Pathwise coordinate optimization

MICHAŁ GROMADZKI

Coordinate Descent – key points

- Used to solve lasso regression
- Iterative method
- Loops through parameters
- Optimizes one coordinate at a time
- Keeps all other coordinates frozen
- Effective for problems with sparse solutions
- Proposed in 2007

Lasso Regression

The lasso solves:

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

The solution also minimizes the Lagrange version:

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

Derivation

Assume that the x_{ij} are standardized so that:

$$\frac{1}{n}\sum_i x_{ij} = 0 \text{ and } \sum_i x_{ij}^2 = 1$$

The objective function can be written as:

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

Derivation

To update β_j, we fix β_{-j} and minimize f with respect to β_j. The partial residual for β_j is defined as:

$$r_i^{(j)} = y_i - \sum_{k \neq j} x_{ik}\beta_k$$

Thus, the objective function can be rewritten for β_j as:

$$f(\beta_j) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \sum_{k \neq j} x_{ik}\beta_k - x_{ij}\beta_j\right)^2 + \gamma|\beta_j| = \frac{1}{2}\sum_{i=1}^{n}\left(r_i^{(j)} - x_{ij}\beta_j\right)^2 + \gamma|\beta_j|$$

Derivation

The objective function can be further simplified to:

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

Note that $r_i^{(j)2}$ is constant with respect to β_j, so it can be ignored for the purpose of the optimization:

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

Derivation

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

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

Substitute $S_j = \sum_{i=1}^{n} r_i^{(j)} x_{ij}$:

$$f(\beta_j) = \frac{1}{2}\left(\beta_j^2 - 2\beta_j S_j\right) + \gamma|\beta_j|$$

Derivation

We can now differentiate to get:

$$\frac{df}{d\beta_j} = \beta_j - S_j + \gamma = 0$$

The solution to this optimization problem is given by the soft-thresholding operator:

$$\beta_j = S(S_j, \gamma), \text{ where } S \text{ is the soft-thresholding operator}$$

Derivation

Substituting back S_j and $r_i^{(j)}$, we get:

$$\beta_j = S\left(\sum_{i=1}^{n} x_{ij}(y_i - \sum_{k \neq j} x_{ik}\beta_k), \gamma\right)$$

Where S is defined as:

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

Implementation

The Pathwise Coordinate algorithm has been implemented as a class in Python.

Attributes:

- **lam** (float) - The regularization parameter
- **tol** (float)- The tolerance level for convergence
- **max_iter** (int) - Maximum number of iterations of the algorithm
- **weights** (array) – The parameters for the regression

Methods:

- **fit()** – This method uses the Pathwise Coordinate optimization algorithm to fit the regression model to the data
- **predict()** - Predicts the output for given input data X using the learned weights
- **get_params()** - This method returns the learned parameters and the regularization parameter

Stopping rule

At the end of each iteration of the algorithms 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

$$\|A\|_{\mathrm{F}} = \sqrt{\sum_{i}^{m}\sum_{j}^{n}|a_{ij}|^2}$$

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

Modification

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.

```
for i in range(X.shape[1]):
    self.weights[i] = self.update_weight(X, y, i)
```

```
idx = np.random.permutation(X.shape[1])
for i in idx:
    self.weights[i] = self.update_weight(X, y, i)
```

Experiments - data

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:

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

$$\beta_j = (-1)^j exp(-2(j-1)/20), Z \sim N(0,1)$$

Experiments

- 3 algorithms:
  - Coordinate descent
  - Coordinate descent – modification
  - LassoLars – sklearn
- 6 values of $\boldsymbol{\rho}$: 0, 0.1, 0.2, 0.5, 0.9, 0.95
- Each experiment computed 15 times

$$\gamma = 1, max\_iter = 1000 \text{ and } tol = 0.0001$$

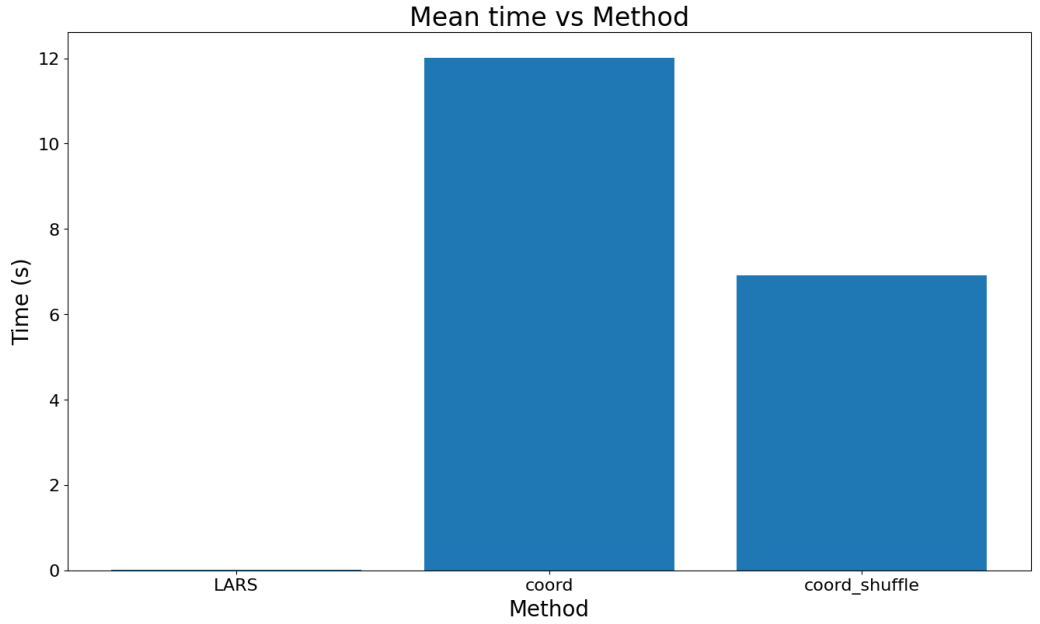
n	100	100	1000	5000
p	1000	5000	100	100

Table 1: Tested sample and feature sizes

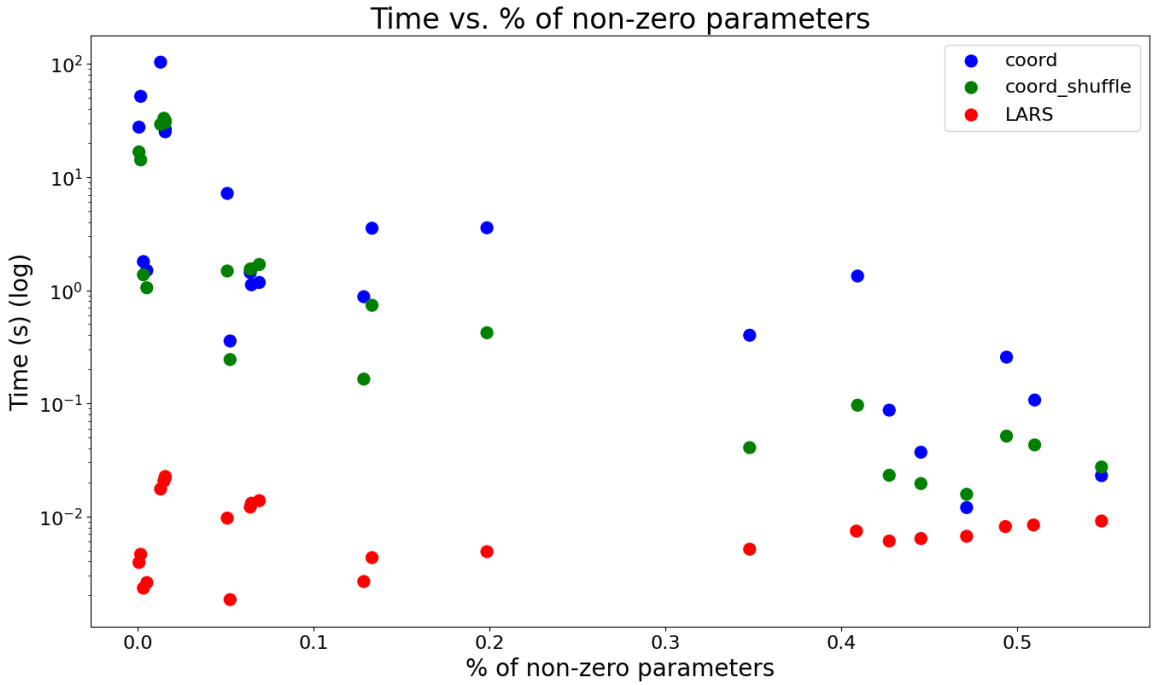
Results

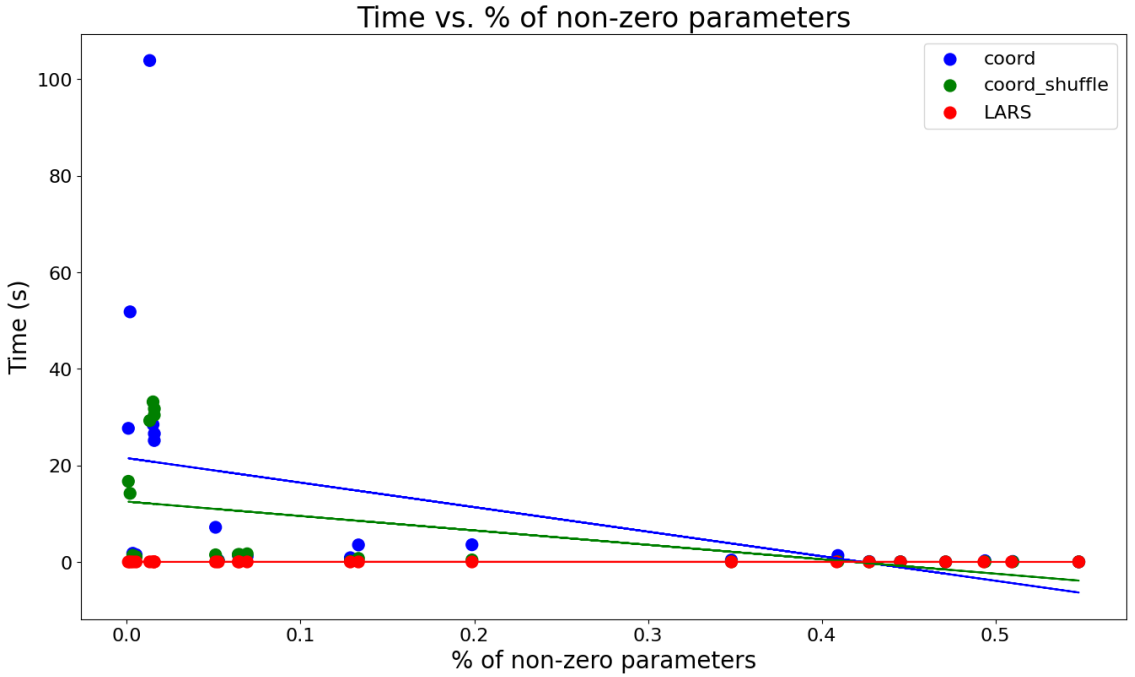
Method	Population correlation between features					
	$n = 100, p = 1000$					
	0	0.1	0.2	0.5	0.9	0.95
coord	1.171	1.117	1.432	7.180	1.495	1.791
coord_shuffle	1.691	1.547	1.537	1.479	1.052	1.369
LARS	0.014	0.013	0.012	0.010	0.003	0.002
	$n = 100, p = 5000$					
	0	0.1	0.2	0.5	0.9	0.95
coord	25.157	26.581	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

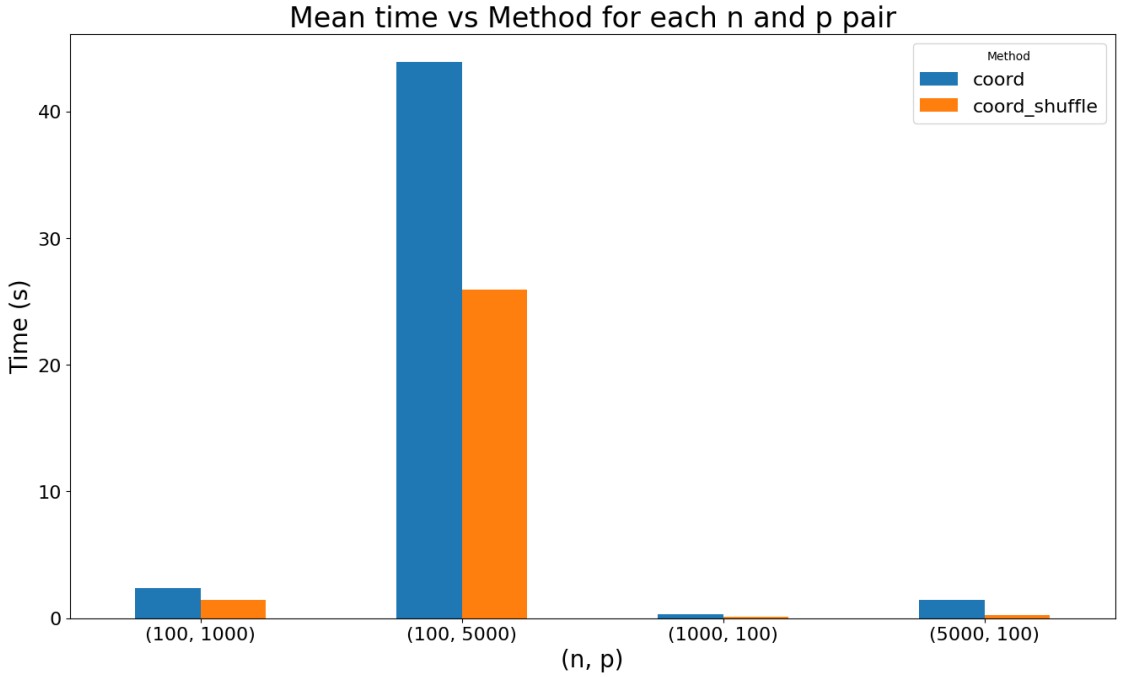


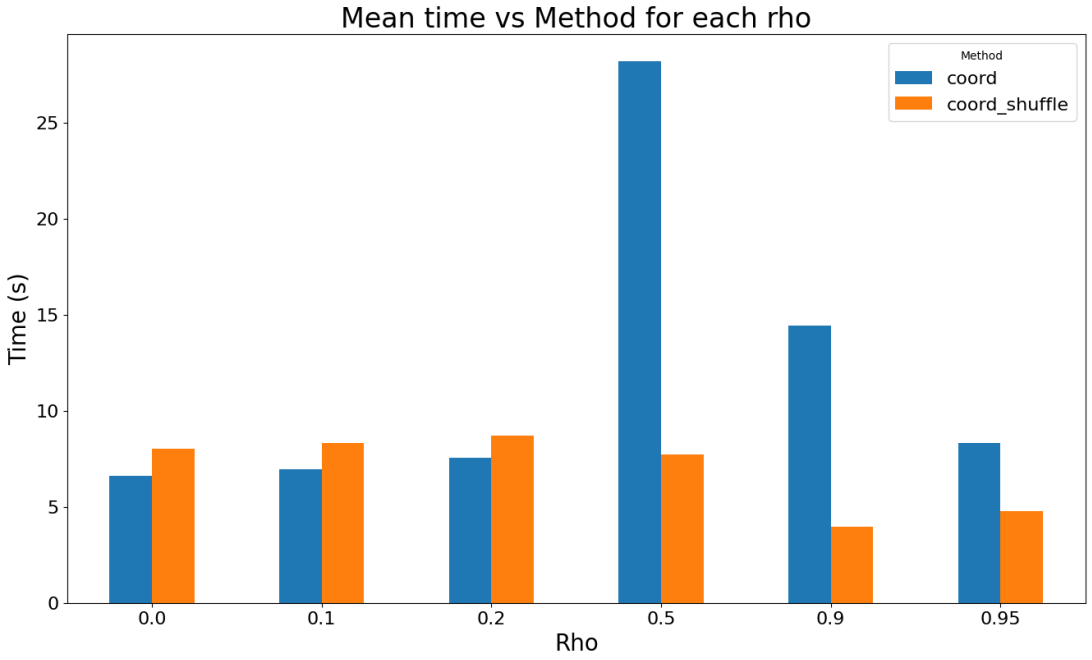
Results


Time vs. % of non-zero parameters
coord
coord_shuffle
LARS
Time (s) (log)
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
0.0
0.1
0.2
0.3
0.4
0.5
% of non-zero parameters

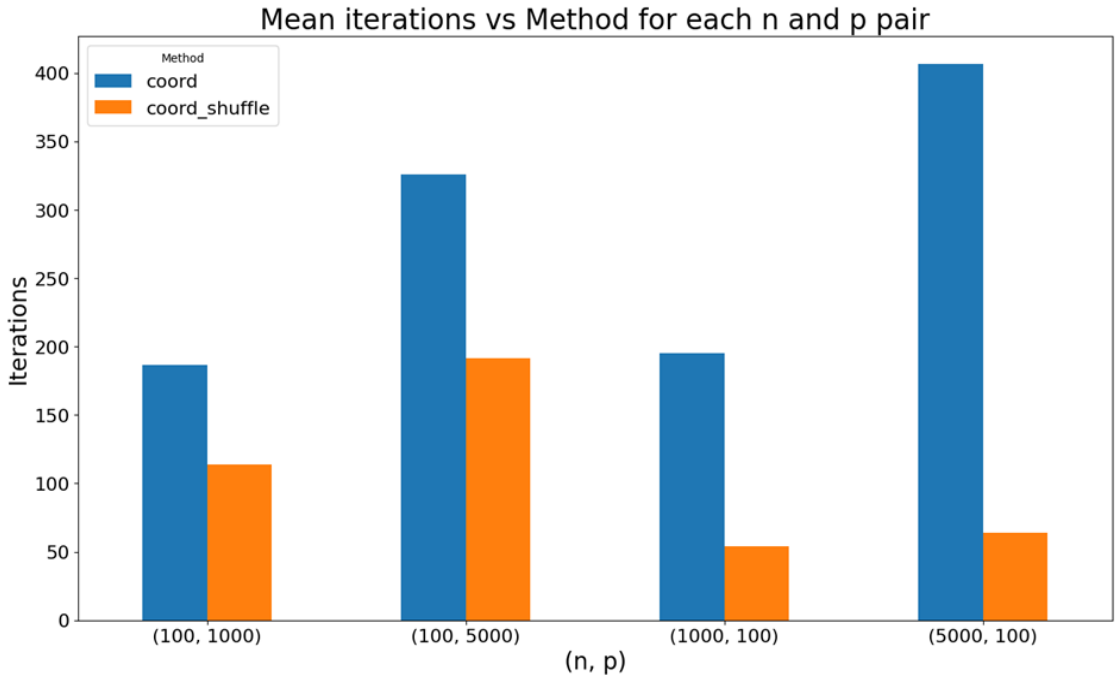

Time vs. % of non-zero parameters
coord
coord_shuffle
LARS
Time (s)
100
80
60
40
20
0
0.0
0.1
0.2
0.3
0.4
0.5
% of non-zero parameters

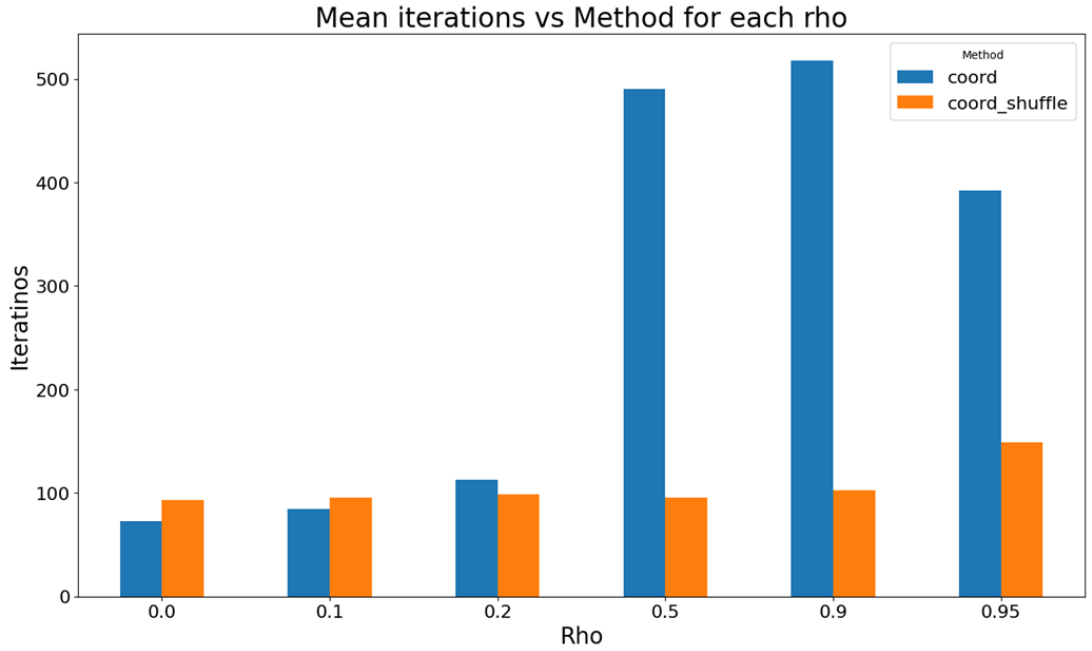
Results


Mean time vs Method for each n and p pair
Method
coord
coord_shuffle
Time (s)
0
10
20
30
40
(100, 1000)
(100, 5000)
(1000, 100)
(5000, 100)
(n, p)

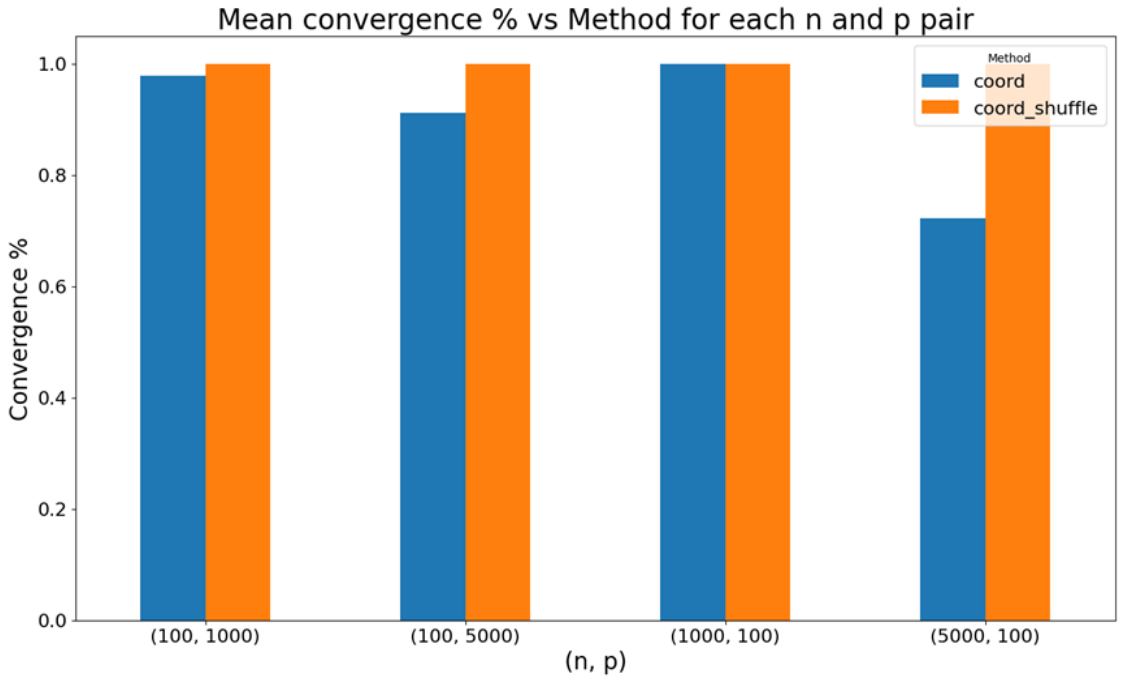

Mean time vs Method for each rho
Method
coord
coord_shuffle
Time (s)
0
5
10
15
20
25
0.0
0.1
0.2
0.5
0.9
0.95
Rho

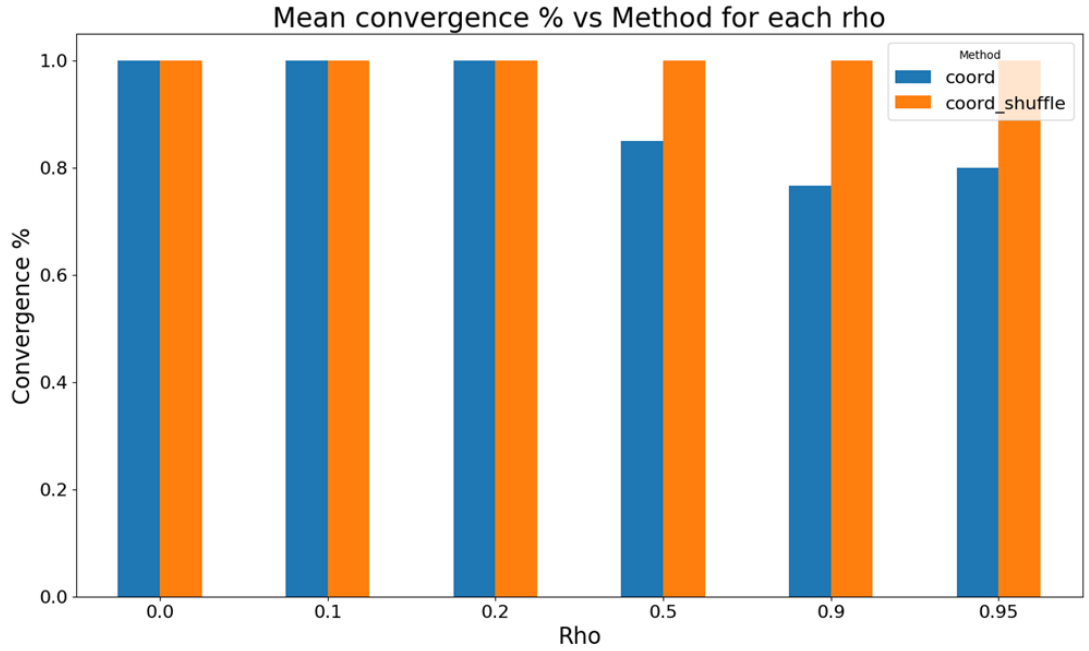
Results


Mean iterations vs Method for each n and p pair
Method
coord
coord_shuffle
Iterations
0
50
100
150
200
250
300
350
400
(100, 1000)
(100, 5000)
(1000, 100)
(5000, 100)
(n, p)


Mean iterations vs Method for each rho
Method
coord
coord_shuffle
Iteratinos
0
100
200
300
400
500
0.0
0.1
0.2
0.5
0.9
0.95
Rho

Results


Mean convergence % vs Method for each n and p pair
Convergence %
1.0
0.8
0.6
0.4
0.2
0.0
(100, 1000)
(100, 5000)
(1000, 100)
(5000, 100)
(n, p)
Method
coord
coord_shuffle


Mean convergence % vs Method for each rho
Convergence %
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.1
0.2
0.5
0.9
0.95
Rho
Method
coord
coord_shuffle

Conclusion

- It is impossible to compare times because of implementations in different languages
- Modified coordinate descent outperforms the original version in most cases
- The higher the % of non-zero parameters the lower the execution time
- The number of features has a much bigger impact on execution times than the number of samples
- LARS execution time decreases with increasing $\boldsymbol{\rho}$

Refrences

Friedman et al, The Annals of Applied Statistics, Vol. 1, 2007, pp. 302322

Least Angle Regression; Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani