Homework 2. Randomized Optimization

Fall 2023, OMSCS, Georgia Institute of Technology

Hossein Sahour, PhD

Abstract

The report examines the effectiveness of four distinct optimization algorithms: randomized hill climbing, simulated annealing, genetic algorithm, and MIMIC (Mutual-Information-Maximizing Input Clustering). It is divided into two parts. In the first part, three different optimization problems (the Traveling Salesman Problem (TSP), The Knapsack Problem, and the Max K-Color problem) are introduced and analyzed to evaluate the aforementioned algorithms. In the second section, three optimization algorithms are employed to find the optimum weights of neural networks used to predict the presence or absence of diabetes in a binary classification task.

Randomized Optimization Algorithms

Randomized optimization (RO) comprises a set of heuristic algorithms designed to address optimization problems where traditional methods may fall short. By introducing randomness in the search process, RO can escape local optima and increase the likelihood of reaching a global optimum.

Randomized Hill Climbing:

Randomized Hill Climbing (RHC) is a type of RO that operates through a loop that continuously moves in the direction of increasing value (uphill) to find the “peak of the mountain”. Unlike the standard hill climbing algorithm, which deterministically selects a neighboring solution based on defined criteria, RHC randomly selects a neighboring solution in each iteration and decides whether to move to it or not by comparing its fitness value.

Simulated Annealing:

Simulated Annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. Inspired by the process of annealing in metallurgy, it simulates the heating and cooling of a material to decrease defects to find a state with a minimum energy. At each step, the SA algorithm evaluates neighboring states of the current solution and probabilistically decides whether to transition to a state with lower energy or remain in the current state.

Genetic Algorithms:

Genetic Algorithms (GA)s are a type of evolutionary algorithm that mimics the process of natural selection to find optimal solutions to problems. It generates a population of candidate solutions and uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection to evolve the solutions over generations.

MIMIC

MIMIC is an optimization algorithm that uses a probabilistic model to guide its search through the solution space. Instead of examining each potential solution individually, MIMIC clusters similar solutions together based on their quality. By doing so, it can identify promising regions in the solution space more effectively.

MIMIC leverages distributions of solutions, to prioritize areas with higher mutual information and, consequently, a greater likelihood of containing optimal or near-optimal solutions.

Part1. Optimization problems

In this section three optimization problems namely the Travelling Salesman Problem (TSP), Knapsack Problem, and Max K-Color problem have been tackled using four different RO algorithms. For each problem, the four RO algorithms were run and results were compared and visualized. For comparison, I created four plots for each problem, including Fitness/Iteration, Fitness/problem size, Function evaluation/Algorithm, and wall clock time plots.

1. **Traveling Salesman Problem (TSP)**

The TSP is a classic optimization problem in which the objective is to find the shortest possible route that visits a set of cities (here coordinates) and returns to the original city. The number of cities (N) in this exercise was 8.

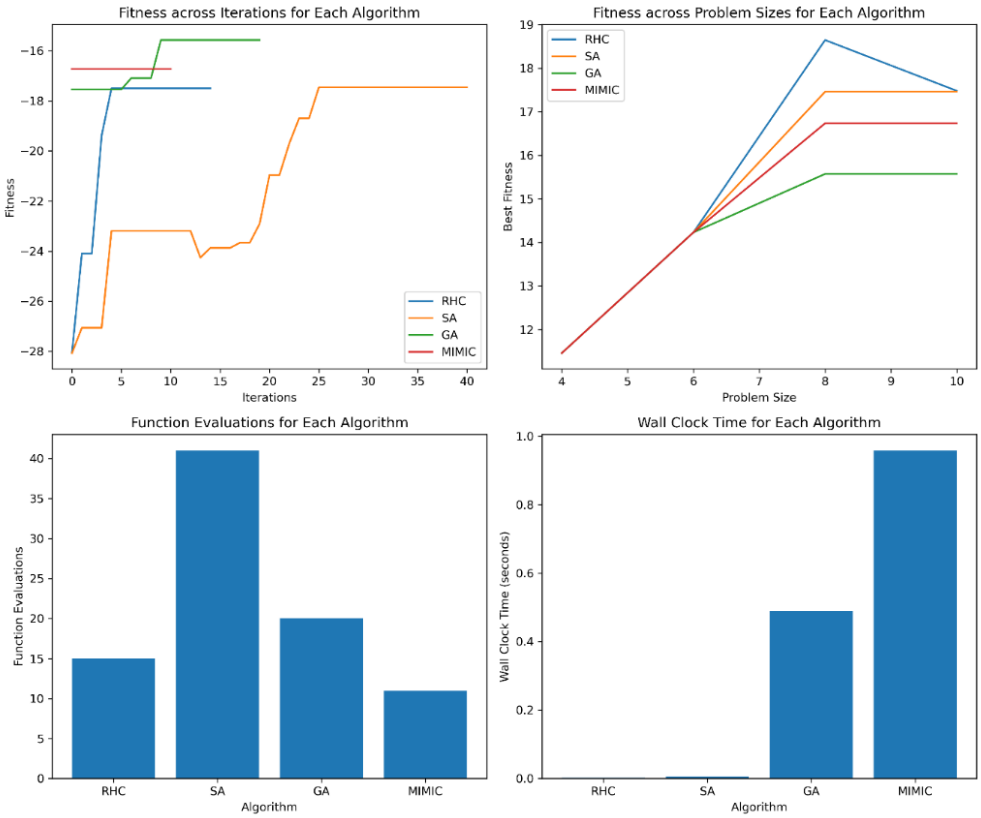

Figure1. TSP problem

According to Figure 1, fitness across iterations, the y-axis shows the fitness values with respect to iterations. In this figure, the highest (least negative) fitness value indicates the shortest route for TSP problem. The GA shows that it converges rapidly to a good solution, meaning that it quickly finds an

efficient solution (route). The SA and RHC also converge, however, it takes more iteration for them to reach the optimal values compared to GA.

The fitness across problem size plot in Figure 1 shows that GA consistently produces the lowest fitness values (it means shortest path which is favorable in the context of TSP) across all problem sizes. This highlights the effectiveness of GA compared to other algorithms to tackle the TSP in this exercise. The other algorithms also provide good solutions, but they do not match the consistency of GA.

The function evaluation bar chart (based on maximum size) helps understand how many times each algorithm evaluated the fitness function to arrive at its solution. Therefore, it is an indication of efficiency. Lower values are preferable because it means the algorithm needs fewer evaluations to converge. GA and MIMIC show the highest function evaluations. The RHC and SA have fewer evaluations which means a faster convergence, but this comes with the risk of missing the global optima.

The Wall Clock Time for Each Algorithm (for maximum size) is a bar chart that shows the time (in seconds) took for an algorithm to reach the solution (near optimum route in TSP). A shorter time is preferable as it indicates faster performance. In this plot, RHC and SA are found to be the fastest algorithms. GA, despite its effectiveness in finding good solutions, takes longer than RHC and SA.

2. Max K-Color problem

In the Max K-Color problem, the objective is to assign one of the K colors to each vertex of a given graph so that the number of adjacent vertices with the same color is maximized. It's a variation of the more commonly known graph coloring problem, where the objective is typically to minimize the number of colors while ensuring no two neighboring vertices share the same color.

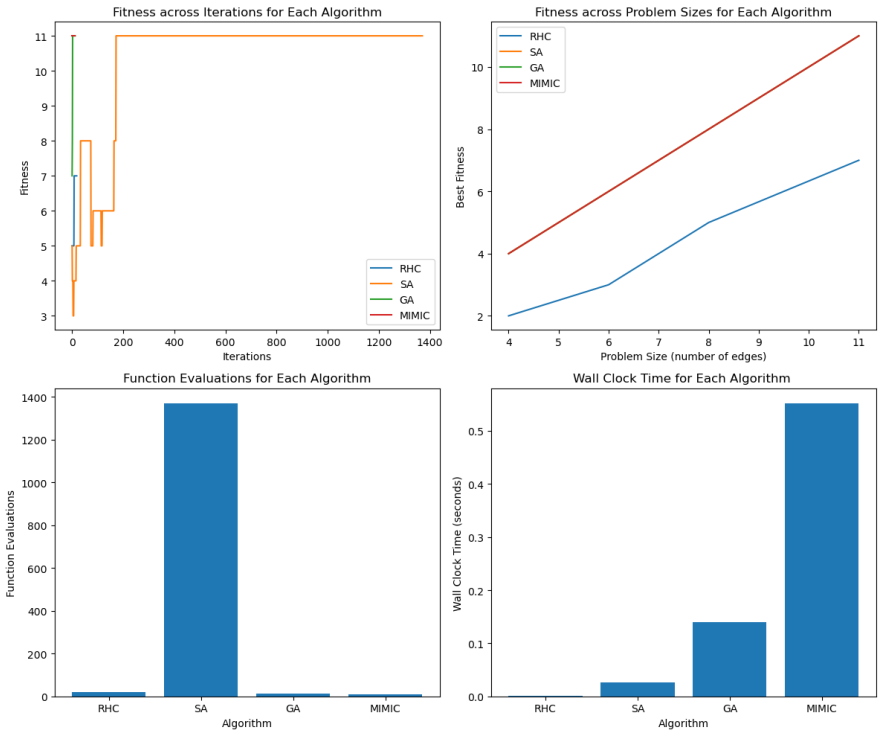

Figure2. The Max K-Color problem

The Max K-Color problem is a combinatorial optimization problem where the goal is to find the best possible coloring configuration that maximizes the number of adjacent vertex pairs with the same color, given a fixed number of available colors (K). In my exercise, I used 4 graphs and max_val of 5 for colors.

According to Figure 2, Fitness across Iterations for Each Algorithm, three algorithms of SA, GA, and MIMIC converge to the highest fitness value highlighting their performance in handling the Max K-Color problem. However, according to the wall clock plot, SA performs significantly faster than others highlighting its outperformance.

3. The Knapsack Problem

The Knapsack Problem is a well-known problem for optimization. For this problem, there are items with weights and values. We need to decide how many of each to put in the knapsack so that weight doesn't go over a limit, and you get the highest possible value.

In Figure 3, the higher fitness values on the y-axis indicate better performance in the context of the Knapsack Problem. The MIMIC and GA algorithms start at a much higher fitness value compared to RHC and SA. However, in the end, MIMIC surpasses GA, suggesting its higher capability to find a good solution compared to GA and the other algorithms. This highlights the effectiveness of MIMIC in finding a solution for the Knapsack Problem in this exercise compared to other algorithms. Also, MIMIC consistently performs the best across varying problem sizes according to figure2, indicating its scalability in finding a good solution for the problem. However, according to Figure 3 the highest performance of MIMIC comes with a cost of requiring more time than other algorithms.

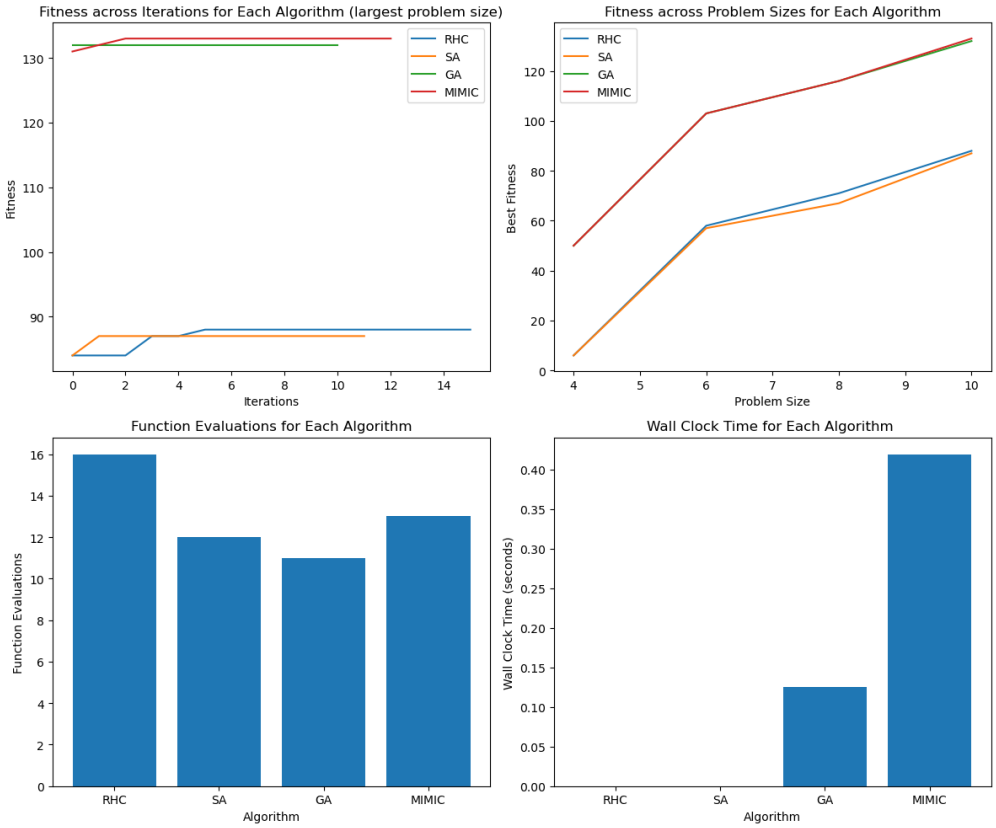

Figure3. The Knapsack Problem

Summary and Discussion for Part 1

Traveling Salesman Problem (TSP):

GA is the most effective, converging rapidly to find a near-optimum path. SA and RHC also converge but require more iterations compared to GA. RHC and SA are the fastest in terms of wall clock time, while GA, despite its effectiveness, takes longer. GA and MIMIC require more function evaluations, indicating they explore the solution space more thoroughly.

Max K-Color Problem:

SA, GA, and MIMIC all converge to the peak fitness values, demonstrating their proficiency in addressing the Max K-Color challenge. Among these, SA stands out due to its significantly faster execution according to the wall-clock figure, making it more time-efficient than the other mentioned algorithms.

Knapsack Problem:

MIMIC consistently outperforms all other algorithms in determining the combination of items that maximizes the total value without surpassing the knapsack's weight limit. While the GA initially shows promise with a superior starting fitness value, its performance is eventually surpassed by MIMIC. It's important to highlight that although MIMIC perfroms better, it demands more computational time compared to other algorithms.

Bias-variance tradeoff properties of the optimization algorithms:

GA shows low bias as it consistently finds good solutions across all three optimization problems. Variance might be higher as it needs more function evaluations, indicating a more thorough exploration of the solution space which can sometimes cause overfitting.

SA shows low to moderate bias as it performs well in TSP and Maximum color problems. It shows lower variance as indicated by its fast convergence across all three problems (It consistently is one of the fastest algorithms in three problems, therefore, it might be producing similar solutions each time, leading to lower variance).

RHC shows high bias in the knapsack problem as it needs more iterations to converge in some problems but lower in the other two problems. RHC provides lower variance given its faster wall clock times in all three problems.

MIMIC shows low bias, especially for the Knapsack problem where it consistently finds optimal solutions. However, it can have higher variance due to its probabilistic nature. This can be tested by running the algorithm several times to see the performance in each run.

Overall, while each algorithm has its strengths and weaknesses, its performance varies based on the specific nature of the optimization problem. For example, MIMIC excels in the Knapsack problem, while GA is more suitable for the TSP. SA's efficiency and speed make it a good choice for many problems.

Part 2. ANN with random optimization

In this section, I applied randomized hill climbing (RHC), simulated annealing (SA), and a Genetic algorithm (GA) to optimize the weights of an ANN model. I used an early-stage diabetes risk prediction dataset. This

dataset includes information on the symptoms of individuals diagnosed with diabetes or those at a higher risk of contracting the condition. The data was gathered through direct patient questionnaires at the Sylhet Diabetes Hospital in Bangladesh and has been medically approved. The dataset comprises 520 records with 17 attributes, which encompass age, gender, and diabetic symptoms. The main outcome variable is a binary distinction between Positive and Negative diabetes diagnoses.

Data preparation and feature analysis

The outcome variable was categorical, labeled as "negative" and "positive". I employed a label encoder to transform these categories into binary classes, 0 and 1 (which was done in assignment1). Following that, the predictor variables were normalized to fall within a range of 0 to 1. The data was then split into training (comprising 75% of the total data) and testing subsets (constituting the remaining 25%). I refrained from oversampling or undersampling because the dataset displayed minimal class imbalance. A correlation matrix was created (Figure 4) to understand the relationships between features and the target variable, as well as the relationship between the individual features. According to Figure 4, variables including Polyuria and Polydipsia, gender, and sudden weight loss have the strongest correlation with diabetes (Pearson coefficient ranging from -0.45 to 0.67), making them significant indicators. Gender and Alopecia are negatively correlated with diabetes, suggesting a potential protective effect or decreased risk associated with these factors. Interestingly, obesity shows a very weak correlation with diabetes, indicating that by itself, obesity cannot be a strong indicator of this disease. Some features have only a weak correlation, so they might not be strong indicators by themselves but could still be informative when combined with other variables.

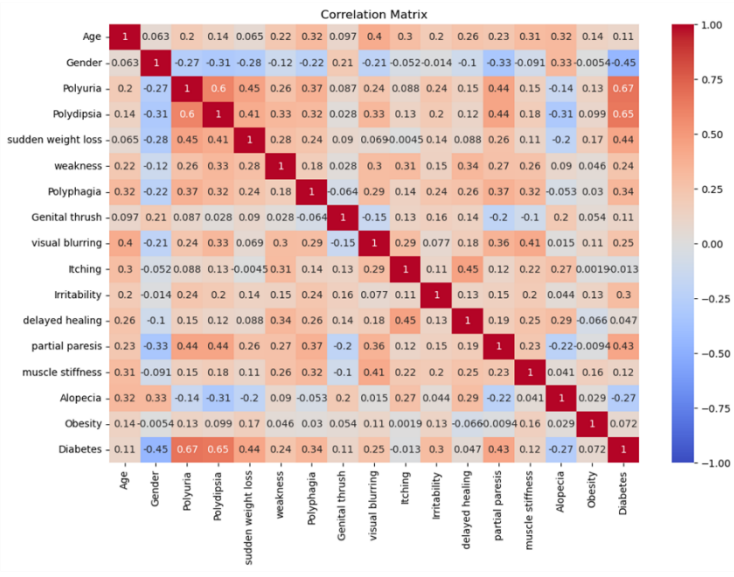

Figure 4. Correlation matrix of the diabetes dataset

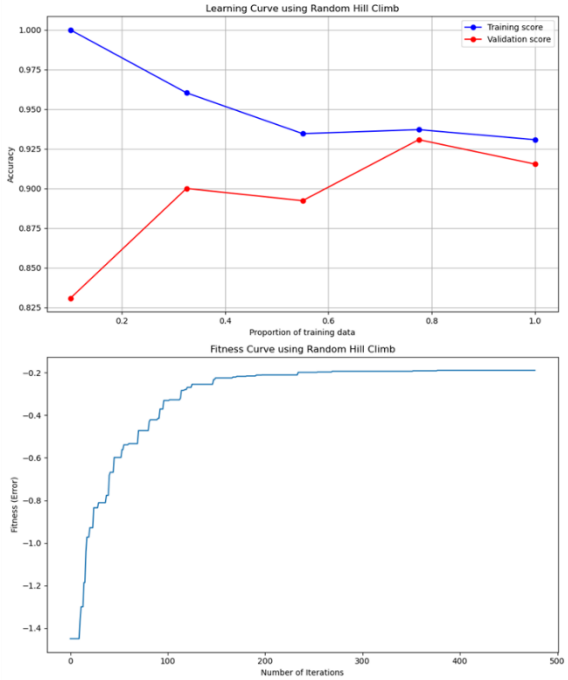

Figure 5. The learning curve and fitness curve of RHC

ANN with optimization algorithms

All three optimization algorithms used the same structure for a better comparison of the RO algorithms used to optimize the weights of the ANN. For each algorithm, I created a learning curve and fitness curve

error plots. The learning curve shows the training and validation accuracy with different proportions of data. The fitness curve shows the negative of error (or cost) over the number of iterations. If the curve is increasing over iterations, it suggests the optimization algorithm is successfully reducing the error. If it is increasing it suggests the optimization algorithm found a local optimum.

1. **ANN with RHC optimization**

Figure 5 shows the two plots for ANN when RHC is used for optimizing weights. In the learning curve, the training Score starts with a very high accuracy close to 1 when using only a small proportion of the training data. As more data is used, the training accuracy decreases as expected because, with limited data, the model can easily overfit to the training set. As more data is used, fitting it perfectly becomes more challenging. The validation score starts with a lower accuracy compared to the training score but increases as more data is added. However, after reaching a peak, the validation score starts to decline. This suggests that the model's ability to generalize improves initially but later on, it starts overfitting to the training data. High bias is typically indicated if the model performs poorly on both the training and testing datasets. According to Figure 5, the model's accuracy on both datasets is relatively high, so bias is not a significant concern. High variance is indicated by a significant gap between training and validation performance. Here, there's a noticeable gap, especially when a larger proportion of training data is used. This can suggest some variance or overfitting. This is indicated by a high training score and a declining validation score as more data is used.

As I mentioned earlier, the fitness curve shows a negative of error over the number of iterations. According to Figure 5, the error decreases sharply in the initial iterations and then levels off, reaching a flat surface. This suggests that the Random Hill Climb optimization algorithm quickly finds a good solution, but after some point, it struggles to find better solutions. It can also be an indication of the RHC being trapped in a local minimum. RHC is a heuristic search algorithm that takes random steps to find an optimal solution. Therefore, it is subjected to getting trapped in a local optima.

The rapid decrease in error during the initial iterations suggests that the algorithm escaped the local minima initially. However, as it advances, it is being converged to a local minimum resulting in the leveling off in the fitness curve.

Given the divergence between training and validation scores in the learning curve, it can be inferred that while the algorithm is minimizing the error on the training data, it might not always lead to the best generalization on unseen data.

2. **ANN with Simulated Annealing**

The plots of ANN with weights optimized with SA are shown in Figure 6. The accuracy of the training starts at a perfect score when only a small proportion of training data is used. As more data is used, the training accuracy decreases suggesting that with fewer data points, the model can easily fit the training data, but as more complexity is introduced, it becomes challenging for the model to fit perfectly. The accuracy on the validation set initially increases, reaching a peak, and then decreases slightly. This shows that the model may be generalizing better to unseen data when trained on a moderate proportion of the training data. However, as it uses the entire training data, its performance on the validation data decreases slightly. The high accuracy when using a small proportion of the training data and the decrease in accuracy as more data is added indicate that the model might have a high variance (or overfitting) when trained on fewer data. However, as more data is used, the training and validation scores start to converge, which suggests

reduced variance. The gap between the training and validation scores towards the end suggests there might still be some variance, but it's less than what we had in the initial stage.

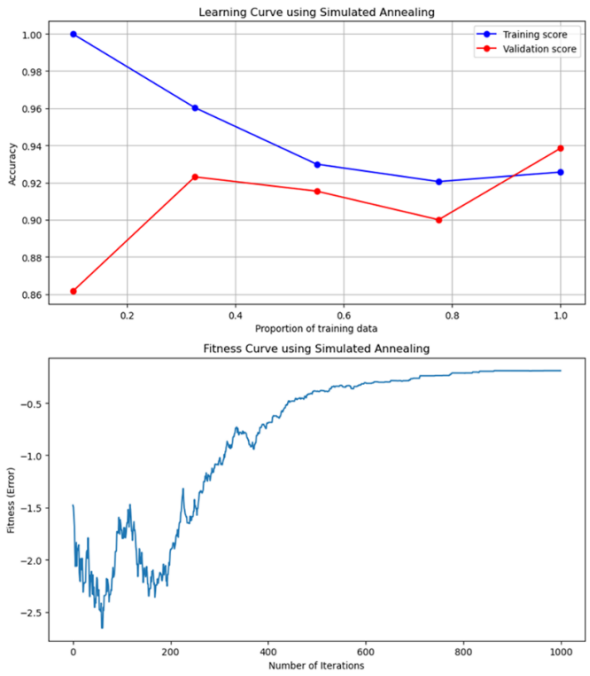

Figure 6. SA optimization for ANN weights

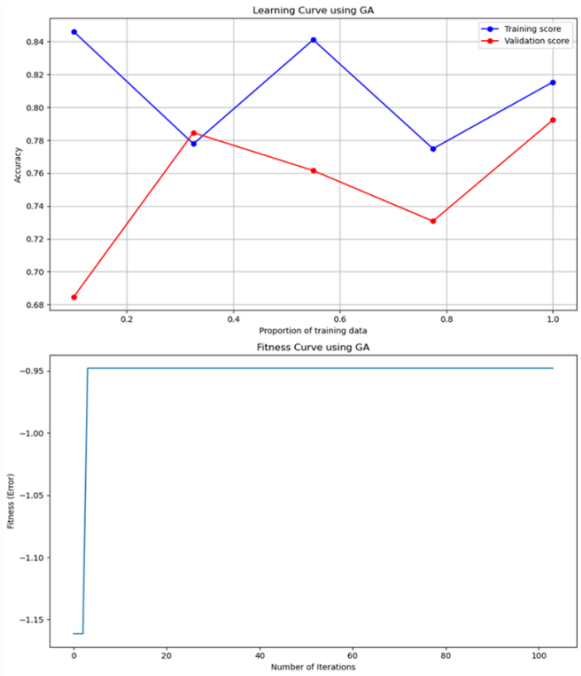

Figure 7. GA optimization for ANN weights

The shape of the fitness curve using SA shows a significant improvement in fitness with iterations, but after approximately 600 iterations, the improvements are marginal, as evidenced by the curve's leveling off.

Looking at the fitness curve, the initial steep decrease shows that SA is effective in quickly finding better weights for the ANN, leading to an abrupt reduction in error. This can be interpreted as the SA navigating the solution space efficiently, potentially escaping local minima. The plateauing of the fitness curve around the 600-1000 iteration could signify that the SA has found a solution that's very close to a global minimum, therefore, further iterations resulted in minimal improvements.

The interaction between bias and variance in this context can be illustrated by the Learning Curve. The initial high performance on training data and relatively poorer performance on validation data (before 0.2 proportion) indicates low bias but high variance. As more data is incorporated, both training and validation scores converge, showing a better balance between bias and variance.

3. ANN with Genetic Algorithm

The accuracy of training for ANN-GA model (Figure 7) starts high when trained on a small proportion of data but decreases as more data is introduced. This is because when a model is trained on smaller data, it can easily fit to the noise and bad data in that small dataset, which might not be representative of a larger

population. The accuracy of validation is lower for smaller datasets but generally increases with more data, peaking around 60% of the training data before dipping again. This indicates that the model benefits from more data up to a certain point but then starts to overfit. High bias is not evident as both scores are reasonably high. Between 20% to 80% of training data proportions a significant gap between training and validation scores is observed which can be an indication of high variance.

The fitness curve for GA (Figure 7) shows a rapid decline in fitness error, which signifies that the GA quickly found a good weight for the model. After around 20 iterations, the error stabilizes. This suggests that the GA has possibly converged to a solution and is not finding better configurations beyond this point. The GA aims to find the best weights for the ANN. It does this by evolving a population of possible solutions over iterations. In our case, GA converges too quickly, which caused the selection of a sub-optimal solution, and increasing bias. This can be observed by the poor performance of ANN-GA in the test set compared to other optimization algorithms in my analysis.

To overcome the issue, I tested additional iterations to find better weights but it did not help. Then I tuned the mutation rates of GA to find its optimum value. This helped GA to find better weights and a better generalization of the ANN model. Figure 8 shows the test accuracy with different mutation probabilities and figure 9 shows an updated learning and fitness curve, showing an improvement after hyperparameter tuning. There is still room for improvement by tuning additional hyperparameters.

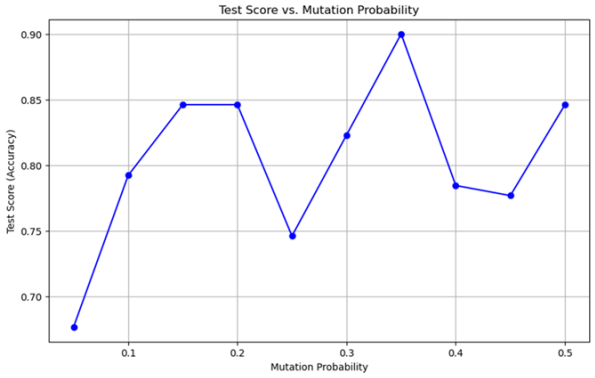

Figure 8. Finding an optimum mutation probability for the GA algorithm. The optimum value was found to be 0.35

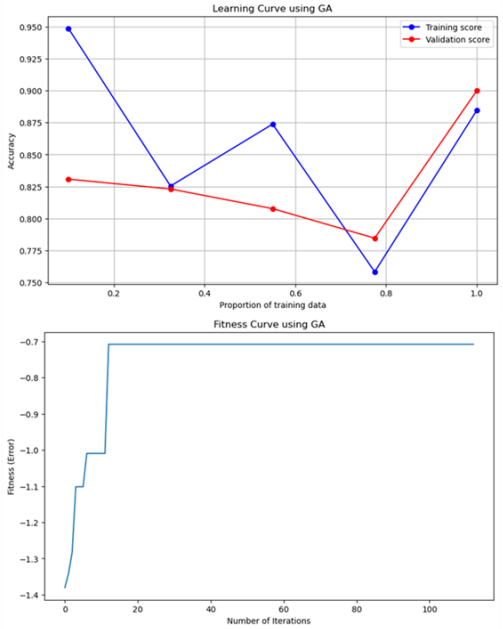

Figure 9. Repeating the ANN-GA with an updated mutation probability

Training time and algorithm comparison

All three algorithms showed high training accuracy when trained on smaller data proportions (Table 1). However, accuracy decreased as more data was introduced. This suggests a high variance scenario, where the model captures not just the underlying data distribution but also random noise. Among the three algorithms optimizing the weights of the ANN model, all performed well, but SA was the best in terms of

both accuracy and efficiency by finding the best solution very quickly. According to Table 1, SA not only outperformed GA and RHC on the validation set but also found the optimum weight more quickly.

Model	Overall accuracy	Training time
ANN-RHC	0.92	1.05
ANN-SA	0.94	0.38
ANN-GA	0.9	16.95

Table 1. Comparison of the three models in the prediction of Diabetes

Conclusion of Part-2

Optimization Behavior: While overall trends for training accuracy are similar across the algorithms, specific behaviors, especially regarding validation accuracy and iterations for convergence, might differ.

There is room for improvement in the optimization algorithms and the classification task as general for my dataset. For example, hyperparameter tuning: As I tested for GA, tuning the hyperparameters can help optimization algorithms find a better solution, especially if they're trapped in local optima.

Feature Engineering: According to Figure 1, several features in the dataset have a low correlation with Diabetes. Removing these and adding more relevant features can help improve classification accuracy.

References

Dataset https://archive.ics.uci.edu/dataset/529/early+stage+diabetes+risk+prediction+dataset

MIrose package:https://github.com/gkhayes/mlrose