizing an Employee Attrition dataset from IBM, which was utilized for Assignment 1. In order to prepare the dataset, several data manipulation steps were performed to ensure the data was in the correct format. First, all categorical variables were converted into dummy variables, to ensure that all data is represented numerically. From there, due to the fact that neural networks have high sensitivities to non-scaled data, a scaler function was applied to normalize all of the input data. The data was then split into a training and test split via a random stratified sample, in order to ensure that the target variable, "Attrition" (indicates that an employee had left their position in the company), is equally represented across all samples.

4.2 Gradient Descent

To provide a means of comparison against the three optimization algorithms, gradient descent was first tested for the neural network.

4.2.1 Fitness Curve

The associated fitness curve appears to show a maximized fitness score after roughly 300 iterations.

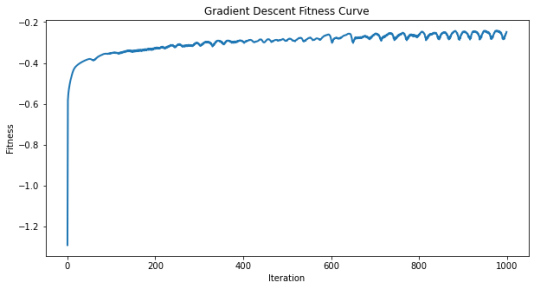

Figure 20: Gradient Descent - Fitness Curve

4.2.2 Precision-Recall

The accuracy-precision F1 score shows a value of 0.67, which appears to be a solid score in terms of predictive ability.This shows that the natural convergence towards local optima works well.

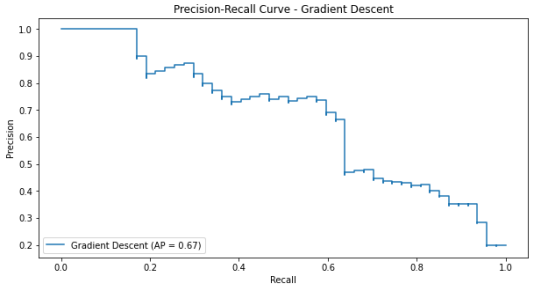

Figure 21: Gradient Descent - Precision Recall

4.2.3 Learning Curve

The associated learning curve shows a cross-validation score of 0.85, whereas the training score is 0.85 as well. These values do converge, and likely benefited from additional iterations to ensure both subsets of data are equally representative in terms of the model's predictive ability.

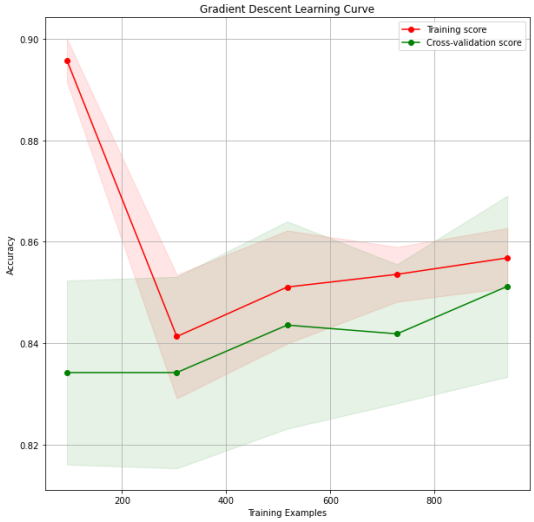

Figure 22: Gradient Descent - Learning Curve

4.2.4 Confusion Matrix

The confusion matrix shows that non-attrition records heavily outweigh the number of employee records where attrition took place; however, the relative accuracy of predicting non-attrition and attrition employees is high.

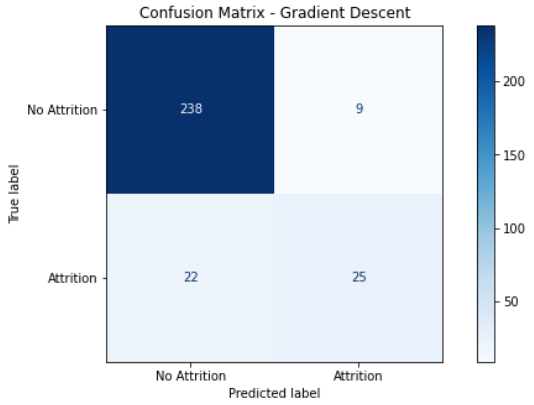

Figure 23: Gradient Descent - Confusion Matrix

4.2.5 Scalability

The model shows nearly a linear trend in terms of training samples and time required for training. Therefore, for the training and validation datasets, one could continue to feed additional data into the model to be sure the learning curve results converge.

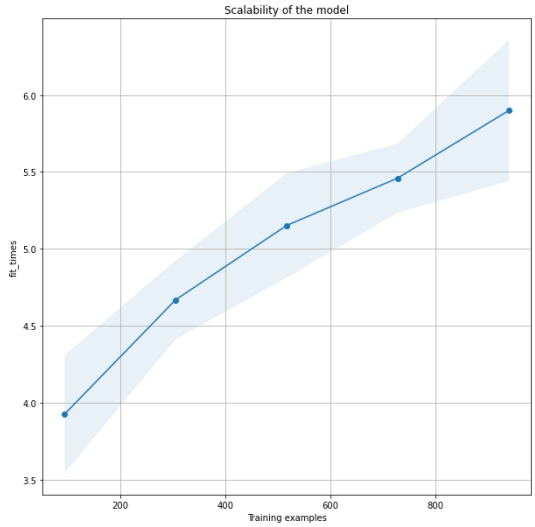

Figure 24: Gradient Descent - Scalability

4.3 Random Hill Climb

This model was tested with 3 initial learning rates (0.001, 0.01, 0.1) and 3 restart parameters for a total of 9 models to train in a GridSearch method. The maximum achieved mean cross-validation score was 0.55. The optimized parameters included an initial learning rate of 0.1, with a maximum of 2500 iterations and 0 restarts.

4.3.1 Precision-Recall

The accuracy-precision F1 score shows a value of 0.5, which indicates worse performance compared to gradient descent. The convergence to local optima, in addition to the inability to step initially into a worse direction for the optimization function likely results in the worse performance compared to gradient descent.

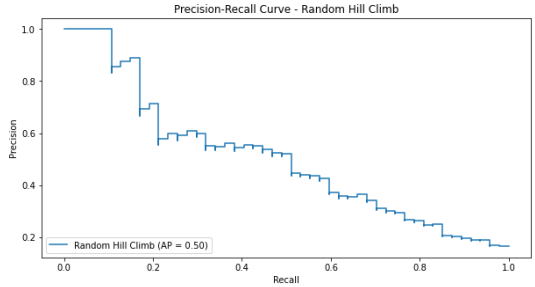

Figure 25: Random Hill Climb - Precision Recall

4.3.2 Learning Curve

The learning curve results converge towards 0.85; however, these curves are not fully converged after 1,000 training examples. This trend suggests that the model's predictive ability fits both the training and test validation set well.

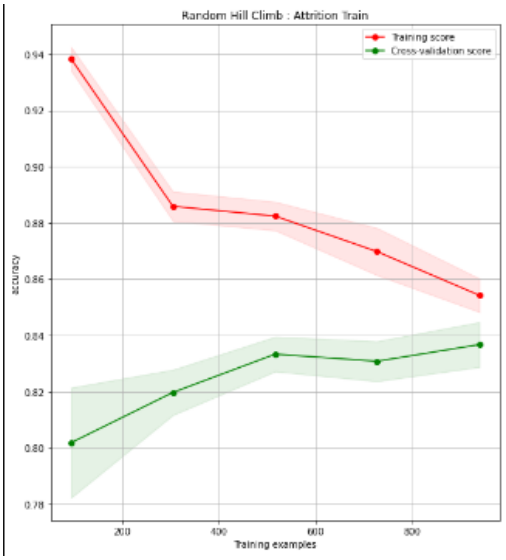

Figure 26: Random Hill Climb - Learning Curve

4.3.3 Confusion Matrix

In looking at the confusion matrix results, the model does well in comparing the non-attribution results; however, the model performs significantly worse in terms of predictive ability of the *attrition* label, given that only 11% of values were accurately predicting for this label (42 of 47 instances).

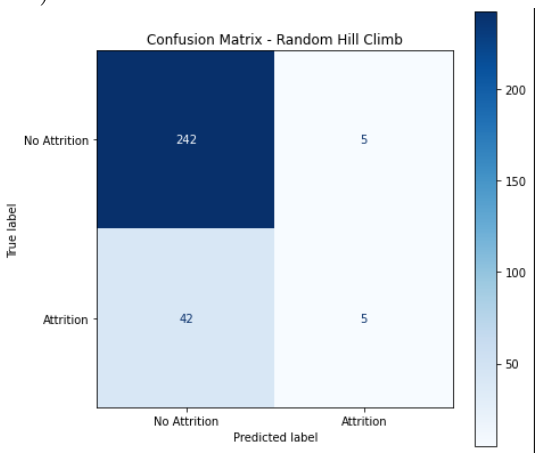

Figure 27: Random Hill Climb - Confusion Matrix

4.3.4 Scalability

The model also shows nearly a linear trend in terms of training samples and time required for training. Therefore, for the training and validation datasets, one could continue to feed additional data into the model to be sure the learning curve results converge and potentially shift the overall accuracy higher.

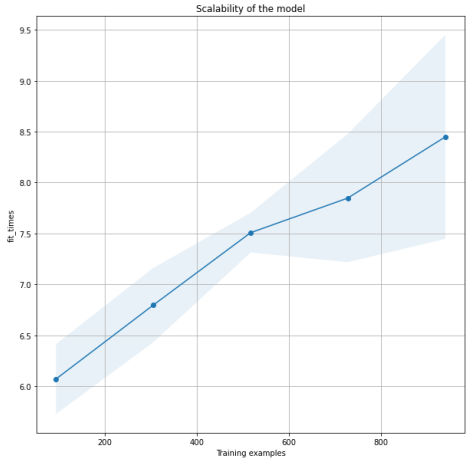

Figure 28: Random Hill Climb - Scalability

4.4 Simulated Annealing

This model was tested with two values for a maximum number of iterations (1000, 2500), three initial learning rates (0.001, 0.01, 0.1), and three parameters for number of restarts (0, 5, 10). Collectively, these parameters were tested via a GridSearch method for training for a total of 18 trained models. The optimally selected parameters included 1000 maximum iterations, an initial learning rate of 0.1, and 0 restarts.

4.4.1 Precision-Recall

The accuracy-precision F1 score shows a value of 0.23. This model performs much worse compared to the random hill climb algorithm and to gradient descent. Simulated annealing avoids the issue of local maxima, and thus showed promising results based on the increased results compared to RHC.

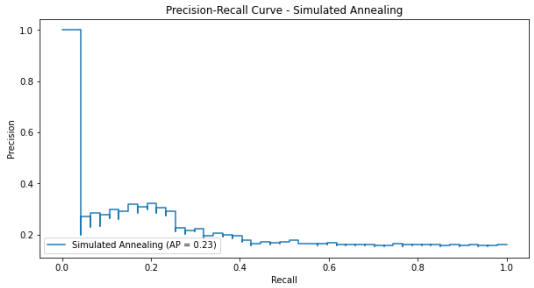

Figure 29: Simulated Annealing - Precision Recall

4.4.2 Learning Curve

The learning curve results converge towards 0.62. The results appear to be fully converged, but the performance is much worse in comparison to Random Hill Climb. This may be the result of parameter tuning that didn't include a wide enough range. For example, the maximum number of iterations could be increased; however, the high learning rate value of 0.1 with increased iterations may result in increased variability of model results.

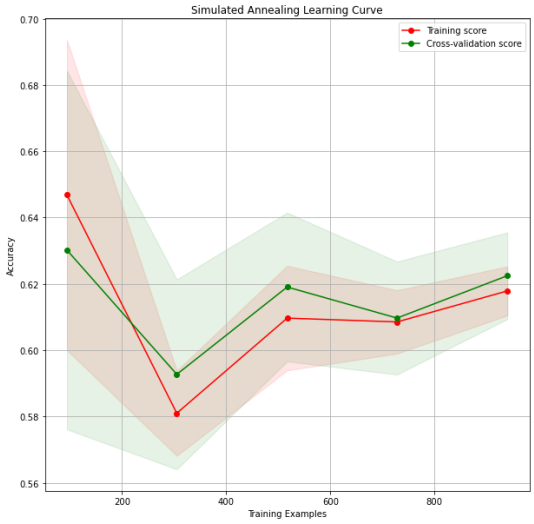

Figure 30: Simulated Annealing - Learning Curve

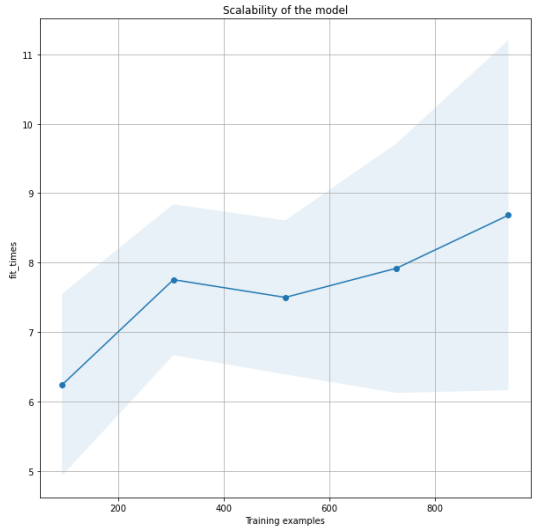

Figure 32: Simulated Annealing - Scalability

4.4.3 Confusion Matrix

The confusion matrix shows very poor performance in terms of predictive ability for both target classes. The performance is nearly split in terms of true random performance.

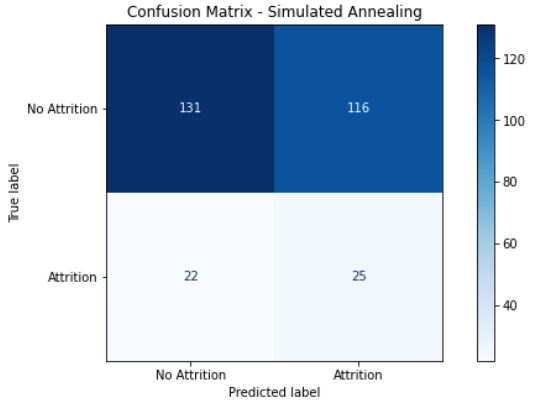

Figure 31: Simulated Annealing - Confusion Matrix

4.4.4 Scalability

The model tends to show a staggered increase in performance regarding model scalability; however, the standard deviation range tends to get wider as training examples increase, thus showing that there's a lesser confidence in the mean values as time goes on.

4.5 Genetic Algorithm

This model was tested with three values for restarting (0, 5, 10), two values for maximum iterations (1000, 2500) and three values for initial learning rate (0.001, 0.01, 0.1). In total, 18 neural network models were trained to identify the optimal parameters. The optimal model parameters included a learning rate of 0.001, 0 restarts, and 1000 maximum iterations.

4.5.1 Precision-Recall

The accuracy-precision F1 score shows a value of 0.64. This result shows optimal performance compared to the other algorithms, in addition to gradient descent. This can potentially be attributed to the combination of good weights to improve the optimization in this algorithm.

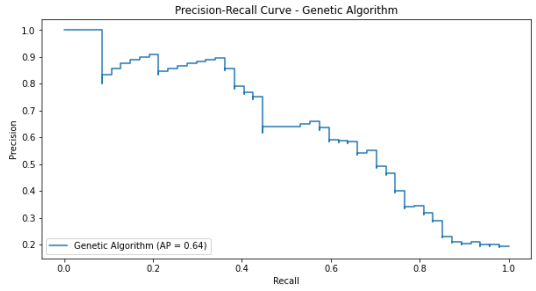

Figure 33: Genetic Algorithm - Precision Recall

4.5.2 Learning Curve

The associated learning curve shows a cross-validation score of 0.85. These curves appear to be fully converged after 1,000 training examples. This shows that the Genetic Algorithm's predictive ability fits the training and testing datasets evenly.

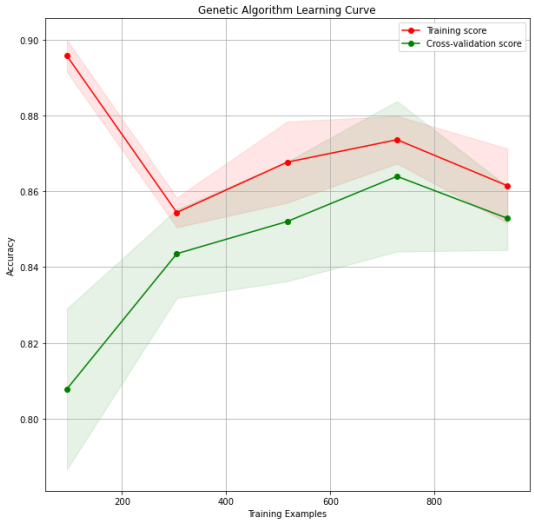

Figure 34: Genetic Algorithm - Learning Curve

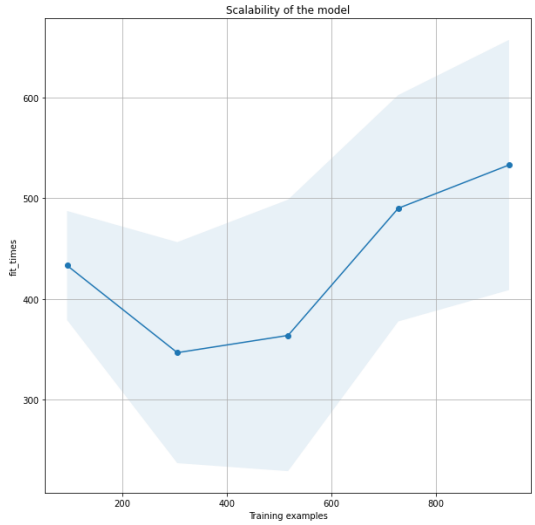

Figure 36: Genetic Algorithm - Scalability

4.5.3 Confusion Matrix

In looking at the confusion matrix results, the Genetic Algorithm performed well for the "No Attrition" label; however, similar to the other models, the model struggles to predict with any confidence the "Attrition" dataset, given that only 44% of this target label class were correctly predicted.

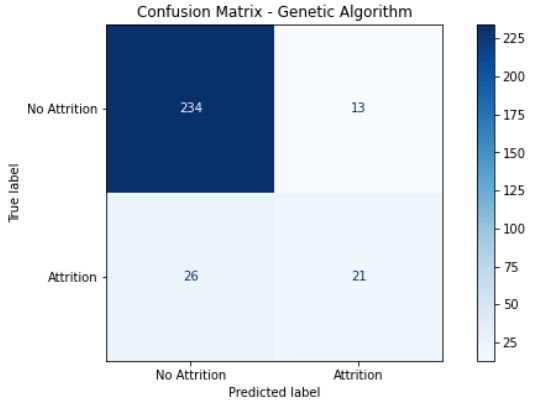

Figure 35: Genetic Algorithm - Confusion Matrix

4.5.4 Scalability

The model's scalability is staggered in terms of training examples; however, the wide range in this parameter leads one to believe that *generally* there's a steady increase in fitness time with additional training data.

4.6 Optimization Function Comparison

5 Conclusion

In terms of the optimization functions, the simulated annealing function performed well based on allowing a function where local optima can be escaped to find the maxima. The genetic algorithm performed well specifically on the Four Peaks algorithm based on the ability to iteratively build on good performance to improve the optimization. The MIMIC algorithm performed well on the Knapsack algorithm based on the usage of density estimators to optimize the problem. The optimization algorithms each had varying levels of performance for the IBM Employee Attrition dataset used from Assignment 1.
Most of the models tested in this work showed promising results; however, there were concerns with results regarding employees within the attrition class. In order to alleviate issues with sensitivities associated with neural network weights, the input data was normalized prior to training. Regardless, the simulated annealing algorithm showed poor performance and likely could've benefited from additional tuning parameters. The final parameters for the Random Hill Climb and Simulated Annealing algorithms ended up being very similar, which thus shows that the neural network weights tend to result in convergence to local optima.

A surprising result in this experiment was the usage of the Genetic Algorithm, which featured strong performance based on utilizing optimal parent node attributes for future child nodes. A consequence of this algorithm being used, is the high computational complexity required to allow the algorithm to run. Nonetheless, the optimal result was obtained using the algorithm, and given the proper computational power, would allow for a powerful neural network model to be created.

6 References

References

[1] De Bonet, J., Isbell, C., Viola, P. (1996). MIMIC: Finding optima by estimating probability densities. Advances in neural information processing systems, 9.

[2] Mallawaarachchi, V. (2020, March 1). Introduction to genetic algorithms - including example code. Medium. Retrieved March 6, 2022, from https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

[3] Shelar, N. (2020, August 9). Simulated annealing in AI: The Engineer. The Polymath Blog. Retrieved March 6, 2022, from https://neelshelar.com/simulated-annealing/

[4] Understanding hill climbing algorithm in Artificial Intelligence. Section. (n.d.). Retrieved March 6, 2022, from https://www.section.io/engineering-education/understanding-hill-climbing-in-ai/

[5] IBM (2017) IBM HR Analytics Employee Attrition and Performance. Kaggle Dataset