Stat243: Problem Set 7, Due Monday Nov. 16

November 2, 2015

Comments:

- This covers Units 9 and 10.
- It's due at the start of class on Nov. 16.
- As usual, your solution should mix textual description of your solution, code, and example output. Feel free to write out answers to the mathematical problems by hand if you like. If you do so, please staple them into any PDF pages in the correct order to avoid having Harold have to hunt around for what problem solution is where.
- Please note my comments in the syllabus about when to ask for help and about working together.
- Please give the names of any other students that you worked with on the problem set.

Questions

1. **(Due Monday Nov. 9)** Per the Piazza announcement, please follow the instructions in *section_questions_nov9.pdf* in the sections directory of the class repository. Write your responses in a file called *ps7_question1.txt*, put it in the top-level directory of your Git repository, and push to Github as you do for your problem set solutions. Also include your responses in your submission of PS7.

2. Details of the Cholesky decomposition.

   (a) Work out the operation count (multiplies and divides) for the Cholesky decomposition, including the constant c, not just the order, for terms involving n^3 or n^2 (e.g., $5n^3/2 + 75n^2$, not $O(n^3)$). You can ignore the square root and any additions/subtractions. You can ignore pivoting for the purpose of this problem. Remember not to count any steps that involve multiplying by 0 or 1. Compare your result to that given in the notes.

   (b) Suppose I've written out the Cholesky calculation based on for loops. If I wanted to save storage space, can I store the Cholesky upper triangular matrix, U, in the storage space that is used for the original matrix as I go along, assuming I'm willing to lose the original matrix, or do I overwrite anything I need later in the calculation of the Cholesky?

   (c) Now, using a test matrix X, compute the Cholesky (using R's *chol()* function) and monitor memory use based on top or based on *mem_used()* or *gc()* in R. Does memory use match that from your answer in part (b)? If not, how much more or less memory is used than you might expect? For a variety of values of n (make sure you have matrices with n in the thousands), find the maximum memory use and the processing time and plot these as a function of n. Empirically how do memory use and processing time scale with n? [Note: make sure the only objects of any substantial size in your workspace are X and the resulting Cholesky matrix so that you only consider memory use from this operation.] For this problem make sure your calculations use only a single thread/core.

Note: You can compute a positive definite test matrix either using the correlation function examples for stochastic (Gaussian) processes we've seen in class, or consider the implications of problem 5a.

3. Compare the speed of $b = X^{-1}y$ using: (a) `solve(X)` followed by '%*%'; (b) `solve(X,y)`; and (c) Cholesky decomposition followed by solving triangular systems. Do this for a matrix of size 5000×5000 using a single thread.

   (a) How do the timing and relative ordering amongst methods compare to the order of computations we discussed in class and the notes? (I don't think I mentioned it anywhere, but the full inversion can be found to take n^3 calculations.)

   (b) Are the results for b the same numerically for the different methods (up to machine precision)? Comment on how many decimal places in b agree, and relate this to the condition number of the calculation.

4. Suppose I need to compute the generalized least squares estimator, $\hat{\beta} = (X^\top \Sigma^{-1} X)^{-1} X^\top \Sigma^{-1} Y$, for X $n \times p$, Σ $n \times n$ and assume that $n > p$. Assume n could be of order several thousand and p of order in the hundreds. First write out in pseudo-code how you would do this in an efficient way - i.e., the particular linear algebra steps and the order of operations. Then write efficient R code in the form of a function, *gls()*, to do this - you can rely on the various high-level functions for matrix decompositions and solving systems of equations, but you should not use any code that already exists for doing generalized least squares.

5. Some practice with matrix manipulations.

   (a) Consider a rectangular matrix, X, with dimensions $n \times p$ and $n > p$. Show that the right singular vectors of X are the eigenvectors of the matrix $X^\top X$ and that the eigenvalues of $X^\top X$ are the squares of the singular values of X. Also show that $X^\top X$ is positive semi-definite (which is good because $X^\top X$ is essentially an empirical covariance matrix, up to scaling and shifting).
   (Sidenote: as I mentioned in class, since the condition number of X is the ratio of the largest and smallest magnitude singular values, this shows why the condition number for using the Cholesky for regression is the square of the condition number for using the QR.)

   (b) Consider an $n \times n$ positive semi-definite matrix X and assume you have already computed the eigendecomposition of X. How can you compute the eigenvalues of $Z = X + cI$ in $O(n)$ arithmetic calculations (including any additions or multiplications), where c is a scalar and I is the identity matrix.