BSTA 670 Statistical Computing: Homework 3

John Pluta

Fall 2014

1 Introduction

Ordinary least squares (OLS) regression is a commonly used method for solving regression problems, by finding coefficient estimates that minimize the residual sum of squared errors. These values make it possible to draw inference on the relationship between the dependent and independent variables. However, there are some cases where OLS is not the preferred method for regression. When the number of predictors, p, approaches or exceeds the number of observations, n, the design matrix is less than full rank and leads to non-unique solutions. Additionally, when predictors are highly correlated, the variance of the estimated coefficients is inflated [3]. Penalized regression refers to a class of techniques that address these issues by placing a constraint on the size of the coefficients and reduces the variance of the estimates, at the expense of some bias. These models take the general form of:

$$\hat{\beta} = (Y - X\beta)'(Y - X\beta) - \phi$$

where ϕ is the penalty term that shrinks the coefficients. There are many types of penalized regression, and each is uniquely determined by its penalty term. One method of particular interest is the least absolute shrinkage and selection operator (LASSO), proposed in [1], which is a form of penalized regression that also enforces variable selection by shrinking some terms to exactly zero.

The LASSO estimate takes the form:

$$\hat{\beta} = argmin_{\beta}(\mathbf{y} - \mathbf{X}\beta)'(\mathbf{y} - \mathbf{X}\beta) - \lambda \sum_{j=1}^{p} |\beta_j|$$

There is no closed form solution to this problem, thus analytical programming methods are required for its solution. In this paper, we examine three such methods: a coordinate descent algorithm [3], a parallel stochastic coordinate descent algorithm [**?**], and the popular least angle regression (LARS) algorithm [2].

Before describing the algorithms, note that by definition, each of them begin with all $\beta_j = 0$. [3] suggests a warm start, where the ordinary least squares coefficient estimates are used as a starting point. In practice, the difference in computation time between the two is negligable, and a warm start will fail in the case where $p > n$. Thus an initialization of $\beta_j = 0$ is used in all cases.

2 Algorithms

2.1 Coordinate Descent: "Shooting"

The coordinate descent algorithm was first introduced in [3] and further refined in [4]. It works by sequentially minimizing the objective function with respect to a single β_j, while holding all other coefficients constant. That is, the partial residual of each covariate is computed and each step, and a soft threshold is applied to adjust the coefficient estimate:

$$\hat{\beta}_j \leftarrow S(X'(y - X_{-j}\beta_{-j})$$

where -j implies the set of covariates not including j, and S is the soft threshold operator:

$S(z, \gamma) = sign(z)(|z| - \gamma)_+$

2.1.1 Number of operations

In optimization problems, we are typically only interested in the operations that are of the highest magnitude. For all of these algorithms, the most computationally expensive operations are matrix multiplication and inversion. Interestingly, these operations take an identical number of flops [**?**], which depends on the method but is generally in the order of n^3. For each algorithm, we only consider these functions for the number of operations.

The coordinate descent algorithm is very appealing as it is arithmetically extremely simple, and thus has a small number of operations. In an optimized version of the algorithm, X'y is computed once (np flops), and the nested loop has two matrix multiplications and one subtractions, for a total of pn + p^2 * N * (pn + p + 1) flops, where N is the number of iterations.

2.1.2 Convergence

Convergence is reached when the difference between the current update and the previous update is below some user-defined threshold (bounded by machine precision). Stated mathematically, converged has been reached when

$\beta_j - \beta_{j-1} < \epsilon$

where ϵ is the threshold. [7] performed a convergence analysis showing that convergence of the coordinate descent algorithm is on the order of np^2; that is, the computation time grows exponentially with the size of the amount of data.

2.2 Parallel Stochastic Descent: "Shotgun"

The parallel stochastic descent algorithm is a parallelized version of the stochastic coordinate descent algorithm described in [5]. In that work, the authors make one change to the usual coordinate descent algorithm. Instead of adjusting each coefficient in sequence, one covariate is updated per iteration, chosen uniformly at random. Parallel stochastic descent further improves upon stochastic coordinate descent by exporting each coefficient update to a different core, so that multiple updates can be processed simultaneously.

2.2.1 Number of operations

All of the calculations are the same as in coordinate descent, except that c cores are used. Instead of each iteration using p loops, $\frac{p}{c}$ are used. We also need to consider the substantial overhead that is added by creating a parallel computing environment, some constant P. Thus, this algorithm uses $P + pn + \frac{p^2}{c} * N * (pn + p + 1)$ flops.

2.2.2 Convergence

Convergence criterion are identical to coordinate descent. Analysis by [**?**] showed that the small modification in stochastic descent has the amazing effect of boosting convergence speed to a factor of p. Thus, the computation time increases only linearly with the size of the problem, instead of exponentially as in coordinate descent. Parallel stochastic descent improves upon this even further, with the convergence rate becoming a factor of $\frac{p}{P}$, where P is the number of cores used [6]. The computation time still grows linearly with the size of the data, but at a fraction of the rate of stochastic descent.

2.3 Least Angle Regression: "LARS"

LARS was first introduced in [?]. It works by beginning with a null set, and iteratively adding or removing variables to the active set one at a time. Only variables in the active set have their coefficient estimates updated in each iteration. This algorithm is unique in that it computes the coefficient paths over the entire range of λ, from 0 to the maximum value that includes all variables. Figure 1 provides an example of coefficient paths from the diabetes dataset. This mapping provides a visual interpretation of how the value of λ effects coefficient estimates, which is of critical importance in setting the tuning parameter for the model.

2.3.1 Number of operations

LARS has two matrix multiplications and one inversion, for a total of $p^3 + 2np$ flops per iteration. It is worth noting that the size of p is the size of the active set, the variables that are currently under consideration, which ranges from 1 to the full set. LARS can add or drop one variable per iteration, but usually achieves convergence in approximately p iterations. Thus we can state that the true number of iterations will be less than $np^2 + p(np + 2p)$.

2.3.2 Convergence

LARS does not convergence in the same way as the prior two algorithms. Instead, the algorithm completes when all variables have been entered into the active set. Note that this does not invalidate the LASSO benefit of variable selection (for a large enough λ, all variables will be non-zero).

3 Methods

Two experiments were conducted:

1) To compare the efficiency of each algorithm, the solution for $\hat{\beta}$ was computed by each method over multiple simulated datasets, and computation time was compared. Simulated datasets were composed of randomly generated numbers from a standard normal distribution, with correlation induced between each pair of columns. The level of correlation was chosen uniformly at random from $\rho = 0.01 - .75$. All design matrices were square and ranged in size from $n = p = 50$ to $n = p = 500$. Outcomes were generated from the same standard normal distribution.

For coordinate descent and parallel stochastic coordinate descent, λ was arbitrarily set to .2; for highly correlated data, the value of lambda is inversely proportional to the level of correlation [9], thus a value close to 0 was chosen. LARS does not take a λ parameter, and instead computes coefficient paths over all values of lambda, from 0 to the value where all variables have entered the model.

2) R is a widely used, open-source programming language, while Matlab is one of the world's most popular and robust commercial packages for scientific computing. To fully appreciate the results of experiment 1, we must also understand the impact of which language is used. To compare efficiency between R and Matlab, coordinate descent was implemented in both packages, and they were run on the same simulated data experiment as in part (1). Run times were then compared.

4 Results

The results of the first experiment are summarized in Figure 1. All algorithms found solutions to all of the simulated datasets, with the computation time of the Shooting and Shotgun algorithms being quite closely related. Shotgun achieved better performance only on the largest dataset. LARS outperformed both other algorithms by a wide margin.

5 Discussion

From this work, the LARS algorithm seems to be the clear winner. In addition to being the most efficient, it is worth noting that the LARS algorithm computes the entire path of each coefficient, as in Figure 2. In

other words, we can see how the choice of λ effects which coefficients are in the model. This has important implications on computation time, as the Shooting and Shotgun algorithms need some other a priori method to select the tuning parameter, typically either cross-validation or a path wise estimation as in [4]. A pathwise begins with the smallest value of lambda that sets all coefficients to zero, and increments lambda over a set range of values (typically with 100 steps) to get the full coefficient path. However, this requires computing the solution 100 times; thus to get an output with similar interpretation to LARS, the computational cost would be 100 times greater than what is displayed in Figure 1.

Though [5] and [6] claim that a stochastic implementation of the coordinate descent algorithm greatly decreases time to performance, this experiment failed to replicate that result.

References

[1] Tibshirani, R. 1996. *Regression shrinkage and selection via the lasso.* J. R. Statist Soc. 58(1):267-288.

[2] Efron, B., Hastie, T., Johnstone, Iain, Tibshirani, R. 2003. *Least angel regression.* The Annals of Statistics. 32(2):407-499.

[3] Fu, W. 1998. *Penalized regression: The bridge versus the lasso.* Journal of Computational and Graphical Statistics. 7(3):397-416

[4] Friedman, J., Hastie, T., Hofling, H., Tibshirani, R. 2007. *Pathwise coordinate optimization.* The Annals of Applied Statistics. 1(2):302-332.

[5] Shalev-Shwartz, S., Tewari, A. 2011. *Stochastic methods for L_1 regularized loss minimization.* Journal of Machine Learning Research. 12:1865-1892.

[6] Bradley, J., Kyrola, A., Bickson, D., Guestrin, C. 2011. *Parallel coordinate descent for L_1 regularized loss minimization.* Proceedings of the 28^{th} International Conference on Machine Learning. Bellevue, WA, USA.

[7] Tseng, P., Yun, S. 2009. *A block-coordinate gradient descent method for linearly constrained non smooth separable optimization.* Journal of Optimization Theory and Applications. 140:513-535.

[8] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein. 2009. *Introduction to Algorithms*, 3rd ed., MIT Press, Cambridge, MA.

[9] Hebiri, M., Lederer, J. 2013. *How correlations influence LASSO prediction.* IEEE Transactions on Information Theory. 50(3):1846-1854.