AI505/AI801, Optimization – Exercise Sheet 03

2026-02-22

Exercise 1 *

Implement the extended Rosenbrock function

$$f(x) = \sum_{n/2}^{i=1} [a(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2]$$

where a is a parameter that you can vary (for example, 1 or 100). The minimum is $\boldsymbol{x}^* = [1, 1, \ldots, 1], f(\boldsymbol{x}^*) = 0$. Consider as starting point $[-1, -1, \ldots, -1]$.

Solve the minimization problem with scipy.optimize using all methods seen in class that are suitable for this task. Observe the behavior of the calls for various values of parameters.

Use the COCO test suite (see article) to carry out this exercise. The advantages of the platform is that it provides:

- a set of problem instances to use, about 1000 to 5000 problems (number of functions $\times$ number of dimensions $\times$ number of instances)
- a collection of results from the literature
- tools to launch and analyze the experiments

The COCO framework considers functions divided in suites. Functions, f_i, within suites are distinguished by their identifier $i = 1, 2, \ldots$. They are further parametrized by the (input) dimension, n, and the instance number, j. We can think of j as an index to a continuous parameter vector setting. It parametrizes, among other things, search space translations and rotations. In practice, the integer j identifies a single instantiation of these parameters. We then have:

$$f_i^j \equiv f[n, i, j] : \mathbb{R}^n \to \mathbb{R} \qquad \boldsymbol{x} \mapsto f_i^j(\boldsymbol{x}) = f[n, i, j](\boldsymbol{x}).$$

Varying n or j leads to a variation of the same function i of a given suite. Fixing n and j of function f_i defines an optimization problem instance $(n, i, j) \equiv (f_i, n, j)$ that can be presented to the solver. Each problem receives again an index within the suite, mapping the triple (n, i, j) to a single number.

Varying the instance parameter j represents a natural randomization for experiments in order to:

- generate repetitions on a single function for deterministic solvers, making deterministic and non-deterministic solvers directly comparable (both are benchmarked with the same experimental setup)
- average away irrelevant aspects of the function definition,
- alleviate the problem of overfitting, and
- prevent exploitation of artificial function properties