Homework #1

CS 169/268 Optimization
Fall 2023
Due: Monday October 16 11:59pm on Canvas

Reading:
Kochenderfer and Walker, Chapter 3;
Also Belegundu and Chandrupatla, Chapter 2;
Also regarding error bar reporting: https://en.wikipedia.org/wiki/Bessel's_correction .
Grad students add:
Bertsekas on optimality and convergence: Section 1.1 .

Recall the Homework Ground Rules in HW0, point #1. In accordance with those rules, …

Problem 1 (undergrads and grads) 1D unconstrained optimization.

1a. *Write*, or else *obtain, edit, instrument, and fully cite*, a functioning one-dimensional black box unconstrained function optimizer, that is, an optimizer of real-valued functions $f(x)$ of one real-valued argument x using only function calls that evaluate $f(x)$ but no information about derivatives $df/dx, d^2f/dx^2$, etc., and no further constraints on the argument x. Here "functioning" just means it always produces some numerical answer.

For gradeability, please use or edit the top-level optimizer itself (including bracketing) to the template provided at the end of this HW.

1b. In code, *choose* $N = 100$ random real-valued starting points $x = \pm \exp(y) \equiv \pm e^y$, where y is chosen with uniform probability distribution from [-10.0,10.0] and the sign is chosen + or - randomly with equal probability 1/2 each.

1c. *Run* your one-dimensional black box unconstrained function optimizer on the test functions

$f(x) = \frac{1}{2}(x-a)^2 = \frac{1}{2}x^2 - ax + \text{constant}$, and

$g(x) = \frac{1}{4}x^4 - ax$, and

$h(x) = e^x + e^{-x} - ax$, *(to avoid under/overflow, pick y from [-2,2] in this case)*

for $a = 2$, starting from *each* of these starting points, and keep track of a convergence measure such as $|x^{(k+1)} - x^{(k)}|$ or $|f(x^{(k+1)}) - f(x^{(k)})|$, the actual absolute value of error (how can you find that out for these particular problems?), the number of function evaluations, and the wall clock running time for each starting point. *Report in a 3x4 table* the (mean ± estimated population standard deviation) of each of these four quantities, aggregated over the $N = 100$ runs.

Problem 2 (grads only - extra credit for undergrads) Coordinate descent.

(a) *Write* a functioning program that repeatedly calls your code of Problem #1 to implement a multidimensional coordinate descent method (repeatedly optimizing one coordinate direction at a time) for unconstrained multidimensional optimization.

(b) *Test* your code on two different functions $f(x, y)$ (where x and y are real-valued arguments to a real-valued function f). E.g. see K&W Appendix B for some good test functions e.g. Rosenbrock's "banana". Use the same test methods and constraints as in Problem 1 above. Report results as in problem 1 above.

Extra credit on either problem (up to 10% extra credit available on entire HW1):
Plot the results of 1(c) or 2(b) as you vary some important numerical parameter eg. governing stopping criterion, or function parameter a, or any other parameter you held fixed previously that you reasonably expect will affect the results. Plot points must have error bars but please explain whether they are errors on our knowledge of means, or estimated population standard deviations.

Julia template for gradeability, Problem 1:

```
function optimizer1D(func, initial_point, initial_step_size)
        # optimization method used = XXX (fill in the XXX with the right name)
        # source(s) of code used = XXX (fill in the XXX with the right name)
        # ***code goes here***
        return final_point  # also return any intermediate statistics required
end
```

Python template for gradeability, Problem 1:

```
def optimizer1D(func, initial_point, initial_step_size):
# optimization method used = XXX (fill in the XXX with the right name)
# source(s) of code used = XXX (fill in the XXX with the right name)
# ***code goes here***
return final_point  # also return any intermediate statistics required
```