Evolutionary Computation, 2019/2020
Programming assignment 3

Important information

- Deadline: 10/Nov/2019, 23:59.
- You should submit through the 'tutoria' a ZIP file containing a PDF file with answers to the questions as well as all the source code that you developed. The ZIP file should also include a 'readme.txt' file explaining how to compile and run the program.
- The name of the zip file must be the letter 'a' followed by your student number, followed by '-P3.zip'. Example: if your student number if 12345, the file name must be `a12345-P3.zip`
- You must discuss your work with the instructor at the lab class on 13/Nov/2019.

About this assignment

The purpose of this programming assignment is to have you program simple evolution strategies. You should report the results obtained on a PDF document, and submit the source code of your experiments.

Generating pseudo-random numbers from the Standard Normal Distribution

Most programming languages provide library functions for generating pseudo-random numbers from various distributions. To implement *Evolution Strategies* you need to generate pseudo-random numbers from the *Standard Normal Distribution*, from now on referred as N(0,1).

If your chosen language has such a function please feel free to use it. Othersiwe, you can write your own function. One way to do that is to implement the *Marsaglia polar method* (see http://en.wikipedia.org/wiki/Marsaglia_polar_method).

Problem A: (1+1)-ES

Implement a (1+1)-ES for real-valued vectors. When doing mutation, each variable is modified by adding the outcome of a sample from N(0,1). Test your algorithm on the so-called *Sphere Model*, an easy unimodal test function defined as follows for n-dimensional vectors.

$$f(\bar{x}) = \sum_{i=0}^{n-1} x_i^2$$

with each x_i initially generated uniformly at random in [-100.0,+100.0]. The optimum value of this function is $x_i = 0.0$ for all i. Test it for different values of n, say $n = 10$ and $n = 100$. How close can you get to the optimum? And how fast?

Problem B: (1+1)-ES with the 1/5 rule

Do the same as problem B but implement the 1/5 rule for adapting the mutation strength. Start with an initial $\sigma = 1$. What differences do you observe comparing with problem A?

Problem C: (μ+λ)-ES and (μ,λ)-ES with one step size (one sigma)

Implement the (μ+λ)-ES and (μ,λ)-ES with one uncorrelated mutation. For adapting the mutation rate, use the recommendation suggested in class (by Hans-Paul Schwefel) and set $\tau = 1/\sqrt{n}$. Test the algorithms in the Sphere Model as well as on Ackley's function defined as follows:

$$f(\bar{x}) = -20 \cdot \exp\left(-0.2 \cdot \sqrt{\frac{1}{n} \cdot \sum_{i=0}^{n-1} x_i^2}\right) - \exp\left(\frac{1}{n} \cdot \sum_{i=0}^{n-1} \cos(2\pi x_i)\right) + 20 + \exp(1)$$

with each x_i initially generated uniformly at random in [-30.0,+30.0]. The optimum value of this function is also $x_i = 0.0$ for all i. Test it for different values of n, say $n = 10$ and $n = 100$, and for different values of μ and λ.

Problem D: (μ+λ)-ES and (μ,λ)-ES with n step size (n sigmas)

Same as problem C but this time your ES has n sigmas, one for each variable. Set the global learning rate to $1/\sqrt{2 \cdot n}$, and the local learning rate to $1/\sqrt{2 \cdot \sqrt{n}}$