BIOSTAT615 Report

Yueying Hu; Sylvan Xu; Chi-Hsiang Yi

December 2023

Github Host: `https://github.com/yueyingh/BIOSTAT-615.git`

1 Introduction

Graphs and networks are frequently employed for visualizing data. In the realm of biology, various biological phenomena are often depicted through graphs, encompassing regulatory networks, metabolic pathways, and protein-protein interaction networks. This proactive utilization of graphs serves as a valuable complement to conventional numerical data like microarray gene expression data. In this project, we delve into an interesting problem in genomic data modeling, sparse fused lasso over a graph, which is a network-constrained regularization procedure for fitting linear regression models and for variable selection. Specifically, we assume that the covariates in the regression model are values of the nodes on a graph, where a link between two nodes may indicate a functional relationship between two genes in a genetic network or physical neighbor between two voxels on brain images (Li & Li, 2010). It can be considered as an optimization problem, and the Alternating Direction Method of Multipliers (ADMM) algorithm can be adapted and applied to solve this kind of problems given its flexibility in simplifying optimization problems and its general convergence properties (Zhu 2017). In this project, we plan to implement the augmented ADMM algorithm proposed by Zhu 2017 as an R package, which can be more general and more efficient in solving sparse fused lasso problems than the standard ADMM.

2 Methodology

In this section, we describe our methodology based on the Augmented ADMM algorithm. We begin with the formulation of the problem, followed by a detailed discussion on the algorithm.

2.1 Problem formulation

Consider the following optimization problem:

$$\min_{x\in\mathbf{R}^p, z\in\mathbf{R}^m} f(x) + g(Dx) \tag{1}$$

We first consider the generalized lasso problem, which is a special form of (1).

$$\min_{x\in\mathbf{R}^p} \frac{1}{2}\|Ax-b\|_2^2 + \lambda\|Dx\|_1. \tag{2}$$

where $A \in \mathbf{R}^{n\times p}$, $b \in \mathbf{R}^n$, and $D \in \mathbf{R}^{m\times p}$. Set $z = Dx$ and rewrite the problem as

$$\min_{x\in\mathbf{R}^p} \frac{1}{2}\|Ax-b\|_2^2 + \lambda\|z\|_1 \tag{3}$$
$$\text{subject to} \quad Dx - z = 0$$

The ADMM algorithm solves this equivalent formulation of (1).

A more generalized form of the problem (1) can be expressed as:

$$\min_{x\in\mathbf{R}^p} \frac{1}{2}\|Ax-b\|_2^2 + \sum_{g=1}^{G} \lambda_g \|D_g x_{\mathcal{I}_g}\|_1, \tag{4}$$

where $\mathcal{I}_g$ is the index set belonging to the gth group of variables, $g = 1, 2, ..., G$. This is an extension of the Lasso penalty which performs variable selection at the group level. The problem can also be rewritten in a similar form as (2).

2.2 Computional challenge and the Augmented ADMM Algorithm

For problem (1), the standard ADMM algorithm gives the following iterative manner:

$$x^{(k+1)} = (A^TA + \rho D^TD)^{-1}(A^Tb + \rho D^T(z^{(k)} - u^{(k)})), \tag{5}$$

$$z^{(k+1)} = S_{\lambda/\rho}(Dx^{(k+1)} + u^{(k)}), \tag{6}$$

$$u^{(k+1)} = u^{(k)} + Dx^{(k+1)} - z^{(k+1)}. \tag{7}$$

One computational challenge comes from the inversion of matrix $A^TA+\rho D^TD$. This could be challenging when $p \gg n$. The augmented ADMM algorithm simplified this inversion to $(A^TA + \rho M)^{-1}$, where M is a user-specified matrix. The only requirement is $M \in \mathbf{R}^{p\times p}$ and $M - D^TD \succeq 0$. According to Zhu (2015), problem (1) can be reformulated as:

$$\min_{x\in\mathbf{R}^p, z\in\mathbf{R}^m} f(x) + g(z) \quad \text{subject to} \quad \begin{bmatrix} D \\ (M - D^TD)^{1/2} \end{bmatrix} x - \begin{bmatrix} z \\ \tilde{z} \end{bmatrix} = 0, \tag{8}$$

where $\tilde{z} = (M - D^TD)^{1/2}$ with $M \in \mathbf{R}^{p\times p}$ and $M - D^TD \succeq 0$. Apply ADMM algorithm to (8), Zhu (2015) gives

$$x_{k+1} = \arg\min_{x\in\mathbf{R}^p} \Big\{ f(x) + (2u^{(k)} - u^{(k-1)})^T Dx + \frac{\rho}{2}(x - x^{(k)})^T D(x - x^{(k)}) \Big\} \tag{9}$$

$$z_{k+1} = \arg\min_{z\in\mathbf{R}^m} \left\{ g(z) + \frac{\rho}{2} \left\| Dx^{(k+1)} - z + \frac{1}{\rho} u^{(k)} \right\|_2^2 \right\} \tag{10}$$

$$u_{k+1} = u_k + \rho\left(Ax_{k+1} - z_{k+1}\right). \tag{11}$$

We apply (9) to (11) on problem (2), and derived the following iteration steps.

$$x^{(k+1)} = (A^TA + \rho M)^{-1}(A^Tb - D^T(2u^{(k)} - u^{(k-1)}) + \frac{\rho}{2}Mx_k + \frac{\rho}{2}M^Tx_k), \tag{12}$$

$$z^{(k+1)} = S_{\lambda/\rho}(Dx^{(k+1)} + \rho^{-1}u^{(k)}), \tag{13}$$

$$u^{(k+1)} = u^{(k)} + \rho(Dx^{(k+1)} - z^{(k+1)}). \tag{14}$$

where $S_{\lambda/\rho}$ is the solf-thesholding operator.

A common choice of M is $c\|D\|_{opt}^2 I$, where $cisaconstant$ and $\|D\|_{opt}$ is the operator norm of D. By replacing D^TD with M, the computational cost will be lower, espicially when $n \gg p$.

Similarly, we derived the iteration steps for problem (3), in a special case when $\mathcal{I}_g$ is the complete set.

$$
\begin{aligned}
x^{(k+1)} &= (A^TA + \rho M)^{-1}(A^Tb - D^T(2u^{(k)} - u^{(k-1)}) \\
&\qquad\qquad + \frac{\rho}{2}Mx_k + \frac{\rho}{2}M^Tx_k), && (15) \\
z_g^{(k+1)} &= S_{\lambda_g/\rho}(D_gx^{(k+1)} + \rho^{-1}u^{(k)}), && (16) \\
u^{(k+1)} &= u^{(k)} + \rho(Dx^{(k+1)} - z^{(k+1)}). && (17)
\end{aligned}
$$

The only difference is that z_g is updated sequentially, with different λ_g. When $\mathcal{I}_g$ is not the complete set, we could arrive at similar iterative manner by filling D_g with 0.

We modified the *admm.genlasso* function from ADMM package to implement iteration $(12) - (14)$ and $(15) - (17)$. The stopping criteria is the same as standard ADMM algorithm.

3 Description of Simulated Data

In the realm of genomic studies, the interplay of genes and their expressions can be complex. Our simulation aimed to capture this complexity by structuring data sets that reflect the attributes of genomic networks. Two distinct simulated datasets were employed: the first, derived from the ADMM package's example, and the second, based on a graph generating procedure (Zhu 2017).

Dataset 1: Standard ADMM Data

The first dataset was designed to reflect typical gene expression data:

- **Dimensions:** The dataset consists of $n = 200$ observations (representing experimental conditions) and $m = 100$ variables (depicting gene expressions).
- **Sparsity:** A sparsity level of $p = 0.1$ was set to simulate the sparse nature of gene expression data.
- **Matrix Generation:**
  - A sparse matrix x_0 was generated as the true underlying data.
  - The matrix A was created as a normalized random matrix to represent gene expressions.
  - The response vector b simulated observed traits or disease states, with added noise to reflect experimental variations.
- **Regularization and Augmentation:**
  - The diagonal matrix D and matrix M were used in the augmented ADMM algorithm to introduce a difference between the standard and augmented methods.
  - A regularization parameter regval was calculated based on the matrix A and the response b.

Dataset 2: Graph-Structured Data

The second dataset was specifically designed to capture the structural relationships in genomic networks:

- **Graph Parameters:**
  - Number of groups in the graph: num.groups = 100.
  - Number of variables per group: num.vars.per.group = 11.
  - Number of active groups (with non-zero coefficients): num.active.groups = 4.
  - Correlation coefficient within groups: cor = 0.7.
  - Error variance: err.var = 0.1.

- **Data Generation:**
  - The dataset X and response Y were generated to reflect the relationships within the graph.
  - Matrices D (incidence matrix) and C (constraint matrix) were created to represent graph structure.
  - The matrix M was utilized for the augmented ADMM algorithm.
- **Dual Lambda Parameters:** $\lambda_1 = 0.5$ and $\lambda_2 = 0.1$ were used to demonstrate the functionality of the algorithms in a graph-structured context.

4 Experimental Setup

We focused on evaluating runtime and precision as primary metrics. For comparing runtime, we ran the functions on different numbers of problem sizes, and kept track of the run time across each problem size. Precision evaluation was approached by comparing the mean squared error (MSE) between the known sparse matrix x0 and the outcomes at each iteration step of the ADMM algorithms until convergence was reached. Figures 1 and 2 depict these comparisons for standard and augmented ADMM functions. Figures 3 and 4 extend these comparisons to specially simulated data and functions tailored to handle graph inputs, allowing for the incorporation of dual lambda parameters.

5 Results and Analysis

Figures 1 to 4 collectively summarize our experimental findings:

- Figure 1 indicates a consistent speed advantage for the Augmented ADMM over the Standard ADMM as problem size increases, highlighting its efficiency in larger datasets.
- In Figure 2, initial iterations show slightly superior precision in Standard ADMM, but as both algorithms converge, their MSE values become almost identical.
- Figure 3 echoes the findings of Figure 1 in the graph representation context, reiterating the computational time advantage of Augmented ADMM.
- Figure 4 shows similar yet slightly different patterns with Figure 2. It reveals a quicker convergence and lower MSE values throughout the iterations for the Standard ADMM in the graph representation scenario.

6 Discussion

For the first two experiments above, we demonstrated the correctness of our implementation through the comparison with the existing ADMM package to solve a generalized LASSO problem. For the last two experiments which are applied to graph problems, our augmented version also achieved a similar result to the modified version of the standard ADMM algorithm. Notably, across the two experimental settings, the introduction of augmented variables significantly accelerated the computation process, a crucial factor in practical applications. It is worth noting, however, that this acceleration came at the cost of a slightly elevated Mean Squared Error (MSE). Despite this trade-off, the overall outcome remained impressive, signifying the potential of our augmented algorithm in addressing graph-related problems.
These empirical results align seamlessly with the findings presented in Zhu's seminal work (2017), thus reinforcing the reliability of our approach. Our interpretation of the author's proposed algorithm, as evidenced by our experiments, reflects a deep comprehension of the underlying principles and methodologies. Our novelty mainly lie in the R package implementation of the augmented algorithm, as it not only enhances the accessibility of our algorithm but also caters specifically to solving graph-related issues. Despite drawing inspiration from Zhu's paper in terms of algorithmic design, our novelty is evident in the translation of these

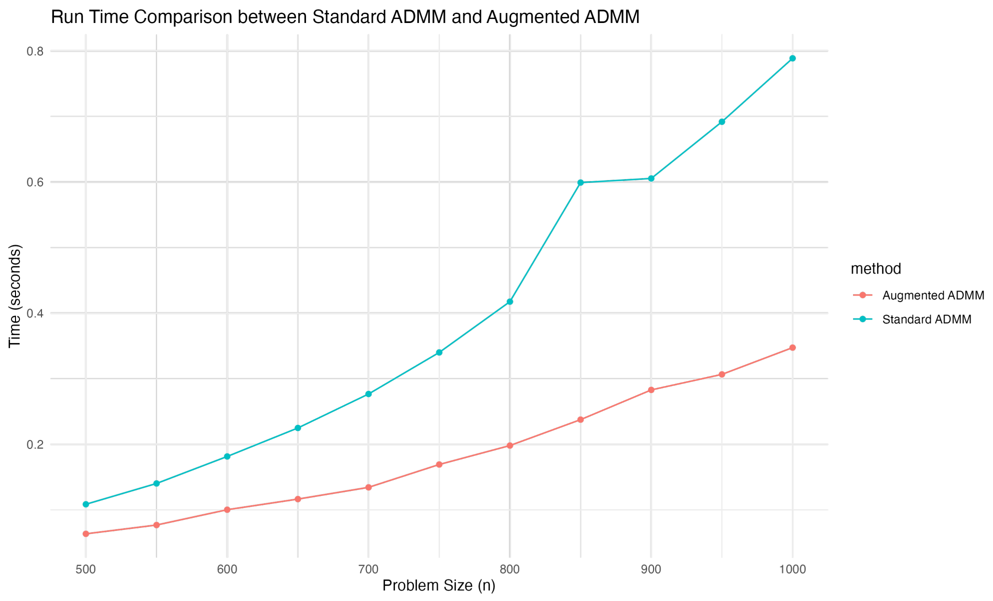

Figure 1: Runtime comparisons for the augmented ADMM (aug_admm_genlasso) and standard ADMM (admm_genlasso)

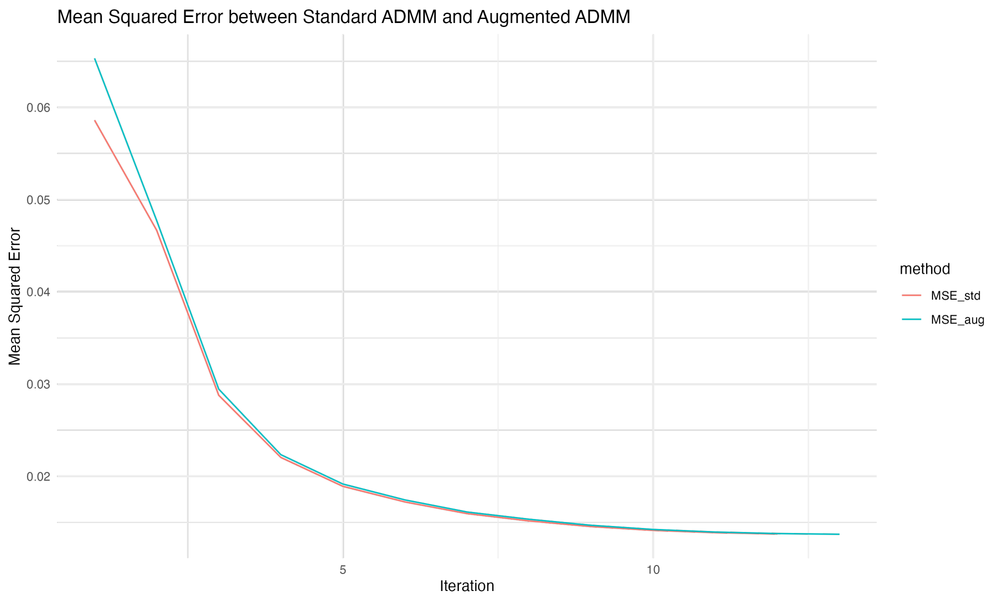

Figure 2: MSE (Mean Squared Error) comparisons for the augmented ADMM(aug_admm_genlasso) and standard ADMM(admm_genlasso)

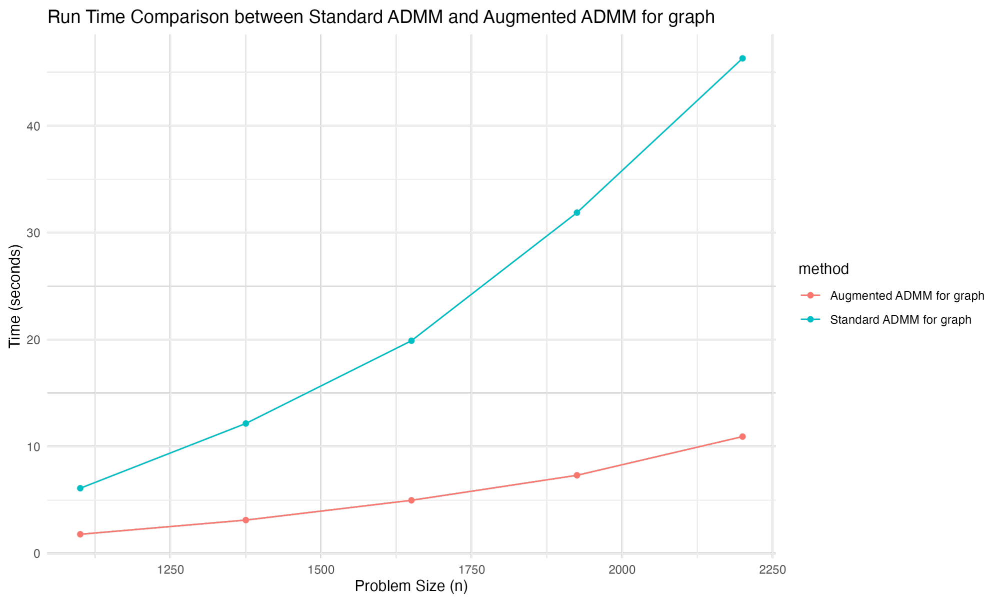

Figure 3: Runtime comparisons for the augmented ADMM(aug_admm_genlasso_for_graph) and standard ADMM(admm_genlasso_for_graph) for graph representation

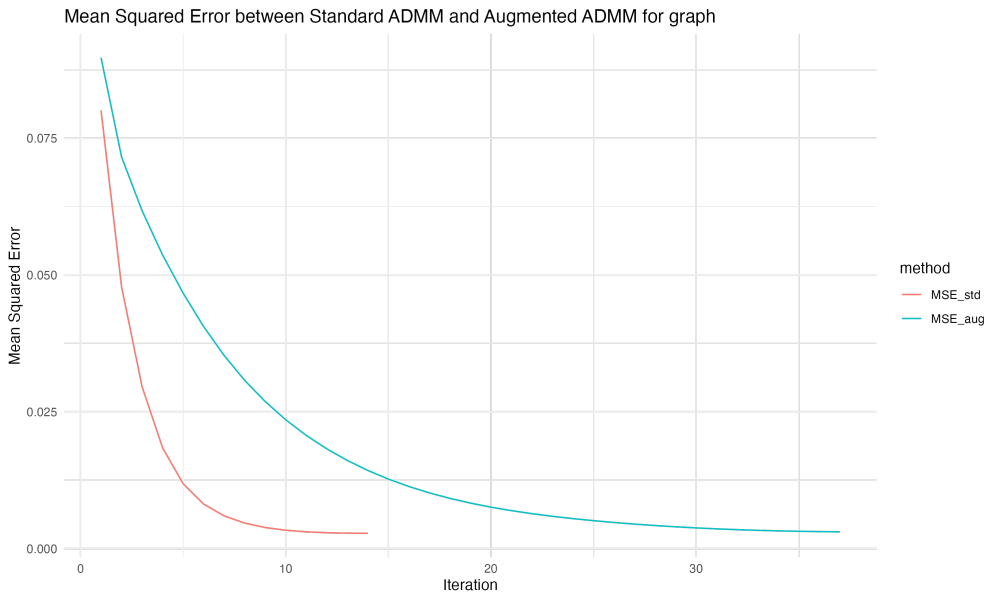

Figure 4: MSE (Mean Squared Error) comparisons for the augmented ADMM(aug_admm_genlasso_for_graph) and standard ADMM(admm_genlasso_for_graph) for graph representation

concepts into a practical, user-friendly R package. This distinction positions our work as a valuable addition to the existing body of knowledge, offering a practical tool for researchers and practitioners dealing with graph-related problems.

Nevertheless, while the current package demonstrates notable functionality, there remain certain potential areas for improvement. Notably, our investigation unveiled that the computational efficiency of the package is influenced by the choice of the augmentation matrix, with variations observed across different selections of parameter M. This observation opens up avenues for future enhancements in the package's performance. One promising direction involves the implementation of an automatic search algorithm to identify the most optimal augmented matrix based on a comprehensive analysis of the underlying rationale behind these efficiency discrepancies. By integrating such a feature, the package could dynamically adapt and optimize its performance, thereby further enhancing its versatility and applicability in various scenarios. Addressing this aspect stands as a key stride towards ensuring the package's continued relevance and effectiveness in diverse computational contexts.

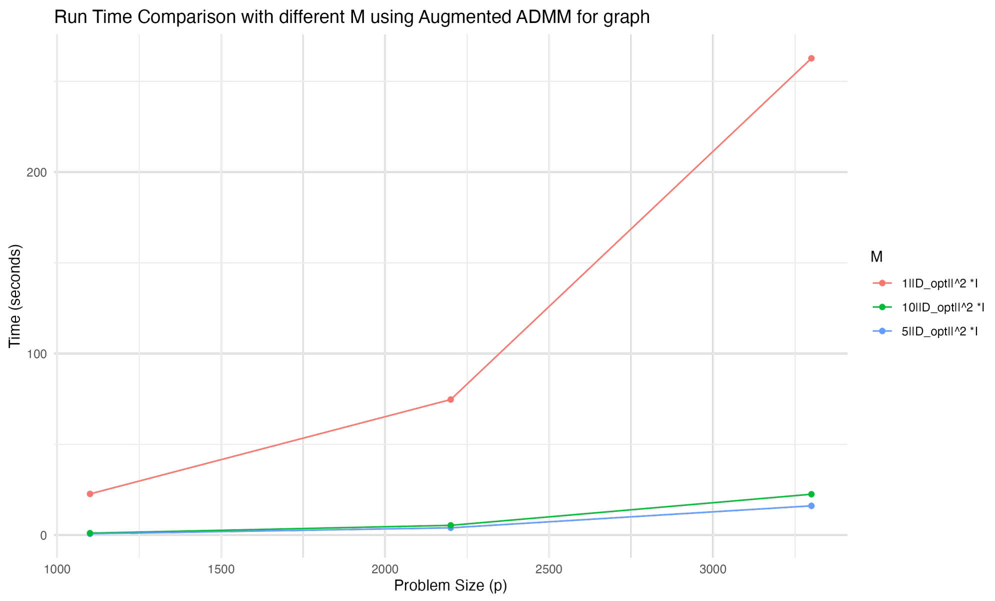

Figure 5: Runtime comparisons for the augmented ADMM (aug_admm_genlasso) with different choices of the augmented matrix

7 Contribution of Members

Theoretical Development: Sylvan Xu
Package Implementation: Yueying Hu
Package testing & Result Plotting: Chi-Hsiang Yi

8 References

Arnold, T. B., and Tibshirani, R. J. (2016), "Efficient Implementations of the Generalized Lasso Dual Path Algorithm," Journal of Computational and Graphical Statistics. [196,200]

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning, 3(1), 1-122.

Li, C., and Li, H. (2010), "Variable Selection and Regression Analysis for Graph-Structured Covariates With an Application to Genomics," The Annals of Applied Statistics, 4, 1498–1516.

Xin, B., Kawahara, Y., Wang, Y., and Gao, W. (2014), "Efficient General- ized Fused Lasso and Its Application to the Diagnosis of Alzheimers Disease," Twenty-Eighth AAAI Conference on Artificial Intelligence, pp. 2163–2169. [200]
Zhu, Y. (2017). An augmented ADMM algorithm with application to the generalized lasso problem. Journal of Computational and Graphical Statistics, 26(1), 195-204.