CSCI 4350 - Open Lab 2

Solving Problems By Local Search

Overview

Develop a software agent in Python to find the maximum value of the Sum of Gaussians (SoG) function

Procedure

1. Create a Python program which uses greedy local search (gradient ascent) to obtain the maximum value of the SoG function, G(), in D dimensions (greedy.py)
    - The program should take 3 command-line arguments [integer: random number seed, integer: number of dimensions (D) for the SoG function, integer: number of Gaussians (N) for the SoG function]
    - The program should start in a random location X in the [0,10] D-cube, where X is a D-dimensional vector
    - The program should use a step size of (0.01*dG(X)/dX) to perform gradient ascent
    - The program should terminate when the value of the function no longer increases (within 1e-8 tolerance) **OR** at a maximum of 100000 iterations
    - The program should print the location (X) and SoG function value (G(X)) at each step (see requirements)
2. Create a Python program which uses simulated annealing (SA) to obtain the maximum value of the SoG function in D dimensions (sa.py)
    - The program should take 3 command-line arguments [integer: random number seed, integer: number of dimensions for the SoG function, integer: number of Gaussians for the SoG function]
    - The program should start in a random location X in the [0,10] D-cube, where X is a D-dimensional vector
    - The program should create an annealing schedule for the termperature (T), and slowly lowering T over time
    - On each iteration, the program should generate a new location Y = X + runif(-0.05,0.05), and choose to accept it or reject it based on the metropolis criterion:

      if G(Y) > G(X) then accept Y; otherwise accept Y with probability e^((G(Y)-G(X))/T)
    - The program should terminate at (a maximum of) 100000 iterations
    - The program should print the location (X) and SoG function value (G(X)) at each step (see requirements)
3. Utilize your programs to analyze the performance of the algorithms
    - Use your greedy program to solve the SoG function for all combinations of D=1,2,3,5 and N=5,10,50,100
    - Use your SA program to solve the SoG function for all combinations of D=1,2,3,5 and N=5,10,50,100
        - Use 100 unique, but corresponding seed values for each case to ensure that you are solving the same problem using both algorithms
    - Calculate the number of times that your SA program **out-performed and/or tied** your greedy program for each condition (within 1e-8 tolerance)
    - Write a report (at least 2 pages, single spaced, 12 point font, 1 inch margins, no more than four pages) describing the SoG function, the code you developled to optimize the function, the annealing schedule you settled on, the performance of the code under various conditions (using the statistics above for justification), any limitations of the overall approach, and describe any additional implementation details that improved the performance of your code
    - Read the submission requirements in the syllabus before submitting your work for grading

Requirements

- You must utilize the numpy.random.ranf() function (import numpy) for generating random numbers.
- You must utilize the SumofGaussians class to set up the problem (download: OLA2-support.zip)
- Use insightful comments in the code to illustrate what is happening on each step
- Include a header in the source code and report with the relevant information for assignments as defined in the syllabus
- Your code should **only** print the current location X, and the value of the SoG function G(X) on each iteration:
    - Example, 1-D output:

```
7.15189366 0.06278163
7.14980459 0.06321930
7.14770360 0.06366198
7.14559057 0.06410975
7.14346539 0.06456269
7.14132794 0.06502088
...
```

    - Example, 2-D output:

```
          5.82914243 6.07541906 0.27940760
          5.82723683 6.08143456 0.28340659
          5.82531476 6.08750207 0.28747493
          5.82337614 6.09362178 0.29161350
          5.82142092 6.09979391 0.29582318
          5.81944905 6.10601862 0.30010479
          ...
```

- Write your report such that a peer NOT taking this course would understand the problem, your approach to solving it, justification of various choices (annealing schedule, gradient step size, etc.), and your final comments.
- Include a table of **all** of the statistics compiled for your report
- Include at least one figure to illustrate the SoG function
- All sources must be properly cited; failure to do so may result in accusations of plagiarism
- Your report should be submitted in PDF format.

Submission

- **Due Date: Thursday, October 7 by 11:00pm**
- Use your PipelineMT credentials to submit your assignment at: https://jupyterhub.cs.mtsu.edu/azuread/services/www/csci4350/assignment-system/public_html/
- A zipped file (.zip) containing:
  - greedy.py
  - sa.py
  - SumofGaussians.py
  - report.pdf