STAT*6801
Computational Methods for LASSO Regression

Vinay Joshy*

November 26, 2025

*By submitting this document for grading, we confirm that this work is our own and that we have adhered to the University of Guelph's Academic Integrity Policies as outlined in the Undergraduate Calendar. We recognize that any violation of these policies, whether intentional or not, may constitute academic misconduct to which we may be subject to an Academic Misconduct Investigation and penalized accordingly if found guilty.

1 Introduction

In the era of high-dimensional data, where the number of predictors (p) often exceeds the number of observations (n), traditional linear regression methods like Ordinary Least Squares (OLS) become ill-posed and prone to overfitting. To address these challenges, penalized regression techniques have become essential tools in modern statistics and machine learning. Among these, the Least Absolute Shrinkage and Selection Operator (LASSO), introduced by [1], has gained widespread popularity due to its ability to perform simultaneous variable selection and regularization.

The LASSO estimator is defined as the solution to the ℓ_1-penalized least squares problem:

$$\hat{\beta}_{\text{LASSO}} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ \frac{1}{2} \|Y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\} \tag{1}$$

The geometry of the ℓ_1 penalty, a diamond-shaped constraint region with sharp corners, encourages sparse solutions where many coefficient estimates are exactly zero. This sparsity is particularly valuable for interpretability in scientific fields such as genomics and finance, where identifying the "true" subset of active features is often as important as prediction itself.

However, computing the LASSO solution path for a sequence of regularization parameters (λ) can be computationally expensive, especially in high-dimensional settings. While the objective function is convex, the non-differentiability of the ℓ_1 norm at zero precludes standard gradient-based optimization. Consequently, specialized algorithms are required. Two of the most prominent approaches are *Coordinate Descent* (CD) [2], which optimizes the objective one coefficient at a time, and *Proximal Gradient Descent* (PGD) [3], which generalizes gradient descent to non-smooth functions.

To further accelerate these algorithms, *Screening Rules* have been proposed to discard inactive predictors before the optimization process begins. By leveraging the Karush-Kuhn-Tucker (KKT) optimality conditions [4], "Strong Rules" [5] can safely identify variables guaranteed to have zero coefficients, thereby solving the optimization problem on a reduced subset of predictors.

In this paper, we implement both CD and PGD algorithms from scratch in `R` to solve the LASSO problem. We further integrate Strong Sequential Screening Rules to evaluate their impact on computational efficiency. Through a comprehensive simulation study varying sample size, dimensionality, and correlation structure, we benchmark these custom implementations against the industry-standard `glmnet` package [2]. Our results highlight the trade-offs between algorithmic simplicity and speed, demonstrating that while Coordinate Descent with screening is generally superior, Proximal Gradient methods remain competitive in specific high-dimensional regimes. Code used in this paper is publicly available on a GitHub repository[1].

2 Methodology

Let us consider the LASSO problem in Equation 1, consists of a response variable $Y \in \mathbb{R}^n$ and a design matrix $X \in \mathbb{R}^{n \times p}$. Our goal is to approximate the regression function by a linear model $\mathbb{E}[Y|X = x] = x^T\beta$. We have n observation pairs (x_i, y_i) and we assume, without loss of generality, that the columns of X are standardized such that $\sum_{i=1}^n x_{ij} = 0$ and $\frac{1}{n}\sum_{i=1}^n x_{ij}^2 = 1$ for all $j = 1, \ldots, p$. It is easy to recognize that Equation 1 is of the form $f(\beta) = g(\beta) + h(\beta)$, where $g(\beta) = \frac{1}{2}\|Y - X\beta\|_2^2$ is the differentiable least-squares loss, and $h(\beta) = \lambda\|\beta\|_1$ is the convex, separable penalty.

[1] `https://github.com/vjoshy/LASSO`

2.1 Coordinate Descent

To solve Equation 1, we employ coordinate descent, which optimizes the objective function with respect to one coefficient β_j at a time while holding all others fixed. Notice that because Equation 1 can be decomposed into smooth $g(\beta)$ and non-differentiable parts $h(\beta)$, the coordinate descent algorithm is guaranteed to converge to a global optimum [6].

The objective function is strictly convex but non-differentiable at $\beta_j = 0$. By analyzing the subgradient (see Appendix for full derivation), the update rule for the j-th coefficient is given by the soft-thresholding operator:

$$\hat{\beta}_j \leftarrow \frac{S(\rho_j, \lambda)}{z_j}$$

where $\rho_j = \sum_{i=1}^{n} x_{ij}(y_i - \hat{y}_i^{(j)})$ is the partial residual correlation (with $\hat{y}_i^{(j)}$ representing the prediction excluding the j-th feature), $z_j = \sum_{i=1}^{n} x_{ij}^2$ is the normalization constant, and $S(\cdot, \lambda)$ is defined as:

$$S(u, \lambda) = \text{sign}(u) \cdot \max(|u| - \lambda, 0)$$

2.2 Proximal Gradient Descent

While coordinate descent optimizes variables sequentially, Proximal Gradient Descent (PGD) updates the entire coefficient vector β simultaneously. The PGD algorithm operates by first taking a gradient descent step on the smooth component $g(\beta)$, and then applying the proximal operator associated with the non-smooth penalty $h(\beta)$. The gradient of the smooth least-squares term is:

$$\nabla g(\beta) = X^T(X\beta - Y)$$

Given a step size $\eta > 0$ (typically chosen such that $\eta \leq 1/L$, where L is the Lipschitz constant of ∇g), the update rule at iteration k is defined as:

$$\beta^{(k+1)} = \text{prox}_{\eta h}\left(\beta^{(k)} - \eta \nabla g(\beta^{(k)})\right) \tag{2}$$

For the LASSO ℓ_1 penalty, the proximal operator $\text{prox}_{\eta\lambda\|\cdot\|_1}$ simplifies to the element-wise soft-thresholding operator derived previously. This yields the iterative update scheme also known as the Iterative Soft-Thresholding Algorithm (ISTA):

$$\beta^{(k+1)} \leftarrow S\left(\beta^{(k)} - \eta X^T(X\beta^{(k)} - Y),\ \eta\lambda\right)$$

where the soft-thresholding function $S(\cdot, \eta\lambda)$ is applied element-wise to the vector argument. Note that the thresholding level is scaled by the step size η.

2.3 Regularization Path

In practice, we do not solve the LASSO problem for a single isolated λ. Instead, we compute the solution path over a grid of decreasing regularization parameters $\lambda_1 > \lambda_2 > \cdots > \lambda_m$. Both the CD and PGD algorithms utilize warm starts to accelerate convergence along this path. Specifically, the converged solution $\hat{\beta}(\lambda_{k-1})$ from the previous step is used to initialize the optimization for the current step λ_k. Because the solution coefficients $\hat{\beta}(\lambda)$ are continuous with respect to λ, the previous solution is typically close to the global optimum for the new λ. This initialization drastically reduces the number of iterations required for the algorithm to converge compared to a "cold start" (initializing at 0). The regularization path is computed over a grid of 100 λ values on a

logarithmic scale. The maximum value $\lambda_{\max}$ is determined by the smallest regularization parameter for which all coefficients are zero:

$$\lambda_{\max} = \max_j |x_j^T Y| \tag{3}$$

The minimum value is set to $\lambda_{\min} = 0.001 \times \lambda_{\max}$, and the full sequence is defined as:

$$\lambda_k = \exp\left(\log(\lambda_{\max}) - \frac{k-1}{99}\left(\log(\lambda_{\max}) - \log(\lambda_{\min})\right)\right), \quad k = 1, \ldots, 100$$

This logarithmic spacing ensures finer resolution at smaller λ values where coefficients transition from zero to non-zero. Convergence for both CD and PGD is declared when the maximum absolute change in coefficients between iterations falls below 10^{-8}.

2.4 Strong Rules for Screening

To improve computational efficiency, particularly in high-dimensional settings where $p \gg n$, we implement *Strong Rules* for predictor screening. These rules allow us to discard predictors that are guaranteed to have zero coefficients at the solution, thereby reducing the dimensionality of the optimization problem before the algorithm begins. The screening rules are derived from the Karush-Kuhn-Tucker (KKT) optimality conditions for the LASSO (See Appendix B). A predictor j has a zero coefficient $\hat{\beta}_j = 0$ if and only if the absolute correlation between the predictor and the residual vector is less than the regularization parameter λ:

$$|\mathbf{x}_j^T (Y - X\hat{\beta})| < \lambda$$

Based on this condition and the assumption that the inner product with the residual is non-expansive with respect to λ, the *Basic Strong Rule* [5] states that we can discard the j-th predictor if:

$$|\mathbf{x}_j^T Y| < 2\lambda - \lambda_{\max} \tag{4}$$

where $\lambda_{\max}$ is the same as from Equation 3 and is the smallest λ for which all coefficients are zero.

However, in practice, we compute the LASSO solution path for a sequence of decreasing regularization parameters $\lambda_1 > \lambda_2 > \cdots > \lambda_m$. We can leverage the solution at the previous step λ_{k-1} to create a tighter bound for the current step λ_k. This is known as the *Sequential Strong Rule* [5]. Predictor j is discarded at step k if:

$$|\mathbf{x}_j^T (Y - X\hat{\beta}(\lambda_{k-1}))| < 2\lambda_k - \lambda_{k-1} \tag{5}$$

By filtering out variables that satisfy this inequality, we solve the KKT system on a much smaller subset of active variables, reducing the computational cost of each iteration.

3 Simulation Study

To evaluate the computational and statistical performance of our implementations, we conducted a comprehensive simulation study. All algorithms were implemented in R and compared against the baseline `glmnet` package. We benchmarked four custom implementations: CD, CD with Screening, PGD, and PGD with Screening. For each scenario, we performed 5-fold cross-validation (CV) to select the optimal λ that minimizes the mean squared error (MSE). All simulations were conducted on a desktop system with an AMD Ryzen 5 5600 processor (3.50 GHz, 6 cores, 12 logical processors) and 48 GB RAM, running Windows 11.

3.1 Data Generation

We generated synthetic datasets according to the Gaussian linear model $Y = X\beta + \epsilon$. For each replication, the rows of the design matrix X were drawn from a multivariate normal distribution $N(0, \Sigma)$. Datasets were simulated across varying parameters:

1. Sparsity (s): Proportion of non-zero coefficients, $s \in \{0.2, 0.4, 0.8\}$.
2. Signal-to-Noise Ratio (SNR): Levels set to $\{1.0, 3.0\}$.
3. Sample Size (n): $n \in \{100, 300, 500\}$.
4. Dimensionality (p): $p \in \{10, 50, 200\}$.
5. Correlation Structure (Σ): Independent vs. Block Diagonal.

The covariance structures are defined as follows: for the independent case, $\Sigma = I_p$ (uncorrelated features); for the block diagonal case, features are grouped into blocks of size 10 with intra-block correlation $\rho = 0.2$, such that $\Sigma_{ij} = 0.2^{|i-j|}$ if variables i, j share a block, and 0 otherwise. Non-zero coefficients in the true coefficient vector β were drawn uniformly from $U([-2, -0.5] \cup [0.5, 2])$. The error term ϵ was drawn from $N(0, \sigma_\epsilon^2)$, with σ_ϵ scaled to maintain the specified SNR levels. The SNR is defined as the ratio of signal variance to noise variance:

$$\text{SNR} = \frac{\text{Var}(X\beta)}{\sigma_\epsilon^2}$$

For each simulated dataset, the noise standard deviation σ_ϵ is calibrated to achieve the target SNR by setting $\sigma_\epsilon = \sqrt{\text{Var}(X\beta)/\text{SNR}}$.

3.2 Performance Metrics

We evaluated the algorithms based on two criteria. Computational efficiency was measured by the total wall-clock time required to compute the full regularization path. Variable selection accuracy was assessed using the F1-score, which balances precision and recall. Precision is defined as $TP/(TP + FP)$, recall as $TP/(TP + FN)$, and the F1-score as

$$\text{F1} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},$$

where TP denotes the number of correctly identified non-zero coefficients, FP the number of zero coefficients incorrectly identified as non-zero, and FN the number of non-zero coefficients incorrectly identified as zero.

4 Results

This section presents results from our simulation study comparing CD and PGD, both with and without strong screening rules, to `glmnet`. Each method was applied to 5,400 datasets (108 scenarios $\times$ 50 replications), with all reported results averaged across the 50 replications within each scenario.

4.1 Computational Efficiency

Figure 1 illustrates the mean wall-clock time required to compute the full regularization path across varying dimensions (p) and sample sizes (n).

As expected, the computational cost increases with dimensionality for all methods. However, the implementation of Strong Rules (orange and blue lines) yields a reduction in runtime compared to the basic implementations. This speedup is most pronounced in the high-dimensional setting ($p = 200$), where the screening rules effectively discard a large majority of inactive predictors before the optimization step.

Comparing the two optimization algorithms, our results show a mixed performance profile. In lower dimensions ($p = 10, 50$), CD generally outperforms PGD. However, in the high-dimensional setting ($p = 200$), particularly with screening enabled, the performance gap narrows, with PGD Screen occasionally matching or slightly edging out CD Screen in efficiency. The baseline `glmnet` (green line) remains the fastest due to its Fortran backend.

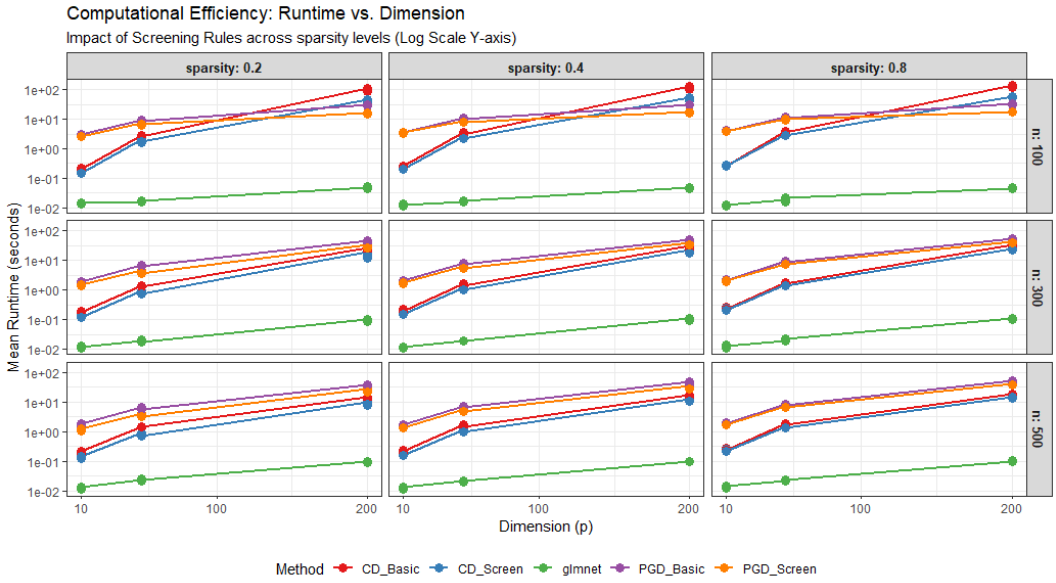

Figure 1: Mean runtime (log scale) vs. dimensionality (p) for different sparsity levels. Screening rules reduce computation time compared to basic implementations.

4.2 Statistical Performance

To assess the quality of the selected models, we examined Precision, Recall, and F1-scores across varying sample sizes and dimensions. Figure 2 displays the performance comparison.

We observe that all methods, including the baseline `glmnet`, achieve nearly identical scores for any given scenario. This confirms that our custom implementations converge to the correct global optimum. As seen in Figure 2, increasing n from 100 to 500 drastically improves all metrics. Performance degrades as p increases; at $p = 200$ with $n = 100$, the models struggle to identify the true support. Interestingly, in this specific high-dimensional, low-sample regime, PGD (both basic and screened) shows a slight advantage in F1-score over the CD implementations.

Figure 3 illustrates the effect of the Signal-to-Noise Ratio (SNR). A lower SNR (1.0, light blue) is expected to consistently reduce the F1-score compared to the high-SNR case (3.0, dark blue). While this degradation is not apparent at $p = 10$ and $p = 50$ but is severe at $p = 200$.

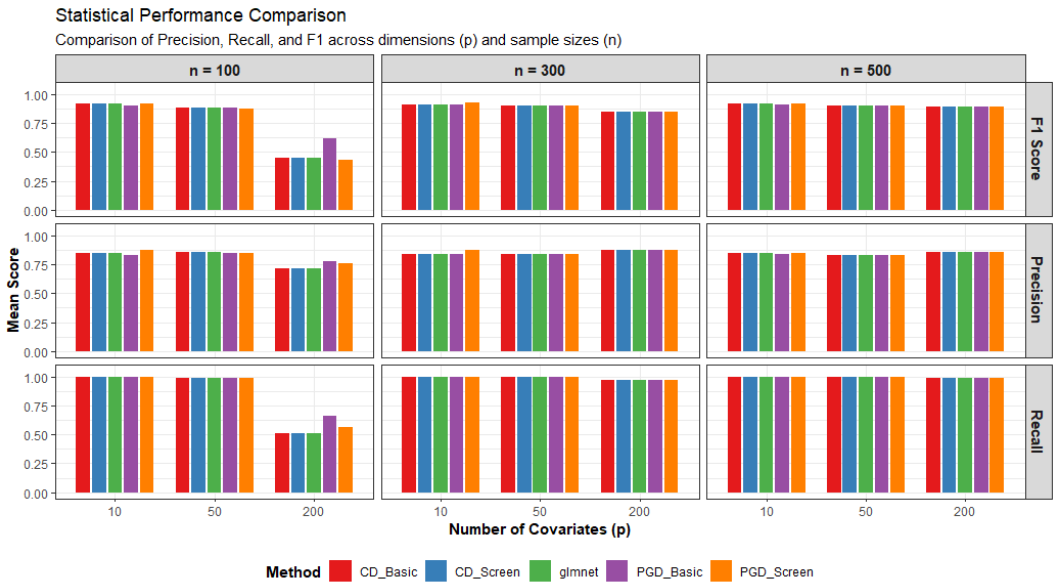

Figure 2: Statistical Performance (Precision, Recall, F1) vs. Number of Covariates (p) across sample sizes (n). Performance improves markedly with larger n. In the hardest scenario ($p = 200, n = 100$), PGD shows slightly higher F1 scores.

Regarding feature correlation, Figure 4 demonstrates that the impact of block-diagonal correlation was less detrimental than hypothesized. The performance (F1-Score) remains largely similar between the Block Diagonal (Red) and Independent (Blue) structures across all complexity levels in our simulation settings.

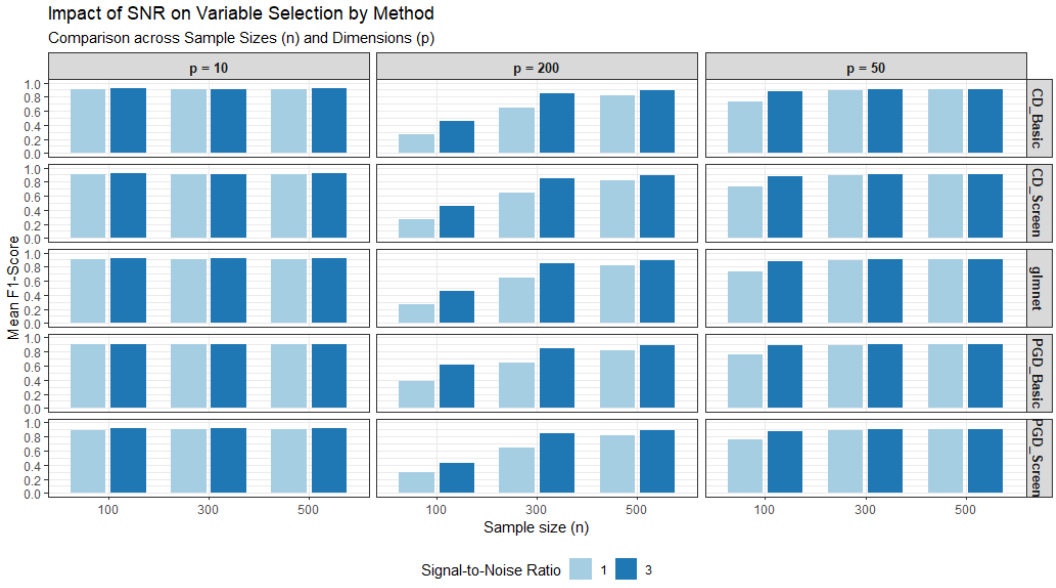

Figure 3: Impact of Signal-to-Noise Ratio (SNR) on F1-Score. Higher noise (SNR=1.0) hampers recovery of the true support in high dimensions.

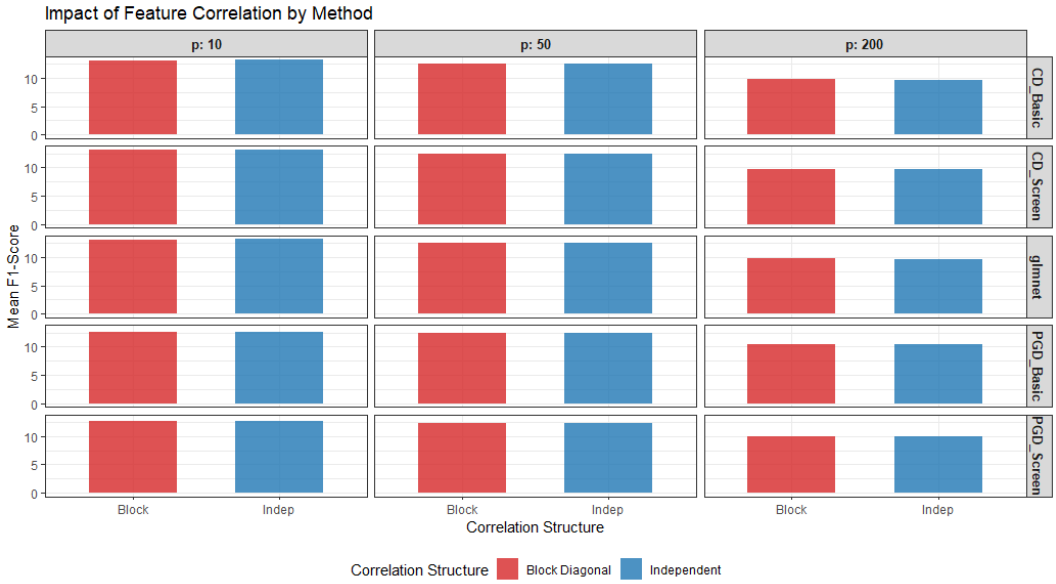

Figure 4: Effect of Correlation Structure on Variable Selection. In our simulation settings, the block diagonal structure showed comparable performance to the independent case.

4.3 Comparison with `glmnet`

Table 1 presents the summary metrics grouped by dimensionality, where test set MSE computed using the CV-selected λ, wall-clock time and F-1 score are presented.

Table 1: Comparison of Performance across Dimensionality (p). Results are presented as Mean (SD). Screening rules provide massive runtime reductions at $p = 200$ (e.g., 51.2s to 26.3s for CD).

Dimension	Method	Test MSE	Runtime (s)	F1-Score
$p = 10$	`glmnet` (Baseline)	5.76 (4.96)	0.01 (0.01)	0.73 (0.17)
	CD Basic	5.76 (4.96)	0.20 (0.04)	0.73 (0.17)
	CD Screen	5.76 (4.96)	0.17 (0.05)	0.73 (0.17)
	PGD Basic	5.77 (4.94)	2.42 (0.91)	0.70 (0.19)
	PGD Screen	5.75 (4.93)	2.16 (0.98)	0.70 (0.19)
$p = 50$	`glmnet` (Baseline)	36.4 (31.2)	0.02 (0.01)	0.69 (0.16)
	CD Basic	36.4 (31.2)	1.96 (0.82)	0.69 (0.16)
	CD Screen	36.4 (31.2)	1.47 (0.70)	0.69 (0.16)
	PGD Basic	36.1 (30.9)	7.76 (2.05)	0.69 (0.16)
	PGD Screen	36.2 (30.9)	6.19 (2.04)	0.68 (0.16)
$p = 200$	`glmnet` (Baseline)	207.0 (155)	0.08 (0.03)	0.53 (0.24)
	CD Basic	207.0 (155)	51.2 (45.4)	0.53 (0.24)
	CD Screen	207.0 (155)	**26.3 (17.6)**	0.53 (0.24)
	PGD Basic	203.0 (152)	40.8 (8.92)	0.58 (0.22)
	PGD Screen	205.0 (154)	**26.5 (9.11)**	0.55 (0.23)

5 Discussion

In this study, we implemented and benchmarked two fundamental optimization algorithms for the LASSO, Coordinate Descent and Proximal Gradient Descent, and evaluated the impact of Strong Screening Rules on their performance.

The most notable finding is the computational speedup provided by Strong Sequential Screening Rules. As shown in Table 1, the benefits of screening are negligible in low-dimensional settings ($p = 10$). However, as dimensionality increases, the advantages become pronounced. In the high-dimensional scenario ($p = 200$), screening reduced the runtime of CD by approximately 49% (from 51.2s to 26.3s) and PGD by 35% (from 40.8s to 26.5s). This confirms that for sparse problems, the overhead of the screening step is far outweighed by the savings gained from solving a smaller linear system.

Contrary to standard theoretical expectations that CD is universally superior for the LASSO [2], our results showed a competitive performance from PGD in high dimensions. While CD was notably faster at $p = 10$ and $p = 50$, PGD with Screening (26.5s) matched the runtime of CD with Screening (26.3s) at $p = 200$. Furthermore, in terms of statistical accuracy (F1-score), PGD slightly outperformed CD in the most difficult scenarios ($p = 200$). This suggests that while CD exploits separability effectively, the global gradient updates of PGD may offer robustness in high-dimensional, low-sample settings.

Critically, our custom implementations achieved mean squared error almost identical to the `glmnet` package across all scenarios, validating the numerical precision of our solvers. The simulation also highlighted the sensitivity of the LASSO to noise; performance degraded under low Signal-to-Noise Ratios (SNR=1.0). The block-diagonal correlation structure did not yield worse performance than the independent case in our specific setup. This might be because of the low intra-block correlation ($\rho = 0.2$) chosen for this simulation framework, a higher magnitude of correlation ($\rho > 0.5$) might lead to worse performance.

While our implementations are algorithmically sound, they remain slower than `glmnet` (e.g., 26.3s vs. 0.08s at $p = 200$). This discrepancy is primarily due to the interpreted nature of R versus `glmnet`'s optimized Fortran backend, although vectorized `R` operations might close some of this gap. Overall, we successfully demonstrated that implementing Strong Sequential Screening Rules accelerates LASSO optimization without compromising statistical accuracy. Among the tested algorithms, Coordinate Descent with Screening proved to be the most efficient approach for lower dimensions, while PGD with Screening remained highly competitive in high-dimensional settings.

References

[1] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. ISSN: 00359246. URL: http://www.jstor.org/stable/2346178.

[2] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Regularization Paths for Generalized Linear Models via Coordinate Descent". In: *Journal of Statistical Software* 33.1 (2010). ISSN: 1548-7660. DOI: 10.18637/jss.v033.i01.

[3] Amir Beck and Marc Teboulle. "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems". In: *SIAM Journal on Imaging Sciences* 2.1 (2009), pp. 183–202. DOI: 10.1137/080716542.

[4] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer, 2009. URL: http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

[5] Robert Tibshirani et al. "Strong Rules for Discarding Predictors in Lasso-Type Problems". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 74.2 (Mar. 1, 2012), pp. 245–266. ISSN: 1369-7412, 1467-9868. DOI: 10.1111/j.1467-9868.2011.01004.x.

[6] Paul Tseng. "Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization". In: *Journal of Optimization Theory and Applications* 109.3 (2001), pp. 475–494. DOI: 10.1023/a:1017501703105.

A Derivation of Coordinate Descent Update

The LASSO objective function is given by:

$$f(\beta) = \frac{1}{2}\|Y - X\beta\|_2^2 + \lambda\|\beta\|_1 \tag{6}$$

This objective is strictly convex (provided columns of X are in general position) but non-differentiable due to the ℓ_1 penalty term $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$. However, the non-smooth part is separable, allowing us to minimize the function with respect to one coordinate β_j at a time while fixing the others.

A.1 Separating the Objective

Let us isolate the terms involving the j-th coefficient β_j. We can rewrite the residual vector $Y - X\beta$ by separating the contribution of the j-th feature x_j:

$$\begin{aligned} Y - X\beta &= Y - \sum_{k=1}^{p} x_k\beta_k \\ &= Y - \sum_{k\neq j} x_k\beta_k - x_j\beta_j \\ &= r_j - x_j\beta_j \end{aligned}$$

where $r_j = Y - \sum_{k\neq j} x_k\beta_k$ represents the *partial residual* calculated without the contribution of predictor j.

Substituting this back into the objective function and discarding terms that are constant with respect to β_j, the coordinate-wise optimization problem becomes:

$$\begin{aligned} L(\beta_j) &= \frac{1}{2}\|r_j - x_j\beta_j\|_2^2 + \lambda|\beta_j| \\ &= \frac{1}{2}(r_j - x_j\beta_j)^T(r_j - x_j\beta_j) + \lambda|\beta_j| \\ &= \frac{1}{2}\left(r_j^T r_j - 2x_j^T r_j\beta_j + \beta_j x_j^T x_j\beta_j\right) + \lambda|\beta_j| \end{aligned}$$

A.2 Subgradient Optimality Condition

Since $L(\beta_j)$ is non-differentiable at 0, we use the subgradient optimality condition. A value $\hat{\beta}_j$ minimizes $L(\beta_j)$ if and only if $0 \in \partial L(\hat{\beta}_j)$, where ∂L is the subdifferential.

The derivative of the smooth quadratic part is $\frac{\partial}{\partial\beta_j}\left(\frac{1}{2}\beta_j^2 x_j^T x_j - \beta_j x_j^T r_j\right) = (x_j^T x_j)\beta_j - x_j^T r_j$. The subgradient of the penalty term $\lambda|\beta_j|$ is λs, where $s \in \text{sign}(\beta_j)$. The sign function is set-valued at 0:

$$\partial|\beta_j| = \begin{cases} \{1\} & \text{if } \beta_j > 0 \\ \{-1\} & \text{if } \beta_j < 0 \\ [-1, 1] & \text{if } \beta_j = 0 \end{cases}$$

Setting the total subgradient to zero:

$$(x_j^T x_j)\beta_j - x_j^T r_j + \lambda s = 0 \tag{7}$$

Let $z_j = x_j^T x_j$ (the normalization constant) and $\rho_j = x_j^T r_j$ (the partial correlation). We analyze the solution in three cases based on the sign of β_j.

Case 1: $\hat{\beta}_j > 0$ In this case, $\text{sign}(\hat{\beta}_j) = 1$. Equation 7 becomes:

$$z_j \hat{\beta}_j - \rho_j + \lambda = 0$$
$$\hat{\beta}_j = \frac{\rho_j - \lambda}{z_j}$$

For this solution to be consistent with the assumption $\hat{\beta}_j > 0$, we must have $\rho_j > \lambda$.

Case 2: $\hat{\beta}_j < 0$ In this case, $\text{sign}(\hat{\beta}_j) = -1$. Equation 7 becomes:

$$z_j \hat{\beta}_j - \rho_j - \lambda = 0$$
$$\hat{\beta}_j = \frac{\rho_j + \lambda}{z_j}$$

For consistency with $\hat{\beta}_j < 0$, we must have $\rho_j < -\lambda$.

Case 3: $\hat{\beta}_j = 0$ If $\hat{\beta}_j = 0$, the optimality condition requires that there exists some $s \in [-1, 1]$ such that $-\rho_j + \lambda s = 0$, which implies $s = \rho_j / \lambda$. This is valid only if $|\rho_j| \leq \lambda$.

A.3 Final Update Rule

Combining the three cases, we can express the solution compactly using the soft-thresholding operator $S(u, \lambda) = \text{sign}(u) \max(|u| - \lambda, 0)$:

$$\hat{\beta}_j = \frac{S(\rho_j, \lambda)}{z_j}$$

If the data is standardized such that $z_j = x_j^T x_j = 1$ (or n), the denominator simplifies accordingly, in this paper $z_j = n$.

B Derivation of Strong Rules

The screening rules are based on the Karush-Kuhn-Tucker (KKT) optimality conditions for the LASSO problem. For the objective function in Equation 1, the KKT conditions [5] state that a coefficient vector $\hat{\beta}$ is a solution if and only if:

$$x_j^T (Y - X\hat{\beta}) = \lambda s_j, \quad \forall j = 1, \ldots, p \tag{8}$$

where s_j is the subgradient of the ℓ_1 norm at $\hat{\beta}_j$:

$$s_j \in \begin{cases} \{\text{sign}(\hat{\beta}_j)\} & \text{if } \hat{\beta}_j \neq 0 \\ [-1, 1] & \text{if } \hat{\beta}_j = 0 \end{cases}$$

From these conditions, we see that if the absolute correlation between predictor j and the residual is strictly less than λ, the coefficient must be zero:

$$|x_j^T (Y - X\hat{\beta})| < \lambda \implies \hat{\beta}_j = 0 \tag{9}$$

B.1 Basic Strong Rule

To derive a screening rule before solving for $\hat{\beta}$, we assume the "unit slope" bound, which posits that the inner product of a predictor with the residual is non-expansive with respect to λ. That is, for any two regularization parameters $\lambda_k < \lambda_{k-1}$:

$$|x_j^T(Y - X\hat{\beta}(\lambda_k)) - x_j^T(Y - X\hat{\beta}(\lambda_{k-1}))| \leq |\lambda_k - \lambda_{k-1}|$$

We want to discard variable j at step λ_k. By the triangle inequality:

$$|x_j^T(Y - X\hat{\beta}(\lambda_k))| \leq |x_j^T(Y - X\hat{\beta}(\lambda_{k-1}))| + |x_j^T(Y - X\hat{\beta}(\lambda_k)) - x_j^T(Y - X\hat{\beta}(\lambda_{k-1}))|$$

Using the non-expansive bound:

$$|x_j^T(Y - X\hat{\beta}(\lambda_k))| \leq |x_j^T(Y - X\hat{\beta}(\lambda_{k-1}))| + (\lambda_{k-1} - \lambda_k)$$

For variable j to be discarded (zero coefficient), we need the LHS to be less than λ_k (from Eq. 9). Therefore, a sufficient condition to guarantee $\hat{\beta}_j(\lambda_k) = 0$ is:

$$|x_j^T(Y - X\hat{\beta}(\lambda_{k-1}))| + (\lambda_{k-1} - \lambda_k) < \lambda_k$$

Rearranging terms yields the **Sequential Strong Rule**:

$$|x_j^T(Y - X\hat{\beta}(\lambda_{k-1}))| < 2\lambda_k - \lambda_{k-1}$$

Any predictor j satisfying this inequality can be safely removed from the optimization at step k.