Stat243: Final group project, Due Friday Dec. 12, 5 pm

November 8, 2014

The project will be done in groups of 3-4, with students assigned randomly but balanced between statistics and non-statistics students.

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.

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

Formatting requirements

Your solution to the problem should have two parts:

1. The code of your solution, including (a) a primary function that carries out the variable selection, (b) a function that carries out the formal tests you applied to your function, and (c) auxiliary functions used by the primary function and testing function. All code should be submitted to Github as a single text file with comments clearly delineating different pieces of the code. Please call the file "genetic.R". Please call your main function "select" and your test function "test". Of course in your development you could use many files, but please combine for the final one.

2. A paper document describing your solution, prepared in LaTeX or R Markdown. The description does not need to be more than a couple pages, but should describe the approach you took in terms of functions/modularity/object-oriented programming, the results on one or more real examples, and the testing that you carried out. 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 the paper document to me - either directly to me, under my door, or in my mailbox. On your paper solution, please indicate the last name of the group member who submitted the R code so that I know where to look on Github.

3. For the help information on your main function, this can either be part of your R code as indicated below or if a PDF, can be stapled to your paper solution.

For a given subtask of the problem, if you find good code available, you may use it as a modular component of your code provided it does not constitute too large a part of your solution and provided you test the code. Consult me with questions on this matter.

You should use Git and Github to manage your collaboration.

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 might also consider an approach called pair programming (see the Wikipedia entry) where two people write code side by side.

Problem

Your task is to implement an genetic algorithm for variable selection in regression problems, including both linear regression and GLMs. Some details are available in Section 3.4 of the Givens and Hoeting book on Computational Statistics - see the electronic book available on Oskicat, or the PDF of Chapter 3 in the class Github repo. You should also be able to find plenty of other information about genetic algorithms online.

1. Your solution should allow the user to provide reasonable inputs, in terms of specifying a dataset and regression model formula, as well as the type of regression. Much of this is information you should just be able to pass along to *lm()* or *glm()*. By default you should just use AIC as your objective criterion/fitness function, but allow users to provide their own fitness 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.

2. Formal testing is required, with a set of tests where results are compared to some known truth. You should have tests for the overall function, and for any modules that do anything complicated. For testing the overall function, since the algorithm is stochastic, you'll need to think carefully about how to set this up. The output of your testing function should be clear and interpretable. I.e., when I run your test function, it should print informative messages of what it is doing and whether the test was passed or failed.

3. 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, in terms of functions vs. OOP methods, naming of objects, etc.

4. In terms of efficiency, the generations are inherently sequential. However, you should try to vectorize as much as possible and allow for shared memory parallel processing when working with the population in a given generation, in particular the evaluation of the fitness function.

5. Show the results of using your implementation on one or more real examples. One possibility is to use it on a subset of the airline data from PS6. For example, you could model delay/non-delay based on predictors such as origin airport, destination airport, month-year combinations, origin-destination pairs, etc. But you don't have to use this dataset.

6. Your solution should include help/manual information as follows: (a) basic doc strings for any function you create (i.e., include comments at the beginning of the function), as well as commenting of code as appropriate. (b) An overall help page for the primary function. This could be in the form of comments in the R code in the form used in the *roxygen2* package (see http://adv-r.had.co.nz/Documenting-functions.html for an example) or simply a PDF (e.g., see the help info on, say, p. 29 of http://www.cran.r-project.org/web/packages/spam/spam.pdf for help on the *chol()* function in the *spam* package). I'd prefer the first (and that is likely to be easier), but the latter is fine too.