Randomized Optimization

Machine learning can also be understood as finding the parameters that fit the data model best to the observed data, i.e. it is an optimization problem. Randomized optimization is a set of optimization method trying to do some kind of random search, which is useful in avoiding being stuck at local optimum when compared to traditional method like gradient decent.

There are two parts in this project. In the first part, four randomized optimization techniques will be introduced first. Then they will be applied to three traditional optimization problems, and will be compared among each other.

In the second part, an artificial neural network model will be fit to the dataset from Assignment 1, and the weights of the neural networks will be estimated by gradient decent and three other randomized optimization methods, and then will be compared among each other.

It will be interesting to see how different randomized optimization algorithm perform in different problem, and which one fits better for a certain type of problems.

Part 1 – Introduction to Randomized Optimization Algorithms

Randomized Hill Climbing

Hill Climbing is an optimization algorithm that search for the optimum by moving the variable X in the directly such that the value of f(x) is increased (decreased) for a maximization (minimization) problem. It is intuitive, and is effective algorithm, however, it is quite possible that it is stuck at a local optimum. Therefore, a nature extension of it is to have a restart at a random point again in order to explore the space to see if there are any better local optimum elsewhere and hoping to find the true global optimum after a sufficient number of trials.

Simulated Annealing

Simulated Annealing is another randomized algorithm which aimed at avoiding being stuck local optimum. By setting a certain probability that it will reject an hill climbing movement and go for a point elsewhere, As time passes, the rejection probability will decrease and hopefully it converge to the global optimum by exploring and exploiting at the same time.

Genetic Algorithm

Genetic algorithm is inspirited by biology. But in the optimization space, it is trying to optimize the objective function f(x) by choose variable x that is performing well; and mixing the x of those are performing well up, and discarding those doing poorly, and hoping will create a better and better generation after sufficient iterations.

MIMIC

MIMIC is an optimizing algorithm that analyse the global structure of the optimization landscape. The historical knowledge of the optimization algorithm is tracked, and the distributional properties are exploited to find a global optimum.

Part 1 – Three Optimization Problems

Three optimization problems are posted in this section. In each of them all four algorithms are applied, and for each of them, different parameters are used, and the one with the best performance from each of them are chosen and compared among each other. The following parameters are tested for each of them:

Randomized Hill Climbing:	10 random restarting
Simulate Annealing:	Temperature parameter at 0.15,0.35,0.55,0.75,0.95
Genetic Algorithm:	10, 30, 50 out of 100 population are allowed to have mutation, and 10, 30, 50 out of 100 populations are kept each iteration.
MIMIC:	50 out of 100 populations is kept each iteration, and the parameter are set at 0.1, 0.3, 0.5, and 0.7.

Traveling Salesman Problem

A salesman has to visit all his/her clients from city to city, one by one, and this traveling salesman problem is find the shortest path that allowing so without missing any of them. In this exercise, a map with 100 random points are used, and each of the 4 algorithms (with different parameters) are used to solve this map

A detailed description of this type of problem can be found at https://developers.google.com/optimization/routing/tsp/tsp

The following table shows the performance of each of the randomized optimization algorithms:

Algorithm	RHC	SA	GA **	MIMIC
Fitness	0.0499	0.0512	0.1044	0.0208
Time	0.0285	0.0120	0.7931	710.1375
Function Evaluation	3011	3011	1452256	331000

Genetic Algorithm is found to be the best fit to this problem. This is because that in the traveling salesman problem, it is very useful to keep the choices with the shortest path, and kick out those with a long distance (high cost). The genetic algorithm is is doing something exactly the same, and thus it perform really well.

Iteration	30	60	150	300
Fitness	0.0816	0.0860	0.0945	0.1044
Time	0.0898	0.1947	0.4268	0.7931
Function Evaluation	15074	29536	72948	1452256

The table below shows that the performance of the genetic algorithm improves quite a lot when number iteration is increased.

Continuous Peaks Problem

A continuous peaks problem is a problem that involve finding of the global optimum point in a space that there more than one local optimums (peaks). This is aimed to test if a certain algorithm is good in avoiding stuck at a local optimum instead of getting the global one.

The following table shows the performance of each of the randomized optimization algorithms:

Algorithm	RHC	SA **	GA	MIMIC
Fitness	98.4	99	92.6	74.4
Time	0.0349	0.0152	0.5882	36.6330
Function Evaluation	5011	5011	349583	551000

The simulated annealing is found to be the best among four, while randomized hill climbing follows really closely. This is because both of them are aimed at avoiding being stuck at a local optimum by restarting at random point or by exploring the space, and thus they fit these multiply continuous peaks problem well.

Flipflop Problem

A flipflop problem is a problem that related to a flipflop machine, which return one when two neighbour nodes are the same, and zero otherwise.

The following table shows the performance of each of the randomized optimization algorithms:

Algorithm	RHC	SA	GA	MIMIC **
Fitness	760.6	783	587.8	804
Time	0.0414	0.0204	3.0211	2849.1470
Function Evaluation	3011	3011	137258	331000

MIMIC is the winner for this case. In this problem, tracking the structure of the system helps a lot, as the flip and flop is exactly about how the signal/data are ordered, and thus MIMIC fit this case well.

Part 2 – Background of Dataset

Only the Dataset 1 from Assignment 1 is used for this exercise. It is a classification problem with only 2 level (true and false) in the dependent variables and it has a mixture of continuous and discrete independent variables. First 50% of the dataset is used as the training set; the second 25% is used as the testing set; and the last 25% is used as the validation set.

Dataset 1 – Default of Credit Card Client

The first dataset is related to the default of credit card clients of customersâ€™ default payments in Taiwan. It will be interesting for banks to know which kind of clients are more likely to default their credit card payment and thus have a better business strategy.

This dataset includes the record of payment history of 30000 clients. The dependent variable is if the client obligated the payment or not, and there are 23 independent variables:

X1: Amount of the given credit including both the individual consumer credit and his/her family (supplementary) credit.
X2: Gender (1 = male; 2 = female).
X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).
X4: Marital status (1 = married; 2 = single; 3 = others).
X5: Age (year).
X6 - X11: History of past payment.
X12 - X17: Amount of bill statement.
X18 - X23: Amount of previous payment.

In this project, only X1 to X11 is considered for modelling for sake of simplicity and the purpose of avoiding cruse of dimensionality.

Source: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

Part 2 – Comparison of Artificial Neural Networks Fitting

A neural network with only 1 hidden layer, and 8 nodes there are fitting, and the weights of that are estimated by different optimization methods. In this case, the variable space are not discrete but continuous, which may be methods like the gradient descent used in a standard artificial neural network works better.

Comparison

The following table shows the performance of each of the randomized optimization algorithms:

Accuracy	GD	RHC	SA	GA
Training Set	0.8925	0.6259	0.3741	0.6259
Testing Set	0.8999	0.6355	0.3645	0.6355
Validation Set	0.7348	0.6987	0.3013	0.6987

Since it is a continuous space, the gradient descent works in this case. Possible because the neural network system is just of one hidden level with eight nodes, the structure is not really complex itself, and thus there are no or not much local optimum to be stuck with, and thus it is not necessary to use random hill climbing or other methods to avoid this issue.

CHAN, Siu Hang (Sammy), Frankfurt am Main, 2018-03-12 04:55