Final Project

Due Friday Dec. 15, 5 pm

PDF

The project will be done in groups of three, with students assigned randomly.

A few comments. First, the project, when split amongst the group members, is not intended to be a huge undertaking. The goals of the project are to give you experience in working collaboratively and developing a well-designed, well-tested piece of software, as well as experience experimenting with a method and comparing to other methods.

The project will be graded as a letter grade and will count for about as much as two problem sets in your final grade.

Please use standard citation practices to cite any papers/code/online resources/people whose ideas you make use of. Please do not consult with class members who are not in your group.

Problem

Your task is to implement and experiment with the use of genetic algorithms for variable selection, including both linear regression and GLMs (though the ideas and implemention would readily generalize to other statistics/ML/DS methods). Some details on genetic algorithms are available in Section 3.4 of the Givens and Hoeting book on Computational Statistics and the baseball data discussed there are in the class GitHub repository as `project/baseball.dat`. You should also be able to find plenty of other information about genetic algorithms online. The result should be an Python package that I can easily use (in particular, I will be testing your code on my own test cases). My grading will be largely based on the following items.

1. Your solution should allow the user to provide sensible inputs. Key inputs of course are the dataset and the type of regression, which you should just be able to pass along to relevant functions from the `statsmodels` or `scikit-learn` packages. By default you should just use AIC as your objective criterion but allow users to provide their own objective function. Similarly you can use the genetic operators described in Givens and Hoeting for variable selection, but ideally your code should be general enough that a user could provide additional operators. For various inputs that control the behavior of the genetic algorithm you should try to come up with good defaults (based in part on point 4 below), but also give users flexibility.
2. Your solution should involve modular code, with functions or OOP methods that implement discrete tasks. You should have an overall design and style that is consistent across the components.

3. In terms of efficiency, the generations are inherently sequential. However, you should try to vectorize as much as possible within a generation. If you like, you can allow for parallel processing on a single machine when working with the population in a given generation, in particular the evaluation of the fitness function, but it is not required.
4. Show the results of using your implementation on several examples (some of these could potentially be simulated data). Also compare results to use of the Lasso (see `sklearn.linear_model.lars_path` or `sklearn.linear_model.lasso_path`).
5. Formal testing is required, with a set of tests where results are compared to some known truth. For testing the overall function, since the algorithm is stochastic, you'll need to think carefully about how to set this up. You should also have unit tests for individual functions that carry out the individual computations that make up the algorithm.
6. You should be writing your own code for essentially everything except the model fitting and other standard functionality available in Python/numpy/scipy. You can also use `scikit` utilities (such as for objective functions). If you'd like to use any other code or packages, please consult with me first.

Formatting requirements and additional information

1. Your solution to the problem should have two parts:
    a. A Python package named `GA`, which can be as simple as a directory named `GA` with .py modules and an `__init__.py`. (If you'd like to get experience with creating a package that could be installed via `pip` or `conda install` you're welcome to explore that but it's not required.) The package should be made available to me via a private repository named `GA-dev` within the Berkeley GitHub account or github.com account of one of the project members. **Make sure to share the repository with me** (my username is `paciorek` in either `github.berkeley.edu` or `github.com`.)

        The package should include:

        i. A primary function called `select` that carries out the variable selection, including appropriate code comments.

        ii. Other supporting code in the same or additional files (please think about clear organization), including appropriate code comments.

        iii. Formal tests set up using `pytest` and included in the package.

        iv. Help information for the main function, in the form of a standard Python doc string for `select`. As part of this, you should have working examples in the example section. You do not need extensive doc strings for your auxiliary functions but there should be a brief doc string just stating what each function does.

    b. A PDF document describing your solution, prepared as a Quarto document or Jupyter notebook. The description does not need to be more than 2-4 pages, but should describe the approach you took in terms of functions/modularity/object-oriented programming, the testing that you carried out, and the results of applying the implementation to the examples. It must include a paragraph describing the specific contributions of each team member and which person/people were responsible for each component of the work. Please submit a

paper copy of the document to me - either directly to me, under my door, or in my mailbox. **On your paper solution, please indicate the URL of the GitHub repository for the project.**

2. You should start the process by mapping out the modular components you need to write and how they will fit together, as well as what the primary function will do. After one person writes a component, another person on the team should test it and, with the original coder, improve it. You could also consider using pair programming for some of your development.
3. You should use Git and GitHub to manage your collaboration.