1 Introduction

1.1 Experiments

This project uses three algorithms: simulated annealing, random hill-climbing and genetic algorithms – on different kinds of problems.

The first problem is training neural network weights on the Mushrooms dataset from Kaggle. Instead of using a standard backprop algorithm like gradient descent, this project investigates the use of the three algorithms for adjusting the weights of a neural network.

The other two problems are more of a pure algorithmic flavor: they are four peaks and k-color.

Four peaks is defined as the following (taken from [1]):

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

where

$tail(b, \vec{x})$ = number of trailing b's in $\vec{x}$

$head(b, \vec{x})$ = number of leading b's in $\vec{x}$

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

We want to maximize this number for a given threshold T.

K-color is a problem that attempts to find a coloring for a graph $G(V, E)$ such that the number of edges that have two vertices of the same color is minimized. K-color attempts to minimize the evaluation function (from [3]):

$$\sum_{\{u,v\} \in E} c_{uv}$$

where for each $\{u, v\}$ in all the edges, $c_{uv} = 1$ if u and v are of the same color and zero otherwise.

2 The algorithms

2.1 Simulated annealing

The simulated annealing algorithms used in this project were taken from the library simanneal. The simulated annealing algorithm uses a move() function (to find a local neighbor) and an energy() function (to evaluate the neighbor) and picks a neighbor if it has a higher energy function or otherwise picks a neighbor with some probability proportional to the current temperature, which decreases after every iteration, and also proportional to the deviation from the current energy. The default parameters in this library are: Tmax: 25000.0, Tmin: 2.5, steps: 50000, updates: 100

2.2 Randomized hill-climbing

The hill climbing was manually implemented. It was implemented with outer iterations and inner iterations, such that for each outer iteration, a random location in the space was chosen. Then, in each inner iteration, several local candidate neigbors were found, and the neighbor with the highest evaluation function (if higher than the current evaluation function) was chosen. The best point over all outer iterations was chosen.

2.3 Genetic algorithm

The genetic algorithms used to solve these problems were taken from scipy.optimize.differentialevolution. This algorithm selects everal local neighbors and selects the popsize * len(x) best at each iteration (where x is the parameter to the fitness function). Between iteration, crossover occurs to generate new points.

3 Results and Analysis

3.1 Neural network

The neural network had one hidden layer with five neurons. The Mushrooms dataset is split such that 60 percent is training data, and 10 percent is testing data.

For this problem, hill climbing performed much better than the other two algorithms. The simulated annealing algorithm produced an accuracy of 70 percent on the training set (which is slightly better than guessing), while the hill-climbing algorithm produced an accuracy of 95.7 percent on the training set. The performances of each of the algorithms on the test set were similar (72 percent for simulated annealing and 95 percent for the hill climbing).

A potential cause for the poor performance of simulated annealing could have been that given a point in space, there aren't that many neighbors for which the loss function of the neural network decreases. Unlike random hill climbing and genetic algorithms, simulated annealing only generates one neighbor at a time, leading to a small likelyhood of getting to a better point in space after each iteration.

3.2 K-coloring

This project considered the graph as a complete graph (a graph where any two vertices are connected), as this forces the vertices to have high degrees and in turn forces a large variation in the colors (if the vertices had low degree, not that many colors would need to be changed to solve the k-color problem).

Here are the results of the costs for clique sizes of 30, 40, 50 and a k-value of 20:

		GA	RHC	SA
Clique size	30	46	40	46
	40	22	20	20
	50	84	80	82

We can see from the table that hill climbing performs considerably better than the other two algorithms. There are also a couple of instances where the genetic algorithm produces a higher cost than the simulated annealing algorithm (for example, when the clique size is 40, the genetic algorithm produces a cost of 22, while the simulated annealing algorithm produces a lower cost of 20).

A possible reason for the good performance of simulated annealing may have been the tendency for uphill moves to occur when the temperature is high. Genetic algorithms are more likely to be stuck in local optima in these instances.

3.3 Four peaks

In this algorithm, hill climbing perfomed much better than the other two algorithms.

This algorithm was run on a list of 10 elements, starting with zeros and containing alternating ones and zeros. The threshold in this problem was equal to 4. The hill climbing algorithm successfully produced a sequence of all zeros, giving the function a value of 10. However, the simulated annealing algorithm produced the sequence [1, 0, 1, 1, 1, 0, 0, 0, 1, 0], while the genetic algorithm produced the sequence [1, 0, 1, 1, 0, 1, 0, 0, 1, 0], which did not have very high evaluations.

From the optimization function that the four peaks problem is trying to solve, it makes intuitive sense the hill climbing algorithm would work. This is because the optimization function favors trailing zeros and leading 1's. The hill climbing algorithm generates 20 neighbors per inner iteration, so it is very likely that at least one of the neighbors will have either a larger number of trailing zeros or a larger number of leading ones, leading to the likely selection of a better neighbor.

In each step of simulated annealing, only one neighbor is produced, and it is not very likely that this neighbor will have at least one more trailing zero or leading 1. Thus, more iterations of simulated annealing would be required to produce a similar result to hill climbing.

Another possible reason that the simulated annealing algorithm and genetic algorithm performed badly could have been since there were not enough iterations. Also, the local neighbors may have need to be chosen differently for the simulated annealing (in this assignment, the local neighbors were chosen by selecting a random index in the array and randomly setting it equal to 0 or 1; this worked well for hill climbing but not for simulated annealing).

4 Sources

[1] https://www.cc.gatech.edu/ isbell/tutorials/mimic-tutorial.pdf

[2] https://www.kaggle.com/uciml/mushroom-classification

[3] http://cedric.cnam.fr/ porumbed/papers/EA07.pdf