Charlie McKnight
Charles Isbell
CS4641
03/11/2017

Randomized Optimization

A comparison of Randomized Hill Climbing,
Simulated Annealing, Genetic Algorithms, and MIMIC

Intro

The data included was retrieved by extending the functionality ABAGAIL. Modified code is included. Raw data in also included as a spreadsheet. If the actual parameters of a function are unspecified the following algorithms were used. Neighbors are chosen for bitstrings by randomly setting one of the bits to a random value. Mutations are performed the same way. Mating is performed by randomly selecting bits from either of the parents.

Phishing Websites Data Set

Description

For the first section I used Randomized Hill Climbing, Simulated Annealing, and A Genetic Algorithm to determine the optimal weights for the nodes in a Neural Network.

For the analysis on the previous datasets part of the assignment I chose to use my Phishing Website Dataset which contains 30 boolean attributes (-1, 1 with zero indicating an unknown value). There were 11055 data-points labeled as either a Phishing website or not. I used 70% for training and 30% for testing as I found this to be the best split in the previous assignment. One benefit to this dataset is that the inputs are already normalized due to the fact that they are booleans.

I chose to have 1 hidden layer with 9 hidden nodes because this was one of the simplest models that yielded good data in the first assignment. I then modified the abalone test in ABAGAIL to conduct the tests (called P2_N1.java). Tests were run with 1000 iterations to determine optimal parameters for each algorithm then final tests were performed 5 times and the results averaged.

Since there is a continuous space that must be searched for the weights it is worth noting how we decide certain factors such as neighbors. In this case a neighbor is found by changing one of the weights at random to a value that is close (randomly selected closeness) to the current value. Mating randomly selects weights from the parents and does not have the issue of a continuous space. Mutation is done by finding a “neighbor” and using that as the mutation. Since there is an infinitely large space to search finding the true min/max is in many cases impossible and in the other cases impractical. In this case I used a fixed number of iterations and decided that a global maximum was determined by performing 100% on the testing set, not achieving 0 SSE.

Back-Propagation

Back-propagation is the benchmark by which I will be judging the Randomized Optimization Algorithms. Back-Propagation on the same training and testing data with the same number of hidden node achieved a training accuracy of 97.89 and a testing accuracy of 95.87. It is worth noting that KNN achieved a testing accuracy of roughly 97%.

RHC

For Randomized Hill Climbing I used the default ABAGAIL implementation. This implementation does not randomly restart but this did not seem necessary because of the results. In that RHC was achieving an average testing accuracy of 99.78%

Simulated Annealing

Unlike with simulated annealing there were two parameters to play with for simulated annealing, The starting temperature and the cooling factor. I chose play with the starting temperature as it appeared to have a larger impact on this dataset. In the table below you can see the results from the test.

Heat	Training Time	Testing Time	Training Accuracy	Testing Accuracy
1.00E+01	31.505	0.083	0.9944	0.99488
1.00E+02	32.283	0.042	0.98863	0.98614
1.00E+03	30.504	0.041	0.98888	0.98553
1.00E+04	30.46	0.045	0.99018	0.98915
1.00E+05	30.517	0.043	0.9642	0.96323
1.00E+06	30.672	0.047	0.99328	0.99066
1.00E+07	30.899	0.041	0.95347	0.95298
1.00E+08	31.033	0.042	0.98178	0.98101
1.00E+09	30.741	0.039	0.97854	0.97257
1.00E+10	29.479	0.042	0.95722	0.95419

It's clear from the table that the initial amount of heat has little impact on the outcome. This is supported by the fact that RHC consistently achieves a similar level of accuracy. This tells me that there are very few local maximums. The data would back this up because each attribute is a boolean and which does not yield itself aswell to local maximums as continuous datasets would. I chose to keep the heat as low as possible because of this and left it at 1E1.

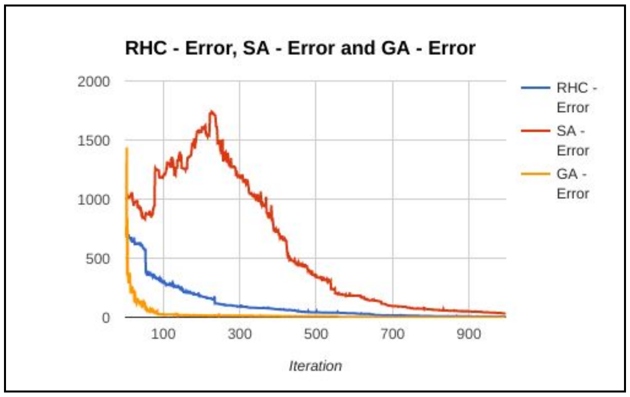
Genetic Algorithm

The genetic algorithm also had a number of parameters to play with, population size, number to mate, and number to mutate. I chose to stay with a population of 100 but increasing the proportion of the other two parameters.

Heat	Training Time	Testing Time	Training Accuracy	Testing Accuracy
N=100, mate= 10 mut = 10	189.164	0.084	100	100
N=100, mate= 30 mut = 30	482.836	0.052	100	100
N=100, mate= 50 mut = 50	661.588	0.05	100	100
N=100, mate= 70 mut = 70	793.782	0.05	100	100
N=100, mate= 90 mut = 90	991.607	0.049	100	100

As you can see no matter what the proportion of mates and mutations the training and testing accuracy were 100. I was shocked to see that not even one in the testing set was miscategorized but after double checking the code this is an accurate result. Seeing as I was already achieving perfect accuracy I decided to not play with the parameters any more.

Comparison



The graph of sum of square error vs iteration is also very informative. It is worth noting that comparing the accuracy at any point between the algorithms is not very informative since

GA perform many more function evaluations per iteration than the other two (which perform 1 evaluation per iteration).

As we can see simulated annealing because of its initial temperature towards the beginning is allowed to progress towards suboptimal solutions while RHC heads the correct direction from the start. This makes it so that what is usually SA's benefit of being able to escape local minimums turns out to be its downfall on a problem with almost no local minimums. This leads to slower optimization than with RHC.

For the genetic algorithm we can see that it learns very quickly. I attribute this to the weights on each input being important in and of themselves. So when two of the population are mated there is a high chance of good weights from one and the other being passed on to the children and the mutation factor insures that good weights are found.

Summary

Every single one of the Randomized Optimization Algorithms showed a large improvement over Back-Propagation. The two best performers being RHC and GA. GA was astounding due to the fact that it found a solution that achieved perfect accuracy on the testing set. RHC was a stand out due to the fact that is is usually regarded as a less refined version of SA but in this case performed better. To select between them it is worth mentioning the differing circumstances under which this training might occur.

	Training Time	Training Accuracy	Testing Accuracy
BackProp	54.98	97.89	95.87
RHC	29.36	0.9989	0.9978
SA	29.22	98.035	97.74
GA	190.09	100	100

Under these circumstances when training time is under a minute for all algorithms and the fitness function runs almost instantaneously it is hard to argue with the results that GA achieved. However, it is easy to envision circumstances under which RHC could become the better of the two algorithms. If the fitness function was much more difficult computationally or perhaps must be remotely called on a different server, RHC would be the better choice. Since even though they were both run for 1000 iterations GA uses roughly 100 times as many function calls per iteration. In this case RHC would run much “faster”.

Counting Ones Problem

Problem

The counting ones problem is fairly straightforward and very easy for humans to wrap their minds around. The input is a bitstring and the fitness function is simply the number of ones contained in said bit string. Meaning a bitstring of length N would have a max fitness of N which corresponds to a bitstring of all ones. It is an interesting problem due to its ease of understanding and the fact that certain algorithms perform very well at it while others perform very poorly.

I evaluated the 4 functions at a variety of input sizes {60, 90, 120, 150, 180} keeping track of their max fitness, number of iterations, training time, and number of fitness function evaluations. These tests were performed by modifying ABAGAILs Counting Ones Problem. The number of iterations were capped at 10,000 and were cut off early if the global maximum was reached.

Ability to Reach Maximum

In all cases the ability to reach the maximum was not affected by the bitstring length. Simulated Annealing, Randomized Hill Climbing, and MIMIC were all able to reach the maximum well within the max number of iterations.

Iterations				
N	RHC	SA	GA	MIMIC
60	369	501	10,000	407
90	745	873	10,000	189
120	1181	1193	10,000	378
150	1617	1415	10,000	262
180	2316	1772	10,000	378

Genetic Algorithms really struggled when it came to its ability to reach the maximum. Through exploring with different parameters none of them were able to achieve the global maximum within a reasonable amount of computation time and population. I attribute GA difficulty in finding the optimal solution to the inherent “randomness” as opposed the other algorithms. The mating of two solutions is just as likely to bring down zeroes as it is to bring down ones and the mutations are just as likely to mutate a 1 to a zero as a zero to a 1. In addition the fitness of an additional 1 is not that great when comparing who in the population lives and who doesn't.

Function Evaluations and Clock Time

Since RHC, SA, and MIMIC were capable of achieving the global optimum the next method of differentiation would be the number of fitness function evaluations. As we can see RHC and SA use an order of magnitude less function evaluations. This in addition to MIMIC being more computationally expensive ends up corresponding to orders of magnitude larger wall clock times.

Function Evals			
N	RHC	SA	MIMIC
60	371	503	20,401
90	747	875	9501
120	1183	1195	18951
150	1619	1417	13151
180	2318	1774	18951

Max Time (S)			
N = 180	0.01544	0.0031	3.65

Analysis

For solving The Counting Ones problem SA/ RHC perform phenomenally. The ability of the two to trace a sort of path to the optimum allowed a very high performance with SA or RHC.

Four Peaks Problem

Problem

Given a bitstring of length N the fitness function is defined as follows.

$$f(\vec{X}, T) = \max\left[tail(0, \vec{X}), head(1, \vec{X})\right] + R(\vec{X}, T) \tag{7}$$

where

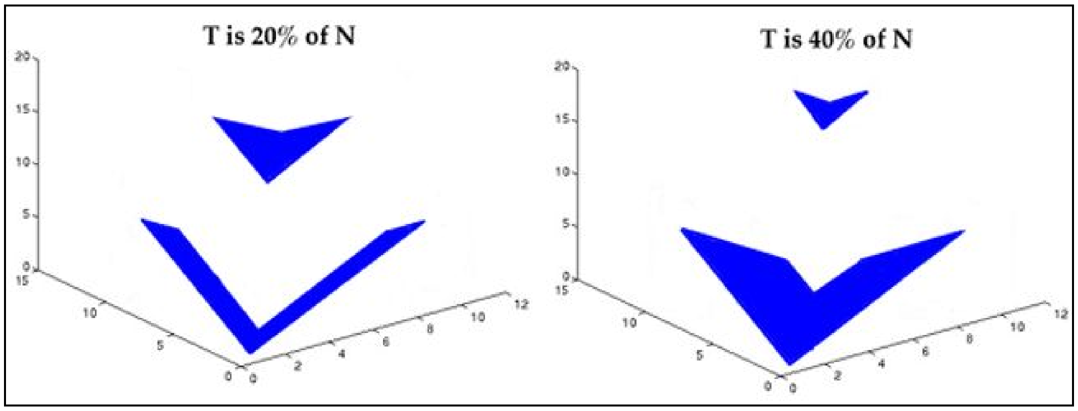
$$tail(b, \vec{X}) = \text{number of trailing } b\text{'s in } \vec{X} \tag{8}$$

$$head(b, \vec{X}) = \text{number of leading } b\text{'s in } \vec{X} \tag{9}$$

$$R(\vec{X}, T) = \begin{cases} N & \text{if } tail(0, \vec{X}) > T \text{ and } head(1, \vec{X}) > T \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Taken From Charles Isbell's paper Randomized Local Search as Successive Estimation of Probability Densities

There are two local minimums for the function. They are at all zero's or all one's. There are in addition to those, two global maximums at T + 1 leading one's or T+1 training zeroes.



Taken from http://www.cc.gatech.edu/~isbell/classes/2003/cs4600_fall/projects/project3/4pks.html

This is an interesting problem because of its discontinuity. Unlike the last problem where there was a trail to follow to get to the global max in this case the trails may lead to the global maximum but there is a good chance (depending on the size of T) that the "trail" will lead to one of the suboptimal local maximums.

Ability To Reach Maximum

As opposed to the previous problem RHC / SA will not perform well on this problem due to the fact that there is a large chance of getting sucked into one of the suboptimal local maximums. Whereas SA annealing is supposed to use the temperature as a way to escape local maximums since the path to the local maximums is so long it would take a lot of random chance to end up escaping to the true maximums. MIMIC and GA on the other hand have a much higher chance of success because they take a large number of initial samples and there is a much higher likelihood of a few of the samples being closer to the true maximum.

This is backed up by the data I acquired. I modified ABAGAIL's Four Peaks Problem to repeat the tests 50 times and average the results. The Algorithms were run until either they reached the Global Maximum or reached the maximum number of iterations (5,000). For these tests GA and MIMIC has populations of 2,000 with GA mating 1,000 and mutating 100 and MIMIC selecting 30 to keep.

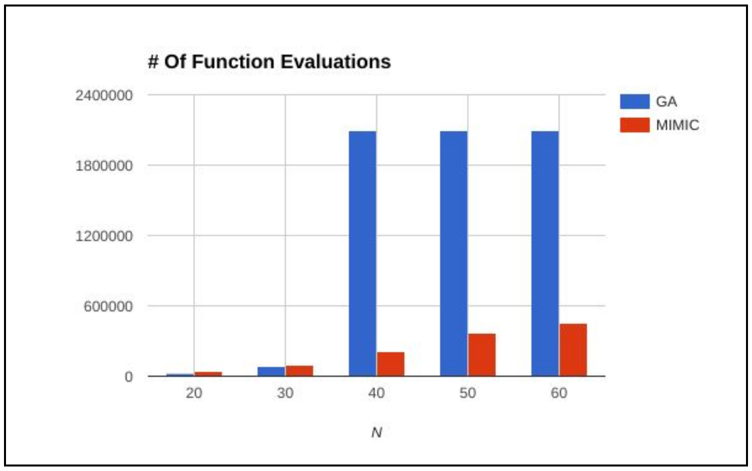
Found Max				
N	RHC	SA	GA	MIMIC
20	0.08	0.54	1	1
30	0.04	0.28	1	1

40	0	0.28	1	0.96
50	0	0.02	1	0.86
60	0	0	1	0.72

As one can clearly see as the bit string size increased the probability of RHC and SA finding the global optimum quickly approached zero because they became increasingly likely to get stuck in a local maximum. While GA and MIMIC were still able to find the global maximums only occasionally getting stuck in the local ones thanks to a large population size.

Function Evaluations

Since RHC and SA almost never actually reached the global maximum I will only be comparing GA and MIMIC. At smaller sizes of N the number of function evaluations that are required is roughly the same but as the size increases GA takes substantially more evaluations, roughly 10 times as many.



Mimic is able to achieve a faster convergence because unlike GA which rely more on chance, mimic learns about its sample space and is able to find the Global Maximum more quickly because of this, assuming there are at least a few good randomly selected samples.

Analysis

For the four peaks problem MIMIC performs the best relative to its number of function evaluations. When it is able to find the solution it does so in very few evaluations and misses could be mitigated by increasing the population size as the problem grows, creating a larger chance of good samples being taken.

Uneven Palindrome Problem

Problem

This is a problem of my own design that I created by extending the existing test in ABAGAIL. The fitness function is defined as follows. A bitstring of length N is taken as input and it is given a score by evaluating it as a palindrome. Matching 1's are awarded 2 points. Matching 0's are awarded 1 point. Meaning there is a maximum score of N. A function could be described as follows

$$\sum_{i=0}^{N/2} eval(i, N-i-1) \quad | \; eval(a,b) \;=\{a \,!= b \;\rightarrow\; 0;\; a == b == 0 \rightarrow 1;\; a == b == 1 \rightarrow 2\;\}$$

This problem is interesting for a number of reasons. The first being that there are many local maximums since if there are two matching zeroes a change in either of them would lead to the score going down and an optimal solution requiring two bits to change. This will present a problem for SA and RHC since there is not a clear path to follow. RHC will get stuck in one of the many local maximums right of the bat. SA will fair better but the sheer number of local maximums will ensure even with a large heat level SA will also get stuck in a local maximum. GA and MIMIC should fair better.

The analysis was performed similarly to the previous problems. 50 iterations were performed and the results of successful iterations averaged. Parameters were played with but general trends seemed to hold regardless of the chosen parameters.

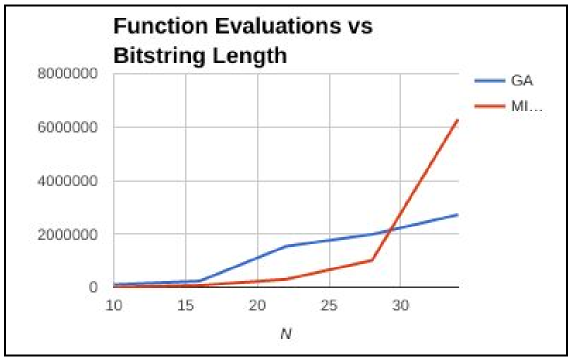
Ability to Reach Maximum

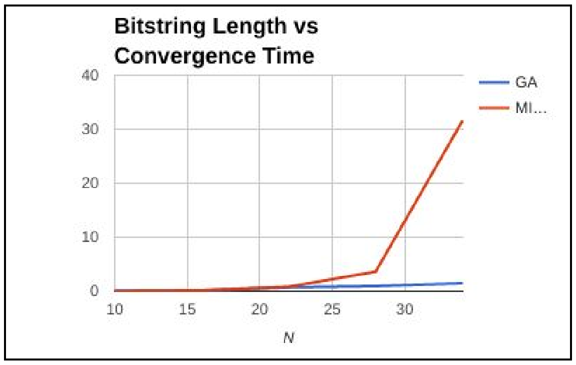
My prior hypothesis ended up being true. Even with a short bitstring of length 10, SA and RHC were rarely able to reach the global maximum. This remained true when changing to higher heat levels and over 9,000,000 iterations. Both algorithms became stuck in local maximums. GA and MIMIC faired much better always reaching the max with populations of just 200.

Reached Maximum				
N	RHC	SA	GA	MIMIC
10	0.02	0.28	1	1
16	0	0	1	1
22	0	0	1	1
28	0	0	1	1
34	0	0	1	1

Function Evaluations and Clock Time

The results became more interesting when differentiating GA and MIMIC by comparing function evaluations. It initially appeared that MIMIC would outperform GA handily. However, as the input size increased the story changed. With shorter bit string GA takes many more function evaluations (with surprisingly a shorter wall clock times) but as the problem size increases the number of MIMICs function evaluations seems to increase exponentially while GA's increases more linearly. Clock time follows this trend as well.





This makes GA the best performer on the Uneven Palindrome problem. While the performance of MIMIC is also improved by increasing the population size this holds true for GA as well. In addition with a constant population size GA outperforms MIMIC as the problem grows.

I attribute this to what was GA's downfall in the previous problems. GA's randomness proves to be a liability in problems when the other algorithms can "learn" the pattern with relative ease. In this case however, since the pattern is difficult for the algorithms to discover and SA / RHC gets stuck in local maximums, GA are able to escape the local maximums through crossover and mutation. This allows GA to escape the 2 bit change required to improve results. With MIMIC the population size is very important because if bad samples are chosen it will take a long time to recover. With GA on the other hand the mutations and crossovers prevent bad samples from being as detrimental.

Conclusion

The different Randomized Optimization problems perform differently depending on the type of problems. With certain algorithms performing extremely well on some problems and very poorly on others.

Randomized Hill Climbing and Simulated Annealing perform very well on problems where there is a sort of path that can be followed to the global optimum. SA's one major benefit over RHC in that it is able to escape local mins/maxs when the heat is high enough. Both algorithms struggle when there are discontinuities in the data such as The Four Peaks Problem when the algorithms can be sucked into the suboptimal local mins/maxs.

MIMIC performs relatively well on problems with discontinuities such as The Four Peaks Problem. Where others get stuck in local mins / maxs, MIMIC is able to get around this by sampling the area and with a large enough population size eventually whittling down the distribution to only the global optimum.

Genetic Algorithms are in many ways the most random of the algorithms which is detrimental in problems where there is a clear path to global optimum such as in the Count Ones problem. However, in some cases where the path is not as evident or there are an inordinate number of Mins / Maxs, genetics algorithms weakness can become its strength and help to discover the global maximum.

So in deciding which algorithm is best, it is essential to know your data and the problem you are trying to solve (or just try all of them).