M1 SCA Université de Lorraine Teacher : Laure Buhry

Mémory and Machine Learning

TP1 – Numerical Optimization and Parameter Estmation

The goal of this computer lab is to implement different deterministic and stochastic optimization methods. You will use *Python* or *iPython* with the SciPy and Numpy library whose documentation is accessible here :
http ://docs.scipy.org/doc/

In order to test and compare the optimization methods, the first part of the work will consist in implementing a benchmark of functions that will try to minimize.

As a first stage, you can plot their 2D or 3D representation using the *Matplotlib Python library*.
Examples of figures and associated scripts are available here :
http ://matplotlib.org/users/screenshots.html
A summary of the *Matplotlib* functions are accessible here :
http ://matplotlib.org/api/pyplot_summary.html

Remark : do ot forget to import *numpy*. You will then need to import :

- scipy.linalg (to manipulate tools of linear algebra type "from scipy import scipy.linalg")
- numpy.random (to maniuplate statistic tools, generate pseudo-random numbers, etc)
- matplotlib.pyplot (graphical representations)

1 Implementation of the testing objective functions

Benchmarck of functions :

$$
\begin{aligned}
f_{00}(x) &= x^4/4 + x^3/3 - 5x^2/2 + 3x - 2,\ -4 \le x \le 3 \\
f_0(x) &= x^3 - 6x^2 + 9x + 1,\ -0.5 \le x \le 4.5 \\
f_1(x_1, x_2, x_3) &= \sum_{i=1}^{3} x_i^2,\ -5.12 \le x_i \le 5.12 \\
f_2(x_1, x_2) &= 100(x_1^2 - x_2)^2 + (1 - x_1)^2,\ -2.048 \le x_i \le 2.048 \\
f_3(x_1, x_2, x_3, x_4, x_5) &= \sum_{i=1}^{5} [x_i],\ -5.12 \le x_i \le 5.12 \\
f_4(x_1, x_2, ..., x_{10}) &= \sum_{i=1}^{10} (i.x_i^4 + \varepsilon_i(x_1, x_2, ..., x_{10}))\ -5.12 \le x_i \le 5.12 \\
& \quad \text{où } \varepsilon_i \text{ is a uniform random variable in}[0\,;1] \\
& \quad \text{whose value is different for each value of } x_i \\
f_5(x_1, x_2) &= \frac{1}{0.002 + \sum_{j=1}^{2} 51/(c_j + \sum_{i=1}^{2}(x_i - a_{ij})^6)} \\
& \quad -65.536 \le x_i \le 65.536
\end{aligned}
$$

where $[x]$ is the floor of x (numpy.floor(x) avec numpy) and where $\gamma = 500$, $c_j = 1 + j$, $a_{1,j} = -16(mod(16, 5) - 2)$ et $a_{2,j} = -16([i/5] - 2)$, mod(x, y) means x modulo y (numpy.mod(x,y)).
One can get a uniform random number with numpy.random (get more info with : numpy.random ?).
Remark : f_1 and f_2 have a global minimum, but f_3 , f_4 et f_5 have several local minima.

a) The function are coded in Python (the files are provided on Arche). Plot the graph of f_0 and f_{00}.
b) Plot the 3D-graphs of the remaining functions. For the functions with n parameters, where n can be larer than 2, only consider que 2 parameters (for instance, for f_3 consider $f_3(X) = \sum_{i=1}^{2}[x_i]$, $-5.12 \le x_i \le 5.12$) and represent the 3D surface where the Z-axis is $f_i(x_1, x_2)$.

2 Implementation of the optimization methods

2.1 Deterministic methods

One only considers here the functions f_0 and f_{00}.

a) Impement the gradient descent algorithm with $\forall k$, $\alpha_k = 0,001$ and test it on f_0 and f_{00} with different initializations : $x(0) = 0,8$ and $x(0) = 1,5$ for f_0 and $x(0) = -2$, $x(0) = 0$ and $x(0) = 1$ for f_{00}. Plot the evolution the evolution of the evaluation of the objective function at each iteration, then display the optimal X found (the one minimizing the objective functions). What do you observe ?
Take a larger value for α_k, for instance, $\alpha_k = 0,5$. what do you observe ?

b) Implement the second order method (Newton) and test it on f_0 and f_{00}. Plot the evolution of the evaluation of the objective function at each iteration, then display the optimal X found (the one minimizing the objective functions).

2.2 Stochastic methods

Same questions with f_0 to f_5 and :

a) the simulated annealing with 150 iterations, $h(T) = 0,9T$, $T_{init} = 500$, $T_{fin} = 0,5$.

b) the differential evolution with a population of 100 individuals and 150 iterations.

c) the particle swarm otimization algorithm with 10 particles and 150 iterations, $\omega = 0.7$, $\phi_1 = 1,5$, $\phi_2 = 1,5$, then with 100 particles and 150 iterations, $\omega = 0.7$, $\phi_1 = 1,5$, $\phi_2 = 1,5$.

d) evolutionary stategy with a population of 100 individuals and 150 iterations, a crossover constant of 0,6 and a mutation constant of 0,1.

e) What could you conclude about the performance of the algorithms with respect to each function of the benchmark ?