CS7641 Homework 2 – Jayden Youngsik Cho

Table of Contents

0. Introduction

The purpose of this analysis is to show different randomized optimization algorithms in machine learning field. Optimization is defined as 'the action of making the best or most effective use of a situation or resource,' where 'best' can be measured with and represented by certain fitness function and 'a situation or resource' can be comprehended as cost function. In this analysis, four different randomized optimization algorithms are depicted: Randomized Hill Climbing, Simulated Annealing, Genetic Algorithm, and MIMIC.

In the previous assignment, the artificial neural network (ANN) algorithm is tested with default weights. However, now those weights for different predictors are calculated or optimized with three optimization algorithms—randomized hill climbing, simulated annealing, genetic algorithm—and results are compared. Then, 3 additional optimization problems are given to show advantages of each of those algorithms plus MIMIC. This analysis will end with evaluation/comparison of performances from those.

All the algorithms are written in Java with ABAGAIL package. To run the code most properly with recommended installations, see README.txt file.

1. Dataset Description

The only one dataset is used in this analysis: breast cancer dataset from the previous assignment. To recall, it is illustrated below in detail.

1.1 Dataset: Breast Cancer

This breast cancer dataset was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. The dataset has 10 attributes/predictors and a response. This dataset originally has 699 data points from 8 different instances, where each instance is recorded in different times, including 16 observations with missing data. All the rows that contains missing data and NA values are omitted before training on algorithms, and therefore the number of data points remains to 683. It is far greater than the number of attributes, and that makes this dataset quite suitable for any classification problem, for the specific purpose of this analysis, artificial neural network (ANN). In terms of its context, the attributes are in regard of breast cancer cells, and they include the size of the cell, the thickness of clump, uniformity of cell size, and so on. The goal here is to correctly classify if a patient has breast cancer cells or not by only using characteristics of the cells. Here is more information about the attributes.

```
#  Attribute                     Domain
   -- -----------------------------------------
   1. Sample code number            id number
   2. Clump Thickness               1 - 10
   3. Uniformity of Cell Size       1 - 10
   4. Uniformity of Cell Shape      1 - 10
   5. Marginal Adhesion             1 - 10
```

```
 6. Single Epithelial Cell Size   1 - 10
 7. Bare Nuclei                   1 - 10
 8. Bland Chromatin               1 - 10
 9. Normal Nucleoli               1 - 10
10. Mitoses                       1 - 10
11. Class:             2 for benign, 4 for malignant
```

Bare nuclei are naked nuclei, which is typically seen in cell degeneration. More information can be found at: https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.names.

2. Algorithms

2.1Finding Optimal Weights for Artificial Neural Network

2.1.1 Initialization

Before running any algorithms, initial parameters are set. The method for each algorithm is different since they require different sets of parameters. For the basic neural network algorithm, the learning rates are fixed to default. The number of hidden layers is decided based on different number of iterations of the algorithm. For simulated annealing, different sets for the cooling rate and the temperature are tested and the best parameters are chosen where its set results in the lowest error. Parameters for the genetic algorithm are population ratio, crossover ratio, and mutation ratio. The same process is conducted here as in the simulated annealing, where each combination of parameters is tested and whichever the set has the lowest error is chosen to be used in training and testing data on the artificial neural network. The default ratio of training and testing sets is 0.3 to 0.7. To re-initialize, set the 'shouldFind' Boolean variables in the code to 'true' and re-run. Those 3 algorithms are then compared with each other and backpropagation method used in the previous assignment as well.

2.1.2 Randomized Hill Climbing

A randomized hill climbing (RHC) is a variant of hill climbing, which is a mathematical optimization technique based on an iterative algorithm that starts with an arbitrary solution to a problem and attempts to find a better solution near it. RHC adds stochastic element to it, so that it randomly chooses a set of possibly better solutions in its neighbor and find the best one among those. The whole iterative process ends when it cannot find a better solution. The random component in RHC somewhat addresses the limitations of a deterministic hill climbing that is likely to get stuck in a local optimum.

Algorithm below shows the iterative process of how it finds the point minimizing the cost function.

```
Input: Iter_max, ProblemSize
Output: Current
Current ⟵ RandomSolution(ProblemSize)
For (iter_i ∈ Iter_max)
   Candidate ⟵ RandomNeighbor(Current)
   If (Cost(Candidate) ≥ Cost(Current))
      Current ⟵ Candidate
   End
End
Return (Current)
```

The known advantages of deterministic hill climbing includes: it uses very little memory and it sometimes finds a reasonable solution in a large search space. Additionally, it is helpful to solve pure optimization problems where the objective is to find the best state according to the objective function. It also requires much less conditions than other search techniques. Therefore, it is widely used in job scheduling, automatic programming, and so on. One of the big disadvantages is that it is usually stuck in a local optimum, however, stochastic element of RHC somewhat alleviates this problem. Though, it is still not guaranteed to find the global optimum.

For this assignment, RHC is used to determine the best set of weights for the artificial neural network (ANN). In other words, it is used to minimize the training and testing errors. In addition to error rates, the time to take until the solution is found is measured because time can sometimes be used as a trade-off, for example, when the fitness value of its solution is low, but it is fast.

Given the breast cancer dataset, RHC is used to find optimal point in ANN. Then, the optimized ANN is used to predict if a patient has breast cancer or not with their predictors.

Here is the result:

```
==========Randomized Hill Climbing=========
Train Error: 34.100%
Test Error: 37.073%
Time Elapsed: 2.720 s
```

It shows that the fitted ANN model has more error when exposed to testing set with optimal weights found by RHC.

2.1.3 Simulated Annealing

The simulated annealing (SA) algorithm goes a step further than RHC. At each step, the SA heuristic considers some neighboring state s' of the current state s, and probabilistically decides between moving the system to state s' or staying in state s. These probabilities ultimately lead

the system to move to states of lower energy. Typically, this step is repeated until the system reaches a state that is good enough for the application, or until a given computation budget has been exhausted. The purpose is the same as the RHC to find the global optimum, but the acceptance method of neighbors is different. The energy of current state, the energy of candidate state, and the temperature calculate the acceptance probabilities. The more detailed introduction to the SA is explained here: http://www.theprojectspot.com/tutorial-post/simulated-annealing-algorithm-for-beginners/6.

Given those properties, the synthetic global temperature plays a crucial role in determining the evolution of states. It is sensitive to coarser energy variations when the temperature is set to high, and it is sensitive to finer variations when it is set to low.

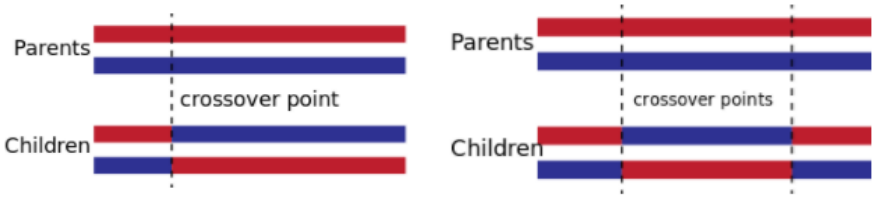
For this analysis purpose, the parameters are tested in the initialization step and best parameters are set. Then, the SA model is used to find the optimal weights for the ANN, and the ANN model is used to predict responses for training and testing datasets.

```
==========Simulated Annealing=========
Train Error: 34.100%
Test Error: 37.073%
Time Elapsed: 2.728 s
```

The result above shows exactly same error for both training and testing sets as with the RHC. It means that they found the same optimal weights. However, the SA took very slightly little more time possibly because it searched the larger space.

2.1.4 Genetic Algorithm

The genetic algorithm (GA) uses a whole different approach to find optimization solutions. The GA is a metaheuristic inspired by the process of natural selection from the larger class of evolutionary algorithms. A typical GA requires a genetic representation of the solution domain and a fitness function to analyze the fitness value of each representation. Generally, bits are used to represent the solution domain. One of the huge advantages of this way of representation is its convenience of moving their parts easily when genetic operators—crossover and mutation—are performed on. Crossover is used to vary the programming of a chromosome or chromosomes from one generation to the next. It is identical to those biological crossover, where genetic algorithms are based upon. There are many ways to perform crossover: single-point, double-point, uniform, and so on. Here are the simple depictions of crossover. The image on the left represents single-point crossover and the right demonstrates double-point crossover.



For this analysis, the single-point crossover method is used.

Another representative genetic operator is called mutation, also identical to the biological mutation. It is mainly used to maintain genetic diversity of genetic algorithm chromosomes when moving toward next generations. Mutation alters one or more gene values in a chromosome from its initial state in random in a hope to find a better solution with higher fitness value than its previous state. The probability of mutation can be defined by a user, and therefore it should be set to a low value. If it is too high, the GA algorithm will simply be a primitive search on entire species.

The algorithm can be initiate with customized population size, population ratio, crossover ratio, and mutation ratio. All the ratios are user-defined and used to determine probabilities of their corresponding genetic operators. The iteration process starts with the default population and generates new population based on possible crossover and mutation and selects best individuals. It keeps iterating until some termination condition is met. The termination condition can be: an optimal solution found, fixed number of generation reached, budget limit exceeded, and so on.

For this assignment, the initial parameters are estimated by the same process as before. Sets of the parameter values are tested and one set is chosen that has best result or minimum error. Then, the weights for the ANN are found and the ANN model is used, again, to predict responses for both training and testing datasets.

```
==========Genetic Algorithms=========
Train Error: 34.100%
Test Error: 37.073%
Time Elapsed: 0.495 s
```

It resulted in the same train and test errors with much less time to compute.

2.2 MIMIC

MIMIC is firstly introduced in 1997 by De Bonet, Isbell, and Viola. Referring to the paper, MIMIC is a framework that analyzes the global structure of the optimization domain, considered as one of the most effective optimization search algorithm. Like other optimization techniques introduced above, MIMIC uses iterative method. It firstly generates population with individuals chosen uniformly across the input domain. Then, it updates the parameters of the density estimator from a sample. After generating more samples from the distribution, it updates its population median fitness that is equal to the Nth percentile of the data. Finally, it keeps

individuals only under the new fitness level. It basically uses much information from the data, so that it will not have to visit the points that have already been visited. More detailed information about MIMIC can be found at the paper: https://www.cc.gatech.edu/~isbell/papers/isbell-mimic-nips-1997.pdf. The paper explains all the assumptions and general comparisons of MIMIC with other optimization algorithms.

In the context of this analysis, MIMIC is only used for the section 2.4 to show what kind of problems MIMIC should be used.

In the next 3 sections, I am going to show advantages of each type of optimization algorithms if used on specific types of problems.

2.3Max-k Coloring Problem

The first problem I want to address is the Max-k Coloring Problem. This problem is very similar to the graph coloring, which is to find a way of coloring the vertices of graph such that no adjacent vertices share the same color. The fitness function of a solution should be the number of the same adjacent colors presented in that solution. This problem captures the advantages of genetic algorithm because it searches parallel from a population of points, and therefore, it can avoid being stuck in a local optimum. Also, since it works on chromosomes, it works better on problems where their solutions can be encoded and represented with discrete values like this problem. The fitness value represents the number of correctly colored areas: higher means better.

The initial set of parameters for this problem consists of the number of vertices, the number of adjacent nodes per vertex, and the number of possible colors. Those are then seeded into the genetic algorithm and used to build a color matrix. With an initial population generated, the genetic algorithm uses three different operators to modify the individuals: swapping neighbors, mutation, and single crossover. As addressed above, these genetic operators can lead into more diverse set of individuals. Then, the iteration of generating next populations terminates when it finds a solution. In addition to the genetic algorithm, the randomized hill climbing, the simulated annealing, and MIMIC are also run to solve this problem for comparison:

```
Randomized Hill Climbing Optimal Fitness Value: 410.0
Failed to find Max-K Color combination !
Total Time Taken: 107
============================
Simulated Annealing Optimal Fitness Value: 410.0
Failed to find Max-K Color combination !
Total Time Taken: 82
============================
Genetic Algorithm Optimal Fitness Value: 450.0
Found Max-K Color Combination !
Total Time Taken: 30
============================
MIMIC Optimal Fitness Value: 450.0
Found Max-K Color Combination !
Total Time Taken: 121
```

Firstly, the randomized hill climbing and the simulated annealing algorithms show a susceptible move in this problem since they fail to find the solution. They also apparently present the lowest fitness values of 410. The genetic algorithm and MIMIC do find the optimal solution with the fitness value of 450, but the genetic algorithm triumphs because the total time to find the solution is significantly lower.

2.4 N-Queens Problem

The N-Queens problem is simulated to show advantages of randomized hill climbing and simulated annealing since they are variants of hill climbing search algorithm. The N-Queens puzzle is the problem of placing *N* chess queens on *NxN* chessboard so that no two queens threaten each other. The optimal solution should have no queens in the same diagonal, row, or column, for which a solution generally exists for all natural numbers N with the exception of 2 and 3. And, because solutions are guaranteed to exist, simulated annealing shines on this problem. The fitness value represents how well the queens are distributed. The only parameter *N* is initialized as 20 to make it complex enough, and then, different algorithms are performed:

```
Randomized Hill Climbing Optimal Fitness Value: 187.0
Total Time Taken: 1
============================
Simulated Annealing Optimal Fitness Value: 189.0
Total Time Taken : 0
============================
Genetic Algorithm Optimal Fitness Value: 184.0
Total Time Taken : 46
============================
MIMIC Optimal Fitness Value: 182.0
Total Time Taken : 101
```

The comparison of results clearly shows the prestige. Both randomized hill climbing and simulated annealing search algorithms took nearly 0 milliseconds to find the optimal solution. Those have advantages over others because there is no local optima and the immediate optimal point they find is the global optimum. Also, those algorithms require much less conditions than others, so that those do not have to undergo additional processes or filters. In addition to that, they reached higher fitness values than others.

2.5 Continuous Peaks Problem

In order to show the advantages of MIMIC, the continuous peaks problem is presented. This problem consists of number of peaks and valleys plus possibly plains. As the name assumes, the solution for this problem is to find the global optimum among the existing peaks and valleys, either maximum or minimum, in a continuous domain. Parameters for this problem include the number of ranges and the number of peaks. The number of peaks is set to the number of ranges divided by 10. By default, the number of ranges is set to 120 to make it complex enough. As a result, MIMIC outperforms:

```
Randomized Hill Climbing Fitness Value: 153.0
Total Time Taken: 144
============================
Simulated Annealilng Fitness Value: 157.0
Total Time Taken: 259
============================
Genetic Algorithm Fitness Value: 149.0
Total Time Taken: 171
============================
MIMIC Fitness Value: 178.0
Total Time Taken: 7067
```

One big number is displayed: the time that MIMIC took. However, this is not a bad happening. Although MIMIC takes a lot longer than any other algorithms, its fitness value is so high that no algorithm could not even get close. Other algorithms stayed around 150 when MIMIC grasped almost 180. It takes so much time only because it tries to learn as much as information and use them. Again, MIMIC thus can be very useful when the cost function is very high. In this case, the fitness value should signify the global optimum.

3. Evaluation

The purpose of this analysis was to evaluate different stochastic search algorithms by depicting advantages of each algorithms when used in an optimal setting. It firstly started off by comparing three algorithms, randomized hill climbing, simulated annealing, and genetic algorithm, to see which one optimizes the weights for artificial neural network. Overall, all of them did well and resulted in the same errors, which means they ended up having the same sets of weights. Secondly, three problems were presented to show advantages of each type of algorithms. Starting with genetic algorithm on the max-k coloring problem, those problems successfully portrayed advantages of each of them.

Since all of them has their own advantages and disadvantages, it is mostly up to users who can choose which to use and customize for their own problem.