IE437 Assignment 2 : Multiple Simulated Annealing with Range Limitation

20246150, LEE Eungyeol

May 11, 2024

Contents

1 Introduction **3**

2 Method **3**
2.1 Object Function Optimization 3
2.1.1 Neural Network Object Function 3
2.1.2 Support Vector Regression Object Function 4
2.2 Gradient Boosting Regressor 4
2.3 Simulated Annealing 4
2.4 Simulated Annealing : Multiple Simulated Annealing 4
2.5 Simulated Annealing : Range Limitation 5

3 Experiments **5**
3.1 Object Function Experiments 5
3.2 Simulated Annealing Experiments 6
3.3 MSA + RL Tuning 7

4 Conclusion and Result **8**

1 Introduction

This assignment is to find the argument that can maximize object value by using 9,992 data from a block box function with 5-dimensional input and 1-dimensional output.

I tried some method based on **Simulated Annealing** to optimize the black box function. The reasons why I use that are, first, it is easier to implement compared to various optimization algorithms, second, it is known to have good performance in finding the global optimal case even when there are various local optimal cases, and finally, I am interested in metaheuristics algorithm.

Simulated Annealing is an approximate method to solve an optimization problem and is used in situations where finding solution is computationally expensive or realistically difficult. This method is inspired by the annealing process in metallurgy, the process of heating a material and then slowly cooling it to create a stable crystal structure.

2 Method

I used a modified and improved method on basic simulated annealing. Below is a figure of my overall algorithm.

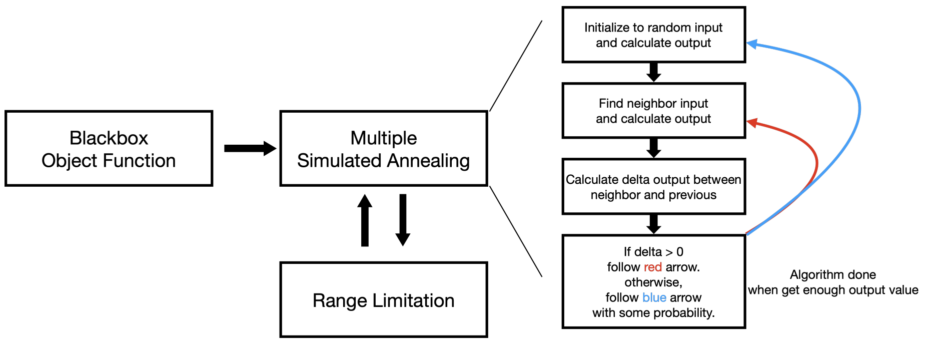
Blackbox Object Function
Multiple Simulated Annealing
Range Limitation
Initialize to random input and calculate output
Find neighbor input and calculate output
Calculate delta output between neighbor and previous
If delta > 0 follow red arrow. otherwise, follow blue arrow with some probability.
Algorithm done when get enough output value

Figure 1: This is the overview own MSA+RL algoithm.

The steps are divided into three. There are the object function step, multiple simulated annealing step, and range limitation step.

2.1 Object Function Optimization

2.1.1 Neural Network Object Function

The Neural Network model consists of following step.

Input linear layer - Batch normalization - Activation function - Hidden linear layer - Batch normalization - Activation function - Output layer

The reason why I consider above model is described in **Section 3.1** below.

2.1.2 Support Vector Regression Object Function

Support Vector Regression is strong machine learning method for various regression task. Especially, support vector with radial basis function kernel is one of the strongest method. By reducing the margin of data, error can be minimized.

2.2 Gradient Boosting Regressor

Gradient boosting regressor is also a powerful classifier based on ensemble learning. This is a classification method that uses several different decision tree classifiers.

2.3 Simulated Annealing

Simulated Annealing is used to optimize the objective function $f(x)$. where x represents the value in the solution space. The goal is to find the value of x that maximize $f(x)$. The algorithm follows below steps.

Step 0 : Simulated annealing start with temperature $T \Leftarrow T_0$ and Initial random solution $x \in [d]$, d is the dimension of input(i.e. in our assignment 2 problem, $d = 5$). Temperature is used during the exploration process to decide whether to continue the search or find a new path, and it changes in each process.

Step 1 : In the existing solution x, the nearest neighbor solution $x' \Leftarrow x + \delta$ is randomly selected.

Step 2 : Calculate $\alpha \Leftarrow f(x') - f(x)$.

Step 3 : If $\alpha > 0$, then substitute the existing solution as a neighboring solution $x \Leftarrow x'$ and return to Step 1. Otherwise, Randomly re-initialize the solution x with the probability $e^{\frac{-\alpha}{T}}$. Finally, Change the temperature according to the cooling rule by $T \Leftarrow Cooling(T)$.

Step 4 : When enough solutions are obtained or the temperature has dropped sufficiently, the algorithm done.

2.4 Simulated Annealing : Multiple Simulated Annealing

Multiple Simulated Annealing is a traditional method that finds the optimal solution while multiple agents share their information. I applied an algorithm that applies simulated annealing through the highest optimal solution of all multiple agents. Through this method, agents that are far from the solution can be quickly initialized and the optimal solution can be found more quickly.

The algorithm only needs small change from the simulated annealing above.

Step 0 : Same as **simulated annealing Step 0**. But the Initial solution is $X \in [d \times n]$, where d is the dimension of input and n is the number of agent. Moreover, global optimal value initialize by $f_{optimal} \Leftarrow 0$.

Step 1 : Same as **simulated annealing Step 1**. In the existing solution x, the nearest neighbor solution $x' = x + \delta$ is randomly selected.

Step 2 : Calculate $\alpha \Leftarrow f(x') - f_{optimal}$.

Step 3 : Same as **simulated annealing Step 3**, but assign $f_{optimal} \Leftarrow f(x')$ when $\alpha > 0$.

Step 4 : Same as **simulated annealing Step 4**. When enough solutions are obtained or the temperature has dropped sufficiently, the algorithm done.

2.5 Simulated Annealing : Range Limitation

Range limitation is the most important element in this algorithm. This is a method to apply multiple search more effectively and explores more deeply the midpoint of the solution of the agent with the highest score. The boundary is gradually reduced so that useless input ranges can be removed. Since it is difficult to find a global optimal value at the beginning of the algorithm, the range limitation only applies a little at the beginning, but becomes more significant towards the end.

Step 0 : Initialize $X \in [d \times n]$ to random value, with range $R = [d \times 2]$. d is the input dimension, and the n is the number of agent. And similarly to multiple simulated annealing, global optimal value initialize by $f_{optimal} \Leftarrow 0$.

Step 1 : Same as **simulated annealing Step 1**. In the existing solution x, the nearest neighbor solution $x' \Leftarrow x + \delta$ is randomly selected. But x'_i that satisfy $R_{i0} \leq x'_i \leq R_{i1}$ for all i.

Step 2 : Same as **simulated annealing Step 2**. Calculate $\alpha \Leftarrow f(x') - f_{optimal}$.

Step 3 : Same as **simulated annealing Step 3**.

Step 4 : When enough solutions are obtained or the temperature has dropped sufficiently, $R \Leftarrow R/r$, where $r = l(i)$, l is the $R \Rightarrow R$ function which return the range limitation value, i is the number of iteration. Function l is more greater when i is more greater. And come back to Step 1 with newly initialized x that satisfy $R_{i0} \leq x_i \leq R_{i1}$ for all i.

Step 5 : When algorithm get enough solution, then algorithm done.

3 Experiments

I experimented object function and algorithm. All source codes are publicly available[1].

3.1 Object Function Experiments

First, I constructed four blackbox object functions which are introduced in **Section 2.1**, Neural Network, Neural Network Batch Normalization, Support Vector Regressor, Gradient Boosting Regressor. Following is the setup condition.

[1] `https://github.com/9tailwolf/Multiple-Simulated-Annealing-Range-Limitation`

Neural Network is the basic model of linear neural network, and it is consist of 3 linear layer, input dense layer(5×64), hidden layer(64×64), and output layer(64×1). Due to the simple data structure of 5 inputs and 1 output, dropout was not included as it was judged to be unnecessary. But batch normalization worked well. There also active function, $ReLU$, between all layer and layer.

Neural Network Batch Normalization is the advanced Model NN, neural network model with batch normalization. Parameters are all same as Model NN.

Support Vector Regressor is initialized with radial basis function, $C = 2, gamma = 1$.

Gradient Boosting Regressor is initialized with, $nestimators = 50, maxdepth = 100, learningrate = 0.1$.

And I experiment above four models. Each experiment was conducted a total of 10 times.

MS Loss	**NN**	**NN+BN**	**SVR**	**GBR**
Mean	2.135	0.414	**0.153**	1.223
Best	1.231	0.351	**0.143**	1.159

Table 1: Benchmark of Neural Network Models.

SVR model shows the most powerful performance. Constant learning was possible with little difference between the average and the best scores. **NN+BN** model also shows good performance. But the other methods are not enough to select as object function.

3.2 Simulated Annealing Experiments

I experiment four simulated annealing algorithm with 2 techniques, multiple simulated annealing(MSA), and range limitation(RL). The **SA** is the vaniala algorithm of simulated annealing, and the **MSA** is the multiple simulated annealing. In this case, the number of agent is 100. **SA+RL** is the simulated annealing with range limitation. this algorithm limit range 100 times. Lastly **MSA+RL** is the multiple simulated annealing with 100 agents, limiting range 100 times.

I choose two ojbect function, **NN+BN** and **SVR**, which show the strong performance in previous experiments.

Object	**SA**	**MSA**	**SA+RL**	**MSA+RL**
NN+BN$_{mean}$	78.640	90.718	78.595	**90.900**
NN+BN$_{best}$	87.521	90.838	82.852	**90.914**
SVR$_{mean}$	79.661	86.146	80.604	**86.153**
SVR$_{best}$	86.142	86.152	83.729	**86.153**

Table 2: Benchmark of various simulated annealing algorithm

The experiments show that the **MSA+RL** method worked effectively compared to the existing simulated annealing. Additionally, I could see that existing SA and MSA were working well. But the SA+RL shows bad performance. Also, due to the property of SVR, it is not sensitive to unobserved or outliers, so it cannot learn small amounts of data that close to the optimal value. Learning seems to more accurate, but the maximum value may be relatively lower compared to other models.

3.3 MSA + RL Tuning

Lastly, I experiment to see if there is any performance change depending on the coefficient in MSA+RL, the number of agent and range limitation. Since the range limitation narrows the input range, when it is too excessive, it may cause to finding optimal solution, and when it is too small, the algorithm may not be performed properly. Every experiments are run 10 times.

Agent— RL	**10**	**30**	**50**	**100**	**200**
10_{mean}	88.456	88.117	88.716	89.170	89.273
30_{mean}	89.041	89.342	89.357	89.395	89.399
50_{mean}	89.125	89.323	89.360	89.402	89.405
100_{mean}	89.319	89.383	89.407	89.407	89.410
200_{mean}	89.354	89.395	89.410	89.410	**89.413**

Table 3: Mean score benchmark of multiple simulated annealing range limitation algorithm with various coefficients. The horizontal axis is the number of range limitation and the vertical axis is the number of agent.

Agent— RL	**10**	**30**	**50**	**100**	**200**
10_{best}	89.147	89.371	89.386	89.395	89.408
30_{best}	89.319	89.414	89.411	89.411	89.413
50_{best}	89.394	89.401	89.408	89.413	89.413
100_{best}	89.398	89.411	**89.416**	89.415	89.413
200_{best}	89.411	89.411	89.413	**89.416**	**89.416**

Table 4: Best score benchmark of multiple simulated annealing range limitation algorithm with various coefficients. The horizontal axis is the number of range limitation and the vertical axis is the number of agent.

The **Table 3** is the mean score of algorithm benchmark, and the **Table 4** is the best score of algorithm benchmark. The result shows when the number of agents and range limitations exceeds a certain level, The optimal solution could be found. In most cases, the optimal value was found, but on average, the number of cases that were closest to the optimal solution increased as the number of agents and range limitations increased.

4 Conclusion and Result

I obtained 10 solutions with the MSA+RL algorithm, the NN+BN objective function, and the SVR objective function, which achieved the best results in the experiment. 5 solutions are from NN+BN objective function, and the other 5 solutions are from SVR objective function.

A	B	C	D	E
0.697	0.561	0.258	0.801	0.432
0.676	0.582	0.257	0.749	0.374
0.756	0.653	0.285	0.747	0.392
0.717	0.572	0.269	0.771	0.429
0.725	0.607	0.159	0.844	0.388
0.724	0.608	0.158	0.845	0.391
0.723	0.608	0.160	0.846	0.391
0.723	0.607	0.160	0.842	0.391
0.727	0.608	0.154	0.847	0.389

Table 5: Final solutions of assignment 2