Advanced Exercise #1:
Invariance in Optimization

Advanced Optimization Lecture -
CentraleSupélec/ESSEC

Dimo Brockhoff
October 27, 2021

Background:

This exercise is meant to be an addition to the course Advance Optimization for those students who have heard the main lecture content already in earlier optimization courses. This exercise is provided to self-teach some of the concepts, discussed later in class.

Learning Objective:

Understand the concept of invariance for optimization algorithms by running some well-known unconstrained optimization algorithms on simple convex-quadratic functions with varying properties and observing when the behavior of an algorithm changes and when not.

Note: The instructions and questions are kept short, vague, and/or open on purpose. The purpose of this advanced exercise is that you explore and have ideas on your own in terms of both implementation details and explanations of what is going on. This openness of the instructions and questions simulates (as realistically as possible in our context) an actual research project on the topic, for example in the form of a Master's thesis.

1 Preparation/Setting

In the following, we will run the Conjugate Gradient (CG) algorithm, the quasi-Newton BFGS algorithm, and the Covariance-Matrix-Adaptation Evolution-Strategy (CMA-ES) in python. The former two are available via the `scipy.optimize` module, the other one via the `pycma` module[1].

Make yourself familiar with how to run those algorithms on simple functions. `scipy.optimize` has the `minimize` function and the CMA-ES can be run via `cma.fmin2`. For all this exploration, keep in mind that you can access the documentation of any module or function by appending a question-mark. For the test functions, you might want to explore the ones in `cma.fitness_functions`.

[1] To install cma, run `pip install cma`

2 First Experiment on Ellipsoid with Varying Condition Number

Please run the above mentioned three algorithms on the (axis-parallel) ellipsoid function

$$f(x) = \sum_{i=1}^{n} \alpha^{\frac{i-1}{n-1}} x_i^2$$

for different values of α, e.g., $\alpha \in \{1, 10, 100, ..., 10^6\}$ and record and display the number of function evaluations needed to reach a certain precision on the f-value (parameter `tol` for CG and BFGS, parameter `tolfun` for CMA-ES). The number of function evaluations used by an algorithm can be obtained from the algorithm object in all three cases.

What do you expect and what do you actually see for the different algorithms? In particular: why do CG and BFGS behave that differently? Which algorithm is invariant to an increase of the condition number?

3 Rotating the Ellipsoid

One additional invariance is invariance to rotations of the search space, i.e. a rotation-invariant algorithm should not be affected when the (axis-parallel) ellipsoid is rotated (`cma.ff.elli` has a corresponding parameter `rot` that allows to easily rotate the function pseudo-randomly).

Run the same experiment as above for all three algorithms and all condition numbers, but now on the rotated ellipsoid. What do you observe? How is the scaling now, compared to the axis-parallel ellipsoid function? Is the comparison the same for even larger condition numbers than 10^6?

4 Monotonic Transformations of the Objective Function

In addition to scaling and rotation invariance, some algorithms do not even use the objective function values $f(x)$ itself, but only the ranking of solutions with respect to f. CMA-ES is one of those algorithms which are so-called function-value-free or invariant under monotonous transformations of f (and thus optimize f and $g \circ f$ in the same way if g is strictly monotonously increasing).

Change the above experiment such that instead of f, the algorithms optimize $f^{1/4}$ (or f^2). What do you expect to happen? Are the resulting functions for example convex? Quadratic? And what do you actually see in the experiments?