Optimization in Data Analysis - Project R3

Michal Gromadzki

June 2024

1 Introduction

Consider the linear regression model, we aim to predict an outcome **y** based on some input variables **X**. The model looks like this:

$$\mathbf{y} = \mathbf{X}\beta + e$$

where **y** is what we're trying to predict, **X** is our input data, β represents the model's parameters, and e is the error term. To find the best β we solve:

$$\min_{\beta} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2, \quad \text{subject to} \quad \|\beta\|_0 \leq k,$$

Here, $\|\beta\|_0$ is the number of non-zero elements in β, and k limits how many parameters can be non-zero. This keeps the model simple and prevents overfitting.

Since this problem can be tough to solve directly, we often use an approximation:

$$\min_{\beta} \left(\frac{1}{2}\|y - X\beta\|_2^2 + \gamma\|\beta\|_1 \right)$$

where γ is a penalty parameter that controls the balance between fitting the data well and keeping the model simple. This version is easier to handle because it's a convex optimization problem. One effective way to solve it is the 'Pathwise Coordinate' method, which updates the parameters iteratively. This approach helps us find a model that is both accurate and not overly complex.

2 Pathwise Coordinate optimization

Pathwise Coordinate optimization is an iterative method used for solving certain convex optimization problems, particularly in the context of L1-penalized regression (lasso). The method works by optimizing one coordinate - variable or parameter at a time while keeping the other coordinates fixed. This process is repeated in a cyclic manner until convergence.

2.1 Derivation

Consider the lasso for regularized regression, We have predictors x_{ij}, $j = 1, 2, \ldots, p$, and outcomes values y_i for the ith observation, for $i = 1, 2, \ldots, n$. Assume that the x_{ij} are standardized so that $\sum_i x_{ij}/n = 0$, $\sum_i x_{ij}^2 = 1$. The lasso solves

$$\min_{\beta} \frac{1}{2} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2, \text{subject to} \sum_{j=1}^{p} |\beta_j| \leq s \tag{1}$$

The bound s is a user-specified parameter, often chosen by a model selection procedure. Equivalently, the solution to 1 also minimizes the "Lagrange" version of the problem

$$f(\beta) = \frac{1}{2} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \gamma \sum_{j=1}^{p} |\beta_j|, \tag{2}$$

where $\gamma \geq 0$. There is one-to-one correspondence between γ and the bound s - if $\hat{\beta}(\gamma)$ minimizes 2, than it also solves 1 with $s = \sum_{j=1}^{p} |\hat{\beta}_j(\gamma)|$.

With a single predictor, the lasso solution is a soft-thresholded version of the least squares estimate $\hat{\beta}$:

$$\hat{\beta}^{lasso}(\gamma) = \begin{cases} \hat{\beta} - \gamma, & \text{if } \hat{\beta} > 0 \text{ and } \gamma < |\hat{\beta}| \\ \hat{\beta} + \gamma, & \text{if } \hat{\beta} < 0 \text{ and } \gamma < |\hat{\beta}| \\ 0, & \text{if } \gamma \geq |\hat{\beta}| \end{cases}$$

This expression arises because the convex-optimization problem 2 reduces to a few special cases when there is a single predictor. Minimizing the criterion 2 with a single standardized x and β simplifies in the following way:

The goal is to simplify the residual sum of squares:

$$\sum_{i=1}^{n} (y_i - x_i\beta)^2 \tag{3}$$

First, recognize that the least squares estimate $\hat{\beta}$ for β is given by:

$$\hat{\beta} = \sum_{i=1}^{n} x_i y_i \tag{4}$$

since it is assumed that x_i is standardized.

Now rewrite 3 using the least squares estimate $\hat{\beta}$:

$$\sum_{i=1}^{n} (y_i - x_i\beta)^2 = \sum_{i=1}^{n} \left(y_i^2 - 2y_i x_i \beta + x_i^2 \beta^2\right)$$

Given the standardization of x_i ($\sum_i x_i^2 = 1$), expression simplifies:

$$\sum_{i=1}^{n} \left(y_i^2 - 2y_i x_i \beta + x_i^2 \beta^2\right) = \sum_{i=1}^{n} y_i^2 - 2\beta \sum_{i=1}^{n} y_i x_i + \beta^2 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i^2 - 2\beta \sum_{i=1}^{n} y_i x_i + \beta^2$$

Since $\hat{\beta} = \sum_{i=1}^{n} x_i y_i$, the expression simplifies to:

$$\sum_{i=1}^{n} y_i^2 - 2\beta \sum_{i=1}^{n} y_i x_i + \beta^2 = \sum_{i=1}^{n} y_i^2 - 2\beta\hat{\beta} + \beta^2$$

The term $\sum_{i=1}^{n} y_i^2$ is constant with respect to β, so it can be ignored for the purpose of the optimization. Thus, the objective function simplifies to:

$$\min_{\beta} \frac{1}{2} \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \gamma|\beta| = \min_{\beta} \frac{1}{2} \left(\beta^2 - 2\beta\hat{\beta}\right) + \gamma|\beta| = \min_{\beta} \frac{1}{2} \left(\beta - \hat{\beta}\right)^2 + \gamma|\beta|$$

The last step was done by completing the square:

$$\frac{1}{2} \left(\beta^2 - 2\beta\hat{\beta}\right) = \frac{1}{2} \left(\beta - \hat{\beta}\right)^2 - \frac{1}{2}\hat{\beta}^2$$

Again, the term $-\frac{1}{2}\hat{\beta}^2$ is a constant and can be ignored for optimization purposes, leading to:

$$\min_{\beta} \frac{1}{2} \left(\beta - \hat{\beta}\right)^2 + \gamma|\beta|, \text{ where } \hat{\beta} = \sum_{i=1}^{n} x_i y_i \tag{5}$$

We can differentiate 5 to get:

$$\frac{df}{d\beta} = \beta - \hat{\beta} + \gamma = 0$$

This leads to solution $\beta = \hat{\beta} - \gamma$ as long as $\hat{\beta} > 0$ and $\gamma < \hat{\beta}$, otherwise 0 is the minimizing solution. Similarly, if $\hat{\beta}$ < 0, if $\gamma < -\hat{\beta}$, then the solution is $\beta = \hat{\beta} + \gamma$, else 0.

With multiple predictors consider a simple iterative algorithm that applies soft-thresholding with a "partial residual" as a response variable. We write 2 as:

$$f(\tilde{\beta}) = \frac{1}{2} \sum_{i=1}^{n} \left(y_i - \sum_{k \neq j} x_{ik} \tilde{\beta}_k - x_{ij}\beta_j\right)^2 + \gamma \sum_{k \neq j} |\tilde{\beta}_j| + \gamma|\beta_j| \tag{6}$$

where all the values of β_k for $k \neq j$ are held fixed at values $\tilde{\beta}_k(\gamma)$. Minimizing w.r.t β_j, we get

$$\tilde{\beta}_j(\gamma) \leftarrow S\left(\sum_{i=1}^{n} x_{ij}(y_i - \tilde{y}_i^{(j)}), \gamma\right) \tag{7}$$

where $S(z, \gamma) = \text{sign}(z)\max(|z| - \gamma, 0)$ is the soft-thresholding function. The update 7 is repeated for $j = 1, 2, \ldots, p, 1, 2, \ldots$ until convergence. An equivalent form of this update is

$$\tilde{\beta}_j(\gamma) \leftarrow S\left(\tilde{\beta}_j(\gamma) + \sum_{i=1}^{n} x_{ij}(y_i - \tilde{y}_i), \gamma\right) \tag{8}$$

2.2 Key Steps in Pathwise Coordinate Optimization

1. **Initialization:** Start with initial values for the coefficients $\tilde{\beta}_j(\gamma)$, which could be set to zeros, random values, or the least-squares estimates.
2. **Cycle Through Coordinates:** For each coordinate β_j:
   - Update $\tilde{\beta}_j$ by minimizing the objective function with respect to β_j while holding the other β_k, for $k \neq j$ fixed.
   - Apply the soft-thresholding operator to update $\tilde{\beta}_j(\gamma)$:

     $$\tilde{\beta}_j(\gamma) \leftarrow S\left(\tilde{\beta}_j(\gamma) + \sum_{i=1}^{n} x_{ij}(y_i - \tilde{y}_i), \gamma\right)$$

     where $S(z, \gamma) = \text{sign}(z)\max(|z| - \gamma, 0)$ is the soft-thresholding function and $\tilde{y}_i$ is the current prediction of y_i.
3. **Repeat Until Convergence:** Continue cycling through the coordinates until the updates to $\tilde{\beta}_j(\gamma)$ are smaller than a predefined threshold, indicating convergence.

3 Implementation

The Pathwise Coordinate algorithm has been implemented as a class in Python. It has the following attributes:

- lam - The regularization parameter γ, which controls the strength of the L1 penalty
- tol - The tolerance level for convergence, determining when to stop the iterative process
- max_iter - The maximum number of iterations allowed for the algorithm to run
- $weights$ - The β parameters for the regression

It also has the following methods:

- $fit(X, y, shuffle)$ - This method uses the Pathwise Coordinate optimization algorithm to fit the regression model to the data. Returns the number of run iterations and a boolean value whether the algorithm has converged.
- $predict(X)$ - This method predicts the output for given input data X using the learned weights.
- $get\_params()$ - This method returns the learned parameters and the regularization parameter.

3.1 Modification

Moreover, a modification of the Pathwise Coordinate algorithm has been implemented and tested. In the modified version instead of iterating through columns from 1 to p. In each iteration, the order is randomly selected. This approach aims to introduce an element of randomness that can potentially improve the convergence properties of the algorithm and avoid cycles that might occur when using a fixed order.

3.2 Stopping rule

At the end of each iteration of the algorithm, the following value is computed:

$$diff = norm(coef - prev\_coef)$$

where:

- norm - The Frobenius norm
- coef - Parameters in the current iteration
- prev_coef - Parameters in the previous iteration

If the $diff$ is smaller than the pre-set tol value the training stops.

4 Experiments

Experiments are based on the original Pathwise Coordinate optimization paper. The Pathwise Coordinate algorithm has been compared to the LassoLars implementation from sklearn package. The training data has been synthetically generated.

4.1 Data generation

Gaussian data was generated with n observations and p predictors, with each pair of predictors X_i, X_j having the same population correlation ρ. The outcome values were generated by

$$Y = \sum_{j=1}^{p} \beta_j X_j + k \cdot Z$$

where $\beta_j = (-1)^j exp(-2(j-1)/20), Z \sim N(0,1)$ and k is chooses so that the signal-to-noise ratio is 3.0. The coefficients are constructed to have alternating signs and to be exponentially decreasing.

4.2 Experiment parameters

Both algorithms have been tested on 4 pairs of sample and feature sizes described in Table 1.

n	100	100	1000	5000
p	1000	5000	100	100

Table 1: Tested sample and feature sizes

Additionally, for each pair of sample and feature sizes, 6 values of ρ have been tested - $0, 0.1, 0.2, 0.5, 0.9, 0.95$. Lastly, each experiment has been computed 15 times to achieve more stable results. All algorithms have been fitted with $\gamma = 1$, $max\_iter = 1000$ and $tol = 0.0001$

5 Analysis

Table 2 presents the mean times for all methods and all combinations of tested data.

Method	Population correlation between features					
	$n = 100, p = 1000$					
	0	0.1	0.2	0.5	0.9	0.95
coord	1.171	1.117	1.432	7.180	1.495	1.791
coord_shuffle	1.691	1.547	1.537	1.479	1.052	1.369
LARS	0.014	0.013	0.012	0.010	0.003	0.002
	$n = 100, p = 5000$					
	0	0.1	0.2	0.5	0.9	0.95
coord	25.157	26.581	28.468	103.883	51.818	27.682
coord_shuffle	30.426	31.737	33.166	29.279	14.208	16.704
LARS	0.022	0.023	0.021	0.017	0.005	0.004
	$n = 1000, p = 100$					
	0	0.1	0.2	0.5	0.9	0.95
coord	0.012	0.037	0.087	0.400	0.876	0.356
coord_shuffle	0.016	0.020	0.023	0.041	0.164	0.244
LARS	0.007	0.006	0.006	0.005	0.003	0.002
	$n = 5000, p = 100$					
	0	0.1	0.2	0.5	0.9	0.95
coord	0.023	0.107	0.256	1.335	3.564	3.532
coord_shuffle	0.027	0.043	0.051	0.096	0.421	0.736
LARS	0.009	0.008	0.008	0.007	0.005	0.004

Table 2: Results of the experiments

For the LARS method, execution time decreases as ρ increases, with times ranging from 0.002 to 0.014 seconds for n=100 and p=1000. This trend holds across all other configurations, with better performance for smaller ρ values. Pathwise Coordinate Optimization generally shows much higher execution times, indicating poorer performance, especially with larger feature sets, such as times ranging from about 25 to over 100 seconds for $n = 100$ and $p = 5000$. Pathwise Coordinate Optimization with shuffling exhibits similar but slightly lower times than the version without shuffling. Overall, LARS performs best with fewer features and more observations. At the same time, Pathwise Coordinate Optimization and its shuffling variant struggle with larger feature sets, demonstrating that higher feature correlation ρ can sometimes improve execution time. However, the impact varies by method and configuration.

5.1 Visualizations

Several visualizations have been prepared to further explore the differences between tested algorithms. Figure 1 presents the mean times for all tested methods.

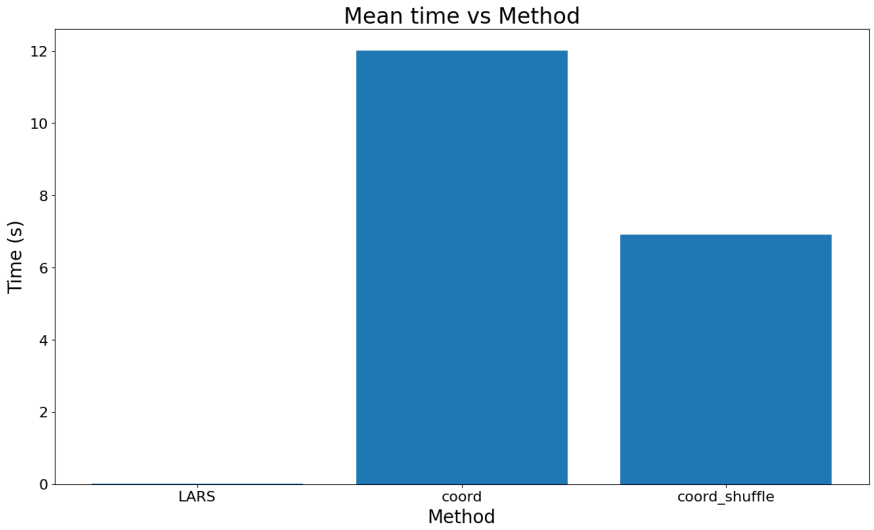


Figure 1: Mean time for each method

LARS heavily outperforms all other methods. The slowest is coord and about 40% faster is coord_shuffle. Figure 2 depicts the times of execution of the methods with respect to $\%$ of non-zero parameters. Note that time is on a logarithmic scale.

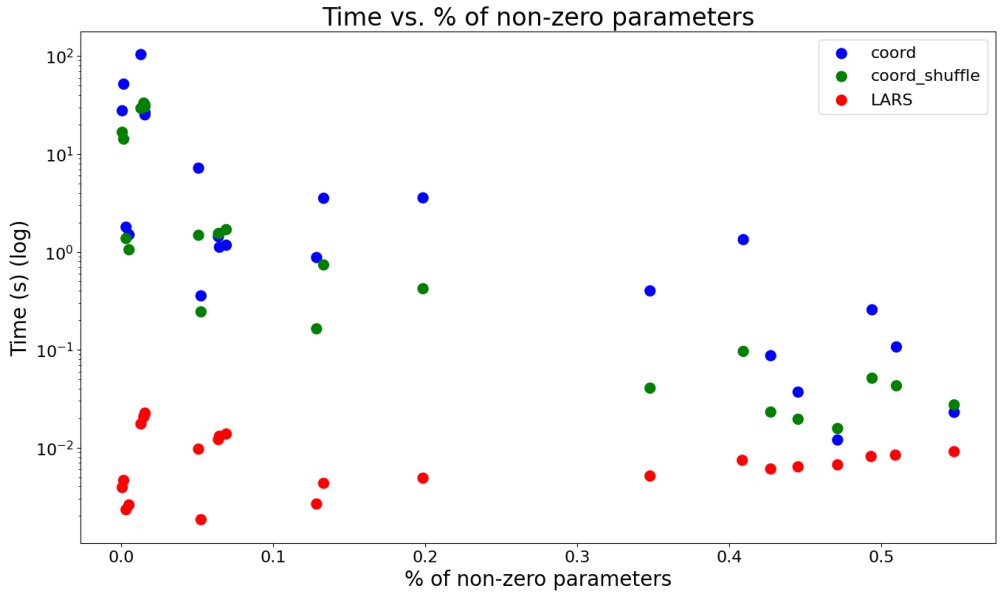


Figure 2: Time vs $\%$ of non-zero parameters

For LARS the time remains relatively constant, while for the Pathwise Coordinate Optimization methods the higher the percentage of non-zero parameters the lower the time. Figure 3 presents the same relationship. However with lines of best fit drawn and no logarithmic scale.

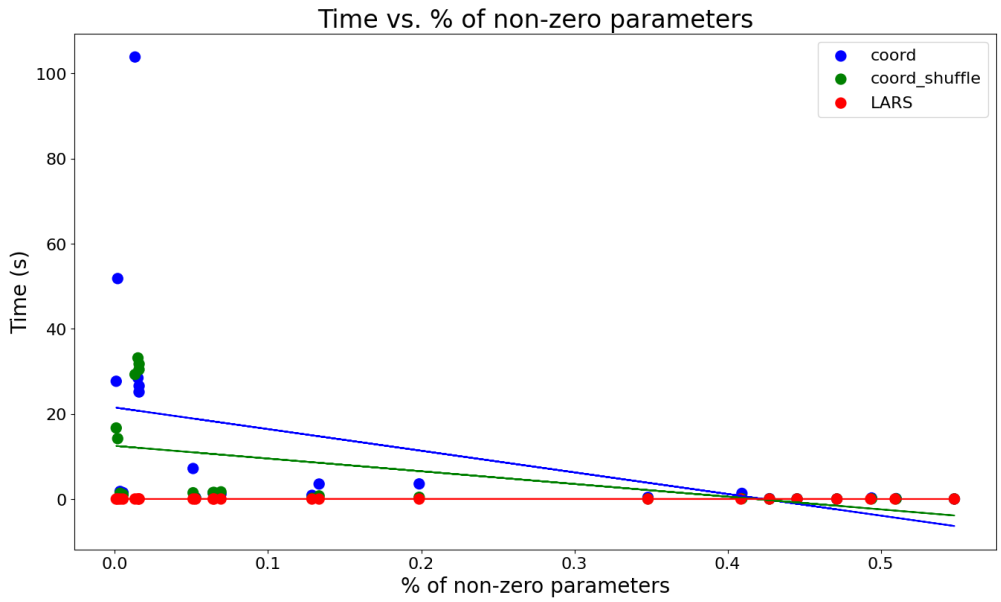
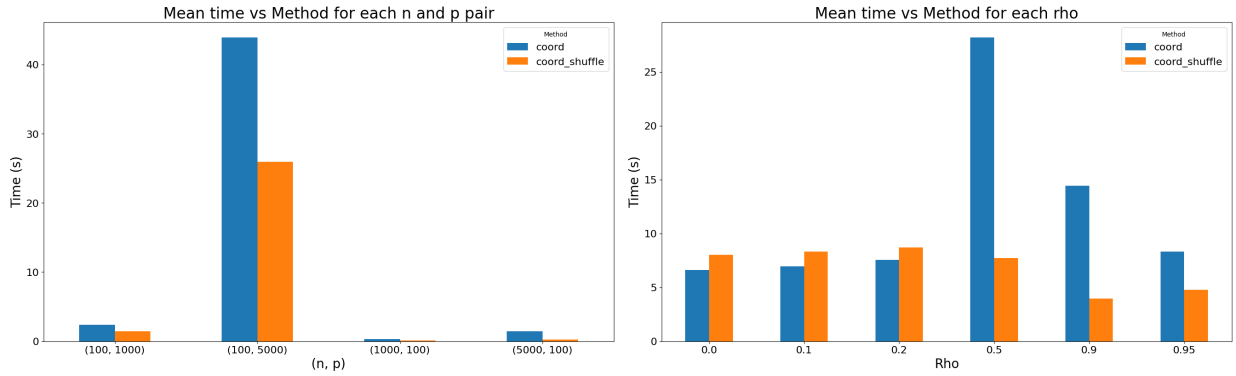


Figure 3: Time vs $\%$ of non-zero parameters

The observations are similar to the ones based on the Figure 2. Figure 4 compares the performance of both versions of the Pathwise Coordinate Optimization method.

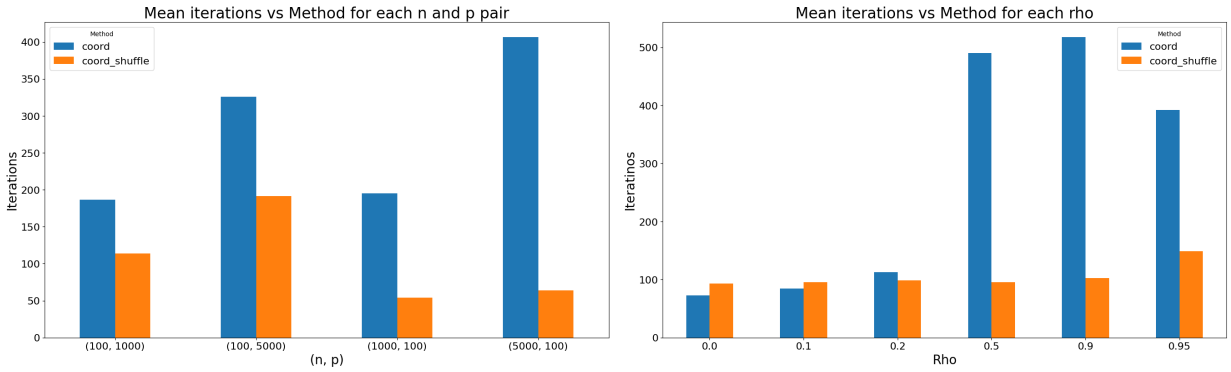


(a) Mean time vs method for each n and p pair

(b) Mean time vs method for each ρ

Figure 4: Mean time vs dataset parameters

Based on Figure 4a it is clear that the version with shuffling outperforms the version without on all dataset sizes. By Figure 5 we, again, see that the version with shuffling is performing better. Special attention should be directed at $\rho = 0.5$ where it outperforms the version without shuffling by over 3 times. Figure 5 presents the mean amount of iterations of the algorithms concerning the dataset's parameters.

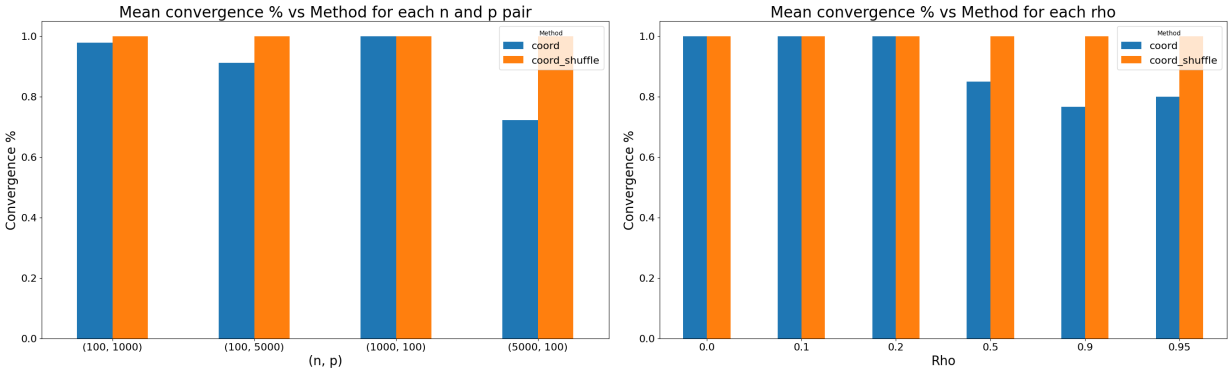


(a) Mean iterations vs method for each n and p pair

(b) Mean iterations vs method for each ρ

Figure 5: Mean iterations vs dataset parameters

As the number of iterations is directly correlated to the execution time the conclusions are going to be similar. Version with shuffling always performs less iterations, with the exception being $\rho = 0$ and $\rho = 0.1$ presented in Figure 5b. Lastly, Figure 6 depicts the mean convergence $\%$ of the algorithms with respect to the dataset's parameters.



(a) Mean convergence $\%$ vs method for each n and p pair

(b) Mean convergence $\%$ vs method for each ρ

Figure 6: Mean convergence $\%$ vs dataset parameters

Again, the version with shuffling outperforms the version without. The first method always converges, while the latter has problems with bigger datasets and higher correlations.

6 Conclusion

In conclusion, the comparison of the LARS method and Pathwise Coordinate Optimization algorithms provides valuable insights into their performance across various configurations. Notably, LARS demonstrates superior execution times, especially in scenarios with fewer features and more observations. As the correlation ρ increases, LARS exhibits decreasing execution times, suggesting that higher feature correlation can sometimes improve performance. It is worth noting that, LARS benefits from being implemented in C, which typically offers faster execution compared to Python implementations due to its lower-level nature and efficiency.

On the other hand, Pathwise Coordinate Optimization methods, particularly the version without shuffling, struggle with larger feature sets, as indicated by significantly higher execution times. However, the introduction of shuffling

improves performance, albeit marginally. The comparison across different dataset parameters consistently shows that the shuffling variant outperforms the non-shuffling one, demonstrating better convergence rates and fewer iterations required for convergence.

Visualizations further illustrate these findings, emphasizing the consistent superiority of LARS over other methods and the favorable impact of shuffling on Pathwise Coordinate Optimization. These insights are valuable for selecting the most suitable algorithm based on dataset characteristics and computational efficiency requirements.