Numbers

The assignment is worth 10% of your final grade.

Read everything below carefully!

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 will be asked to implement or steal several randomized search algorithms. In addition, you will be asked to exercise your creativity in coming up with problems that exercise the strengths of each.

As always, you may program in any language that you wish insofar as you feel the need to program. As always, *it is your responsibility* to make sure that we can actually recreate your narrative if necessary.

Read everything below carefully!

The Problems You Give Us

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 create (for sufficiently loose values of "create" including "steal", though it's fairly easy to come up with simple problems on your own in this case) three optimization problem domains. 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 choice 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.*

You will apply all four search techniques to these three optimization problems. 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.

The Problems Given to You

In addition to analyzing discrete optimization problems, you will also 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 problems you created for assignment #1. Notice that this assignment is about an optimization problem and about supervised learning problems. That probably means that looking at only the loss or only the accuracy won't tell you the whole story. Luckily, you have already learned how to write an analysis on optimization problems and on supervised learning problems; now you just have to integrate your knowledge.

Because we are nice, we will also let you know about some pitfalls you might run into:

- The weights in a neural network are continuous and real-valued instead of discrete so you might want to think a little bit about what it means to apply these sorts of algorithms in such a domain.
- There are different loss and activation functions for NNs. If you use different libraries across your assignments, you need to make sure those are the same. For example, if you used scikit-learn and don't modify the ABAGAIL example, they are not.

What to Turn In

You must submit:

1. A file named *README.txt* that contains instructions for running your code
2. a file named yourgtaccount-*analysis.pdf* that contains your writeup.

The file yourgtaccount-*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? Feel free to include any supporting graphs or tables. And by "feel free to", of course, I mean "do".
- a description of your optimization problems, and why you feel that they are interesting and exercise the strengths and weaknesses of each approach. Think hard about this.
- analyses of your results. Beyond answering why you got the results you did you should compare and contrast the different algorithms. 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 writeup is limited to 10 pages total.

Coding Resources

Just like in assignment 1, you can use any library as long as it wasn't specifically written for this class, particularly for automating away your personal analysis. While looking at libraries, you might want to take a look at ABAGAIL:

https://github.com/pushkar/ABAGAIL (链接到外部网站。)

Grading Criteria

At this point you are not surprised to read that you are being graded on your analysis more than anything else. I will refer you to this section from assignment #1 for a more detailed explanation. On the other hand, I will also point out that implementing some of these algorithms is very easy (almost not worth stealing the code, but please feel free to do so anyway (I would steal the code)) but at least one of them requires some time (luckily, there are now versions of this algorithm out there to steal).

You should start now.

您将对这三个优化问题应用所有四种搜索技术。第一个问题应该突出遗传算法的优点，第二个问题是模拟退火，第三个问题是 MIMIC 的优点。要有创造力和思考能力。这并不要求问题是复杂或痛苦的。它们可以很简单。例如，4 峰和 k 色问题相当简单，但却相当简洁地说明了相对优势。给你的问题除了分析离散优化问题外，您还将使用前三种算法来为神经网络找到合适的权值。特别地，你们会用它们来代替在作业 1 中使用的神经网络的支撑至少在作业 1 中创建的一个问题上使用。注意，这个作业是关于优化问题和监督学习问题的。这可能意味着只看损失或只看准确性并不能告诉你整个情况。幸运的是，您已经学习了如何编写关

于优化问题和监督学习问题的分析;现在你只需要整合你的知识。因为我们是好人，所以我们也会让你知道你可能会遇到的一些陷阱:•神经网络中的权值是连续的实数，而不是离散的，所以你可能想要思考一下，在这样的领域中应用这些算法意味着什么。•网络有不同的损失和激活函数。如果你在作业中使用不同的库，你需要确保它们是相同的。例如，如果你使用 scikit-learn 而不修改 ABAGAIL 的例子，它们就不是。该交什么?你必须提交:1. 一 个 名 为 README.txt 的文件，其中包含运行代码的指令 2. 一个名为 yourgtaccount-analysis.pdf 的文件，里面包含了你的书面记录。文件 yourgtaccount-analysis.pdf 应该包含:•你在网络上运行算法得到的结果:为什么你会得到你做的结果?为了提高性能，您可能会对这些算法进行哪些更改?请随意包含任何支持图表或表格。我说的"feel free to"，当然是指"do"。•描述你的优化问题，为什么你觉得它们很有趣，并练习每种方法的优缺点。仔细想想。•分析你的结果。除了回答为什么你得到了你所做的结果，你应该比较和对比不同的算法。他们的挂钟时间有多快?迭代?哪个算法表现最好?你如何定义最好?要有创造力，想尽可能多的问题，以及尽可能多的答案。你知道规矩。注:分析记录总数为 10 页。编码资源就像作业 1 一样，你可以使用任何库，只要它不是专门为这门课写的，特别是自动化你的个人分析。在查看库时，你可能想看看 ABAGAIL:https://github.com/pushkar/ABAGAIL(链接到外部网站)。评分标准在这一点上，当你读到你的分析比其他任何东西都重要时，你不会感到惊讶。我将让你参考作业#1 的这一部分，以获得更详细的解释。另一方面,我也将指出实现这些算法很容易(几乎不值得偷的代码,但请这样做(我将偷代码))但至少其中一个需要一些时间(幸运的是,现在版本的这个算法偷)。你应该现在就开始。