Cours scientifique - IN104: Projet informatique

Natalia Díaz Rodríguez, Vladimir Paun

10 avril 2018

1 Summary of the project : Genetic algorithms

This is one of the projects proposed for IN104 course Projet Informatique [1].

Genetic algorithms take inspiration from two concepts of the theory of evolution : the definition of a selection process and the use of random mutations to obtain a new set of solutions from solutions previously envisaged. The project aims to implement a genetic algorithm and use it to determine the largest subset of a set of relative integers such that the sum of the elements of this subset is 0.

The project will be implemented in Python 3. Support slides on GA are in the website of the project [2].

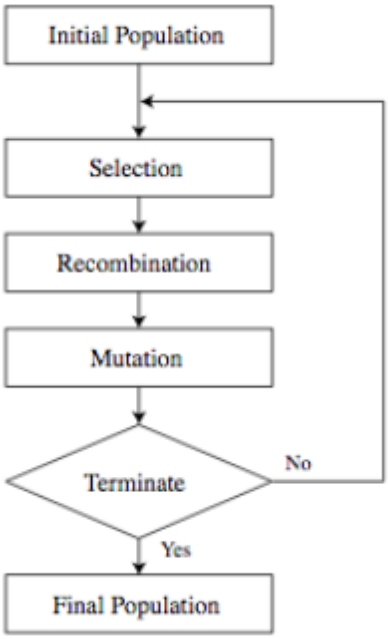

FIGURE 1 – Example of genetic algorithms population evolution cycle

2 Genetic Algorithms

Because it can be challenging to search using methods like exhaustive search or exact optimization (e.g., linear programing) for finding the optimal solution in a high dimensional space, we draw inspiration

1. http://perso.ensta-paristech.fr/~paun/ENSTA/IN104/introduction.pdf https://synapses.ensta-paristech.fr/catalogue/ue/16/in104-projet-informatique
2. https://perso.ensta-paristech.fr/~tbernard/Ens/IN104ProgDet.html

from the field of genetic algorithms, which has been shown to achieve efficient search across a variety of similar problems. The proposed design system of GAs can be shaped to mimic the processes that underlie these stochastic optimization methods [2].

Genetic algorithms are a collection of methods inspired by genetics and natural selection. Within the metaphor, each candidate solution is conceptualized as having a chromosome composed of genes, where each gene's value is an allele.

— Chromosome : the string codification of a candidate solution (the code for an individual from the population)

— An allele : a feature (component, in our case an integer) that can be included as part of a chromosome string. In Biology, an allele is a variant form of a gene. Some genes have a variety of different forms, which are located at the same position, or genetic locus, on a chromosome. Humans are called diploid organisms because they have two alleles at each genetic locus, with one allele inherited from each parent.

2.1 Genetic Algorithm Operators

Crossover and mutation are two basic operators of GA. Performance of GA very often depends on them. Type and implementation of operators depends on encoding and also on the problem. There are many ways how to do crossover and mutation [1] :

— Selection :
A fitness function filters (*selects*) a set of most fit individuals according to a score function of each individual

— Crossover : e.g., in *single point crossover*, one crossover point is selected ; the binary string from the beginning of the chromosome to the crossover point is copied from one parent, and the rest is copied from the second parent. Example :
11001011+11011**111 = 11001111**

— Mutation : An operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. For instance, *Bit string inversion* selected bits are flip (inverted) at random positions. E.g.
Example 1 :
1**1**001001⇒ 1**0**001001
Example 2 :

1	0	1	0	0	1	0
				↓		
1	0	1	0	1	1	0

The probability of a mutation of a bit is 1/l where l is the length of the binary vector.

— Uniform : This operator replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes.

2.2 An industry case study in fashion design at Stitch Fix Inc.

To illustrate how the GA operators work in a real problem, we consider the case in designing clothing from Stitch Fix [2]. In a fashion design use case, a blouse's chromosome is a vector of attributes corresponding to the dimensions of the search space [2]. Generating a new design can then be framed as an evolutionary process searching over a population of possible blouses, through a series of generations. Each generation passes through three stages :

— Selection : Each individual of the current generation is evaluated for its fitness. This implies access to an explicit objective function mapping attributes to outcomes or empirical measurements of each individual in the current generation. For instance, blouses might be selected to maximize style and

fit feedback. These individuals from the current generation can then enter a mating pool using not-so-natural selection procedures such as selecting the N fittest individuals, or selecting individuals with a probability proportional to their fitness (e.g., biased roulette-wheel selection).

— Recombination (Crossover) : Individuals in the mating pool are bred to produce new individuals that are hopefully desirable and novel because they were created by decomposing two good parents and using the resulting elements to compose a child. There are a variety of procedures for implementing recombination. They essentially boil down to using a random mix of features from each parent. An example of the latter in our system might be passing sleeve type, sleeve length, and sleeve fabric as a unit. We can then propose which genes to select from each parent using this broader knowledge.

— Mutation : Diversity and novelty are introduced into new generations by randomly changing alleles. These mutations search the vicinity of the solution space via a random walk. This vicinity can be more or less local depending on the type of mutations that are employed. For instance, the values of different alleles can be assigned proportional to the observed distribution of alleles or forced into their extremes to achieve a more aggressive search (e.g., boundary mutation). Finding a compromise between these is practical to retrieve underrepresented regions of the feature space.

3 Evaluation

The components of the final grade will be aggregated to provide a final mark according to the ENSTA scale.

— 25% **Source code** : features, tests, documentation, etc. Source code must be documented, the report including a link to the repository must be in a .zip file. The repository will contain the sources, as well as a plain text file README.md which indicates the actual operational features and limitations. Python code should compile, the teaching assistants are not supposed to make significant corrections for it to compile : a code with few features, but that compiles and does not crash will be preferred to a more complete code but which is not directly operational !

— 25% **Defense** (10 minutes presentation, 10 minutes questions) : The formal quality of the presentation will be an important element. The defense needs to include a demonstration on the basis of the source code, an analysis of the difficulties encountered and implemented solutions. It will not include a presentation of the problem or the method of resolution that the teaching assistant obviously knows already well. The defense is open to everyone (subject to the acceptance of the pair that will present), the chronological order of defense will be given by the list of each group.

— 20% **Practice Analysis** -Max 5 pages : You will return a critical report before the defense in which you analyze and criticize the progress of your project and its success and failure factors. This evaluation component also includes the oral treatment of this question during the defense.

— 30% **Progress** of practical work during the practical lab sessions (based on git commits).

For this particular project we will evaluate the implementation of key methods such as *get_fitness(), select(), mutate(), crossover()*, plots based on the population size and the demo of the code running and changing diverse parameters.

3.1 Reporting and deadlines

Note : In the French version of the AlgoGen_expected_format.pdf indicates to provide a visualization. This is not needed ; instead, you need to provide a regular static plot (e.g., using matplotlib) of the evolution of the population. Optionally, it will be positively valued if you get inspired from "The Era of Deep Neuroevolution" blog post in order to apply some new concepts to your project solution.

A zip file containing the pdf report should be sent to Natalia Díaz Rodríguez (natalia.diaz (at) ensta (dot) fr). The report must include a link to the github (preferred) or gitlab repository with the code and must be called surname-first-team-member_surname-second-team-member(.zip and .pdf, respectively). Deadlines are in the course website [3].

3. http://perso.ensta-paristech.fr/~tbernard/Ens/IN104ProgDet.html

4 Getting started

Test driven development is strongly recommended as a development cycle. Some of the files included in the project repository [4] :

— Presentation of the project : IN104_slides_PAUN.pdf
— Detailed subject : IN104_AlgoGen_sujet_PAUN.pdf
— Expected deliverable format : AlgoGen_expected_format.pdf
— Given test input data files : inputSet.zip
— Code for input set generation : genEnsDep.py.zip
— Both gitlab.ensta.fr and github.com can be used. However, the latter is recommended (in which case, the project created needs to strictly be set as private). If you can't create a private repo, follow instructions in Appendix.

5 Useful links and references

1. www.obitko.com/tutorials/genetic-algorithms/crossover-mutation.php
2. Genetic algorithms in Fashion Personalization at Stitch Fix : http://multithreaded.stitchfix.com/blog/2016/07/14/data-driven-fashion-design/
3. https://en.wikipedia.org/wiki/Crossover_(genetic_algorithm)
4. Test Driven Development and Git versioning : http://perso.ensta-paristech.fr/~paun/ENSTA/MO101/
5. GIT :
   — Antonin Raffin tutorials - Intro to Git : http://slides.com/antoninraffin/git and Git intermediate : http://slides.com/antoninraffin/git-intermediate
   — http://users.humboldt.edu/smtuttle/s12cis492/492guide-to-git.pdf
   — https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf
   — https://github.com/git-tips/tips#everyday-git-in-twenty-commands-or-so
   — https://tutorialzine.com/2017/11/10-useful-git-tips
6. Install Python libraries and Master Python : http://musicinformationretrieval.com/python_basics.html
7. Python Numpy http://cs231n.github.io/python-numpy-tutorial/ and IPython tutorials http://cs231n.github.io/ipython-tutorial/
8. Iterate fast installing Jupyter notebooks http://jupyter.org/install and get good at IPython : http://musicinformationretrieval.com/get_good_at_ipython.html
9. The quartet of NumPy, SciPy, Matplotlib, and IPython is a popular combination in the Python world. Numpy Basics : http://musicinformationretrieval.com/numpy_basics.html, Numpy Tutorial : http://scipy.github.io/old-wiki/pages/Tentative_NumPy_Tutorial
10. Intro to genetic algorithms with an industry case (data-driven fashion design) :https://multithreaded.stitchfix.com/blog/2016/07/14/data-driven-fashion-design/
11. Nice machine learning tutorials and notebooks : Image processing, k-means, matplotlib and numpy (French) : http://wwwdi.supelec.fr/liesse/html/tp_k_means/tp_k_means.html
12. Useful for permutations and combinations with replacement : itertools : https://docs.python.org/2/library/itertools.html
13. On the length limits of a Python string : http://stackoverflow.com/questions/1739913/what-is-the-max-length-of-a-python-string

4. https://github.com/NataliaDiaz/IN_104-Projet-Informatique

6 Appendix

6.1 Recap from pre-requirement lecture MO101 : Intro to Test Driven Development (TDD) and Git versioning control

This TD on TDD+GIT must follow the instructions [5]. It will be evaluated on the work done during the class (the game on add, commit, push and pull) and also, the work done until next following 7 days- i.e., until Tuesday 3rd April 23.59h. This is in order to ensure you have had time to understand the concepts behind and are in shape to put it in practice during the IN104 projet informatique.

Remember to add your ssh-key to your gitlab/github account (each of the team members) using [1]. There are some subtleties found to work easier in Github versus Gitlab ; therefore, github is recommended :

1. Create a new account as in [1] and a repository
2. Add your ssh-key as in [2].
3. Provide master access to your team member and me (username 'diaz') via the *Team* option settings of the project.

For those using gitlab, access denied problems should be solved by giving your partner (and your teaching assistant), *master* access instead of *developer* access.

An editor easier to use than *vi* is *Atom* (open source, recommended) [3] or *sublime* [4].

6.1.1 Test Driven Development

unittest has some internal magic that :
— discovers all classes inheriting from *unittest.TestCase*
— then runs its *setUp* function
— then runs all methods that are named *test_**
— then prints a summary (passed vs failed tests)

And more. We've already seen other methods to do unitary tests (using the *__name__==__main__* trick in the program containing some *_test_func*). unittest is just more practical when there is a large set of tests as here. See all unittest frameworks online [6].

6.1.2 Installing the testing frameworks

```
conda install -c anaconda pytest
```

Additionally, although it should not be needed :

```
conda install -c anaconda unittest2
```

6.1.3 Running tests

Option A) (preferred) To run a given test :

```
python -m unittest test_name
```

e.g. :

```
python -m unittest tests_binary
```

if tests_binary.py is in the main binary repository

Option B) Run all tests in tests folder (-v for verbose mode) :

```
py.test -v
```

5. https://gitlab.ensta.fr/paun/mo101-agile-tdd/tree/master
6. http://docs.python.org/3/library/unittest.html

6.1.4 Links

[1] Github (alternative to gitlab, easier) : [7] ->request discount for student -> individual, using your ensta email. Option b : create gitlab account [8].

[2] Create and add your ssh key to gitlab [9] or github [10]

[3] Atom editor [11]

[4] Sublime editor [12]

[5] Python overloading operator API : https://docs.python.org/3.4/library/operator.html

6.1.5 Troubleshooting

— Q : Using gitlab :

```
bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
fatal: protocol error: bad line length character: No s
```

A : You can try setting the locale language :

```
export LC_ALL=fr_FR.UTF-8
```

you can also try run the command *ssh-add* (this commands adds the ssh key to the ssh agent).

7. https://education.github.com/
8. http://gitlab.ensta.fr/
9. https://gitlab.ensta.fr/help/ssh/README
10. https://help.github.com/articles/adding-a-new-ssh-key-to-your-github-account/
11. https://atom.io
12. https://www.sublimetext.com/3