Peer Review 1 for Devon Humphries

Spencer Woody
SDS 385

September 14, 2016

1 Introduction

Hi Devon,

Thanks for meeting with me. In this review I will provide my overall impressions of comments on both your written report and the R code you provided on you github. This will essenitially be a written version of what we discussed today after class.

2 Review of Report

2.1 Linear Regression

First off, this section should really be called "Linear Weighted Regression" or something similar. Since we are dealing with matrices and vectors throughout this problem, you don't need to bolden all you variables for the matrices and vectors. Doing so takes extra time and also can lead to confusion. For example, you bolden both y but not β even though they are both vectors. On the bottom of page 1 you say "Taking the partial derivative with respect to β ...," however, we are taking the *gradient* of this expression since β is a vector. Additionally the notation $\frac{\partial}{\partial \beta}$ is invalid and so you should instead use ∇_β throughout. However, besides these nitpicks your derivation of $\hat{\beta}$ is very clear.

I suggest trying to code up the SVD and Cholesky decompositions in addition to the QR decomposition. Also, why is Cholesky more unstable than QR? Expand on that. Your pseudocode should look a little more like actual

R code. Here's a generic example:

Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while *not at end of this document* **do**
  read current;
  if *understand* **then**
    go to next section;
    current section becomes this one;
  else
    go back to the beginning of current section;
  end
end

There are several packages you can use to do this. You can learn more about incorporating pseudocode on the LaTeX Wikibooks page for Algorithms.

You do a good job of including your R code inline with the rest of your report. However, I would still recommend using syntax highlighting which will make your code more readable at a first glance. Look into the `listings package` within LaTeX.

In your section talking about sparse matrices, perhaps expand a bit more on how exactly R handles spare matrices. I believe that R does this by treating a matrix as an element list of non-zero entries. Also, I did not see any benchmarking done across your various solve functions. This is key to understanding the comparative computational advantages of your functions.

2.2 GLM

I noticed that you have only shown the derivation of the gradient of the likelihood, but I will still review what you have. You can use the `\ln` command within your LaTeX code to have it appear as "ln" instead of "ln". It is generally understood that logarithms are usually natural logarithms unless otherwise noted, so it is more common to see log used instead of ln. Also, $e^{-x_i^T\beta}$ looks pretty congested. Use `\exp(-x_ i^T\beta)` instead to have it appear as $\exp(-x_i^T\beta)$. If you use the `align` environment (read about it on the LaTeX Wikibooks page for Advanced Mathematics) then you can align all your equations so that the equal signs all line up. The rest of this section looks good though. I like that you apply the chain rule by first finding the

gradient of w_i and then using that in finding the full gradient of the log-likelihood.

3 Review of R Code

Overall, your R code looks very clean and organized. The spacing is done well so that it is easy to read. A big help, though, would be to include plenty of comments so that a reader can get through your code more efficiently.

You sometimes put a lot of matrix commands in one line. For example, in one line you write `betahat = solve(t(X) %*% W %*% X) %*% t(X) %*% W %*% y`. I'm not sure, but it may be more computationally efficient to break this off into two lines. Do some benchmarking with matrix operations and see what works most efficiently. This page has some general tips on the best ways to perform matrix multiplication in R. For example, `crossprod(X,y)` is faster than `t(X) %*% y`, and `A %*% (B %*% y)` is faster than `A %*% B %*% y`.

Within your function for solving sparse matrices, be sure that the first thing you do is redefine `X` as `X <- matrix(X, sparse = T)`. Otherwise R won't treat `X` as sparse!

I like the cleanness in how you define your functions. For example, you define a sigmoid function to make calculations of w_i easier later on. It looks very nice because it makes your subsequent functions less cluttered. It's a good application of function-oriented programming.

There is a very easy way to turn `Y` into a series of zeros and ones instead of writing an inefficient `for` loop. Just do `Y <- Y == "B"`. Technically this will be a vector of Boolean values (`TRUE` and `FALSE`) but they are still handled mathematically as zeros and ones. You may find it useful to read up on how R handles logicals on the documentation page for Logic in R.

When you are going through the iterations of a `for` loop and updating a list for each iteration, it is more efficient to instead define a list *before your loop* of `NA` values with the same length as the number of iterations, and then changing the ith entry in your list during the ith iteration of your loop. You can do this with `mylist <- rep(NA, n.iterations)`. This is because it is

inefficient in R to expand a data structure in a loop. Preallocation avoids this pitfall.

4 Conclusion

I hope this review has been helpful. As always, you can reach out to me via email or in my office if you would like to further discuss this assignment. If you update report or R code, possibly to include more about the second problem, I am happy to review that as well.