

Le Lasso

La solution component wise

S.Canu
APPS ITI /INSA de Rouen Normandie

2 octobre 2023

Le Lasso

- Données

- ▶ $X \in \mathbb{R}^{n \times p}$ les p variables observées n fois
- ▶ $y \in \mathbb{R}^n$ la variable réponse

- Inconnues

- ▶ $\beta \in \mathbb{R}^p$

- hypothèses : pas de biais (pas de terme constant)

- ▶ $\text{mean}(X) = 0$
- ▶ $\|X_j\|^2 = 1$
- ▶ $\text{mean}(y) = 0$

le cout pénalisé

$$J_\lambda(\beta) = \frac{1}{2} \|X\beta - y\|^2 + \lambda \|\beta\|_1 ,$$

Les conditions d'optimalité

$$0 \in \partial J_\lambda(\beta) = X^\top(X\beta - y) + \lambda \partial(\|\beta\|_1) ,$$

La notion de chemin de régularisation

$$\min_{\beta \in \mathbb{R}^p} J_\lambda(\beta) \quad \text{avec} \quad J_\lambda(\beta) = \frac{1}{2} \|X\beta - y\|^2 + \lambda \sum_{j=1}^p |\beta_j|$$

$$\partial J(\beta) = X^\top(X\beta - y) + \lambda s(\beta) \quad \text{avec } s_j(\beta) = \begin{cases} 1 & \text{si } \beta_j > 0 \\ [-1, 1] & \text{si } \beta_j = 0 \\ -1 & \text{si } \beta_j < 0 \end{cases}$$

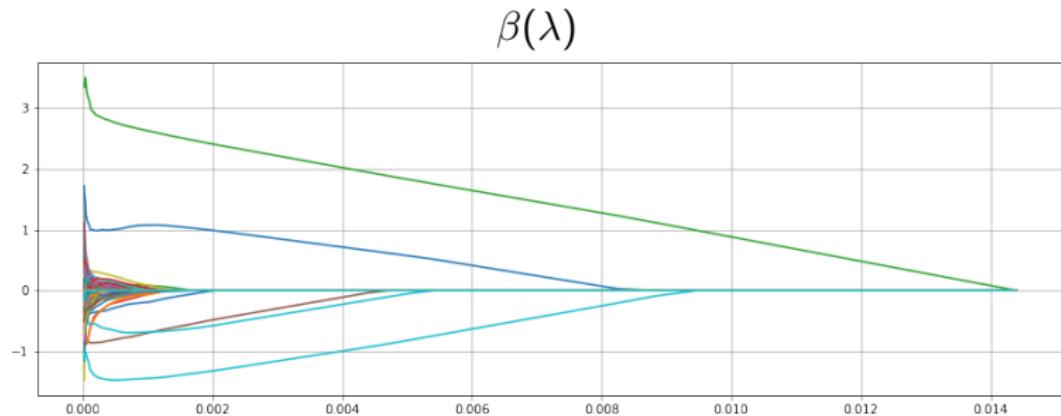
β optimal si $0 \in \partial J(\beta)$ c'est-à-dire si pour $g(\beta) = X^\top(X\beta - y)$

$$\begin{cases} g_j(\beta) + \lambda = 0 & \text{si } \beta_j > 0 \\ -\lambda \leq g \leq \lambda & \text{si } \beta_j = 0 \\ g_j(\beta) - \lambda = 0 & \text{si } \beta_j < 0 \end{cases}$$

Un cas intéressant :

$$\text{si } \lambda \geq \max_j (X^\top y)_j \quad \text{alors} \quad \beta = 0$$

La notion de chemin de régularisation



λ large $\Rightarrow \beta = 0$ so that $|X^\top y| \leq \lambda$

the first non zero component for vector β appears for $\lambda = \max |X^\top y|$

the regularization path is piecewise linear

Algo CW pour le Lasso

$$J_\lambda(\beta) = \frac{1}{2} \|X\beta - y\|^2 + \lambda \sum_{j=1}^p |\beta_j|,$$

Algo CW pour le Lasso

itérer, tant que on n'a pas convergé :

Pour $j = 1, p$

fixer toutes les variables sauf β_j
résoudre le Lasso en une dimension :

$$\hat{\beta}_j = \arg \min_{\beta \in \mathbb{R}} J_\lambda^{(j)}(\beta)$$

(1)

Le cout du Lasso en une dimension

$$J_\lambda^{(j)}(\beta) = \frac{1}{2} \|X_j \beta - (y - X \beta^{(-j)})\|^2 + \lambda |\beta|$$

Le Lasso en 1d

$$\min_{\beta \in \mathbb{R}} J_{\lambda}^{(1d)}(\beta) \quad \text{avec} \quad J_{\lambda}^{(1d)}(\beta) = \frac{1}{2} \|\mathbf{x}\beta - \mathbf{r}\|^2 + \lambda |\beta|$$

$$\partial_{\beta} J_{\lambda}^{(1d)}(\beta) = \mathbf{x}^{\top}(\mathbf{x}\beta - \mathbf{r}) + \begin{cases} \lambda\alpha & \text{if } \beta = 0 \\ \lambda\text{sign}(\beta) & \text{else.} \end{cases}$$

$$0 \in \partial_{\beta} J_{\lambda}^{(1d)}(\beta) \Leftrightarrow \begin{cases} \exists \alpha \in [-1, 1], -\mathbf{x}^{\top}\mathbf{r} + \lambda\alpha = 0 & \text{if } \beta = 0 \\ \|\mathbf{x}\|^2\beta - \mathbf{x}^{\top}\mathbf{r} + \lambda\text{sign}(\beta) = 0 & \text{else.} \end{cases}$$

The differential part :

$$\begin{cases} \text{if } \beta > 0 & \beta = \frac{\mathbf{x}^{\top}\mathbf{r} - \lambda}{\|\mathbf{x}\|^2} \quad \text{it works when } \mathbf{x}^{\top}\mathbf{r} > \lambda \\ \text{if } \beta < 0 & \beta = \frac{\mathbf{x}^{\top}\mathbf{r} + \lambda}{\|\mathbf{x}\|^2} \quad \text{it works when } \mathbf{x}^{\top}\mathbf{r} < -\lambda \end{cases}$$

It remains the non differential part : when $-\lambda < \mathbf{x}^{\top}\mathbf{r} < \lambda$ then $\beta = 0$

Le Lasso en 1d

The differential part :

$$\begin{cases} \text{if } \beta > 0 & \beta = \frac{\mathbf{x}^\top \mathbf{r} - \lambda}{\|\mathbf{x}\|^2} \quad \text{it works when } \mathbf{x}^\top \mathbf{r} > \lambda \\ \text{if } \beta < 0 & \beta = \frac{\mathbf{x}^\top \mathbf{r} + \lambda}{\|\mathbf{x}\|^2} \quad \text{it works when } \mathbf{x}^\top \mathbf{r} < -\lambda \end{cases}$$

It remains the non differential part : when $-\lambda < \mathbf{x}^\top \mathbf{r} < \lambda$ then $\beta = 0$

if $\|\mathbf{x}\|^2 = 1$

$$\hat{\beta}_{Lasso1d} = \begin{cases} 0 & \text{if } |\mathbf{x}^\top \mathbf{r}| \leq \lambda \\ \text{sign}(\mathbf{x}^\top \mathbf{r})(|\mathbf{x}^\top \mathbf{r}| - \lambda) & \text{else.} \end{cases}$$

Or in one line :

$$\text{sign}(\mathbf{x}^\top \mathbf{r}) \max((|\mathbf{x}^\top \mathbf{r}| - \lambda), 0)$$

Algo CW pour le Lasso

Pour $j = 1, p$: $\hat{\beta}_j = \arg \min_{\beta \in \mathbb{R}} J^{(j)}(\beta)$

$$\begin{aligned} J^{(j)}(\beta) &= \frac{1}{2} \|y - X\beta^{(-j)} - X_j\beta\|^2 + \lambda|\beta| \\ &= \frac{1}{2} \|\mathbf{r} - X_j\beta\|^2 + \lambda|\beta| \end{aligned} \tag{2}$$

avec $\mathbf{r} = y - X\beta^{(-j)}$ et $\beta^{(-j)} = (\beta_1, \dots, \beta_{j-1}, 0, \beta_{j+1}, \dots, \beta_p)$

Maths

Pour $j = 1, p$

$$\beta^{(-j)} = (\beta_1, \dots, \beta_{j-1}, 0, \beta_{j+1}, \dots, \beta_p)$$

$$\mathbf{r} = y - X\beta^{(-j)}$$

$$\hat{\beta}_j = \arg \min_{\beta \in \mathbb{R}} J^{(j)}(\beta, X_j, z)$$

fin de pour

Info

for j in range(p) :

$$bj = beta; bj[j] = 0;$$

$$\mathbf{r} = y - X @ bj$$

$$beta[j] = \text{sign}(x^\top \mathbf{r}) \max(|x^\top$$

end

Codage

- ➊ let's begin with $\beta = 0$
- ➋ in that case $r \leftarrow y$ since $X\beta = 0$
- ➌ fit a single (the j th) component of vector β :
$$\beta[j] = \text{sign}(\mathbf{x}^\top \mathbf{r}) \max((|\mathbf{x}^\top \mathbf{r}| - \lambda), 0)$$
- ➍ update r : $r \leftarrow r - X[:, j]\beta[j]$
- ➎ choose another component j'
- ➏ $r \leftarrow r + X[:, j']\beta[j']$
- ➐ fit the other component j' : $\beta[j'] = \text{sign}(\mathbf{x}^\top \mathbf{r}) \max((|\mathbf{x}^\top \mathbf{r}| - \lambda), 0)$
- ➑ update r : $r \leftarrow r - X[:, j']\beta[j']$
- ➒ ... iterate (go to 5)

Sklearn et le Lasso

<https://scikit-learn.org/stable/>

The screenshot shows the official scikit-learn website. At the top, there's a navigation bar with links like 'de', 'Adele', 'Drive', 'cantine', 'Galaxie_CS', 'Eval2020_25Norm...', 'Commite_IA', 'DARI', 'Microsoft Word - m...', 'retraite', 'Gaelle_J', 'Firefox Monitor', 'quentin', and 'les articles'. Below the navigation bar is the scikit-learn logo and a menu with 'Install', 'User Guide', 'API', 'Examples', and 'More'. The main content area has a blue header with the text 'scikit-learn' and 'Machine Learning in Python'. Below this are three cards: 'Classification' (Identifying which category an object belongs to, with applications in spam detection and image recognition, and algorithms like SVM, nearest neighbors, random forest), 'Regression' (Predicting a continuous-valued attribute associated with an object, with applications in drug response and stock prices, and algorithms like SVR, nearest neighbors, random forest), and 'Clustering' (Automatic grouping of similar objects, with applications in customer segmentation, and algorithms like k-Means and spectral clustering). To the right of the classification card is a list of bullet points: 'Simple and efficient tools for predictive data analysis', 'Accessible to everybody, and reusable in various contexts', 'Built on NumPy, SciPy, and matplotlib', and 'Open source, commercially usable - BSD license'.

- Lasso
- LassoCV
- lasso_path

- LassoLars
- LassoLarsCV
- lars_path

- sklearn.decomposition.sparse_encode

sklearn.linear_model.Lasso

```
class sklearn.linear_model.Lasso(alpha=1.0, *, fit_intercept=True, normalize=False, precompute=False, copy_X=True, max_iter=1000, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic')
```

[source]

Linear Model trained with L1 prior as regularizer (aka the Lasso)

The optimization objective for Lasso is:

```
(1 / (2 * n_samples)) * ||y - Xw||^2 + alpha * ||w||_1
```

Technically the Lasso model is optimizing the same objective function as the Elastic Net with `l1_ratio=1.0` (no L2 penalty).

Read more in the [User Guide](#).

Parameters:

`alpha : float, default=1.0`

Constant that multiplies the L1 term. Defaults to 1.0. `alpha = 0` is equivalent to an ordinary least square, solved by the `LinearRegression` object. For numerical reasons, using `alpha = 0` with the `Lasso` object is not advised. Given this, you should use the `LinearRegression` object.

`fit_intercept : bool, default=True`

Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered).

`normalize : bool, default=False`

This parameter is ignored when `fit_intercept` is set to False. If True, the regressors `X` will be normalized before regression by subtracting the mean and dividing by the l2-norm. If you wish to standardize, please use `sklearn.preprocessing.StandardScaler` before calling `fit` on an estimator with `normalize=True`.

3.2.4.1.3. sklearn.linear_model.LassoCV

```
class sklearn.linear_model.LassoCV(*, eps=0.001, n_alphas=100, alphas=None, fit_intercept=True, normalize=False, precompute='auto', max_iter=1000, tol=0.0001, copy_X=True, cv=None, verbose=False, n_jobs=None, positive=False, random_state=None, selection='cyclic')
```

[source]

Lasso linear model with iterative fitting along a regularization path.

See glossary entry for [cross-validation estimator](#).

The best model is selected by cross-validation.

The optimization objective for Lasso is:

```
(1 / (2 * n_samples)) * ||y - Xw||^2 + alpha * ||w||_1
```

Read more in the [User Guide](#).

Parameters:

`eps : float, default=1e-3`

Length of the path. `eps=1e-3` means that `alpha_min / alpha_max = 1e-3`.

`n_alphas : int, default=100`

Number of alphas along the regularization path

Lasso et Elastic Net

Le Lasso

$$J_\lambda(\beta) = \frac{1}{2} \|X'\beta - y'\|^2 + \lambda \|\beta\|_1 ,$$

Elastic Net

$$J_{\text{el}}(\beta) = \frac{1}{2} \|X\beta - y\|^2 + \lambda \|\beta\|_1 + \gamma \|\beta\|_2^2 ,$$

Lasso = Elastic Net with $\gamma = 0, X = X', y = y'$

Elastic Net = Lasso with $X' = (X^\top, \sqrt{\gamma}I_p)^\top, y' = (y^\top, 0_p)^\top$

$$\|X\beta - y\|_2^2 + \underbrace{\gamma \|\beta\|_2^2}_{\|\sqrt{\gamma}\beta\|_2^2} = \left\| \underbrace{\begin{bmatrix} X \\ \sqrt{\gamma}I_p \end{bmatrix}}_{=:X'} \beta - \underbrace{\begin{bmatrix} y \\ 0_p \end{bmatrix}}_{=:y'} \right\|_2^2 = \|X'\beta - y'\|_2^2 .$$

Reweighted least square (Lasso and Ridge)

$$J_\lambda(\beta) = \frac{1}{2} \|X\beta - y\|^2 + \lambda \sum_{j=1}^p |\beta_j| = \frac{\beta_j^2}{|\beta_j|},$$

the idea : iterate the weighted ridge towards a fixed point

$$\beta^{(k+1)} = \arg \min_{\beta} \frac{1}{2} \|X\beta - y\|^2 + \lambda \sum_{j=1}^p \frac{\beta_j^2}{|\beta_j^{(k)}|} = w_j \beta_j^2,$$

with $w_j = 1/|\beta_j^{(k)}|$, that is

$$\beta^{(k+1)} = (X^\top X + \lambda W)^{-1} X^\top y \quad \text{with } \text{diag}(W) = \frac{1}{|\beta_j^{(k)}|}$$

$W = \text{Identité}$

Tant que on n'a pas convergé

$$\beta = \arg \min \frac{1}{2} \|X\beta - y\|^2 + \lambda \beta^\top W \beta$$

$$W = \text{diag}(1/|\beta|)$$

fin de tant que

The adaptive Lasso

$$J_\lambda(\beta) = \frac{1}{2} \|X\beta - y\|^2 + \lambda \sum_{j=1}^p w_j |\beta_j|,$$

$$w_j = \frac{1}{|\hat{\beta}_j^{(ls)}|^\gamma}$$

```
w = np.linalg.solve(X.T@X,X.T@y)
w = 1/np.sqrt(np.abs(w))
X_c = X/w
c = mon_lasso(X_c,y,lambda)
beta = c/w
```

The Adaptive Lasso and Its Oracle Properties, H. Zou, JASA, 2012