Stat243: Problem Set 7, Due Monday November 16

November 7, 2020

This covers Units 9 and 10.
It's due **as PDF submitted to Gradescope** and submitted via GitHub at 10 am on Nov. 16.
Comments:

1. The formatting requirements are the same as previous problem sets.

2. Please note my comments in the syllabus about when to ask for help and about working together. In particular, **please give the names of any other students that you worked with on the problem set and indicate in comments any ideas or code you borrowed from another student.**

Problems

1. Suppose I have a statistical method that estimates a regression coefficient and its standard error. I develop a simulation study and have m = 1000 simulated datasets that each give me an estimate of the coefficent and its estimated standard error. How would I determine if the standard error provided by the statistical method properly characterizes the uncertainty of the estimated regression coefficient? You could answer this generally or by talking specifically about the SE columns in Table 1 of the Cao et al. paper. Note your answer can be as simple as a sentence or two describing what quantities to consider.

2. Experimenting with importance sampling.

   (a) Use importance sampling to estimate the mean (i.e., $\phi = E_f X$) of a truncated t distribution with 3 degrees of freedom, truncated such that $X < (-4)$. Have your sampling density be a normal distribution centered at -4 and then truncated so you only sample values less than -4 (this is called a half-normal distribution). You should be able to do this without discarding any samples (how?). Use $m = 10000$ samples. Create histograms of the weights $f(x)/g(x)$ to get a sense for whether Var$(\hat{\phi})$ is large. Note if there are any extreme weights that would have a very strong influence on $\hat{\phi}$. Estimate Var$(\hat{\phi})$. Hint: remember that your $f(x)$ needs to be appropriately normalized or you need to adjust the weights per the class notes.

   (b) Now use importance sampling to estimate the mean of the same truncated t distribution with 3 degrees of freedom, truncated such that $X < (-4)$, but have your sampling density be a t distribution, with 1 degree of freedom (not 3), centered at -4 and truncated so you only sample values less than -4. Again you shouldn't have to discard any samples. Respond to the same questions as above in part (a). In addition, compute a 95% uncertainty interval for your estimate, using the Monte Carlo simulation error, $\sqrt{\hat{\text{Var}}(\hat{\mu})}$.

3. 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.

4. We've seen how to use Gaussian elimination (i.e., the LU decomposition) to solve $Ax = b$ and that we can do the solution in $n^3/3$ operations (plus lower-order terms). Suppose I want to know how inefficient it is to explicitly invert the matrix A and then multiply, thereby finding $x = A^{-1}b$ via matrix-vector multiplication. If we look at R's *solve.default()*, we see it solves the system $AZ = I$ to find $Z = A^{-1}$. Next note that *help(solve)* indicates it calls a Lapack routine DGESV (http://www.netlib.org/lapack/explore-html/d7/d3b/group__double_g_esolve_ga5ee879032a8365897c3ba91e3dc8d512.html), which uses the LU decomposition.
   Count the number of computations for

   (a) transforming $AZ = I$ to $UZ = I^*$ (where I^* is no longer a diagonal matrix),
   (b) for solving for Z given $UZ = I^*$, and
   (c) for calculating $x = Zb$.

   Then compare the total cost to the $n^3/3$ cost of what we saw in the class notes and video.

   Notes: In counting the computations you should be able to make use of various results we derived in class concerning the Gaussian elimination computations and computations involved in a backsolve, so your answer should be able to simply combine together results we've already discussed without any detailed new derivation.
   Second note: Given that R's call to *dgesv* doesn't take account of the special structure of I on the right-hand side, you do not need to take account of the fact that because I has zeroes and ones, one can actually save some computation. You can simply count the calculations as if I were filled with arbitrary values. (Note: if we did actually try to be careful about making use of the structure of I, it turns out we could save $n^3/3$ calculations.)

5. Two-stage least squares (2SLS) is a way of implementing a causal inference method called instrumental variables that is commonly used in economics. Consider the following set of regression equations:

$$\begin{aligned} \hat{X} &= Z(Z^\top Z)^{-1} Z^\top X \\ \hat{\beta} &= (\hat{X}^\top \hat{X})^{-1} \hat{X}^\top y \end{aligned}$$

   which can be interpreted as regressing y on X after filtering such that we only retain variation in X that is correlated with the instrumental variable Z. An economics graduate student asked how he could compute $\hat{\beta}$ if Z is 60 million by 630, X is 60 million by 600, and y is 60 million by 1, but both Z and X are sparse matrices.

   (a) Describe briefly why I can't do this calculation in two steps as given in the equations, even if I use the techniques for OLS discussed in class for each stage.
   (b) Figure out how to rewrite the equations such that you can actually calculate $\hat{\beta}$ on a computer without a huge amount of memory. You can assume that any matrix multiplications involving sparse matrices can be done on the computer (e.g., using the *spam* package in R). Describe the specific steps of how you would do this and/or write out in pseudo-code.

Notes: (1) The product of two sparse matrices is not (in general) sparse and would not be sparse in this case. (2) As discussed in Section 6.2 of Unit 10, there are R packages (and software packages more generally) for efficiently storing (to save memory) and efficiently doing matrix manipulations (to save computation time) with sparse matrices.

6. **(Extra credit)** In class we saw that the condition number when solving a system of equations, $Ax = b$, is the ratio of the largest and smallest magnitude eigenvalues of A. Show that $\|A\|_2$ (i.e., the matrix norm induced by the usual L2 vector norm; see Section 1.6 of Unit 10) is the largest of the absolute values of the eigenvalues of A for symmetric A. To do so, find the following quantity,

$$\|A\|_2 = \sup_{z:\|z\|_2=1} \sqrt{(Az)^\top Az}.$$

If you're not familiar with the notion of the supremum (the *sup* here), just think of it as the maximum. It accounts for situations such as trying to find the maximum of the numbers in the open interval (0,1). The max is undefined in this case since there is always a number closer to 1 than any number you choose, but the *sup* in this case is 1.

Hints: when you get to having the quantity $\Gamma^\top z$ for orthogonal Γ, set $y = \Gamma^\top z$ and show that if $\|z\|_2 = 1$ then $\|y\|_2 = 1$. Finally, if you have the quantity $y^\top Dy$, think about how this can be rewritten given the form of D and think intuitively about how to maximize it if $\|y\|_2 = 1$.

7. **(Extra credit)** In Unit 10 we discussed that having a 0 as an eigenvalue of a covariance matrix amounts to having a constraint (one of the eigenvectors has zero weight). In (optional) Section 2.4, I say a bit about having a zero eigenvalue of a precision matrix, where a precision matrix is the inverse of the covariance matrix.

   (a) What is the relationship between the eigenvalues of a covariance matrix, Σ, and the eigenvalues of the corresponding precision matrix, Σ^{-1}?

   (b) Consider the following autoregressive style model:

$$y_i \sim N(y_{i-1}, \sigma^2)$$

The likelihood (joint distribution for $(y_1, \ldots, y_n)$) is

$$\prod_{i=2}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(y_i - y_{i-1})^2\right)$$

which gives us the sum of squares:

$$\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - y_{i-1})^2$$

We can equivalently represent the model as

$$(y_1, \ldots, y_n) = Y \sim N(0, \sigma^2\Sigma)$$

where $Q = \Sigma^{-1}$ is the precision matrix and looks like this (for the specific case of $n = 5$):

$$Q = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

Show that with the joint distribution based on the multivariate representation, $Y \sim N(0, \sigma^2 Q^{-1})$, you get the same sum of squares as above.

(c) Show that if you add a constant value to every y_i, the sum of squares is unchanged. This makes sense because we can see the sum of squares as involving contrasts (differences) of adjacent data values. The interpretation is that this model says nothing about the overall level of the y_i values, only about their contrasts.

(d) For the $n = 100$ case, create Q and find the eigendecomposition in R and show that one of the eigenvalues is 0 and that the corresponding eigenvector is a vector with each value equal to $1/\sqrt{n}$.

(e) To generate a random vector $Y = \Gamma \Lambda^{1/2} z$, where Γ and Λ have the eigenvectors and eigenvalues of Σ, we would need to invert Q. But we can't do that because an eigenvalue is 0. Instead, we would use the pseudo-inverse described at the start of Section 2.4. For the case of $n = 100$ carry out this algorithm and generate and plot a small number of random vectors Y from this autoregressive model. You should see that the mean of each vector Y is 0, which is consistent with having imposed a constraint by using the pseudo-inverse.