Peer Review 1 for Yinan

Spencer Woody
SDS 383D

February 8, 2017

1 Introduction

Hi Jared,

Thanks for letting me review your SGD implementation. In this review I will provide my general impressions of your C++ and R code and some suggestions that I have for you.

2 Code Performance and Lazy Updating

Overall I am impressed by your code's performance. You ran your whole script over the dataset in about 12 seconds which is a pretty good time. Your negative log-likelihood converges in a similar fashion to what I saw in my own code. My only concern is about the way that you apply lazy updating. I'm not sure why of your justification for using the "hack" lazy updating that you do. It's clear that your code functions just fine regardless and shrinks the coefficients (judging by your predictive power plot). But still, a more detailed explanation would show just how good an approximation it is, and how much it affects your coefficients compared to penalizing them for each iteration.

One thing I suggest is adding a few more comments to your code. There are three for-loops in your code, so it is easy to get lost in what is happening in which loop, etc. So perhaps you can give a comment above each one saying what exactly is happening in each one, or give an overview of the whole looping mechanism right before you start your first loop. This would help a reader with less working knowledge of SGD to parse through and understand your code

3 Cross-Validation

I like the graph you produced for measuring predictive power. It's impressive that you were able to measure accuracy, sensitivity, and specificity over a set of λ values ranging from e^{-15} to e^{5}.

However, I would encourage you to run k-fold cross-validation on your model, where you break up the whole dataset into k bins and measure predictive power k times, each time treating one bin as the testing set and the data outside that bin as the training set. Cross-validation is more robust than doing one split the data into one training set and one testing set, and it will also give you an associated standard error for the predictive power for each value of λ. Given the large sample size we are working with, you can probably get away with having only 5 folds. Granted, this means performing five times the number of computations as you normally would, but you can choose a smaller set of λ values to cross-validate over.

When I did this, I saw that having no penalty led to the greatest predictive power. I have run into similar results when applying ridge regularization to linear regression, where out-of-sample mean squared error is smallest without any penalty. This is because the ridge penalty introduces bias for your coefficient estimates. Still, you might choose to include a penalty, and follow the one standard error rule, which states that you choose the most restricted model (i.e., the one with the largest penalty) with an associated error rate less than the lowest error rate plus its standard error. Robert Tibshirani has a great set of slides on cross-validation on his CMU page. Look at what he mentions on the one standard error rule. This is also discussed in 3.3 of *The Elements of Statistical Learning* by Hastie, Tibshirani, and Friedman.

Once again, nice work with your implementation! Feel free to reach out to me with any questions you have or if you'd like to discuss this further.

Spencer