Assignment 2: Randomized Optimization

Due: March 11th 11:55pm
Please submit via T-Square

Why?

The purpose of this project is to explore random search. As always, it is important to realize that understanding an algorithm or technique requires more than reading about that algorithm or even implementing it. One should actually have experience seeing how it behaves under a variety of circumstances.

As such, you need to implement or find implementations of several randomized search algorithms. In addition, you will need to exercise creativity in coming up with problems that exercise the strengths of each.

You may program in any language that you wish (we do prefer java, python, matlab, Lisp or C++). In any case *it is your responsibility* to make sure that the code runs on the standard CoC linux boxes.

Read everything below carefully!

The Problems Given to You

You must implement four local random search algorithms. They are:

1. Randomized hill climbing
2. Simulated annealing
3. A genetic algorithm
4. MIMIC

You will then use the first three algorithms to find good weights for a neural network. In particular, you will use them instead of backprop for the neural network you used in assignment #1 on at least one of the classification problems you created for assignment #1. Notice that weights in a neural network are continuous and real-valued instead of discrete so you might want to think about what it means to apply these sorts of algorithms in such a domain.

The Problems You Give Us

In addition to finding weights for a neural network, you must create three optimization problems on your own. For the purpose of this assignment an "optimization problem" is just a *fitness function* one is trying to *maximize* (as opposed to a cost function one is trying to minimize). This doesn't make things easier or harder, but picking one over the other makes things easier for us to grade.

Please note that *the problems you create should be over discrete-valued parameter spaces. Bit strings are preferable.*

The first problem should highlight advantages of your genetic algorithm, the second of simulated annealing, and the third of MIMIC. Be creative and thoughtful. It is not required that the problems be complicated or painful. They can be simple. For example, the 4-peaks and k-color problems are rather straightforward, but illustrate relative strengths rather neatly.

What to Turn In

You must submit via T-Square a tar or zip file named *yourgtaccount*.{zip,tar,tar.gz} that contains a single folder or directory named *yourgtaccount*. That directory in turn contains:

1. a file named *README.txt* containing instructions for running your code
2. your code
3. a file named *analysis.pdf* containing your writeup
4. any supporting files you need

The file *analysis*.pdf should contain:

- The results you obtained running the algorithms on the networks: why did you get the results you did? What sort of changes might you make to each of those algorithms to improve performance? Include supporting graphs or tables.
- A description of your optimization problems, why you feel that they are interesting, and why they demonstrate the strengths and weaknesses of each approach. Think hard about this.
- Analyses of your results. Why did you get the results you did? Compare and contrast the different algorithms. What sort of changes

might you make to each of those algorithms to improve performance? How fast were they in terms of wall clock time? Iterations? Which algorithm performed best? How do you define best? Be creative and think of as many questions you can, and as many answers as you can. You know the drill. **Note: Analysis write-up is limited to 10 pages.**

One example of analysis is available in the resource section of T-Square.

Grading Criteria

This assignment is out of **10 points**, each representing a percentage point of your final grade. The breakdown for grading is as follows:
1 point: A working implementation of all approaches listed above for your two classification problems.
9 points: Your analysis. You are being graded on your analysis more than anything else. However, analysis without proof of working code will harm your analysis grade.

The key thing is that your explanations should be both thorough and concise. Dig into many angles of analysis, but do not dig too deeply into any one part.

Follow the directions carefully. Failure to turn in files without the proper naming scheme, or anything else that makes the grader's life unduly hard will lead to an ignored assignment. There will be no late assignments accepted.