Bio-Inspired Artificial Intelligence

Prof. Giovanni Iacca
giovanni.iacca@unitn.it

Week 1 Lab Exercises

Getting up and running with Linux and Python

Linux. The labs for this course will be executed (and have been tested) on Linux machines. In case you have not used a computer running Linux before, don't fret. You should be able to log in with your UNITN username and password. Once logged in you will see a graphical interface that is not so different from what you would see on a computer running Windows or Mac OS. Take a few minutes to explore the operating system. At the bottom of your window you should see several icons. The important ones are:

- The File Manager, where you can view, modify, open, create, and delete files.
- The Terminal, where you can gain access to the command line interface that you will be using for many of the exercises. If you've never used a Linux/Unix command line interface before, have a read through:

  `https://tinyurl.com/commandlineforbeginners`

  and/or search online for other intros to the terminal.
- The Web Browser, just like you have on Mac/Windows (e.g. Firefox, Chrome, Safari, etc.). You will use the web browser to download materials.

Python. Most of the lab exercises we will see in this course are implemented in the Python programming language. In particular, many of the labs in this course employ the *inspyred*[1] and *deap*[2] frameworks for Python. Additional packages will be used in some specific labs. You will not have to do much programming, but a basic understanding of Python is required to complete the exercises. If you have never used Python before, you should look through:

`http://hetland.org/writing/instant-python.html`

or follow the tutorials listed here:

`http://wiki.python.org/moin/BeginnersGuide/Programmers`

Additionally, if you are a MATLAB user, you may want to look at: `https://docs.scipy.org/doc/numpy/user/numpy-for-matlab-users.html`

[1] `https://pypi.org/project/inspyred/`
[2] `https://pypi.org/project/deap/`

Why Python? There are many reasons that we choose to use Python for this course. For starters, unlike MATLAB, it is free and open. Hoyt Koepke does a good job of summarizing many other reasons here:

`http://www.stat.washington.edu/~hoytak/blog/whypython.html`

Installation. If you want to run Python on your own computer then by far the easiest method to get up and running is to install Anaconda `https://www.anaconda.com/download` (you will want to choose the correct version of **Python 3.x**[3] for your CPU + Operating System) Then, for the class exercises you will need to install *inspired* and *deap*. At a terminal, run[4]:

```
$>pip install inspyred --user
$>pip install deap --user
```

Alternatively, you can download the source files and install them manually:

`https://pypi.python.org/packages/source/i/inspyred/inspyred-1.0.1.tar.gz`

`https://pypi.python.org/packages/source/d/deap/deap-1.2.2.tar.gz`

extract these archives, and then in each of the directories where you extracted them run:

```
$>python setup.py install --user
```

[3]Be sure to download Python 3.x and not 2.x, since this is the version for which the software used for lab exercises has been tested with.

[4]Please note that in this document `$>` represents your command prompt, do not re-type these symbols!

Introduction

Goal. The goal of this lab is to perform some preliminary experiments aimed at understanding some advantages and pitfalls of Evolutionary Algorithms (EAs). In particular, you will observe the effects of mutations and problem dimensionality, and reflect to what extent these observations also apply to biological evolution.

Getting started. Download the file `01.Exercises.zip` from Moodle and unzip it. The unzipped folder contains a sub-folder named `src`. Depending on where you downloaded the zip file (typically, your home folder or the `Downloads` folder), you should be able to use the terminal to move to the `src` folder (e.g., by typing `cd ~/Downloads/01.Exercises/src`).

Each exercise has a corresponding `.py` file. To solve the exercises, you will have to open, edit, and run these `.py` files.

Note that, unless otherwise specified, in this week's exercises we will use real-valued genotypes. I.e., an individual is a vector of real-valued parameters $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$ (however, keep in mind that other types of individual representations are possible, such as trees or bit strings, which will not be explored in this lab). The fitness of an individual is given by the fitness function $f(\mathbf{x})$. The aim of the algorithms will be to *minimize* the fitness function $f(\mathbf{x})$, i.e. to find the vector $\mathbf{x}_{min}$ that has the lowest value $f(\mathbf{x})$. In other words, lower values $f(\mathbf{x})$ correspond to a better fitness!

Exercise 1

In this first exercise, we will not yet run a complete EA. Instead, we consider a single parent individual $\mathbf{x}_0$, from which a number of offspring individuals are created using a Gaussian mutation operator (which adds a random number from a Gaussian distribution with mean zero and standard deviation σ to each parameter x_i of the parent). The fitness function, shown in Figure 1 for $N = 2$ variables, is defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{N} x_i^2$$

This function is usually defined to as "sphere" function[5] and is one of the most used benchmark functions in continuous (real-valued) optimization. This fitness function is unimodal, i.e. it has a single global minimum at the origin. Furthermore, this function is "scalable", i.e. it can be defined for any arbitrary number of variables ($N = 1, 2, 3, ...$, i.e. $N \in \mathbb{N}$). We will analyze the effects of mutations on the fitness depending on the value of the parent $\mathbf{x}_0$, the mutation magnitude[6] (the standard deviation σ), and the number of dimensions N of the search space.

[5] Note that its contour lines, i.e. the loci of points for which the function has a constant value, are N-dimensional "spheres" centered in $\mathbf{0}$. For instance, in 2-D, the contour lines are curves described by $x_1^2 + x_2^2 = k$, which correspond to a circle (the equivalent of a sphere in 2 dimension) with radius $\sqrt{k}$ and center in $\{0, 0\}$. In 3-D, the contour lines are curves described by $x_1^2 + x_2^2 + x_3^2 = k$, which correspond to a sphere with radius $\sqrt{k}$ and center in $\{0, 0, 0\}$. For $N > 3$, each contour line corresponds to a "hyper-sphere", i.e. a generalization of a sphere.

[6] In the following, "mutation magnitude" indicates a generic measure of the mutation effect on the genotype. E.g. in continuous optimization with Gaussian mutation $x' = x + \mathcal{N}(0, \sigma)$ the "mutation magnitude" is simply σ. This is different from the "mutation probability", that is the chance that a given loci would be mutated. The combination of these two aspects, magnitude and probability, may be considered the overall "mutation rate".

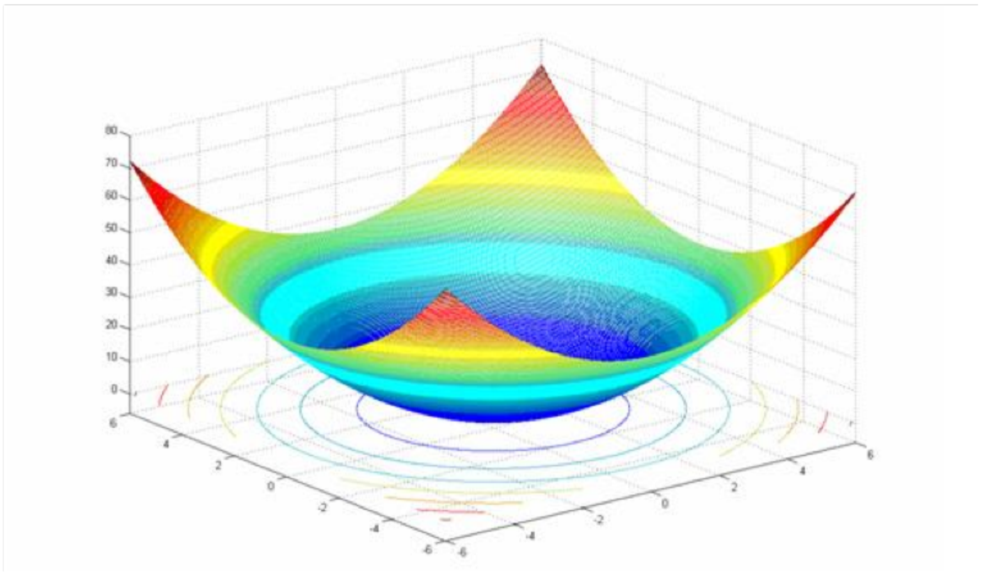

Figure 1: A graphical representation of the sphere function for $N = 2$ variables

To start the experiments, from a command prompt run[7]:

```
$>python exercise_1.py
```

to generate offspring from a single parent $\mathbf{x}_0$ using a Gaussian mutation operator (which adds a random number from a Gaussian distribution with mean zero and standard deviation σ to the parent). Generate offspring from different parents (e.g. $\mathbf{x}_0$=0.1, 1, 10) using different mutation magnitudes (standard deviations σ). First consider the one-dimensional case, then two dimensions, and finally many dimensions (e.g. $N = 100$). For one or two dimensions, the fitness landscape with the parent and the offspring is shown. For more dimensions, a boxplot[8] with the fitness of the offspring is shown where the green, dashed line is the fitness of the parent.

Try to answer the following questions:

- Do the mutations tend to improve or worsen the fitness of the parent?
- Are low or high mutation magnitudes best for improving the fitness? How does this depend on the initial value of the parent and on the number of dimensions of the search space?

Exercise 2

In this exercise we will try to confirm the observations that we did qualitatively in the previous exercise, by plotting boxplots side-by-side to evaluate the statistical significance of observed differences.

[7]For all exercises in this lab you may follow the name of the `.py` file with an integer value, which will serve as the seed for the pseudo-random number generator. This will allow you to reproduce your results. Also, please note that in this document `$>` represents your command prompt, do not re-type these symbols.

[8]Visit `http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.boxplot` if you are unfamiliar with boxplots

Run the script `exercise_2.py`, and compare different values for:

- the number of dimensions of the search space;
- the value of the parent (how close it is to the optimum);
- the mutation magnitude σ;

and try to confirm the answers that you gave in the previous exercise. See the comments in the script for more details.

NOTE: If you vary one of these three parameters, *make sure that you set the other two at a constant value* (otherwise it may be difficult to interpret your results).

Exercise 3

We will now use an EA to find the minimum of the unimodal fitness function defined in the previous exercise and analyze the effect of the mutation magnitude and the dimensionality of the search space on the results.

Run `exercise_3.py` to run a basic, mutation-only EA on the 1-D sphere function first.

- How close is the best individual from the global optimum?

Increase the dimensionality of the search space to two and more.

- How close are the best individuals now from the global optimum?
- Can you get as close as in the one-dimensional case by modifying the mutation magnitude and/or the number of generations?

Exercise 4

In this exercise we will try to confirm the observations that we did qualitatively in the previous exercise, by plotting boxplots side-by-side to evaluate the statistical significance of observed differences.

Run the script `exercise_4.py` to do three batches of 30 runs of the EA with different mutation magnitudes (it may take a minute). The boxplot compares the best fitness values obtained (at the end of each run) in the three conditions.

- Did you see any difference in the best fitness obtained? Try to explain the result.

Instructions and questions

Concisely note down your observations from the previous exercises (follow the bullet points) and think about the following questions.

- What is the genotype and what is the phenotype in the problems considered in this lab?
- What are the advantages and disadvantages of low/high mutation magnitudes in EAs?

- Based on the previous observations, do you think there is an optimal mutation magnitude for a biological organism? Do mutations typically improve or worsen the fitness of a biological organism? In which situations do you think low/high mutation rates are advantageous for a population of bacteria?