CS7641 Machine Learning (Fall 2020)

Assignment 2 – Randomized Optimization

Sergiy Palguyev (gtid: spalguyev3)

I. Abstract

The following paper present two analyses. The first analysis explores four optimization search algorithms against three optimization problems. The second analysis uses the first three search algorithms to find best weights for a Neural Network, specifically against the dataset which was explored in Assignment 1.

II. Random Search Algorithms Introduction

The four search algorithms presented for evaluation are Randomized Hill Climbing, Simulated Annealing, Genetic Algorithm and MIMIC. The problems chosen to evaluate these search algorithms are the Four Peaks problem, the Flip Flop problem, and the Knapsack problem.

A. Randomized Hill Climbing

Randomized Hill Climbing (RHC) is a search algorithm which is a variant of the "generate and test" algorithm. At any point in state space, the algorithm will attempt to find the local maximum in the direction which optimizes the cost function. RHC can find the local maximum but not necessarily the global maximum if it enters into one of several regions. At a local maximum, the algorithm stops since all neighboring states have values which are worse than the current state. On a plateau, all neighbors have the same value, thus choosing a direction is not possible. Any point on a ridge can seem like a peak because all possible directions are downward.[1] For testing RHC the values varied are:

Attempts	2	8	32	128	512
Restarts	5	25	45	65	85

B. Simulated Annealing

Simulated Annealing (SA) simulated the statistical process of growing crystals using the annealing process. This is an effective and general form of optimization, useful in finding global optima in presence of large numbers of local optima. A basic implementation of SA is the same as RHA, except instead of picking the next best move, SA picks a random move. A probability associated with the badness of a move decreases as subsequent moves get better and better.[2] For testing SA the values varied are:

Attempts	2	8	32	128	512
Decay	geom	exp	arith		

C. Genetic Algorithm

Genetic Algorithm (GA), bases its optimization strategy on natural selection. The GA algorithm modifies a population of individual solutions evolving them over time to arrive as a optimal final solution. The GA algorithm is specifically well suited for discontinuous, non-differentiable, stochastic or nonlinear problems.[3] For testing GA the values varied are:

Attempts	4	32	256
Population	10	100	1000
Keep %	0.001	0.01	0.1

D. MIMIC Algorithm

Mutual Information Maximizing Input Clustering (MIMIC), also belongs to the larger class of evolutionary algorithms. MIMIC attempts to exploit the underlying structure of a problem to eliminate re-exploration of sub-optimal portions.[4] For testing MIMIC the values varied are:

Attempts	4	32	256
Population	25	125	250
Keep %	0.01	0.1	0.3

III. Optimization Problems

A. Four Peaks

Four Peaks (FP) problem has two local optima with wide basins. This problem should show poor performance for RHC and SA optimization compared to evolutionary algorithms due to the large troughs between peaks. This should be a difficult problem for

CS7641 Machine Learning (Fall 2020)

Assignment 2 – Randomized Optimization

Sergiy Palguyev (gtid: spalguyev3)

hill climbing algorithms since a point in the basin cannot know the best next direction to travel.

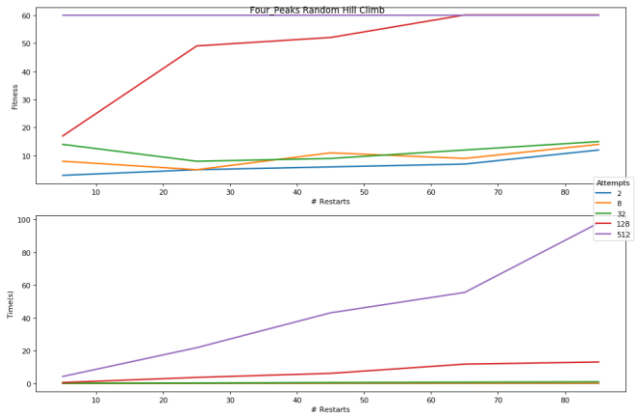

Figure 1: Four Peaks RHC Fitness and Time Graphs

The RHC optimization algorithm for Four Peaks problem shows very interesting behavior. For small attempts, fitness remains small. For 128 attempts and 256 attempts, the Fitness graph shows a plateau reached at a fitness of 60. For all Attempt values a higher number of restarts (>60) lead to higher fitness values. This is a somewhat fast running algorithm which calculates data points at most within 100 seconds notably increasing with # of Attempts.

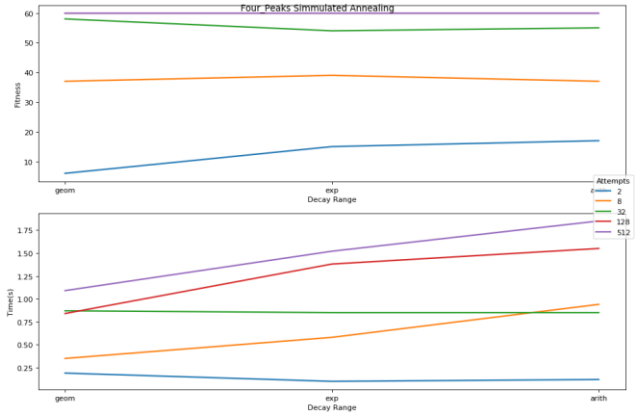

Figure 2: Four Peaks SA Fitness and Time Graphs

The SA optimization algorithm for Four Peaks looks very similar for the RHC algorithm as section II.B describes SA as a randomized version of the RHC algorithm. As such, we see slightly better performance as 128 Attempts perform well with all decay schedules, reaching a maximum fitness of 60, the same as 512 Attempts. The best performing Decay function, however, is arithmetic. This algorithm is very fast, at maximum taking 2 seconds to run.

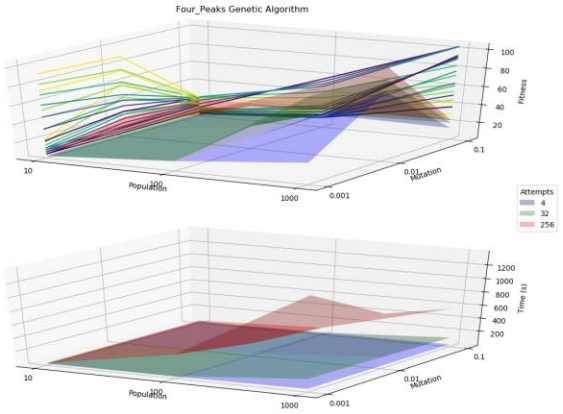

Figure 3: Four Peaks GA Fitness and Time Graphs

The GA optimization algorithm for Four Peaks immediately shows a difference in fitness. The initial theory, at the beginning of this section, assumed evolutionary algorithms should be better than RHC/SA algorithms. Observing the fitness graph, it is apparent that the 256 Attempts graph reach well above a fitness of 60, attaining a maximum fitness of 104 with population of 1000 and mutation value of 0.001. Observing the general upward trend, it is evident that increasing Attempt # maybe continue increasing fitness value as well.

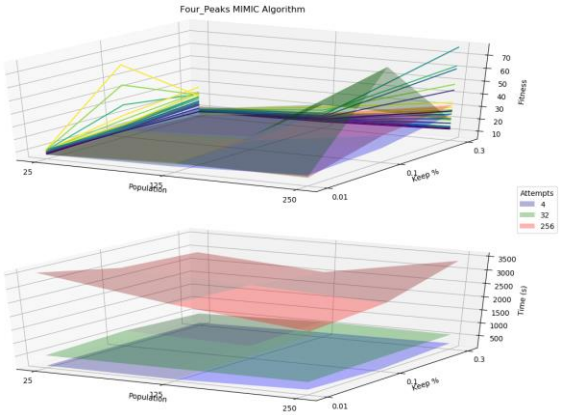

Figure 4: Four Peaks MIMIC Fitness and Time Graphs

The MIMIC algorithm performed mediocre, at best. This may be due to the range of values being picked being wrong or the generally poor fit of this algorithm to the problem. The best fitness value of 77 was attained with a population of 250, keep % of 0.1

and at 32 Attempts. The underlying problem space is not complex and thus may be a poor fit for MIMIC optimization.

B. Knapsack

Knapsack (KS) problem seeks to determine the optimum number of items to pack into a collection so the total weight of the items is less than or equal to a given limit which the value of the items is as high as possible. This problem has a complex problem space and should benefit most from GA or MIMIC optimization. Since both weight and value must be optimized up to a threshold, hill climbing algorithms should perform poorly in this task.

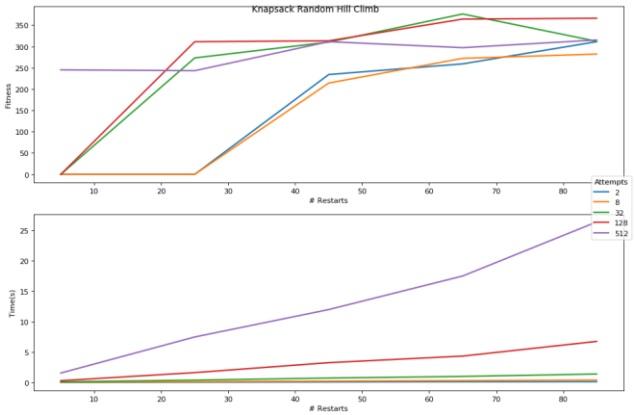

Figure 5: Knapsack RHC Fitness and Time Graphs

RHC for Knapsack problem shows a very promising fitness value of 376 at 65 restarts with 32 attempts. This algorithm is relatively fast, but begins to slow down as the number of restarts increases with larger attempts number. The values appear to plateau with higher restart and attempt values. As such, this graph most likely displays the limits of RHC optimization for the KA problem.

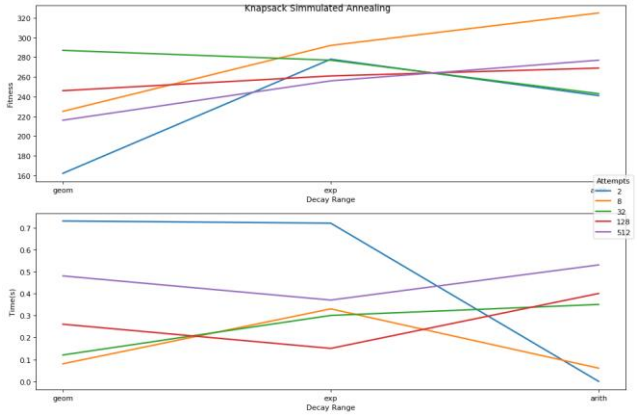

Figure 6: Knapsack SA Fitness and Time Graphs

SA for Knapsack problem performs worse than RHC, with the maximum fitness of value of 325 attained at 8 attempts with arithmetic decay function. Observing the fitness graph closely, it is evident that even with higher attempt numbers, the fitness values do not grow higher. This is a poor optimization algorithm for this problem.

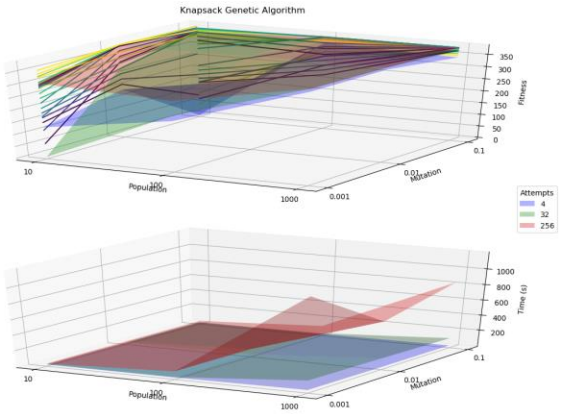

Figure 7: Knapsack GA Fitness and Time Graphs

GA for Knapsack problem performs fairly well, reaching a maximum fitness value of 381 with 32 attempts at a population of 1000 and mutation of 0.001. Considering the convergence of the various attempt graphs, population and mutation values around this fitness result indicated this may be the best fitness score attainable by GA for this problem. For high attempt and population values, GA execution slows significantly.

CS7641 Machine Learning (Fall 2020)

Assignment 2 – Randomized Optimization
Sergiy Palguyev (gtid: spalguyev3)

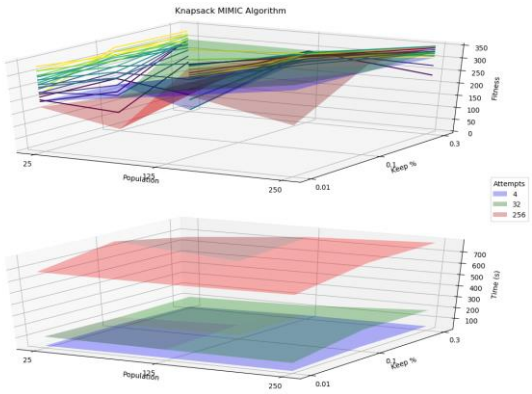

Figure 8: Knapsack MIMIC Fitness and Time Graphs

MIMIC for Knapsack problem performed rather poorly, attaining a fitness value of 350 with 4 attempts, a population of 250 and keep % of 0.01. Considering the highest value was reached at the highest population value, in addition to the general upward trend observed on the contour projections against the XZ plane, it is evident that there is general upward trend with higher population values. Although the initial assumption that MIMIC should provide better results failed to be demonstrated, the general trends in the graphs point to the fact the MIMIC may still prove itself with higher property values for population.

C. Flip Flop

Flip Flop (FF) problem counts the number of time bits alternate in a bit string. This problem should be a fairly good candidate for SA due to randomized hill climbing nature. Evolutionary algorithms such as GA and MIMIC may overcomplicate the problem since the problem space is not complex while the simple RHC algorithm would not be able to locate a global optimum easily.

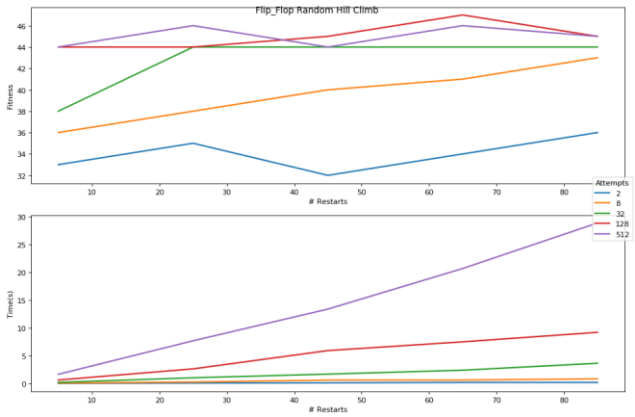

Figure 9: Flip-Flop RHC Fitness and Time Graphs

RHC for FlipFlop problem attained highest fitness value of 47 with 128 attempts and 65 restarts. The general trend in the graph, demonstrates a plateau of fitness reached with increasing values. It is likely that GA cannot perform better with higher values.

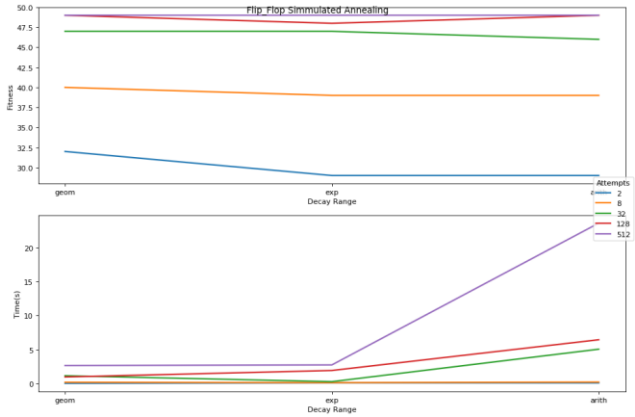

Figure 10: Flip-Flop SA Fitness and Time Graphs

SA for FlipFlop problem attained highest fitness value of 49 with 128 attempts with geometric decay function. Observing the graph, similarly to RHC, demonstrates it is unlikely better fitness can be attained with higher attempt values. SA attained the best fitness score for the FlipFlop problem, validating the initial hypothesis.

CS7641 Machine Learning (Fall 2020)

Assignment 2 – Randomized Optimization
Sergiy Palguyev (gtid: spalguyev3)

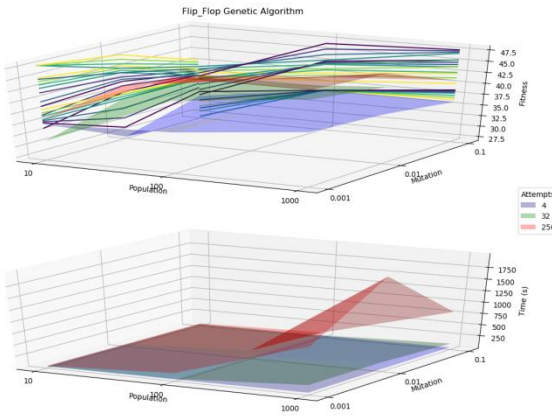

Figure 11: Flip-Flop GA Fitness and Time Graphs

GA for FlipFlop problem did not receive the best score, only attaining a value of 48 with 246 attempts population of 1000 and mutation of 0.001. It is evident form XA and YZ projections that the general upward trend suggests small mutation values and higher population values may lead to better fitness score.

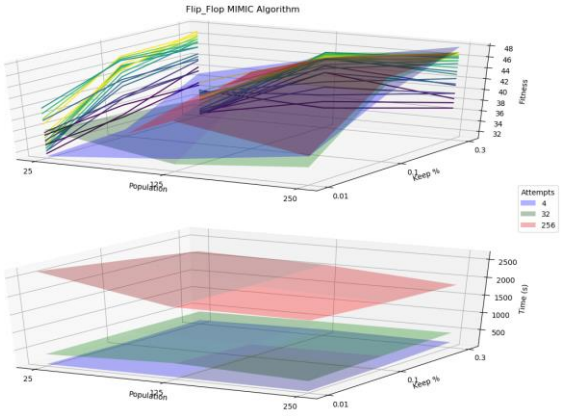

Figure 12: Flip-Flop MIMIC Fitness and Time Graphs

MIMIC for FlipFlop problem, similarly attains best fitness score of 48 with 4 attempts at population of 250 and keep percent of 0.3%. This result similarly indicates that higher parametric values may lead to better fitness values. It is also interesting that neither parameter value affects execution time as much as attempts number, which slows down the algorithm significantly.

IV. Neural Networks Weights Optimization

A. Introduction to Dataset

The Banknote Authentication dataset [1] consists of four continuous values representing image attributes used for evaluation of an authentication procedure for banknotes. The classification is binary, classifying each banknote as authentic, or not. This dataset is very interesting as it must have high accuracy and precision in determining authenticity.

B. Neural Network Back-propagation

First, the best hyper-parameters are calculated by prebuild sklearn algorithm RandomizedSearchCV. The output of this function will be used as values that may be locked while others are varied, to see effects on model accuracy, as well as to determine a set of "optimized" parameters for the rest of the analysis to compare to.

BanknoteAuthentication NeuralNet calculated BEST parameters are (76, 80) Layers, 0.1 Alpha with 1.0 score at 26% Train and 74% Test

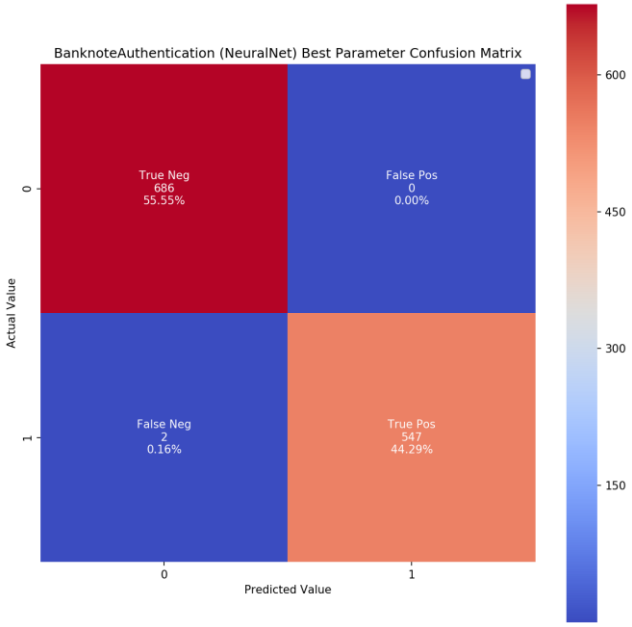

Figure 13: NeuralNet Best Params Confusion Matrix

The Confusion Matrix with these parameters, determined that the model is predicting with 100.0% Train Accuracy, 99.2% Test Accuracy, 99.7% CV Accuracy, 98.3% Precision Score, 100.0% Recall Score and 99.1% F1 Score

CS7641 Machine Learning (Fall 2020)

Assignment 2 – Randomized Optimization
Sergiy Palguyev (gtid: spalguyev3)

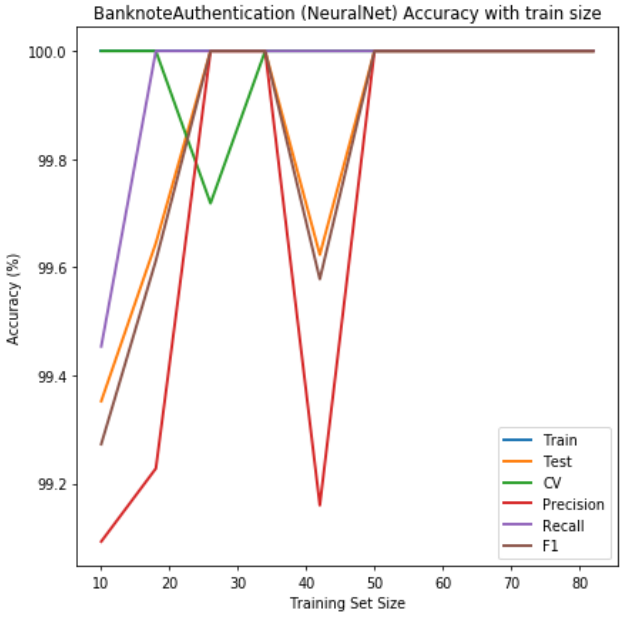

Figure 14: Neural Nets Backpropagation

Banknote dataset display a very interesting behavior. As seen, all accuracy metrics reach 100% peak with a very small training set. This is in line with what sklear.RandomizedSearchCV values present with a Train to Test ratio of 26/74. The graph displays very sharp drops and swings in accuracy metrics in higher ratios, most likely pointing to overfitting caused by large numbers of hidden neural networks.

C. Randomized Hill Climbing Optimization for NN

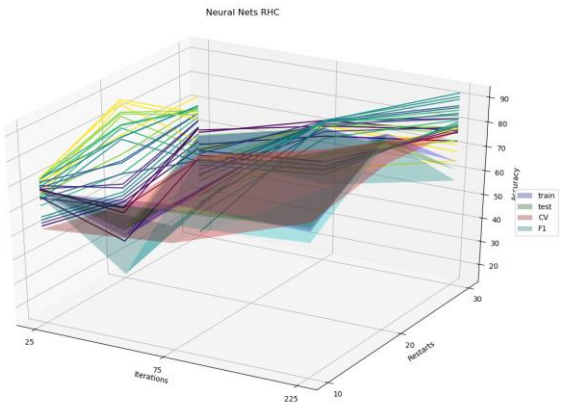

Figure 15: Neural Nets RHC

In exploring the Randomized Hill Climbing algorithm for Neural Nets, it is evident that higher iterations as well as higher restarts perform better on all metrics. However, we do not see the same high accuracy observed in Assignment 1. With RHC algorithm, the best F1 accuracy reaches about 85%. It is possible that the range of values chosen to explore are not exhaustive and larger values of Restarts or Iterations could yield better results. The general upward trend observed in the projection against the XZ plane indicates that higher values of iterations may significantly increase accuracy.

D. Simulated Annealing Optimization for NN

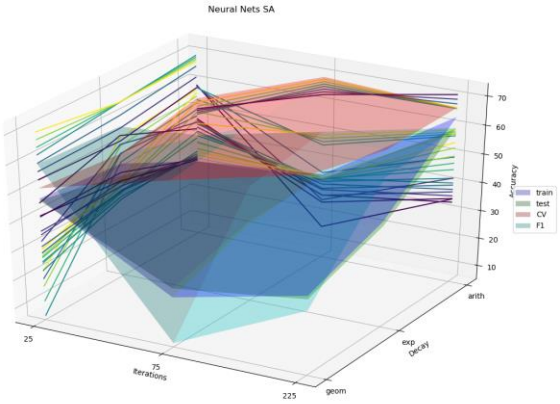

Figure 16: Neural Nets SA

Simulated Annealing fared worse than RHC in this test. Large iterations in geometric decay caused severe F1 drops. All accuracy metrics fare best with arithmetic decay and large iterations. However, all metrics remain at accuracy scores between 70% and 80%, which is rather low. Flat trend lines in all projections suggest increasing parametric values will most likely do very little to improve accuracy scores for any metric.

E. Generic Algorithm Optimization for NN

Genetic Algorithm Optimization varied number of iterations between 25, 75 and 225.

CS7641 Machine Learning (Fall 2020)

Assignment 2 – Randomized Optimization
Sergiy Palguyev (gtid: spalguyev3)

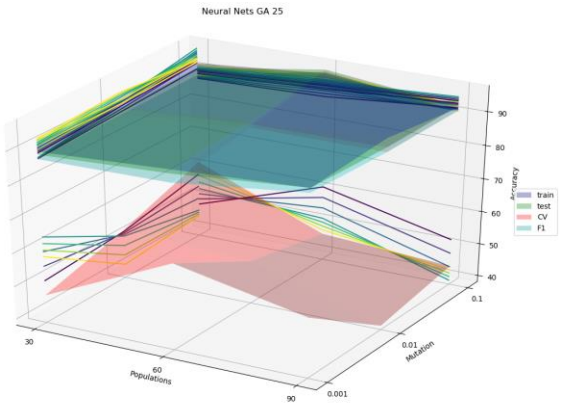

Figure 17: Neural Nets GA at 25 iterations

GA with 25 iterations fared much better than RHC or SA maintaining high accuracy values around 90 for all metrics, except CV which remained considerably low around the 40-50 range. This is a poor result as CV is an indicator of how well the Neural Net model performs on real-world data.

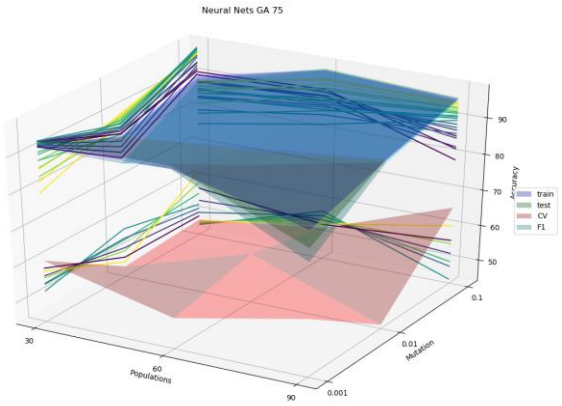

Figure 18: Neural Nets GA at 75 iterations

GA with 75 iterations increased its accuracy values. Most metrics improved slightly, but CV saw a significant boost, rising to the 50-60 accuracy range. This is still a low score, but a general trend upwards is forming with increased iteration numbers.

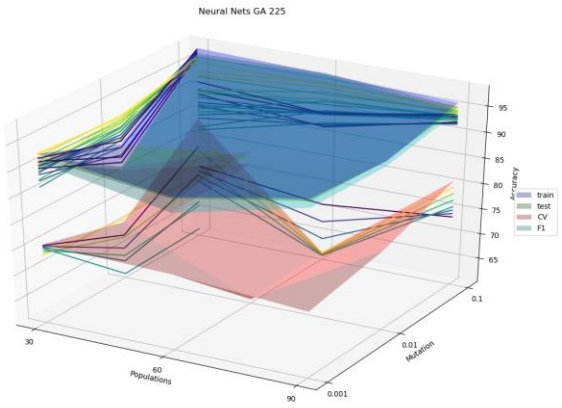

Figure 19: Neural Nets GA at 225 iterations

GA with 225 iterations again increased CV accuracy scores to the range of 65%-85%. Other metric scores increased as well, reaching above accuracy of 95%. This result continues to support the theory that increased iteration counts will improve accuracy scores across the board. The general trend of graph projections on the XZ and YZ planes demonstrate that higher population and mutation values also contribute to increase in accuracy score across all metrics.

V. Time Analysis

The plots included in Sections III.A, III.B and III.C indicate, across all problems that RHC is an easy algorithm, and executed fairly quickly, even with increased numbers of restarts and iterations. SA, based on RHC is also a fast algorithm and often runs faster than RHC according to the time plots from Sections III.A, III.B and III.C.

The evolutionary algorithm GA tends to slow down with population increase across all problems analyzed in this paper. The number of iterations, and mutation values, however do not significantly affect execution time. The MIMIC algorithm, also based on evolutionary theory is the slowest running algorithm of all four. This algorithm does not appear to be affected population number of keep percentage as much as Iteration number. The run time increases significantly with larger iteration numbers.

CS7641 Machine Learning (Fall 2020)

Assignment 2 – Randomized Optimization

Sergiy Palguyev (gtid: spalguyev3)

VI. Conclusion Part 1

As predicted, the Four Peaks problem performed best using the GA algorithm. GA can be very slow. Instead of maintaining a population GA maintains a probability vector, throwing away the population once analyzed. For this reason, is difficult to determine how cautiously to treat the results or whether we have reached the global optimum.

The Knapsack problem, failed the initial hypothesis predicting the MIMIC should produce best results, instead GA producing higher fitness values. However, the general behavior and trends observed by the graph, it is estimated that with further parameter tuning, MIMIC can produce much better fitness values than GA. As such, it is evident that evolutionary algorithms are best for this problem and although GA performed better, MIMIC should still be preferred for optimization, only with a larger set of parametric values to test.

The Flip-Flop problem performed as predicted, attaining the highest fitness value using the SA algorithm. However, further analysis is required as MIMIC also showed promising results with higher parameter values.

In summary, the optimization algorithms analyzed in this paper can be summarized as such:

Algorithm	Pro	Con
RHC	Fast No tuning	Not suitable for complex problems.
SA	Fast Finds global optima	Need to tune parameters
GA	Somewhat slow	Need to tune Can be slow for complex problems High volatility
MIMIC	Good for complex problems. Good at finding global optima	Need to tune parameters Very slow.

VII. Conclusion Part 2

Optimizing Neural Net weights is a complex problem which initially appears to best be served by evolutionary algorithms. Observing the graphs from Sections IV.C and IV.D it is evident that RHC and SA do not perform as well as the weight picked in Assignment 1, summarized in Sections IV.A and IV.B. GA algorithm graphs in Section IV.E, although lagging in performance behind our Backpropagation graph from Assignment 1, show a very promising trend towards better performance. Along the three graphs evaluated, it appears that with more iterations, the general trend towards higher and higher accuracy scores across all metrics can be achieved.

Although the scores in Assignment 1 were all at or near 100%, the graph in Section IV.B points to significant overfitting occurring at the Neural Network. Thus weights picked by GA optimization with a sufficiently high iteration count, and optimal population and mutation values, would be significantly more trustworthy. Since the GA algorithm is good at finding global maxima for complex problems, This optimization technique would produce values of much higher confidence.

VIII. References

1. https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/
2. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/learn-43/lib/photoz/.g/web/glossary/anneal.html
3. https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html
4. https://www.cc.gatech.edu/~isbell/papers/isbell-mimic-nips-1997.pdf
5. Banknote Authentication Dataset – Retrieved from https://archive.ics.uci.edu/ml/datasets/banknote+authentication
6. Source Code
   The source code and data can be found at : https://github.com/sergiypalguyev/Fall2020CS7641-Assignment2