# Méthodes pénalisées & Lasso

Master parcours SSD - UE Apprentissage Statistique II

Pierre Mahé - bioMérieux & Université de Grenoble-Alpes

# Ce cours

Méthodes pénalisées :

- un cadre plus général que les SVMS
- basé (ici) sur des modèles linéaires
- mettant en jeu un critère de régularisation/pénalisation
  - joué par la marge des SVMs
- dont la pénalité Lasso
- qui conduit à des modèles parcimonieux
  - sélection de variables
- ainsi que ses extensions
  - elastic-net, group-lasso, ...

# Plan

1. SVMs & méthodes pénalisées
2. Régressions Ridge
3. Pénalité Lasso
4. Pénalité Elastic-Net
5. Pénalité Goup-Lasso
6. Mise en oeuvre en `Python`

# SVMs & méthodes pénalisées

# SVMs

## Soft-margin SVM : problème primal

$$(w^*, b^*) = \underset{\substack{w\in\mathbb{R}^p, b\in\mathbb{R},\\ \xi\in\mathbb{R}^n}}{\operatorname{argmin}} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad y_i(\langle w, x_i\rangle + b) \geq 1-\xi_i, \quad i = 1,\dots,n,$$
$$\xi_i \geq 0, \quad i = 1,\dots,n.$$

On notera $\boxed{f(x) = \langle w, x\rangle + b}$ :

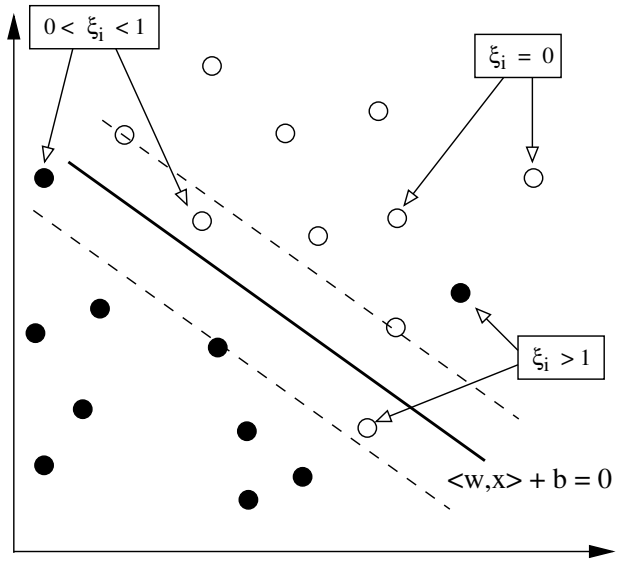
- $y_i f(x_i) > 0 \Leftrightarrow (x_i, y_i)$ bien classé
- $y_i f(x_i) > 1 \Leftrightarrow (x_i, y_i)$ bien classé en dehors de la marge

$\Rightarrow$ la variable $\boxed{\xi_i = 1 - y_i f(x_i),\ \xi_i \geq 0}$ :

- quantifie l'erreur faite au point $(x_i, y_i)$
- est une variable "ressort" ("slack" variable)

# Illustration - slack-variables

Slack-variable $\boxed{\xi_i = 1 - y_i f(x_i),\ \xi_i \geq 0}$ :

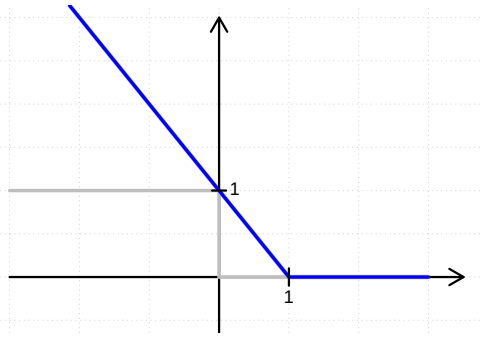



- $\xi_i = 0$ : point $(x_i, y_i)$ bien classé en dehors de la marge
- $0 < \xi_i < 1$ : point $(x_i, y_i)$ bien classé mais dans la marge
- $\xi_i > 1$ : point $(x_i, y_i)$ mal classé

# Hinge loss

Fonction de coût associée = hinge loss ("charnière") :

$$\begin{aligned} h\big(y, f(x)\big) &= \max\big(0, 1 - yf(x)\big) \\ &= \big(1 - yf(x)\big)_+ \end{aligned}$$




$\Rightarrow \xi_i = 1 - y_i f(x_i),\ \xi_i \geq 0$

Interprétation :

- pas de perte si $yf(x) \geq 1$ : hors de la marge
- perte linéaire dès qu'on rentre dans la marge

Remarque : toujours $\geq$ au coût 0/1

- plus facile à manipuler car continue
- $\sum_i h\big(y_i, f(x_i)\big) > \#$ erreurs de classification

# SVMs & hinge-loss

## Soft-margin SVM : problème primal

$$
\begin{aligned}
(w^*, b^*) = \underset{\substack{w\in\mathbb{R}^p, b\in\mathbb{R}, \\ \xi\in\mathbb{R}^n}}{\operatorname{argmin}} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \\
\text{s.t.} \quad & y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i, \quad i = 1, \dots, n, \\
& \xi_i \geq 0, \qquad\qquad\qquad\quad i = 1, \dots, n.
\end{aligned}
$$

$\Rightarrow$ ré-écriture du problème SVM :

$$
\boxed{(w^*, b^*) = \underset{w\in\mathbb{R}^p, b\in\mathbb{R}}{\operatorname{argmin}} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} h\big(y_i, f(x_i)\big)}
$$

où $f(x) = \langle w, x\rangle + b$ et $h\big(y, f(x)\big) = \big(1 - yf(x)\big)_+$.

# SVMs & risque empirique pénalisé

On peut alors écrire :

$$
\begin{aligned}
(w^*, b^*) &= \underset{w\in\mathbb{R}^p, b\in\mathbb{R}}{\operatorname{argmin}} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} h\big(y_i, f(x_i)\big) \\
&= \underset{w\in\mathbb{R}^p, b\in\mathbb{R}}{\operatorname{argmin}} \quad \frac{1}{2C}||w||^2 + \sum_{i=1}^{n} h\big(y_i, f(x_i)\big) \\
&= \underset{w\in\mathbb{R}^p, b\in\mathbb{R}}{\operatorname{argmin}} \quad \sum_{i=1}^{n} h\big(y_i, f(x_i)\big) + \frac{1}{2C}||w||^2 \\
&= \underset{w\in\mathbb{R}^p, b\in\mathbb{R}}{\operatorname{argmin}} \quad \sum_{i=1}^{n} h\big(y_i, f(x_i)\big) + \lambda||w||^2
\end{aligned}
$$

$\Rightarrow$ formulation de risque empirique pénalisé.

# SVMs & risque empirique pénalisé

Formulation de risque empirique pénalisé :

$$(w^*, b^*) = \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} h\big(y_i, f(x_i)\big) + \lambda ||w||^2$$

- risque empirique : erreur faite sur le jeu d'apprentissage
  - selon la fonction de coût $h\big(y, f(x)\big)$
- penalisation : pour contrôler le sur-apprentissage
  - risque empirique seul = risque de sur-apprentissage
    - e.g., haute dimension ou classes de fonctions complexes
  - pénalisation = régularisation
  - pour les SVMs : liée directement à la notion de marge

# Risque empirique pénalisé

Risque empirique pénalisé : formulation générale

$$(w^*, b^*) = \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} L\big(y_i, f(x_i)\big) + \lambda \Omega(w)$$

où :

- $L\big(y, f(x)\big)$ est une fonction de coût / de perte
  - grand quand $y$ et $f(x)$ sont différents
- $\Omega(w)$ est une fonction de régularisation

# Risque empirique pénalisé

Risque empirique pénalisé : formulation générale

$$(w^*, b^*) = \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} L\big(y_i, f(x_i)\big) + \lambda \Omega(w)$$

où :

- $L\big(y, f(x)\big)$ est une fonction de coût / de perte
  - grand quand $y$ et $f(x)$ sont différents
- $\Omega(w)$ est une fonction de régularisation

$\Rightarrow$ ici on considèrera :

- des modèles linéaires : $f(x) = \langle w, x \rangle + b$
- des fonctions de perte et de régularisation convexes

# Fonctions de perte classiques

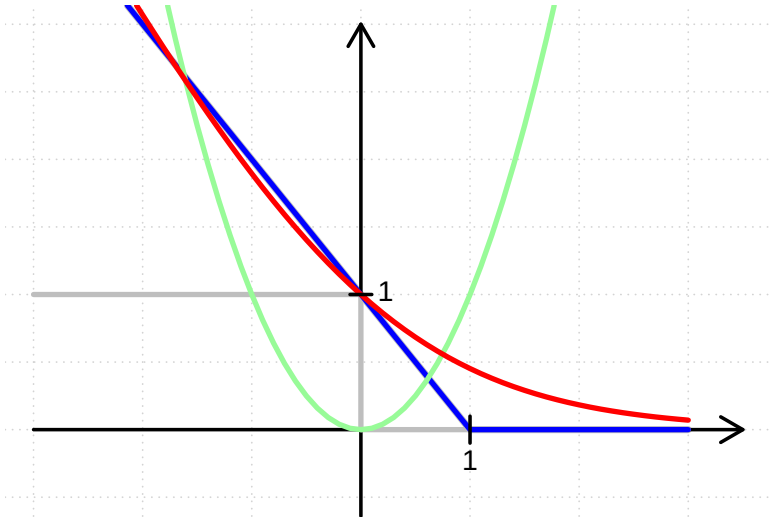
Régression :

- perte quadratique : $L\big(y, f(x)\big) = (y - f(x))^2$

Classification :

- perte hinge : $L\big(y, f(x)\big) = \big(1 - yf(x)\big)_+$
- perte logistique : $L\big(y, f(x)\big) = \log\left(1 + e^{-yf(x)}\right)$

(la quantité $yf(x)$ est parfois appelée la **marge** de $(x, y)$)



# Perte logistique

## Régression logistique :

$$\begin{cases} P(Y=1|x) = \sigma(f(x)) = \frac{1}{1+e^{-f(x)}} \\ P(Y=-1|x) = 1-\sigma(f(x)) = \frac{1}{1+e^{+f(x)}} \end{cases}$$

$\Rightarrow$ on a donc $\forall (x_i, y_i)$ : $\boxed{P(Y=y_i|x_i) = \frac{1}{1+e^{-y_i f(x_i)}}}$

# Perte logistique

Régression logistique :

$$\begin{cases} P(Y=1|x) = \sigma(f(x)) = \frac{1}{1+e^{-f(x)}} \\ P(Y=-1|x) = 1-\sigma(f(x)) = \frac{1}{1+e^{+f(x)}} \end{cases}$$

$\Rightarrow$ on a donc $\forall (x_i, y_i)$ : $\boxed{P(Y=y_i|x_i) = \frac{1}{1+e^{-y_i f(x_i)}}}$

Solution du maximum de (log) vraisemblance :

$$\begin{aligned} (\hat{w}, \hat{b}) &= \operatorname{argmax} \prod_{i=1}^{n} \frac{1}{1+e^{-y_i f(x_i)}} \\ &= \operatorname{argmin} \sum_{i=1}^{n} \log(1+e^{-y_i f(x_i)}) \end{aligned}$$

# Perte logistique

## Perte logistique :

$$L\big(y, f(x)\big) = \log\left(1 + e^{-yf(x)}\right)$$
$$= -\log\big(P(Y = y|x)\big).$$

$\Rightarrow$ utiliser la perte logistique = faire une régression logistique

# Perte logistique

Perte logistique :

$$L\big(y, f(x)\big) = \log\left(1 + e^{-yf(x)}\right) = -\log\big(P(Y = y|x)\big).$$

$\Rightarrow$ utiliser la perte logistique = faire une régression logistique

Conséquence : prédiction probabiliste :

$$\boxed{P(Y = 1|x) = \frac{1}{1 + e^{-f(x)}}} \Rightarrow \text{critère de confiance}$$

# Perte logistique

Perte logistique :

$$L\big(y, f(x)\big) = \log\left(1 + e^{-yf(x)}\right) = -\log\big(P(Y = y|x)\big).$$

$\Rightarrow$ utiliser la perte logistique = faire une régression logistique

Conséquence : prédiction probabiliste :

$$\boxed{P(Y = 1|x) = \frac{1}{1 + e^{-f(x)}}} \Rightarrow \text{critère de confiance}$$

En pratique :

- logistique et hinge sont proches : $\sim$ performances
- logistique dérivable partout
- "charnière" des SVMs $\Rightarrow$ vecteurs supports
  - pas de VS en logistique, mais inutile dans le primal

# Fonctions de pénalisation incontournables

Pénalité Ridge (ou $L_2$) : $\boxed{\Omega_{\text{Ridge}}(w) = ||w||_2^2 = \sum_{j=1}^{p} w_j^2}$

Pénalité Lasso (ou $L_1$) : $\boxed{\Omega_{\text{Lasso}}(w) = ||w||_1 = \sum_{j=1}^{p} |w_j|}$

# Fonctions de pénalisation incontournables

Pénalité Ridge (ou $L_2$) : $\boxed{\Omega_{\text{Ridge}}(w) = ||w||_2^2 = \sum_{j=1}^{p} w_j^2}$

Pénalité Lasso (ou $L_1$) : $\boxed{\Omega_{\text{Lasso}}(w) = ||w||_1 = \sum_{j=1}^{p} |w_j|}$

Même effet : pénaliser les valeur élevées $\Rightarrow$ régularisation

- intuition : pente élevée $\rightarrow$ $y$ varie + vite quand $x$ varie

mais avec une géométrie différente :

- **lasso** : des coefficients $w_j$ **exactement** $= 0$
- **ridge** : coefficients $w_j$ petits mais **jamais nuls**

$\Rightarrow$ Lasso = méthode parcimonieuse, sélection de variables

# Pertes & régularisation

Combinaisons pertes / pénalités :

| | Quadratique | Hinge | Logistique |
|---|---|---|---|
| Ridge | ridge regression | SVM | ridge logistic-regression |
| Lasso | Lasso | L1-SVM | L1 logistic-regression |

$\Rightarrow$ pour la suite on s'intéressera principalement aux pertes quadratiques et logistiques

- cadre des **modèles linéaires généralisés**
- logistique et SVM très proches

# Régression Ridge

# Ridge régression

Régression linéaire avec perte quadratique et pénalité $L_2$ :

$$(w^*, b^*) = \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 + \lambda \left|\left|w\right|\right|_2^2$$

$$= \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - b - \sum_{j=1}^{p} w_j x_{ij}\right)^2 + \lambda \sum_{j=1}^{p} w_j^2$$

# Ridge régression

Régression linéaire avec perte quadratique et pénalité $L_2$ :

$$(w^*, b^*) = \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 + \lambda \left|\left|w\right|\right|_2^2$$

$$= \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - b - \sum_{j=1}^{p} w_j x_{ij}\right)^2 + \lambda \sum_{j=1}^{p} w_j^2$$

Pour le résoudre : centrer les variables $\rightarrow \tilde{x}_{ij} = x_{ij} - \bar{x}_j$

1. on estime alors l'intercept $b$ par $b^* = \hat{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$
2. on obtient $w$ avec le même problème sans intercept.
3. solution : $w^* = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^\top y$ où $\mathbf{X}[i,j] = \tilde{x}_{ij}$

# Ridge regression

Régression linéaire vs régression ridge :

$$w^* = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top y \quad \Rightarrow \quad w^* = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^\top y$$

# Ridge regression

Régression linéaire vs régression ridge :

$$w^* = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top y \quad \Rightarrow \quad w^* = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^\top y$$

La matrice $\mathbf{X}^\top \mathbf{X}$ (de taille $p \times p$) est :

- non inversible si $p > n$
  - résolution du problème impossible
- mal conditionnée si descripteurs corrélés
  - instabilité numérique, forte variance dans l'estimation

# Ridge regression

Régression linéaire vs régression ridge :

$$w^* = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top y \quad \Rightarrow \quad w^* = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^\top y$$

La matrice $\mathbf{X}^\top \mathbf{X}$ (de taille $p \times p$) est :

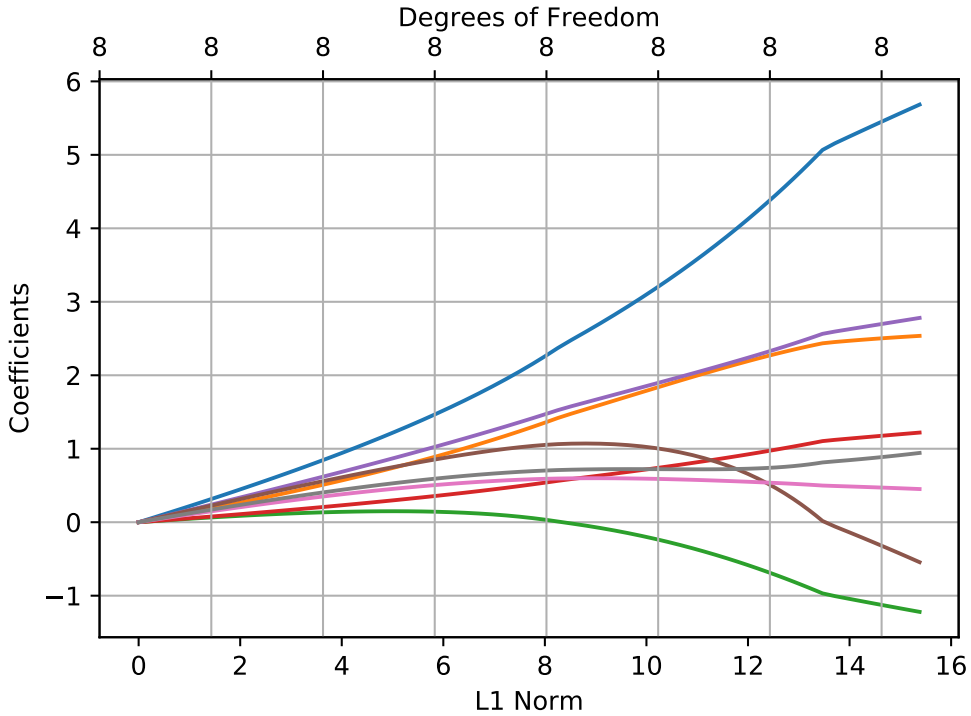
- non inversible si $p > n$
  - résolution du problème impossible
- mal conditionnée si descripteurs corrélés
  - instabilité numérique, forte variance dans l'estimation

$\Rightarrow$ ajout du terme $\lambda \mathbf{I}$ = régularisation :

- permet de travailler en haute dimension ($p >> n$)
  - sans faire de sélection de variables, parfois instable
- stabilise l'estimation en présence de variables corrélées
  - poids "partagé" entre variables fortement corrélées

# Ridge regression - illustration

Illustation : `Prostate` dataset [1]




$\Rightarrow$ "shrinkage" : réduction de l'amplitude des coefficients

$\Rightarrow$ regularisation path : coefficients en fonction de $\lambda$ ou $||w||_1$

1. https://web.stanford.edu/~hastie/ElemStatLearn/

# Ridge regression - remarques

Méthode simple, efficace et stable (e.g., en haute dimension)

- **question clé** : régler le paramètre de régularisation

# Ridge regression - remarques

Méthode simple, efficace et stable (e.g., en haute dimension)

- **question clé** : régler le paramètre de régularisation

Inteprétation bayésienne :

- moindres carrés = maximum de vraisemblance sous hypothèse de bruit Gaussien : $y \to \mathcal{N}(\langle w, x \rangle + b\,;\ \sigma^2)$
- régularisation $L_2$ = a priori $w_j \to \mathcal{N}(0, \sigma_0^2)$

$\Rightarrow$ moindres carrés pénalisé : estimateur du MAP
$\Rightarrow \lambda \sim 1/\sigma_0^2$

# Ridge regression - remarques

Méthode simple, efficace et stable (e.g., en haute dimension)

- **question clé** : régler le paramètre de régularisation

Inteprétation bayésienne :

- moindres carrés = maximum de vraisemblance sous hypothèse de bruit Gaussien : $y \to \mathcal{N}(\langle w, x \rangle + b\,;\, \sigma^2)$
- régularisation $L_2$ = a priori $w_j \to \mathcal{N}(0, \sigma_0^2)$

$\Rightarrow$ moindres carrés pénalisé : estimateur du MAP
$\Rightarrow \lambda \sim 1/\sigma_0^2$

Ridge logistic-regression : même mécanisme

- pas de "closed form" solution, algorithme itératif

# Ridge regression - remarques

Méthode simple, efficace et stable (e.g., en haute dimension)

- **question clé** : régler le paramètre de régularisation

Inteprétation bayésienne :

- moindres carrés = maximum de vraisemblance sous hypothèse de bruit Gaussien : $y \to \mathcal{N}(\langle w, x \rangle + b\,;\, \sigma^2)$
- régularisation $L_2$ = a priori $w_j \to \mathcal{N}(0, \sigma_0^2)$

$\Rightarrow$ moindres carrés pénalisé : estimateur du MAP
$\Rightarrow \lambda \sim 1/\sigma_0^2$

Ridge logistic-regression : même mécanisme

- pas de "closed form" solution, algorithme itératif

Limite : pas de sélection de variables

- coefficients "shrinkés" vers 0 mais jamais nuls

# Lasso

# Régression Lasso

Régression linéaire avec perte quadratique et pénalité $L_1$ :

$$(w^*, b^*) = \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 + \lambda \left|\left|w\right|\right|_1$$

$$= \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - b - \sum_{j=1}^{p} w_j x_{ij}\right)^2 + \lambda \sum_{j=1}^{p} |w_j|$$

# Régression Lasso

Régression linéaire avec perte quadratique et pénalité $L_1$ :

$$(w^*, b^*) = \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 + \lambda \left|\left|w\right|\right|_1$$

$$= \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - b - \sum_{j=1}^{p} w_j x_{ij}\right)^2 + \lambda \sum_{j=1}^{p} |w_j|$$

Par rapport à pénalité ridge :

- même effet de régularisation : shrinkage des coefficients
- mais conduit à des coefficients exactement = 0

$\Rightarrow$ solution parcimonieuse (sparse) : sélection de variables.

# Régression Lasso & sélection de variables

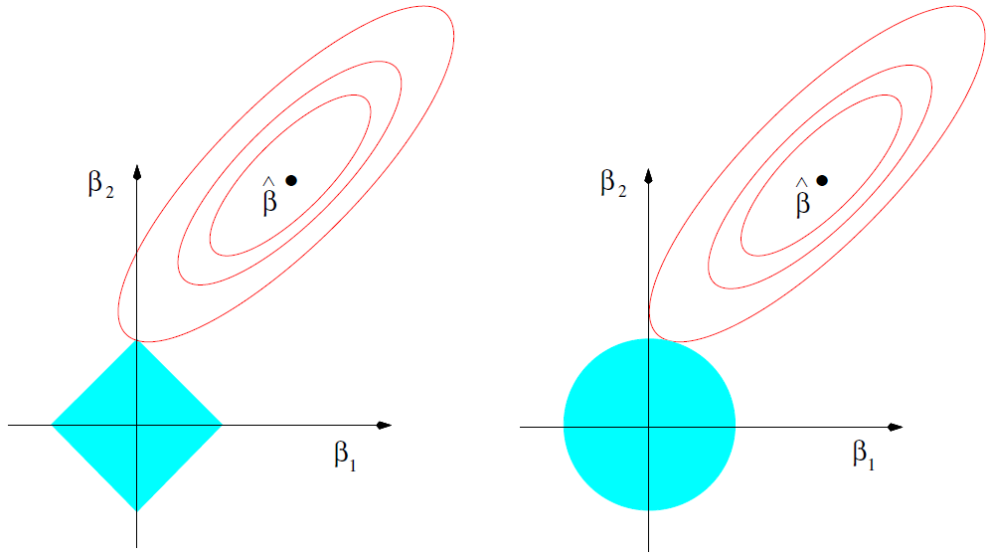
## Formulation équivalente :

$$(w^*, b^*) = \underset{w \in \mathbb{R}^p, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 \text{ tel que } \left|\left|w\right|\right|_1 \leq t.$$

- $\forall\, \lambda, \exists\, t$ tels que les deux solutions soient identiques.
  - i.e., mêmes chemins de régularisation.
- valable aussi pour ridge (et autres).
- pas de relation directe entre $t$ et $\lambda$.
- en pratique, $\lambda$ plus simple à optimiser que $t$.
  - la solution varie moins vite en fonction de $\lambda$ que de $t$.
- version sous contrainte : aide à intepréter la pénalisation.

# Régression Lasso & sélection de variables

Illustration : Lasso vs Ridge (image de Hastie et al. (2001)) :



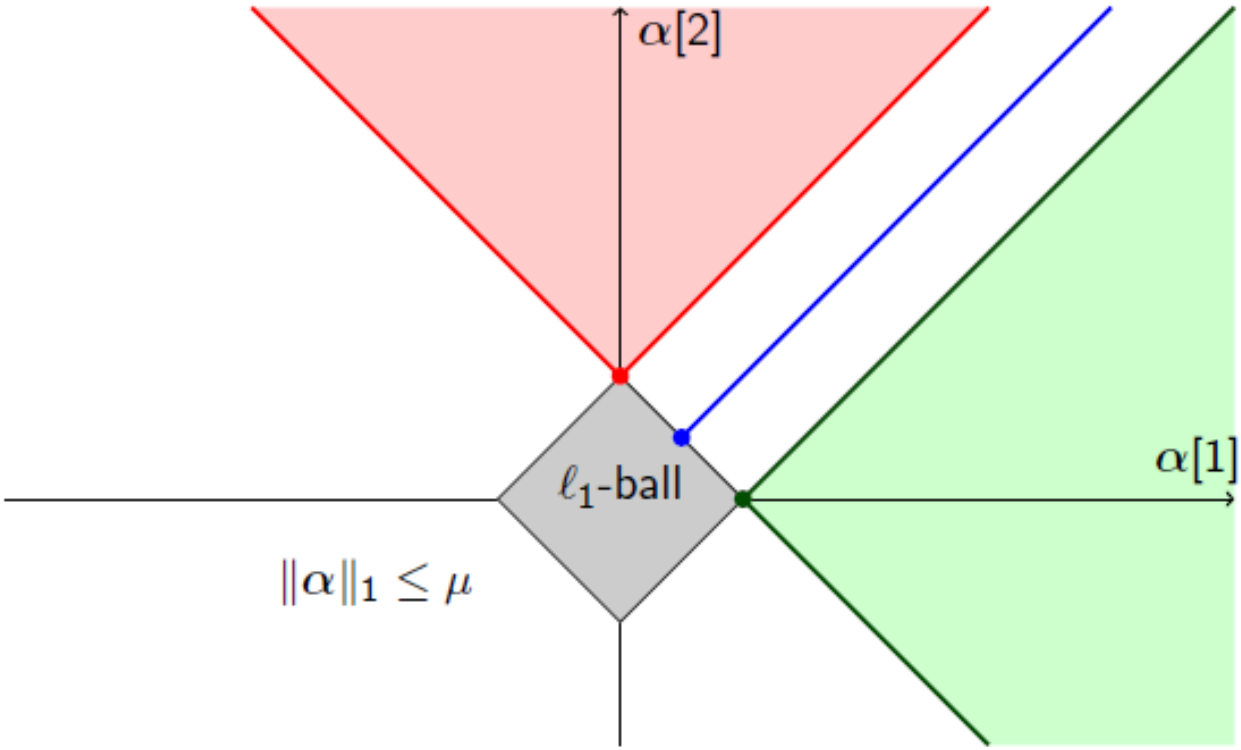

- problèmes à 2 dimensions, coefficients $\beta = [\beta_1\ \beta_2]$
- $\hat{\beta}$ : solution des moindres carrés (fonction quadratique)
- régions admissibles : $||\beta||_1 \leq t$ et $||\beta||_2^2 \leq t$

$\Rightarrow$ solution = projection de $\hat{\beta}$ sur les régions admissibles
$\Rightarrow$ points de singularité de $||.||_1$ = solution parcimonieuse

# Pénalité Lasso & sélection de variables [2]

## Why does the $\ell_1$-norm induce sparsity?

### Regularizing with the $\ell_1$-norm




The projection onto a convex set is "biased" towards singularities.

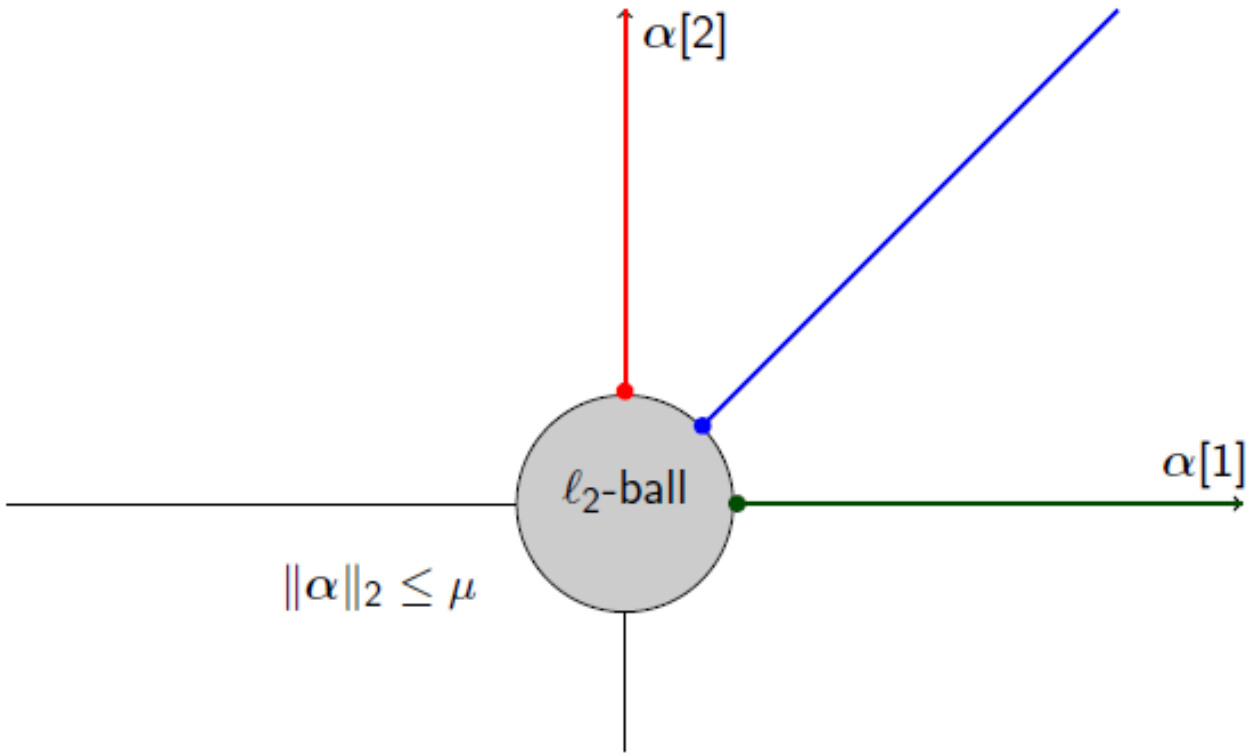


2. Slide tirée d'une présentation de Julien Mairal (INRIA Grenoble)

# Pénalité Lasso & sélection de variables [3]

## Why does the $\ell_1$-norm induce sparsity?

### Regularizing with the $\ell_2$-norm



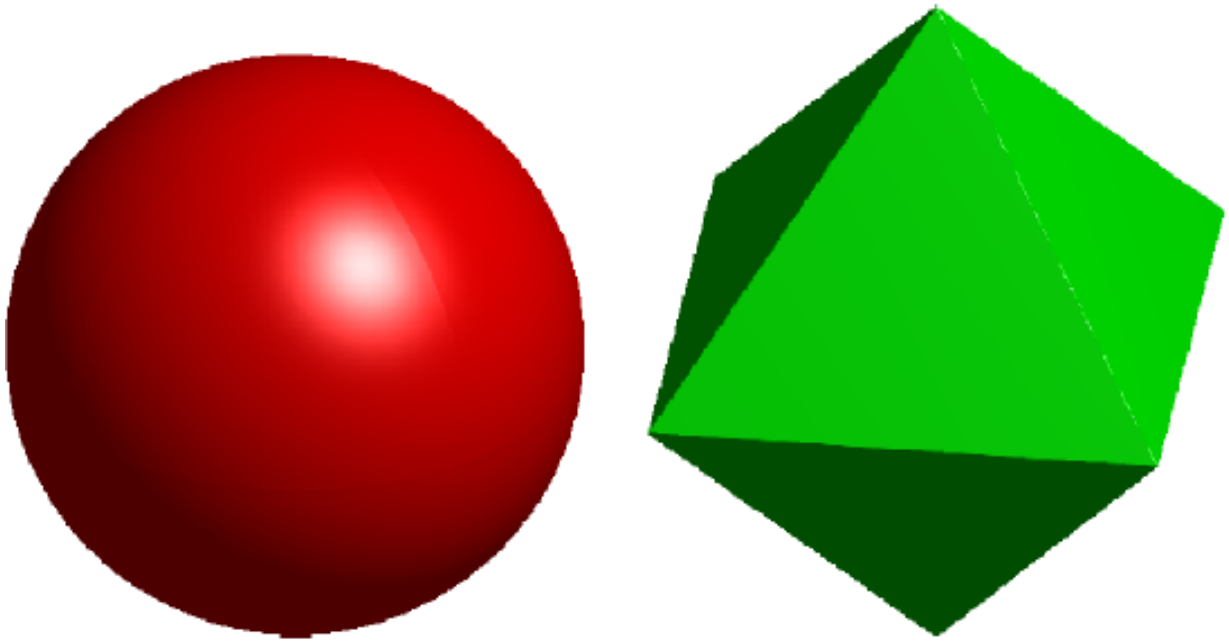

The $\ell_2$-norm is isotropic.



3. Slide tirée d'une présentation de Julien Mairal (INRIA Grenoble)

# Pénalité Lasso & sélection de variables

Boules $L_1$ et $L_2$ en 3D : [4] :



(a) $\ell_2$-norm ball (b) $\ell_1$-norm ball

$\Rightarrow$ sphère vs diamant.

4. Image tirée de Bach et al. (2012)

# Lasso - résolution

On va considérer une version sensiblement différente :

$$w^* = \underset{w \in \mathbb{R}^p}{\operatorname{argmin}} \quad \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \langle w, x \rangle\right)^2 + \lambda \left|\left| w \right|\right|_1$$

où les $y_i$ sont centrés et les $x_{ij}$ standardisés.

- dans ce cas, on peut travailler sans intercept $b$
- le terme $1/2n$ rend l'effet de $\lambda$ indépendant de $n$

# Lasso - résolution

On va considérer une version sensiblement différente :

$$w^* = \underset{w \in \mathbb{R}^p}{\operatorname{argmin}} \quad \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \langle w, x \rangle\right)^2 + \lambda \left|\left| w \right|\right|_1$$

où les $y_i$ sont centrés et les $x_{ij}$ standardisés.

- dans ce cas, on peut travailler sans intercept $b$
- le terme $1/2n$ rend l'effet de $\lambda$ indépendant de $n$

Résolution :

- contrairement à ridge : pas de solution analytique
- mais problème convexe (QP) : différents algorithmes

$\Rightarrow$ ici : algorithme de "coordinate descent"

- simple, intuitif et efficace - à la base du package `glmnet`

# Lasso - résolution

Pour un seul prédicteur le problème s'écrit :

$$w^* = \underset{w}{\operatorname{argmin}} \frac{1}{2n} \sum_{i=1}^{n} (y_i - w x_i)^2 + \lambda |w|$$

Approche standard : calculer la dérivée selon $w$ et l'annuler

Problème : la fonction $|.|$ n'est pas dérivable en 0.

# Lasso - résolution

Pour un seul prédicteur le problème s'écrit :

$$\boxed{w^* = \underset{w}{\operatorname{argmin}} \frac{1}{2n} \sum_{i=1}^{n} (y_i - w x_i)^2 + \lambda |w|}$$

Approche standard : calculer la dérivée selon $w$ et l'annuler

Problème : la fonction $|.|$ n'est pas dérivable en 0.

$\Rightarrow$ On sépare les cas $\{w < 0; w = 0; w > 0\}$ et on obtient :

$$w^* = \begin{cases} \frac{1}{n}\langle x, y\rangle - \lambda & \text{si} \quad \frac{1}{n}\langle x, y\rangle > \lambda, \\ 0 & \text{si} \quad \frac{1}{n}|\langle x, y\rangle| \leq \lambda, \\ \frac{1}{n}\langle x, y\rangle + \lambda & \text{si} \quad \frac{1}{n}\langle x, y\rangle < -\lambda. \end{cases}$$

# Lasso - résolution

Solution pour un prédicteur unique :

$$w^* = \begin{cases} \frac{1}{n}\langle x, y\rangle - \lambda & \text{si} \quad \frac{1}{n}\langle x, y\rangle > \lambda, \\ 0 & \text{si} \quad \frac{1}{n}|\langle x, y\rangle| \leq \lambda, \\ \frac{1}{n}\langle x, y\rangle + \lambda & \text{si} \quad \frac{1}{n}\langle x, y\rangle < -\lambda. \end{cases}$$

# Lasso - résolution

Solution pour un prédicteur unique :

$$w^* = \begin{cases} \frac{1}{n}\langle x, y\rangle - \lambda & \text{si} \quad \frac{1}{n}\langle x, y\rangle > \lambda, \\ 0 & \text{si} \quad \frac{1}{n}|\langle x, y\rangle| \leq \lambda, \\ \frac{1}{n}\langle x, y\rangle + \lambda & \text{si} \quad \frac{1}{n}\langle x, y\rangle < -\lambda. \end{cases}$$

$\Rightarrow$ interprétation :

1. $\frac{1}{n}\langle x, y\rangle$ = solution des moindres carrés
   - régression linéaire classique avec $x$ standardisé
2. on le "shrinke" vers 0 d'une valeur $\lambda$ :
   - $\frac{1}{n}\langle x, y\rangle > 0 \Rightarrow \frac{1}{n}\langle x, y\rangle - \lambda$
   - $\frac{1}{n}\langle x, y\rangle < 0 \Rightarrow \frac{1}{n}\langle x, y\rangle + \lambda$
3. s'il est trop petit : on le met à zéro
   - $\frac{1}{n}|\langle x, y\rangle| \leq \lambda$

# Lasso - résolution

Solution pour un prédicteur unique :

$$w^* = \begin{cases} \frac{1}{n}\langle x, y\rangle - \lambda & \text{si} \quad \frac{1}{n}\langle x, y\rangle > \lambda, \\ 0 & \text{si} \quad \frac{1}{n}|\langle x, y\rangle| \leq \lambda, \\ \frac{1}{n}\langle x, y\rangle + \lambda & \text{si} \quad \frac{1}{n}\langle x, y\rangle < -\lambda. \end{cases}$$

# Lasso - résolution

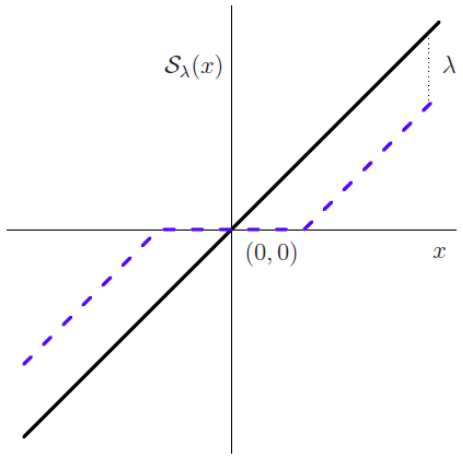
Solution pour un prédicteur unique :

$$w^* = \begin{cases} \frac{1}{n}\langle x, y\rangle - \lambda & \text{si} \quad \frac{1}{n}\langle x, y\rangle > \lambda, \\ 0 & \text{si} \quad \frac{1}{n}|\langle x, y\rangle| \leq \lambda, \\ \frac{1}{n}\langle x, y\rangle + \lambda & \text{si} \quad \frac{1}{n}\langle x, y\rangle < -\lambda. \end{cases}$$

Notons $\beta = \frac{1}{n}\langle x, y\rangle$ :

$$w^* = \begin{cases} \text{sign}(\beta)(|\beta| - \lambda) & \text{si } |\beta| > \lambda, \\ 0 & \text{sinon.} \end{cases}$$

$$\Rightarrow \boxed{w^* = \text{sign}(\beta)(|\beta| - \lambda)_+ = \mathcal{S}_\lambda(\beta)}$$




La fonction $\mathcal{S}_\lambda(x)$ est le "soft-thresholding operator".

# Lasso - résolution

Si $x_i \in \mathbb{R}^p$, on "boucle" sur les $p$ descripteurs :

- on met à jour le coefficient $w_j$ en gardant les autres fixés
- on applique le même principe sur les résidus :

$$\boxed{w_j^* = \mathcal{S}_\lambda\left(\frac{1}{n}\langle x^{(j)}, r^{(j)}\rangle\right)} \quad \text{où} \quad r_i^{(j)} = y_i - \sum_{k \neq j} w_k x_{ik}$$

(au lieu de $w^* = \mathcal{S}_\lambda(\frac{1}{n}\langle x, y\rangle)$ si un seul descripteur $x$)

# Lasso - résolution

Si $x_i \in \mathbb{R}^p$, on "boucle" sur les $p$ descripteurs :

- on met à jour le coefficient $w_j$ en gardant les autres fixés
- on applique le même principe sur les résidus :

$$\boxed{w_j^* = \mathcal{S}_\lambda\left(\frac{1}{n}\langle x^{(j)}, r^{(j)}\rangle\right)} \quad \text{où} \quad r_i^{(j)} = y_i - \sum_{k \neq j} w_k x_{ik}$$

(au lieu de $w^* = \mathcal{S}_\lambda(\frac{1}{n}\langle x, y\rangle)$ si un seul descripteur $x$)

$\Rightarrow$ algorithme de coordinate-descent.

- converge sur ce problème (convexe)
- simple à mettre en oeuvre, pas de paramètres
- **"warm start"** pour obtenir un chemin de régularisation

# Lasso - résolution

Coordinate descent : ré-écriture du problème selon $w_j$ :

$$\frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} w_j x_{ij}\right)^2 + \lambda \sum_{j=1}^{p} |w_j|$$

$$\Leftrightarrow \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \sum_{k\neq j} w_k x_{ik} - w_j x_{ij}\right)^2 + \lambda \sum_{k\neq j}^{p} |w_k| + \lambda |w_j|$$

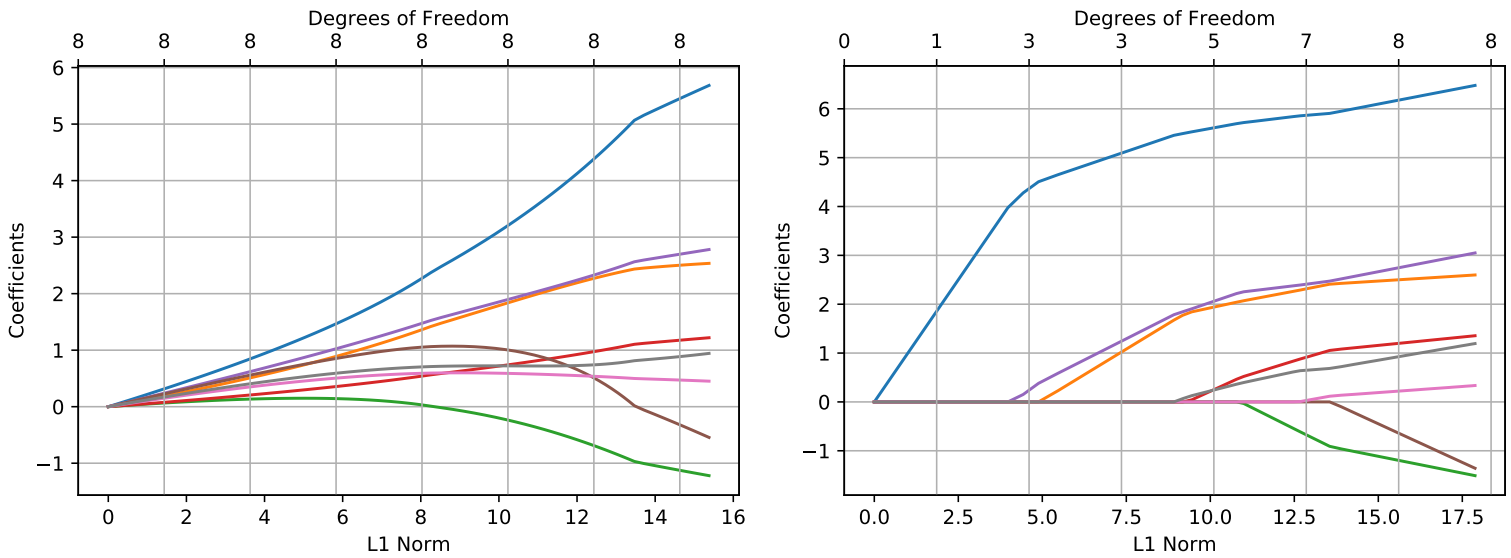
$$\Leftrightarrow \frac{1}{2n}\sum_{i=1}^{n}\left(r_i^{(j)} - w_j x_{ij}\right)^2 + \lambda \sum_{k\neq j}^{p} |w_k| + \lambda |w_j|$$

$\Rightarrow$ proche de $\boxed{\frac{1}{2n}\sum_{i=1}^{n}(y_i - w x_i)^2 + \lambda |w|}$ (pour 1 variable)

- $\sum_{k\neq j}^{p} |w_k|$ disparait quand on dérive par rapport à $w_j$

# Lasso - illustration

Illustation : `Prostate` dataset, ridge vs lasso




$\Rightarrow$ au bout du chemin : même solution = moindres carrés

- pas de pénalisation

$\Rightarrow$ Lasso : inclusion graduelle des variables

# Lasso - remarques

Lasso : méthode de sélection de variables "embedded"

- réalisée lors de la construction du modèle

# Lasso - remarques

Lasso : méthode de sélection de variables "embedded"

- réalisée lors de la construction du modèle

Nombreux développements théoriques

- consistence, optimisation, ...

# Lasso - remarques

Lasso : méthode de sélection de variables "embedded"

- réalisée lors de la construction du modèle

Nombreux développements théoriques

- consistence, optimisation, ...

Coordinate descent : **1** algorithme d'optimisation

- alternatives : proximal, LARS/homotopie, ...
- adaptaté à la régression logistique

# Lasso - remarques

Lasso : méthode de sélection de variables "embedded"

- réalisée lors de la construction du modèle

Nombreux développements théoriques

- consistence, optimisation, ...

Coordinate descent : 1 algorithme d'optimisation

- alternatives : proximal, LARS/homotopie, ...
- adaptaté à la régression logistique

Limites :

- sélectionne au plus $\min(n, p)$ variables
- solution pas toujours unique
  - e.g., si prédicteurs identiques ou antagonistes
- {prédicteurs corrélés} : tend à prendre 1 représentant

# Elastic Net

# Pénalité Elastic-Net

Pénalité elastic-net : compromis $L_1/L_2$

$$\boxed{\Omega_\alpha(w) = \alpha\left|\left|w\right|\right|_1 + (1-\alpha)\left|\left|w\right|\right|_2^2}$$

ou :

$$\boxed{\Omega_\alpha(w) = \alpha\left|\left|w\right|\right|_1 + \frac{(1-\alpha)}{2}\left|\left|w\right|\right|_2^2}$$

$\Rightarrow \alpha = 1$ : Lasso , $\alpha = 0$ : Ridge.

# Pénalité Elastic-Net

Pénalité elastic-net : compromis $L_1/L_2$

$$\boxed{\Omega_\alpha(w) = \alpha\left|\left|w\right|\right|_1 + (1-\alpha)\left|\left|w\right|\right|_2^2}$$

ou :

$$\boxed{\Omega_\alpha(w) = \alpha\left|\left|w\right|\right|_1 + \frac{(1-\alpha)}{2}\left|\left|w\right|\right|_2^2}$$

$\Rightarrow \alpha = 1$ : Lasso , $\alpha = 0$ : Ridge.

Intérêts :

- stabilise l'estimation du Lasso
- permet de sélectionner plus que $n$ variables
- prend en compte de la corrélation entre variables
  - tend à sélectionner des "blocs" de variables corrélées

# Pénalité Elastic-Net

Régression Lasso avec deux variables identiques $x_1$ et $x_2$ :

- $\{w_1 = 0\ ;\ w_2 = \beta\}$ ou $\{w_1 = \gamma\beta\ ;\ w_2 = (1-\gamma)\beta\}$ équivalent en terme de loss et pénalité

$\Rightarrow$ solution mal définie (ridge : répartit le poids - $\beta/2$)

5. Exemple tiré de Hastie et al. (2015).

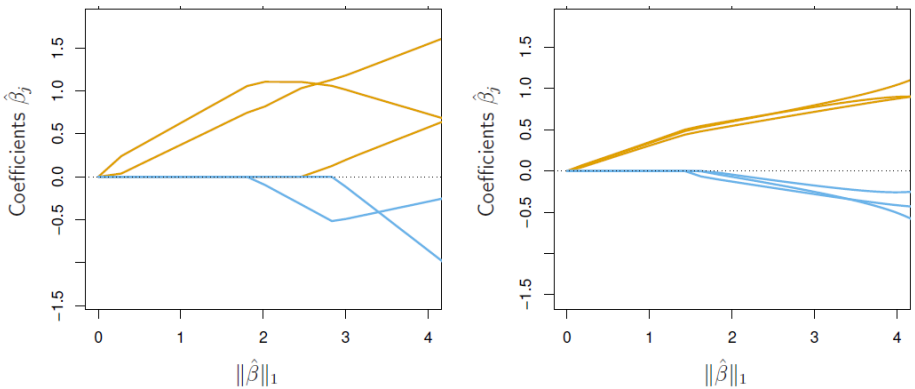
# Pénalité Elastic-Net

Régression Lasso avec deux variables identiques $x_1$ et $x_2$ :

- $\{w_1 = 0\ ;\ w_2 = \beta\}$ ou $\{w_1 = \gamma\beta\ ;\ w_2 = (1-\gamma)\beta\}$ équivalent en terme de loss et pénalité

$\Rightarrow$ solution mal définie (ridge : répartit le poids - $\beta/2$)

Plus généralement : instable si variables fortement corrélées [5]




- $2 \times 3$ variables très corrélées ($\sim 0.97$)
- gauche : Lasso
- droite : elastic-net

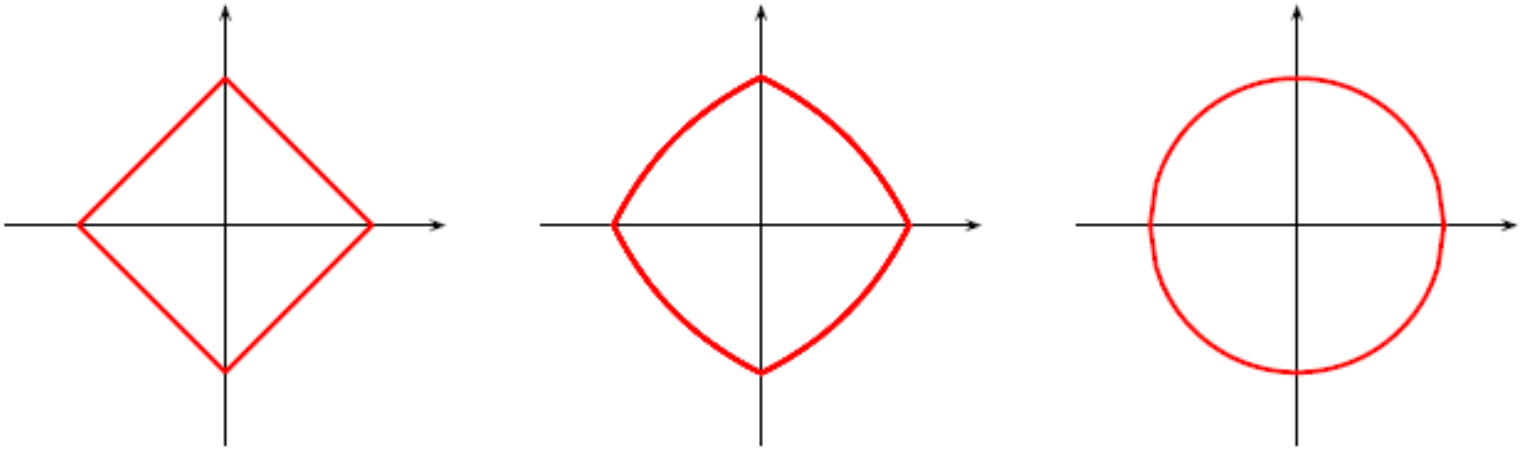
$\Rightarrow$ terme ridge : stabilise le lasso

$\Rightarrow$ tend à sélectionner des groupes de variables corrélées

5. Exemple tiré de Hastie et al. (2015).

# Pénalité Elastic-Net

Illustration : boules Lasso, Enet et Ridge en 2D

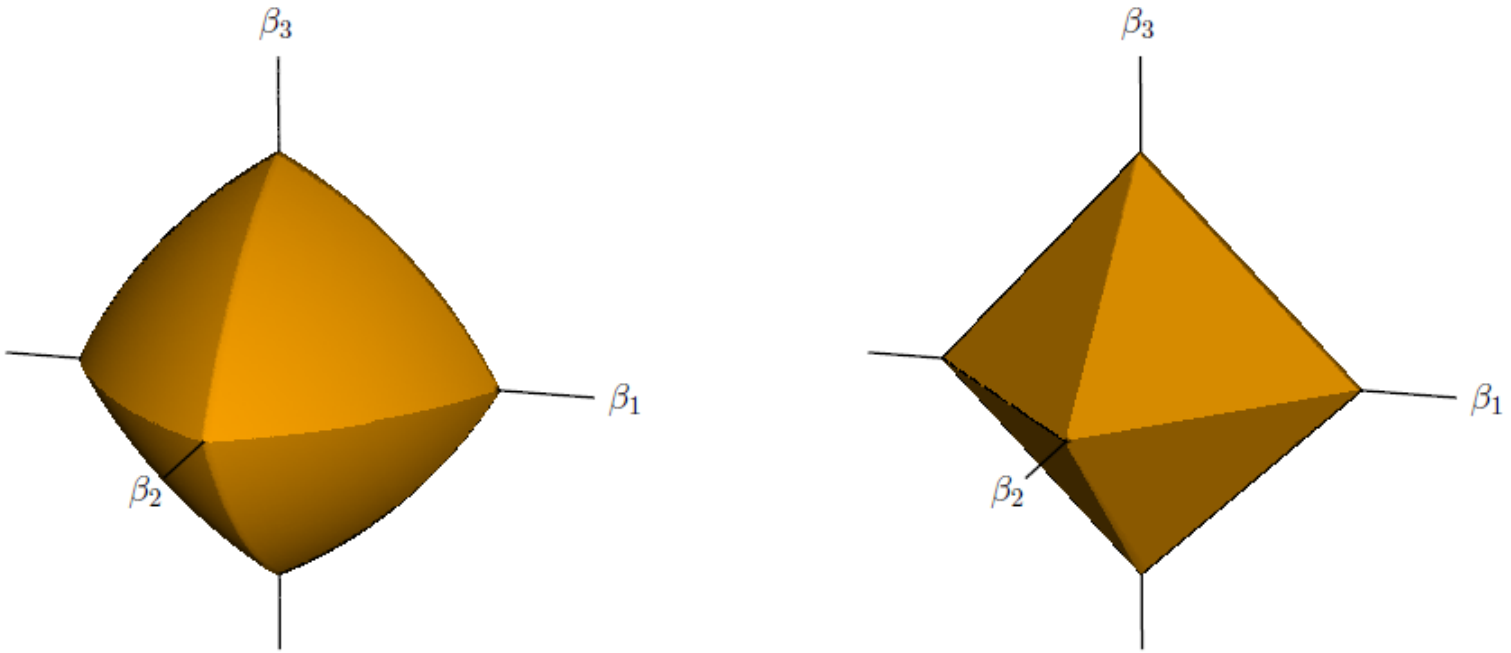


(a) $\ell_1$-ball, 2-D (b) elastic-net, 2-D (c) $\ell_2$-ball, 2-D

(image tirée d'une présentation de Julien Mairal)

# Pénalité Elastic-Net

Illustration : boules Lasso et elastic-net en 3D

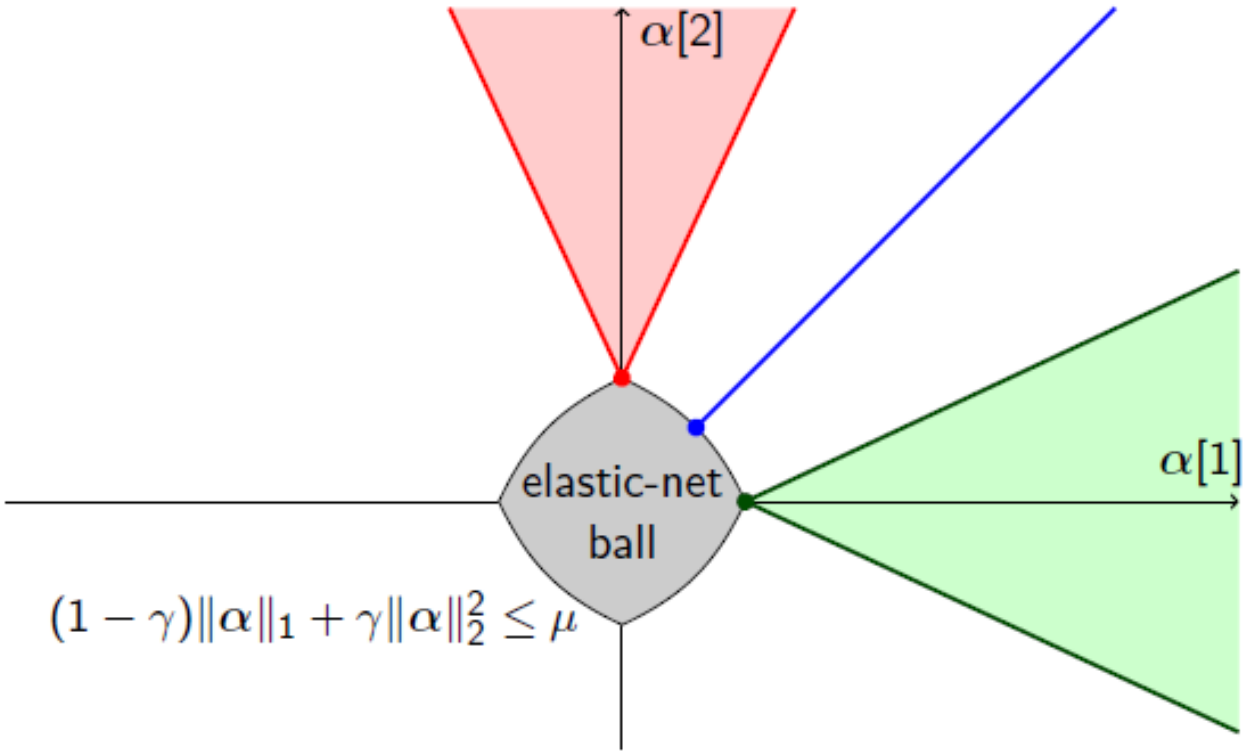



(image tirée de (Hastie et al., 2015), $\alpha = 0.7$)

# Pénalité Elastic-Net & sélection de variables [6]

## The elastic-net

vs other penalties

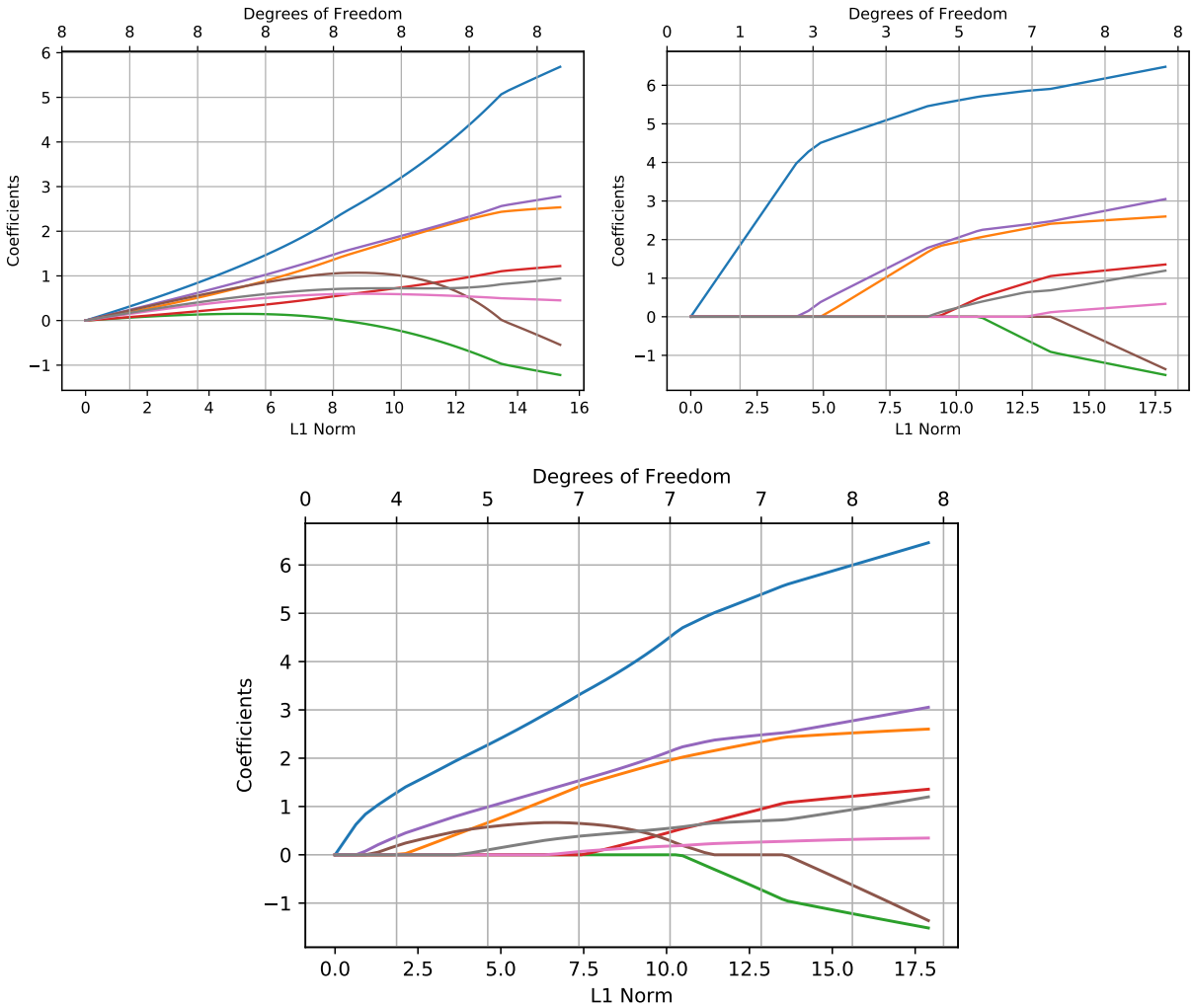



Julien Mairal | Sparse Estimation for Image and Vision Processing | 38/187

6. Slide tirée d'une présentation de Julien Mairal (INRIA Grenoble)

# Elastic-net - illustration

## Illustation : `Prostate` dataset, ridge & Lasso vs eNet



# Elastic-net - remarques

- Pénalité elastic-net : compromis Lasso / Ridge
  - stabilise le Lasso
  - permet de retenir plus que $n$ variables
  - effet de "grouping"
- Sélection de "groupes" de variables corrélées
  - relation $\alpha$ et degré de corrélation
- Un paramètre de plus à régler
- Solution unique dès qu'on considère un terme ridge
  - plus de solution mal définies quand prédicteurs identiques avec le Lasso
- Implémentation par coordinate-descent similaire
- Intéressant notamment pour données post-génomiques
  - fortes corrélations entre les variables (e.g., pathways)
  - compromis performance / inteprétabilité

# Group-Lasso

# Group-Lasso

Les données sont parfois intrinsèquement groupées :

- gènes intervant dans un même "pathway" biologique
- variables qualitatives encodées par one hot encoding
- ...

$\Rightarrow$ intéressant / souhaitable de les sélectionner ensemble

- interprétabilité et pertinence du modèle

# Group-Lasso

Les données sont parfois intrinsèquement groupées :

- gènes intervant dans un même "pathway" biologique
- variables qualitatives encodées par one hot encoding
- ...

$\Rightarrow$ intéressant / souhaitable de les sélectionner ensemble

- interprétabilité et pertinence du modèle

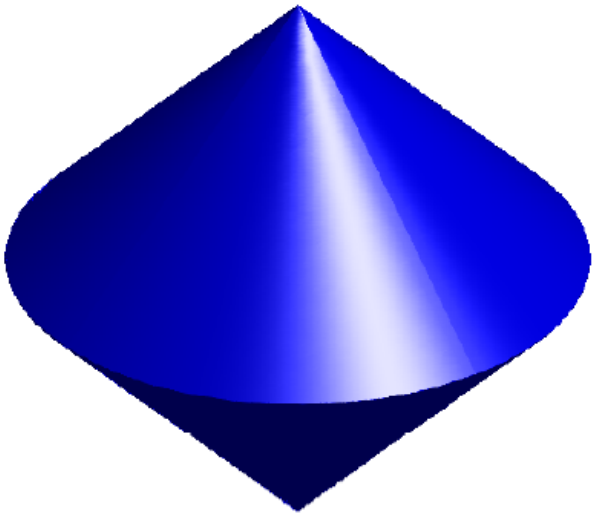
Pénalité Group-Lasso :

$$\boxed{\Omega_{\mathcal{G}}(w) = \sum_{g \in \mathcal{G}} \left|\left| w[g] \right|\right|_2}$$

où $\mathcal{G}$ définit une partition des variables en groupes.

# Group-Lasso - illustration

Illustration tirée de Bach et al. (2012) :



- 3 variables $x_1, x_2, x_3$
- $\mathcal{G} = \{[1,2],[3]\}$

$$\Rightarrow \Omega_{\mathcal{G}}(w) = ||w[1,2]||_2 + ||w[3]||_2$$
$$= ||w[1,2]||_2 + |w[3]|$$

$\Rightarrow$ disque sur le plan $(x_1, x2)$ :

- encourage $w_3 = 0$
- active $(w_1, w_2)$ en même temps (comportement ridge)

$\Rightarrow$ "pointe" selon $x_3$ :

- encourage $(w_1 = 0\ ;\ w_2 = 0)$

# Group-Lasso - interprétation

Pénalité Group-Lasso :

$$\boxed{\Omega_{\mathcal{G}}(w) = \sum_{g \in \mathcal{G}} \left|\left| w[g] \right|\right|_2}$$

où $\mathcal{G}$ définit une partition des variables en groupes.

Interprétation / remarques :

- peut-être interprétée comme la norme $L_1$ des groupes
- si groupes = variables uniques : le Lasso
  - $\mathcal{G}\{[1], [2], ..., [p]\}$, $\left|\left| w_i \right|\right|_2 = |w_i|$
- en pratique : pondération des groupes
- par rapport à elastic-net : groupes définis a priori

# Group-Lasso & apprentissage multi-tâches

Régression multivariée :

- $K$ réponses : $y_i = [y_i^{(1)}, ..., y_i^{(K)}] \in \mathbb{R}^K$
- chaque réponse = 1 modèle linéaire $w^{(k)} \in \mathbb{R}^p$
- fonction de perte = moindres carrés (sur les $K$ réponses)

# Group-Lasso & apprentissage multi-tâches

Régression multivariée :

- $K$ réponses : $y_i = [y_i^{(1)}, ..., y_i^{(K)}] \in \mathbb{R}^K$
- chaque réponse = 1 modèle linéaire $w^{(k)} \in \mathbb{R}^p$
- fonction de perte = moindres carrés (sur les $K$ réponses)

Stratégie classique : apprendre les modèles séparemment

$\Rightarrow$ peut s'écrire comme un problème global :

$$W = \operatorname{argmin} \sum_{k=1}^{K} \sum_{i=1}^{n} \left( y_i^{(k)} - \langle W_{.,k}, x_i \rangle \right)^2 + \lambda \sum_{k=1}^{K} \left|\left| W_{.,k} \right|\right|_1,$$

où $W \in \mathbb{R}^{p \times K}$ et $W_{.,k} = w^{(k)}$.

$\Rightarrow$ problèmes découplés (résolus indépendamment)

# Group-Lasso & apprentissage multi-tâches

Régression multivariée : approche "mono-tâche"

$$W = \text{argmin} \sum_{k=1}^{K} \sum_{i=1}^{n} \left( y_i^{(k)} - \langle W_{.,k}, x_i \rangle \right)^2 + \lambda \sum_{k=1}^{K} ||W_{.,k}||_1$$

$\Rightarrow$ chaque tâche "sélectionne" ses propres variables.

# Group-Lasso & apprentissage multi-tâches

Régression multivariée : approche "mono-tâche"

$$W = \text{argmin} \sum_{k=1}^{K} \sum_{i=1}^{n} \left( y_i^{(k)} - \langle W_{.,k}, x_i \rangle \right)^2 + \lambda \sum_{k=1}^{K} \left|\left| W_{.,k} \right|\right|_1$$

$\Rightarrow$ chaque tâche "sélectionne" ses propres variables.

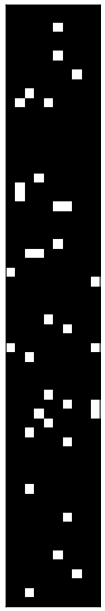
Pénalisation group-lasso selon les tâches :

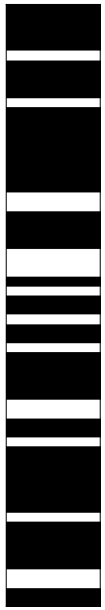
$$\sum_{k=1}^{K} \left|\left| W_{.,k} \right|\right|_1 \quad \Rightarrow \quad \sum_{j=1}^{p} \left|\left| W_{j,.} \right|\right|_2$$

- coefficients d'une variable groupés selon les tâches
- actives ou inactives dans toutes les tâches à la fois

$\Rightarrow$ apprentissage multi-tâche (résolu conjointement)

# Group-Lasso & apprentissage multi-tâches





$$\Omega(W) = \sum_{k=1}^{K} \left|\left| W_{.,k} \right|\right|_1 \qquad \Omega(W) = \sum_{j=1}^{p} \left|\left| W_{j,.} \right|\right|_2$$

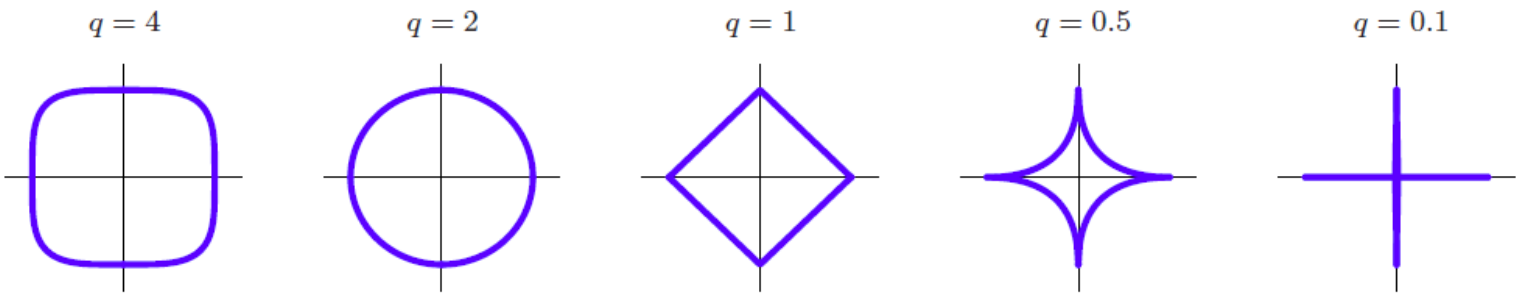
Applications :

- régression multivariée
- classification multiclasse par régression multinomiale
  - $P(Y = k|x) = e^{\langle w^{(k)}, x\rangle + b_k} / \sum_{l=1}^{K} e^{\langle w^{(l)}, x\rangle + b_l}$

# Extensions

# Pénalités $L_q$ et pénalités non convexes.

Pénalités $L_q$ [7] : $\boxed{\Omega_q(w) = \sum_{j=1}^{p} |w_j|^q}$



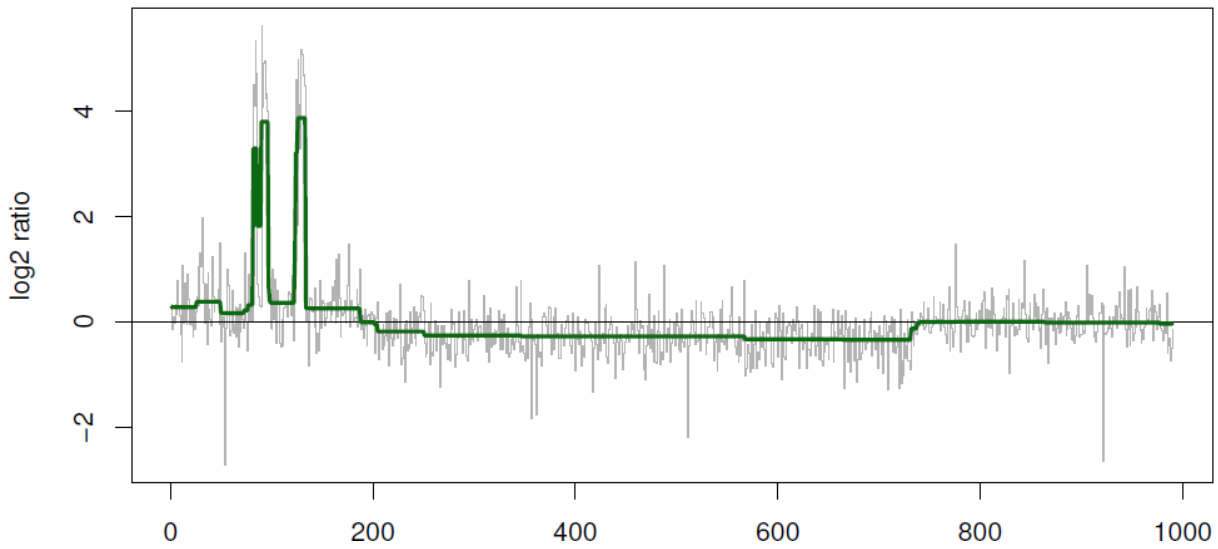

- $q = 2$ : ridge ; $q = 1$ : Lasso
- $q = 0$ : compte le nombre de variables $\neq 0$
  - problème combinatoire
- $q < 1$ : non-convexes
  - + sparse, mais plus de solution unique

$\Rightarrow$ Lasso : plus petite relaxation convexe de $L_0$.

7. Image tirée de Hastie et al. (2015)

# Fused Lasso

Fused-Lasso [8] : $$\Omega_{\text{FL}}(w) = \lambda_1 \sum_{j=1}^{p} |w_j| + \lambda_2 \sum_{j=2}^{p} |w_j - w_{j-1}|$$




- selectionne blocs de variables contigues
- applications en génomique et imagerie (extension 2D)

8. image tirée de Hastie et al. (2015)

# Et bien d'autres...

- Group-Lasso & "structured sparsity"
  - groupes "chevauchants" , imbriqués, organisés spatialement, ...
- Lasso & stability selection
  - procédure de ré-échantillonnage pour stabiliser la sélection de variables
- Lasso & inférence en haute dimension
  - "selective inférence" : de la sélection de variables à l'estimation de $p$-valeurs
- ..

# Mise en oeuvre Python

# Mise en oeuvre Python

`Scikit-Learn` : module `linear_model`

- Lasso, ridge, elastic-net
  - + versions par chemin de régularisation & validation croisée
  - + versions multi-tâches
- eco-système habituel
  - i.e., `fit, predict`
- différentes stratégies d'optimisation
  - coordinate descent, LARS
- ...

MAIS : essentiellement pour la régression

- moins développé pour régresion logistique

# Mise en oeuvre Python

Package `glmnet` :

- implémenté en Fortran, interfacé en `R` et `Python`
  - package python : **glmnet_python** [9]
  - package R : **glmnet**
- pénalité elastic-net (dont Lasso et ridge)
- modèles linéaires généralisés
  - linéaire, logistique, multinomiale
  - + versions multi-tâches
- approche de chemin de régularisation
  - optimisation par coordinate-descent + warm start
- fonctions de visualisation
- optimisation de $\lambda$ par validation croisée

$\Rightarrow$ TP basé sur `glmnet`.

9. https://web.stanford.edu/~hastie/glmnet_python/

# glmnet_python

Fonctions principales :

- `glmnet()` & `cvglmnet()`
  - construit les modèles + estime performance par VC
- `glmnetPlot()` & `cvglmnetPlot()`
  - visualisation des résultats
- `glmnetCoef()` & `cvglmnetCoef()`
  - extraits les coefficients pour une valeur de $\lambda$
- `glmnetPredict()` & `cvglmnetPredict()`
  - réalise la prédiction pour une valeur de $\lambda$

$\Rightarrow$ exemple détaillé de mise en oeuvre ici :
https://github.com/bbalasub1/glmnet_python/blob/master/test/glmnet_examples.ipynb

# glmnet_python, en pratique...

Installation de glmnet_python parfois difficile...

- compatibilité python-3, requiert librairie fortran, ...

On pourra faire le TP en R.

Ou alors essayer cette implémentation :
https://github.com/civisanalytics/python-glmnet

- compatible "écosystème" scikit-learn
- mais moins complète...

# glmnet_python : équivalent en R

Fonctions principales de la librairie R :

- glmnet() & cv.glmnet()
  - construit les modèles + estime performance par VC
- plot.glmnet() & plot.cv.glmnet()
  - visualisation des résultats
- coef.glmnet() & coef.cv.glmnet()
  - extraits les coefficients pour une valeur de $\lambda$
- predict.glmnet() & predict.cv.glmnet()
  - réalise la prédiction pour une valeur de $\lambda$

$\Rightarrow$ exemple détaillé de mise en oeuvre ici : https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html

# glmnet_python

## glmnet() & cvglmnet() - point clés :

- arguments : X, y, family et alpha
  - family : gaussian, binomial, multinomial, ...
  - + **ptype** pour CV : critère de performance (e.g., class ou auc pour classif)
- définit automatiquement une grille de valeurs pour $\lambda$
  - 100 valeurs définis par une heuristique (paramétrable)
- renvoie un dictionnaire contenant :
  - lambdau : grille de $\lambda$
  - beta : coefficients obtenus ($p \times 100$ pour régression)
  - en + pour cvglmnet() :
    - cvm : performance moyenne par fold (pour les $\lambda$)
    - cvsd, cvlo, cvup : écart-type + intervalle
    - lambda_min : $\lambda$ donnant meilleure performance
    - lambda_1se : $\lambda >$ lambda_min donnant meilleure performance à 1 SE près.

# glmnet_python

`glmnet()` & `cvglmnet()` - point clés :

- en pratique, `cvglmnet()` commence par fitter un modèle global, puis estime les performances par VC.
- les deux fonctions considèrent la même grille de valeurs
- `cvglmnet()`peut réaliser ensuite la prédiction à partir (1) du modèle global et (2) du meilleur $\lambda$.

# glmnet_python

`glmnetPlot()` & `cvglmnetPlot()` - point clés :

- `glmnetPlot()` : trace chemin de régularisation
- `cvglmnetPlot()` : trace performances de validation croisée en fonction de $\lambda$

# glmnet_python

`glmnetCoef()` & `cvglmnetCoef()` : extraction des coefficients obtenus à partir :

1. du modèle obtenu (par `glmnet()` ou `cvglmnet()`)
2. de la valeur de $\lambda$ souhaitée :
   - `glmnetCoef()` : une valeur numérique
   - `cvglmnetCoef()` : une valeur numérique ou `'lambda_min'` ou `'lambda_1se'`

# glmnet_python

`glmnetPredict()` & `cvglmnetPredict()` : extraction des prédictions obtenues à partir :

1. du modèle obtenu (par `glmnet()` ou `cvglmnet()`)
2. de nouvelles données
3. de la valeur de $\lambda$ souhaitée :
   - `glmnetPredict()` : une valeur numérique
   - `cvglmnetPredict()` : une valeur numérique ou `'lambda_min'` ou `'lambda_1se'`
4. du type de prédiction souhaité :
   - `ptype='link'` : score du modèle linéaire
   - `ptype='response'` : probabilité associée (idem `link` pour régression)
   - `ptype='class'` : classe prédite (pas valable pour régression ou cox)

# Références

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Structured sparsity through convex optimization. *Statistical Science*, 27 :450–469, 2012.

T. Hastie, R. Tibshirani, and J.. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity*. CRC Press, 2015.