Advanced Optimization

Master AIC - Paris Saclay University

Exercices - Stochastic Continuous Optimization

Anne Auger and Dimo Brockhoff
anne.auger@inria.fr

I Pure Random Search (PRS)

We consider the following optimization algorithm.

```
[Objective: minimize $f : [-1,1]^n \to \mathbb{R}$
$X_t$ is the estimate of the optimum at iteration $t$
Input $(U_t)_{t \geq 1}$ independent identically distributed each $U_t \sim \mathcal{U}_{[-1,1]^n}$ (unif. distributed in $[-1,1]^n$) ]
1. Initialize $t = 1$, $X_1 = U_1$
2. while not terminate
3.         $t = t + 1$
4.         If $f(U_t) \leq f(X_{t-1})$
5.                 $X_t = U_t$
6.         Else
8.                 $X_t = X_{t-1}$
```

1. Show that for all $t \geq 1$
$$f(X_t) = \min\{f(U_1), \ldots, f(U_t)\}$$

2. We consider the simple case where $f(x) = \|x\|_\infty$ (we remind that $\|x\|_\infty := \max(|x_1|, \ldots, |x_n|)$). Show the convergence in probability of the PRS algorithm towards the optimum of f, that is prove that for all $\epsilon > 0$
$$\lim_{t \to \infty} \Pr\left(\|X_t\|_\infty \geq \epsilon\right) = 0$$
Hint: Prove and use the equality
$$\{\|X_t\|_\infty \geq \epsilon\} = \cap_{k=1}^{t} \{\|U_k\|_\infty \geq \epsilon\}$$

3. Let $T_\epsilon = \inf\{t | X_t \in [-\epsilon, \epsilon]^n\}$ (with $\epsilon > 0$) be the first hitting time of $[-\epsilon, \epsilon]^n$.
Show that T_ϵ follows a geometric distribution with a parameter p that we will determine.
Deduce the expected value of T_ϵ, that is the expected hitting time of the PRS algorithm.

4. When we implement an optimization algorithm (without derivatives), the cost of the algorithm is the number of calls to the objective function. Write a pseudo-code of the PRS algorithm where at each iteration the objective function f is called only once.

II Adaptive step-size algorithms

We are going to test the convergence of several algorithms on some test functions, in particular on the so-called sphere function

$$f_{\text{sphere}}(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{x}_i^2$$

and the ellipsoid function

$$f_{\text{elli}}(\mathbf{x}) = \sum_{i=1}^{n} (100^{\frac{i-1}{n-1}} \mathbf{x}_i)^2 \ .$$

1. What is the condition number associated to the Hessian matrix of the functions above? Are the functions ill-conditioned?

2. Use Matlab or Python to implement the functions. We can create two functions `fsphere.m` and `felli.m` that take as input a vector $\mathbf{x}$ and returns $f(\mathbf{x})$.

The $(1+1)$-ES algorithm is on of the simplest stochastic search method for numerical optimization. We will start by implementing a $(1+1)$-ES with constant step-size. The pseudo-code of the algorithm is given by

```
Initialize $\boldsymbol{x} \in \mathbb{R}^n$ and $\sigma > 0$
while not terminate
    $\mathbf{x}' = \mathbf{x} + \sigma \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
    if $f(\mathbf{x}') \leq f(\mathbf{x})$
        $\mathbf{x} = \mathbf{x}'$
```

where $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ denotes a Gaussian vector with mean $\mathbf{0}$ and covariance matrix equal to the identity.

1. Implement the algorithm. You can write a function that takes as input an initial vector $\mathbf{x}$, an initial step-size σ and a maximum number of function evaluations and returns a vector where you have recorded at each iteration the best objective function value.

2. Use the algorithm to minimize the sphere function in dimension $n = 5$. We will take as initial search point $\mathbf{x}^0 = (1, \ldots, 1)$ [`x=ones(1,5)`] and initial step-size $\sigma = 10^{-3}$ [`sigma=1e-3`] and stopping criterion a maximum number of function evaluations equal to 2×10^4.

3. Plot the evolution of the function value of the best solution versus the number of iterations (or function evaluations). We will use a log scale for the y-axis (`semilogy`).

4. Explain the three phases observed on the figure.

   To accelerate the convergence, we will implement a step-size adaptive algorithm, i.e. σ is not fixed once for all. The method to adapt the step-size is called one-fifth success rule. The pseudo-code of the $(1+1)$-ES with one-fifth success rule is given by:

```
Initialize $\boldsymbol{x} \in \mathbb{R}^n$ and $\sigma > 0$
while not terminate
    $\boldsymbol{x'} = \boldsymbol{x} + \sigma \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
    if $f(\boldsymbol{x'}) \leq f(\boldsymbol{x})$
        $\boldsymbol{x} = \boldsymbol{x'}$
        $\sigma = 1.5\, \sigma$
    else
        $\sigma = (1.5)^{-1/4} \sigma$
```

5. Implement the (1+1)-ES with one-fifth success rule and test the algorithm on the sphere function $f_{\text{sphere}}(x)$ in dimension 5 ($n = 5$) using $\mathbf{x}^0 = (1, \dots, 1)$, $\sigma_0 = 10^{-3}$ and as stopping criterion a maximum number of function evaluations equal to 6×10^2. Plot the evolution of the square root of the best function value at each iteration versus the number of iterations. Use a logarithmic scale for the y-axis. Compare to the plot obtained on Question 3. Plot also on the same graph the evolution of the step-size.

6. Use the algorithm to minimize the function f_{elli} in dimension $n = 5$. Plot the evolution of the objective function value of the best solution versus the number of iterations. Why is the $(1+1)$-ES with one-fifth success much slower on f_{elli} than on f_{sphere} ?

7. Same question with the function

$$f_{\text{Rosenbrock}}(x) = \sum_{i=1}^{n-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \ .$$

8. We now consider the functions, $g(f_{\text{sphere}})$ and $g(f_{\text{elli}})$ where $g : \mathbb{R} \to \mathbb{R}, y \mapsto y^{1/4}$. Modify your implementation in Questions 5 and 6 so as to save at each iteration the distance between $\mathbf{x}$ and the optimum. Plot the evolution of the distance to the optimum versus the number of function evaluations on the functions f_{sphere} and $g(f_{\text{sphere}})$ as well as on the functions f_{elli} and $g(f_{\text{elli}})$. What do you observe? Explain.

III Running and Understanding CMA-ES

Download the MATLAB code (or Python code) of the CMA-ES algorithm (cmaes.m) on the webpage of Nikolaus Hansen (the main author of the algorithm):

`http://cma.gforge.inria.fr/cmaes_sourcecode_page.html`

1. Run the algorithm in dimension 10 to minimize the following functions

   - $f_{\text{elli}}(\mathbf{x}) = \sum_{i=1}^{n} ((10^3)^{\frac{i-1}{n-1}} \mathbf{x}_i)^2$
   - $f_{\text{tablet}}(\mathbf{x}) = 10^6 \mathbf{x}_1^2 + \sum_{i=2}^{n} \mathbf{x}_i^2$

   Use the option `LogPlot='on'` that shows the typical graphical output of the algorithm that displays in particular the evolution of the mean vector, step-size and covariance matrix adapted in the CMA-ES algorithm.

2. Explain the different plots that appear on the screen.

3. Identify and explain the two main (convergence) phases observed.

4. What is the relationship between the eigenvalues of the covariance matrix in the end and the eigenvalues of the Hessian matrix of the functions?

5. Connect the asymptotic convergence rate on the convex quadratic function that corresponds to the slope of the last part of the convergence graph with the convergence rate on the sphere function. Explain.

6. The function $f_{\text{ellirot}}(\mathbf{x})$ is defined by $f_{\text{ellirot}}(\mathbf{x}) = f_{\text{elli}}(P\mathbf{x})$ where P is a rotation matrix (sampled uniformly among the rotation matrices). Run the CMA-ES algorithm on f_{ellirot} et f_{elli} (`cmaes('felli',x0,sigma0)` et `cmaes('fellirot',x0,sigma0)`). Understand and explain the differences observed in the graphical output.

7. Compare now the convergence rate of CMA-ES and of the $(1+1)$-ES with one-fifth success rule on $f_{\text{elli}-2}(\mathbf{x}) = \sum_{i=1}^{n} ((10^2)^{\frac{i-1}{n-1}} \mathbf{x}_i)^2$ for $n = 10$. For this you can for instance report the number of function evaluations that both algorithms need to reach 10^{-6} for 6 different runs. Explain the differences observed.

8. We consider now the function

$$f_{\text{rastrigin}}(\mathbf{x}) = 10n + \sum_{i=1}^{n}(\mathbf{x}_i^2 - 10\cos(2\pi\mathbf{x}_i))$$

Show that the function is multimodal (We remind that a function to be minimized is multimodal if it has more than one local optimum).

9. The default population size (the parameter λ) in CMA-ES equals $4 + \lfloor(3\log(n))\rfloor$. Run 5 times the CMA-ES algorithm with its default population size to minimize the Rastrigin function in dimension 10 starting with a point sampled according to `rand(10,1)` and with initial step-size equal to 10: (`cmaes('frastrigin',rand(10,1),10)`). Mesure the success probability. Realize the same experiment by multiplying the default population size by $2, 4, 8, 16, 32$. What do you observe? Explain.