1. Optimization Problems

1.1 Count Ones

Suppose we an array with length N, each element in the string is a binary number (i.e., 0 or 1).The problem is to find the array that has the maximum number of 1s.

Hence, we can define this problem over discrete-valued space. The fitness function to maximize can be defined as follow:

$$f = \sum_{i=1}^{N} x_i$$

1.2 Four Peaks

The fitness of this problem consists of counting the number of 0s at the start, and the number of 1s at the end and returning the maximum. If both the number of 0s and the number of 1s are above some threshold value T then the fitness function gets a bonus of 100 added to it. This is where the name “four peaks” comes from: There are two small peaks where there are lots of 0s, or lots of 1s, and then there are two larger peaks, where the bonus is included.

To formally define the fitness function:

$$f(\vec{X}, T) = \max\left[tail(0, \vec{X}), head(1, \vec{X})\right] + R(\vec{X}, T)$$

where

$$tail(0, \vec{X}) = \text{ number of trailing 0's in} \vec{X}$$

$$head(1, \vec{X}) = \text{ number of leading 1's in} \vec{X}$$

$$R(\vec{X}, T) = \begin{cases} N & \text{if } tail(0, \vec{X}) > T \text{ and } head(1, \vec{X}) > T \\ 0 & \text{otherwise} \end{cases}$$

There are two global maxima for this function. They are achieved either when there are $T + 1$ leading $1's$ followed by all 0's or when there are $T + 1$ trailing 0's preceded by all 1's. There are also two sub-optimal local maxima that occur with a string of all $1's$ or all 0's.

1.3 Knapsack

The Knapsack problem is a constrained optimization problem: given a set of items, each with a weight and a value, determined the number of each item to include in a collection so that the total weight is no larger than a given limit and the total value is as large as possible.

To formally define this problem, let there be n items, $z1$ to zn where zi has a non-negative value vi and non-negative weight wi. xi is the number of copies of the item zi. This is subject to the constraint W which is the maximum weight.

Hence, we can define this problem over discrete-valued space. The fitness function to maximize can be defined as follow:

$$f = \sum_{i=1}^{N} v_i x_i, s.t. \sum_{i=1}^{N} w_i x_i \leq W$$

$$s.t., \sum_{i=1}^{N} w_i x_i \leq W, x_i \geqslant 0$$

2. Comparison of four algorithms on solving above problems

2.0 Parameter Tuning

For Randomized Hill Climbing, we allow a maximum of 100000 iterations, and the maximum number of attempts to find a better neighbor at each step is 10.

For Simulated Annealing, we allow a maximum of 100000 iterations, and the maximum number of attempts to find a better neighbor at each step is 10. Besides, we will be using a exponential decay function.

$$T(t) = \max(T_0 e^{-rt}, T_{min})$$

Where T_0 = 1, r = 0.005, T_min = 0.001.

For Genetic Algorithm, we allow a maximum of 100000 iterations, and the maximum number of attempts to find a better neighbor at each step is 10. Besides, we set population size to 20, probability of a mutation at each element of the state vector during reproduction to 0.1.

For MIMIC, we allow a maximum of 100000 iterations, and the maximum number of attempts to find a better neighbor at each step is 10. Besides, we set population size to 20, proportion of samples to keep at each iteration to 0.2.

2.1 Count Ones

This is a simple problem with a single global optima. Hence, we would expect all four algorithms to find the best solutions, but Simulated Annealing should excel.

Here we train each algorithm on different size of input vector x. On each input size, we trained each algorithm ten times and take the average on the metrics.

Below is result of final fitness function value each algorithm found, or we can call it optimal solution found by each algorithm. Each row represent a different input vector size.

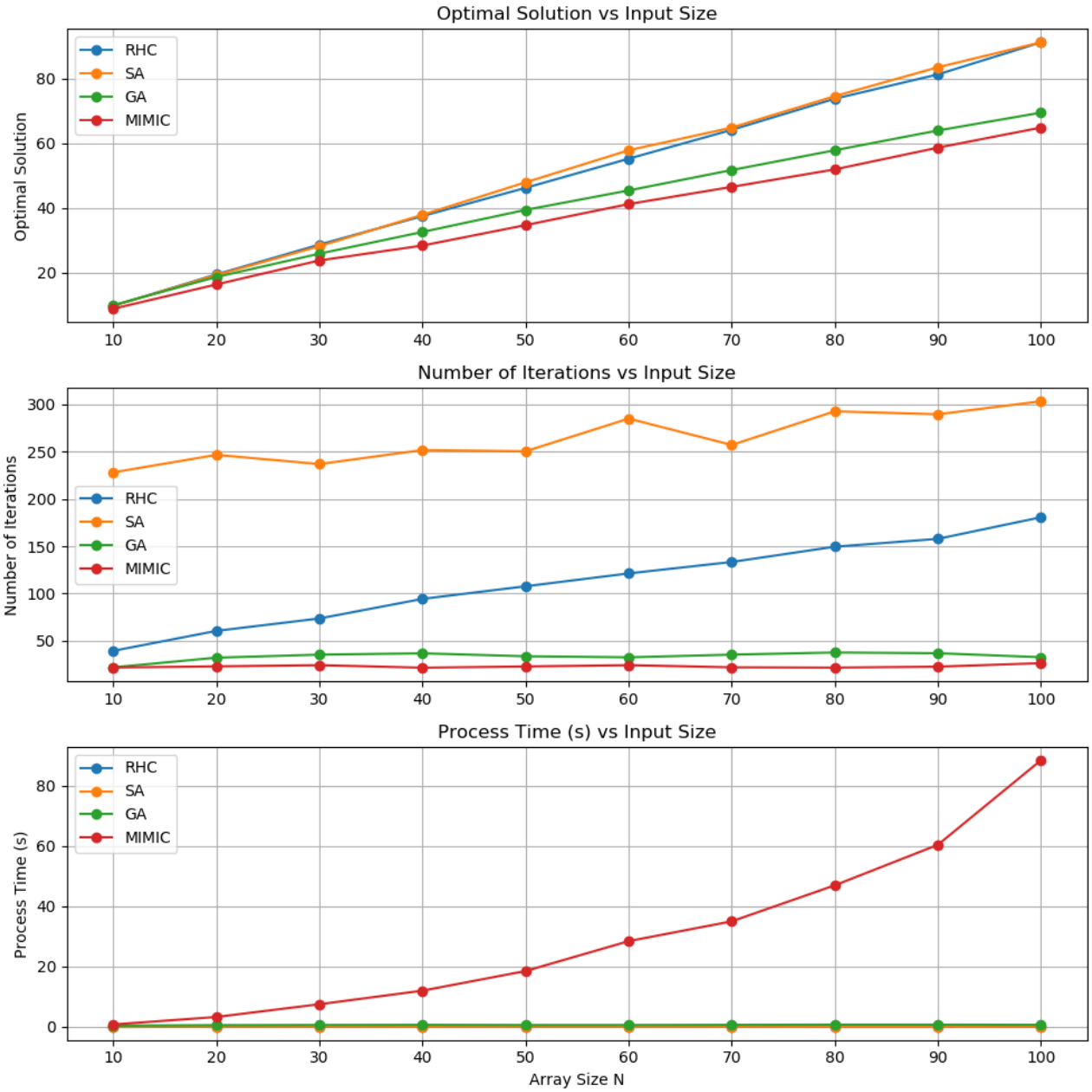
	RHC	SA	GA	MIMIC
10	9.8	9.8	10	8.9
20	19.5	19.3	18.7	16.4
30	28.7	28.2	25.9	23.8
40	37.5	37.9	32.6	28.4
50	46.2	47.9	39.4	34.7
60	55.2	57.8	45.4	41.2
70	64.1	64.8	51.7	46.5
80	73.7	74.5	57.8	51.9
90	81.2	83.4	63.9	58.6
100	91.1	91.1	69.4	64.8

	RHC	SA	GA	MIMIC
10	39.5	228	22.1	21.9
20	60.6	246.5	32.3	23.1
30	73.7	236.8	35.5	24.4
40	94.4	251.6	36.9	21.7
50	107.7	250.3	33.8	23
60	121.3	284.8	32.7	24.4
70	133.4	257	35.5	22.1
80	149.5	292.5	37.8	21.8
90	157.8	289.4	37	22.8
100	180.6	303.1	32.8	26.5

Below is result of total number of iterations each algorithm took, or we can call it optimal solution found by each algorithm. Each row represent a different input vector size.

	RHC	SA	GA	MIMIC
10	0.001	0.008	0.332	0.730
20	0.002	0.008	0.507	3.215
30	0.002	0.008	0.565	7.445
40	0.002	0.008	0.617	11.940
50	0.003	0.008	0.551	18.457
60	0.003	0.009	0.550	28.370
70	0.003	0.008	0.614	34.943
80	0.004	0.009	0.677	46.896
90	0.004	0.009	0.670	60.337
100	0.004	0.010	0.621	88.385

Below is result of total process time each algo took. Each row represent a different input size. Below is the plot.



Apparently, Simulated Annealing performed best here in terms of finding the optimal solution and process time.

For the nature of this Count One problem, Simulated Annealing's evaluation functions is inexpensive to compute and there are no local optima in which to get stuck.

Randomized Hill Climbing is equally fast or even faster than Simulated Annealing, but as the input vector size become large, RHC won;t find the best solution because the amount of randomness added to the exploration is not as good as Simulated Annealing.

Genetic Algorithm did not gain much from mutation, MIMIC's knowledge on previous explorations did not help much either.

When there is no structure or pattern in the search space or past points, GA and MIMIC won;t perform very well. This Count one problem is a typical example.

2.2 Four Peaks

This is a problem with two local optima and two sharp global optima at the edges of the problem space. Genetic Algorithms are more likely to find these global optima than other methods. Let's see the result.

Below is result of final fitness function value each algorithm found, or we can call it optimal solution found by each algorithm. Each row represent a different input vector size.

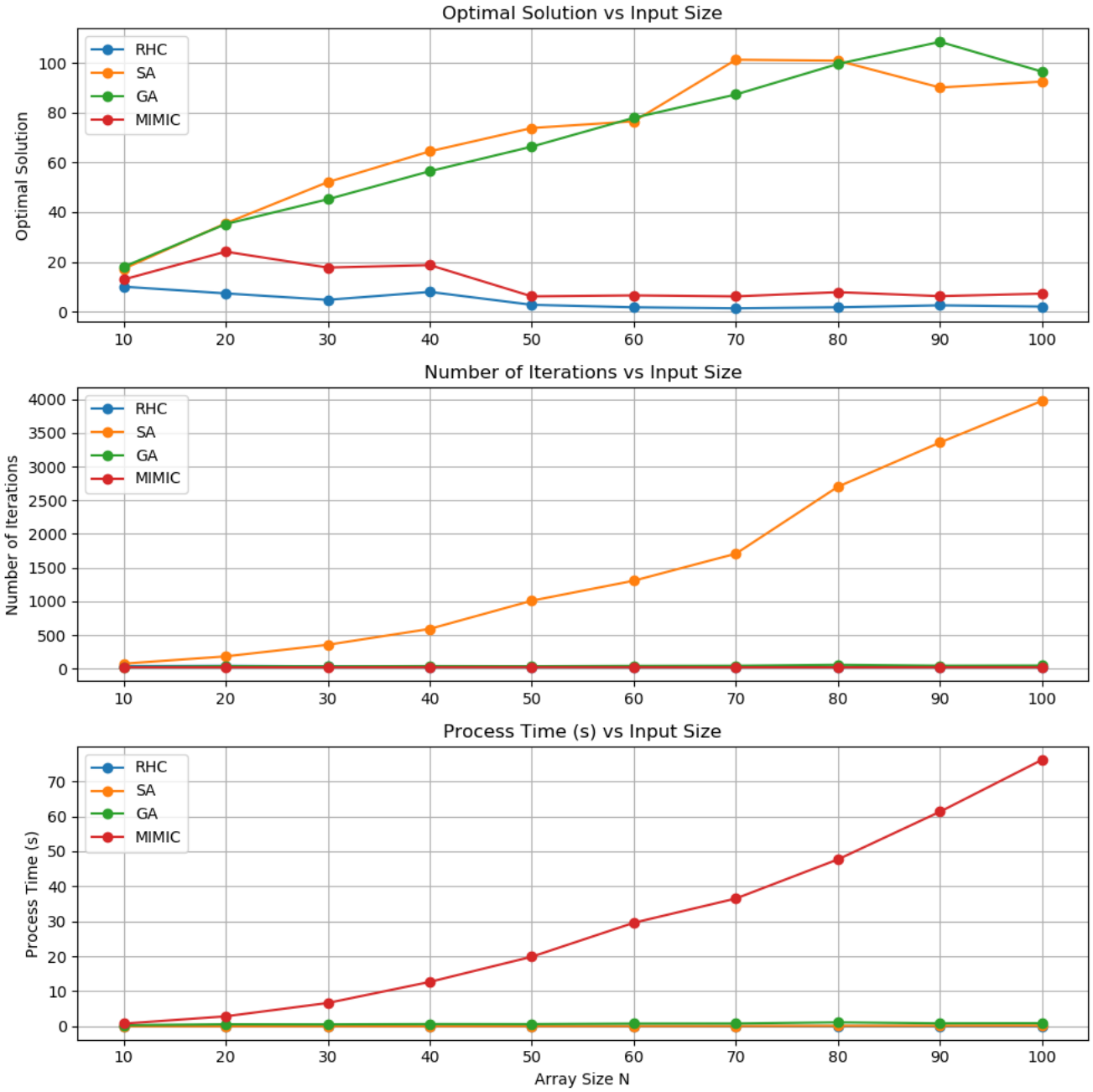
	RHC	SA	GA	MIMIC
10	10	17.2	18	12.9
20	7.3	35.5	35.2	24.1
30	4.7	52.1	45.2	17.7
40	7.9	64.5	56.5	18.7
50	2.7	73.9	66.4	6.1
60	1.7	76.6	78	6.5
70	1.3	101.4	87.4	6.1
80	1.7	101	99.7	7.8
90	2.5	90.2	108.6	6.2
100	2	92.6	96.6	7.2

	RHC	SA	GA	MIMIC
10	37.3	74.5	21.1	21.2
20	41.6	182.6	35.9	21
30	33.4	354.9	33.2	21.2
40	28	591	37.9	22.6
50	27.2	1009.4	35.5	23.2
60	21.5	1306.6	40.9	22.6
70	24.2	1708	42.7	21.1
80	23.6	2702.6	56.9	21.3
90	23.6	3358.5	43.7	21.8
100	23.5	3977.7	45.9	21.5

Below is result of total number of iterations each algorithm took, or we can call it optimal solution found by each algorithm. Each row represent a different input vector size.

	RHC	SA	GA	MIMIC
10	0.001	0.002	0.318	0.745
20	0.001	0.006	0.551	2.832
30	0.001	0.013	0.526	6.647
40	0.001	0.020	0.616	12.675
50	0.001	0.036	0.602	19.876
60	0.001	0.050	0.757	29.585
70	0.001	0.067	0.787	36.516
80	0.001	0.109	1.113	47.696
90	0.001	0.133	0.831	61.334
100	0.000	0.163	0.879	76.163

Below is result of total process time each algorithm took. Each row represent a different input vector size.



Below is the plot.

As we can see from above, GA is able to find the best solution among the four algorithm, and it runs very fast, takes fairly small amount of iterations to find the optima.

As we have mentioned above, GA excels here because the nature of mutation and evolution in its searching, it will avoid the local optima and add randomness to its search, and eventually find the global optima efficiently. Both RHC and SA tend to stuck at the local optima, this is what happens when we apply a greedy algorithm to a problem like Four Peak, which has local optima.

2.3 Knapsack

This is a classic NP-Hard optimization problem with no polynomial time solution. The strength of MIMIC was highlighted in this context, as it exploited the underlying structure of the problem space that was learned from previous iterations.

Below is result of final fitness function value each algorithm found, or we can call it optimal solution found by each algorithm. Each row represent a different input vector size.

	RHC	SA	GA	MIMIC
20	124.5	126.7	142.4	158.3

	RHC	SA	GA	MIMIC
20	29.1	38.1	27.1	21

Below is result of total number of iterations each algorithm took, or we can call it optimal solution found by each algorithm. Each row represent a different input vector size.

	RHC	SA	GA	MIMIC
20	0.001	0.002	0.484	2.753

Below is result of total process time each algorithm took. Each row represent a different input vector size.

As we can see here, MIMIC can find the best solution, and took the fewest iterations to achieve it. Thus when having history knowledge would help, MIMIC would be the best option.

3. Training Neural Network with random search algorithm

Below is the problem I defined in Assignment 1

3.0 Wine Recognition (Data link)

The data of Wine Recognition is the results of a chemical analysis of wines grown in the same region in Italy by three different cultivators. There are thirteen different measurements taken for different constituents found in the three types of wine.

With 13 different statistics, we would like to predict which type a glass of wine belong to.

The data contains 13 numeric features and each data point belongs to one of 3 classes.

Following features are included:

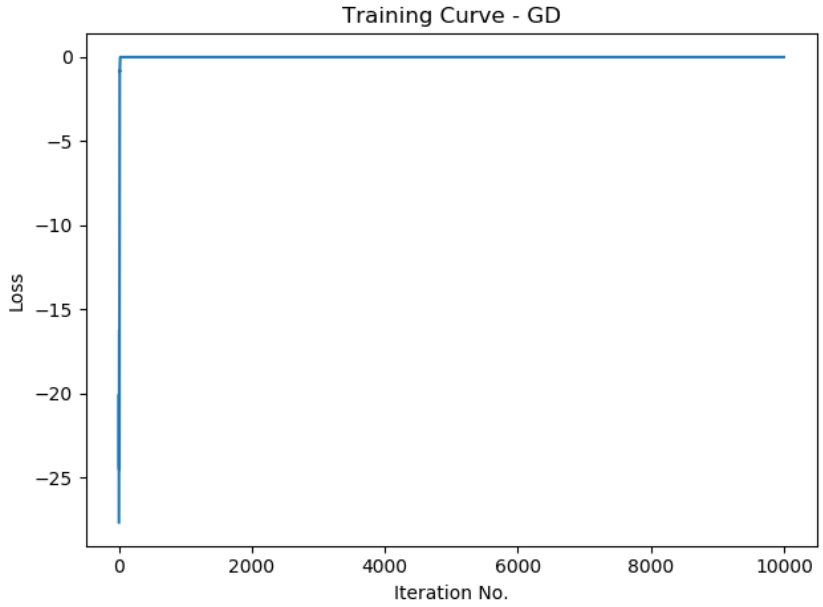
- Alcohol
- Malic acid
- Ash
- Alcalinity of ash
- Magnesium
- Total phenols
- Flavanoids
- Nonflavanoid phenols
- Proanthocyanins
- Color intensity
- Hue
- OD280/OD315 of diluted wines
- Proline

Now let's use RHC, SA, GA to train the weights in neural networks, and compare the result with that in traditional backprop with Gradient Descent.

Here for all different algorithms, we use the same simple neural network structure, with 5 hidden layers, each layer contains 100 units. The activation function is ReLU.

3.1 Traditional Gradient Descent

Here we allow a maximum of 10000 iterations, and a learning rate of 1e-5.

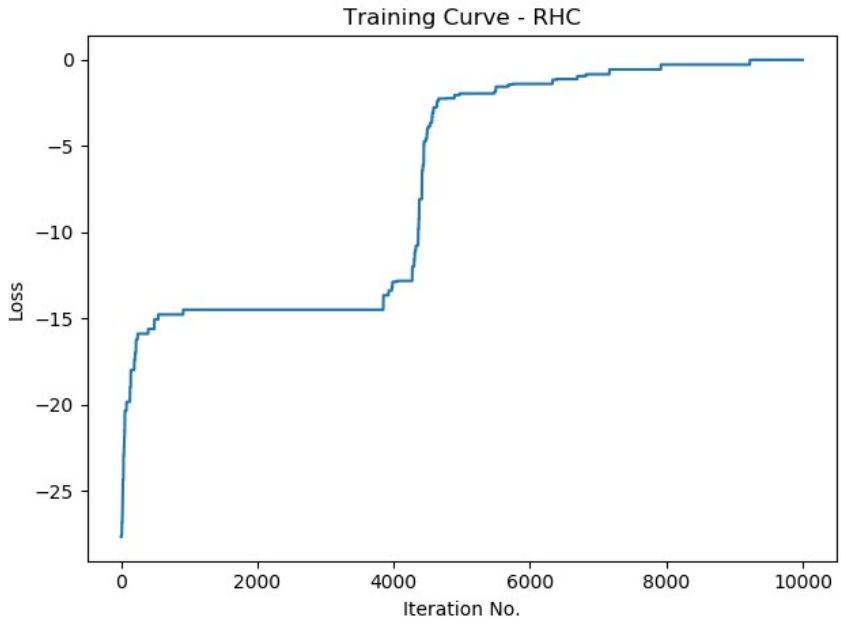


As we can see, this algorithm converges very fast, at about 100 iterations, it can find the optima.

3.2 Randomized Hill Climbing

Here we allow a maximum of 10000 iterations, and the maximum number of attempts to find a better neighbor at each step is 100, and a learning rate of 10.

Below is the training curve.



As we can see, even if RHC can somehow find the best solution, but not very efficient. It merely converges to optima at max 10000 iterations. This bad performance by RHC is as expected since it only randomly search its neighbor without any further information or mutation steps.

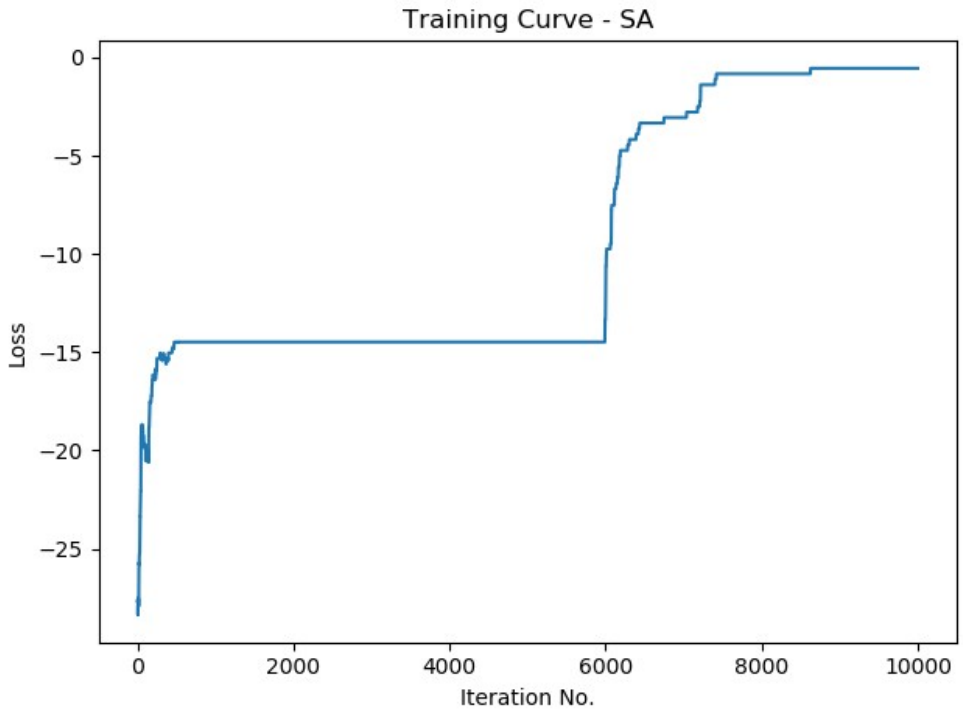
3.3 Simulated Annealing

Here we allow a maximum of 10000 iterations, and the maximum number of attempts to find a better neighbor at each step is 100, and learning rate 10. Besides, we will be using a exponential decay function.

$$T(t) = \max(T_0 e^{-rt}, T_{min})$$

Where T_0 = 1, r = 0.005, T_min = 0.001.

Below is the training curve.

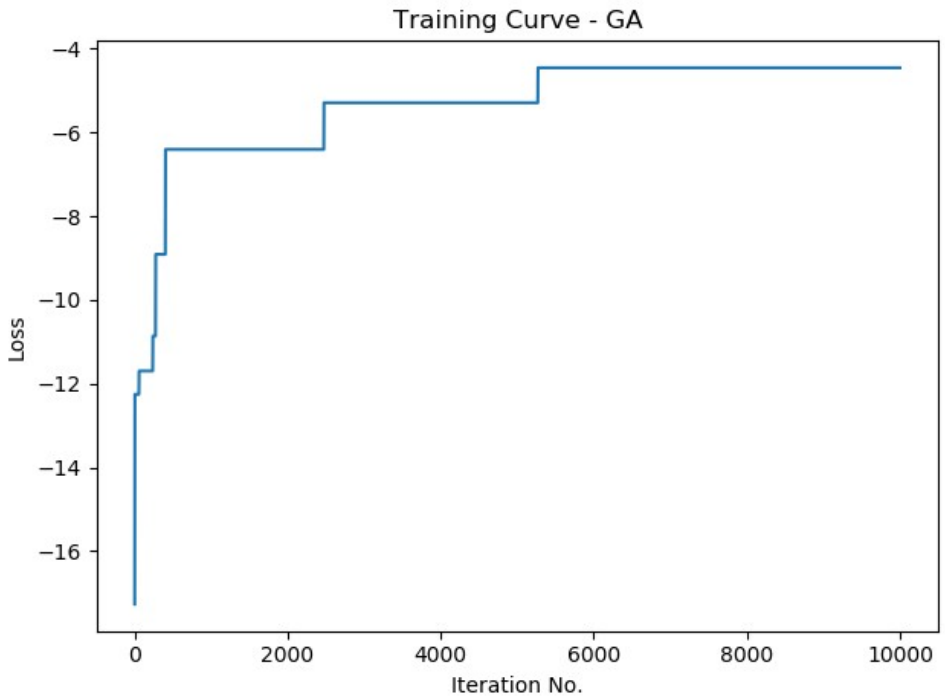


As we have expected, its performace is similar to RHC, merely converges at max iterations. But we can clearly the improvement compared with RHC. It took less iteration to find the global maximum. Again, because of its nature of only randomly search nearby space, it won't be a great algorithm.

3.4 Genetic Algorithm

Here we allow a maximum of 10000 iterations, and the maximum number of attempts to find a better neighbor at each step is 10, and learning rate 10. Besides, we set population size to 20, probability of a mutation at each element of the state vector during reproduction to 0.1.

Below is the training curve.



As we can see, GA performs better than RHC and SA because this problem of find best weights for nn is not a easy task. There surely will be some local optima. GA's nature of mutation and exploration gives the power to converge faster.

3.5 Analysis

Below is the summary of all four algorithms.

	RHC	SA	GA	GD
Train Accuracy	100.00%	98.39%	87.10%	100.00%
Test Accuracy	90.74%	85.19%	90.74%	96.30%
Process Time	111.9	171.4	13,529.5	620.7
Final Loss	0.000	-0.557	-4.457	0.000
# of Iterations before convergence	10000	8000	6000	100

For GA, it is most time consuming, it need large amount of calculations in each iteration. From Iteration 0 to 6000, GA is improving in step wise function. The reason is that at certain iteration, it suddenly finds a better offspring mutations that improves a lot. We also different annealing temperature decay function, but the result and training curve is not very sensitive to that.

For RHC and SA, there are large amount of iterations where the loss is not improved. If we allow early stops, the algorithm will mostly likely converge at local optima. And both algorithm converge very slow. One way to improve it is use larger learning rate, but it can only help to a degree.

For all four algorithm, we also tried different learning rate, the smaller the learning rate is, the slower the training process will be, but it can eventually find the global optima given infinite time. The larger the learning rate is, the faster the training process will be, but less likely to find global optima, because it will miss the global optima with a large step.