Homework #1

CS 169/268 Optimization
Fall 2018
Due: Tuesday October 9 11:59pm on EEE

Allowed: Reading (but not copying) pseudocode and code in the three recommended/optional books listed first in the class Syllabus. Also, reading external pseudocode after you have tried to write your own code.
Not allowed: Other outside/external code reading or code use (eg. copying or execution).

Turn in: 1-page written description of your approach and results, together with source code and I/O files proving your results.
Programming language: Python. OK to use numpy, matplotlib, and iPython; but not scipy or sympy.
Formats: Writeups in .doc, .docx, or .pdf. Code in .py or .ipynb .

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

(a) *Write* 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.

(b) *Test* your code of part (a) on (at least) two different 1D unconstrained optimization problems *f(x)*, with optimization starting points drawn from a probability distribution which you choose and document. *Report* sampled average values, together with estimated error arising from the probabilistic sampling (using $x \pm y$ notation in any tables, and error bars or box plots in any plots), for the following: (i) Number of function evaluations *f(x)*, (ii) wall-clock running time, and (iii) relative distance between output optimum and true mathematical optimum. *Format* these results in a table, eg one row per problem. At most one of your objective functions *f(x)* can be quadratic; otherwise, you choose them and document your choice. You may choose functions with desirable properties such as continuity and/or convexity. *Report* any numerical parameters you used, eg in stopping criteria.

For 1a above and 2a below, try to stay close to the following template, for TA grading automation. Of course some optimizers may require different arguments after "func".

```
def optimizer1D(func, initial_point, initial_step_size):
# optimization method used = XXX (fill in the XXX with the right name)
# ***student work goes here***
return final_point
```

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*). Same constraints as 1(b) above. Report results as in problem 1(b) above.

Extra credit on either problem (up to 10% extra credit available on entire HW1):
(c) *Plot* the results of 1(b) as you vary some important numerical parameter eg. governing stopping criterion, or any other parameter you held fixed previously that you reasonably expect will affect the results.
(d) *After* all of your other work on HW1 is frozen for submission to EEE, you may *get* source or executable code (despite the "not allowed" list above) for some other 1D black box unconstrained function optimizer (problem 1a) and run it through exactly exactly the same steps as you used in 1(b) or 2(b) above. *Report* average + expected error results of your optimizer vs the external optimizer in tabular format: problems x methods x quantity measured.