

# Modèles linéaires pénalisés & Lasso

Master parcours SSD - UE Apprentissage Statistique I

Pierre Mahé - bioMérieux & Université de Grenoble-Alpes

## Modèles linéaires pénalisés :

- ▶ un cadre général d'**apprentissage supervisé**
  - ▶ classification et régression
- ▶ basé sur des **modèles linéaires**
  - ▶ dont la régression logistique et les SVMs
- ▶ mettant en jeu un critère de **régularisation/pénalisation**
  - ▶ lié à la norme du vecteur de coefficients
- ▶ dont la pénalité **Lasso**
- ▶ ainsi que ses **extensions**
  - ▶ elastic-net, group-lasso, ...
- ▶ qui conduisent à des **modèles parcimonieux**
  - ▶ sélection de variables

Modèles linéaires  
pénalisés

Ridge

Lasso

Elastic Net

Group-Lasso

Extensions

Mise en oeuvre R

Références

1. Modèles linéaires pénalisés
2. Régressions Ridge
3. Pénalité Lasso
4. Pénalité Elastic-Net
5. Pénalité Goup-Lasso
6. Mise en oeuvre en R

# Modèles linéaires pénalisés

# Risque empirique pénalisé

Formulation de **risque empirique pénalisé** :

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

- ▶ **risque empirique** : erreur faite sur le jeu d'apprentissage
  - ▶ selon la fonction de coût  $L(y, f(x))$
- ▶ **penalisation** : pour contrôler le sur-apprentissage
  - ▶ risque empirique seul = risque de sur-apprentissage
    - ▶ e.g., haute dimension ou classes de fonctions complexes
  - ▶ pénalisation = régularisation

# Risque empirique pénalisé

Formulation de **risque empirique pénalisé** :

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

- ▶ **risque empirique** : erreur faite sur le jeu d'apprentissage
  - ▶ selon la fonction de coût  $L(y, f(x))$
- ▶ **penalisation** : pour contrôler le sur-apprentissage
  - ▶ risque empirique seul = risque de sur-apprentissage
    - ▶ e.g., haute dimension ou classes de fonctions complexes
  - ▶ pénalisation = régularisation

⇒ 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

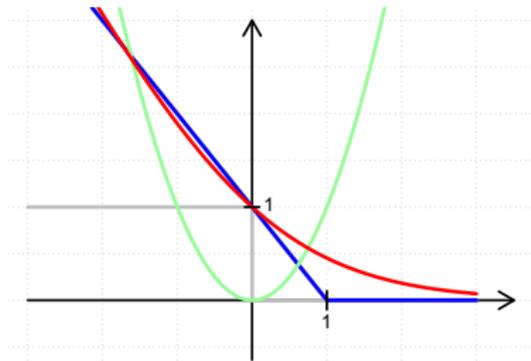
## Régression :

- ▶ **perte quadratique** :  $L(y, f(x)) = (y - f(x))^2$

## Classification :

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

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



## Régression logistique :

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

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

# Perte logistique

Régression logistique :

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

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

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 :

$$\begin{aligned}L(y, f(x)) &= \log(1 + e^{-yf(x)}) \\ &= -\log(P(Y = y|x)).\end{aligned}$$

⇒ utiliser la **perte logistique** = faire une **régression logistique**

# Perte logistique

Perte logistique :

$$\begin{aligned}L(y, f(x)) &= \log(1 + e^{-yf(x)}) \\ &= -\log(P(Y = y|x)).\end{aligned}$$

⇒ 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 :

$$\begin{aligned}L(y, f(x)) &= \log(1 + e^{-yf(x)}) \\ &= -\log(P(Y = y|x)).\end{aligned}$$

⇒ 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 : ~ performances
- ▶ logistique dérivable partout
- ▶ "charnière" des SVMs ⇒ vecteurs supports

# Fonctions de pénalisation incontournables

Pénalité Ridge (ou  $L_2$ ) :

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

Pénalité Lasso (ou  $L_1$ ) :

$$\Omega_{\text{Lasso}}(w) = \|w\|_1 = \sum_{j=1}^p |w_j|$$

# Fonctions de pénalisation incontournables

Pénalité Ridge (ou  $L_2$ ) :

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

Pénalité Lasso (ou  $L_1$ ) :

$$\Omega_{\text{Lasso}}(w) = \|w\|_1 = \sum_{j=1}^p |w_j|$$

**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

Combinaisons pertes / pénalités :

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

⇒ pour la suite on s'intéressera principalement aux **pertes quadratiques et logistiques**

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

# Régression Ridge

Régression linéaire avec perte quadratique et pénalité  $L_2$  :

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

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

Régression linéaire avec perte quadratique et pénalité  $L_2$  :

$$\begin{aligned} (w^*, b^*) &= \operatorname{argmin}_{w \in \mathbb{R}^p, b \in \mathbb{R}} \sum_{i=1}^n (y_i - f(x_i))^2 + \lambda \|w\|_2^2 \\ &= \operatorname{argmin}_{w \in \mathbb{R}^p, b \in \mathbb{R}} \sum_{i=1}^n (y_i - b - \sum_{j=1}^p w_j x_{ij})^2 + \lambda \sum_{j=1}^p w_j^2 \end{aligned}$$

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 :  $w^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top y$  où  $\mathbf{X}[i, j] = \tilde{x}_{ij}$

# Ridge regression

Régression linéaire vs régression ridge :

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

# Ridge regression

Régression linéaire vs régression ridge :

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

La matrice  $\mathbf{X}^T \mathbf{X}$  (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 :

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

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

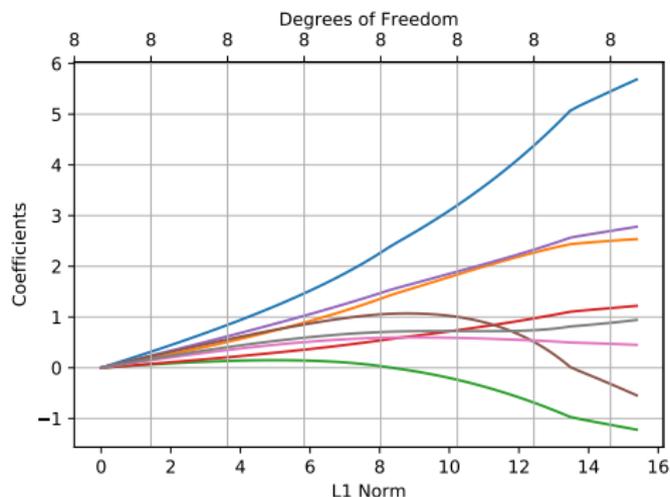
- ▶ non inversible si  $p > n$ 
  - ▶ résolution du problème impossible
- ▶ mal conditionnée si descripteurs corrélés
  - ▶ instabilité numérique, forte variance dans l'estimation

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

- ▶ permet de travailler en haute dimension ( $p \gg n$ )
  - ▶ 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

Illustration : Prostate dataset<sup>1</sup>



⇒ "shrinkage" : réduction de l'amplitude des coefficients

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

# 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

Interprétation bayésienne :

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

$\Rightarrow$  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

**Intepétation bayésienne** :

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

⇒ moindres carrés pénalisé : estimateur du MAP

⇒  $\lambda \sim 1/\sigma_0^2$

**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 :  $y \rightarrow \mathcal{N}(\langle w, x \rangle + b; \sigma^2)$
- ▶ régularisation  $L_2$  = a priori  $w_j \rightarrow \mathcal{N}(0, \sigma_0^2)$

⇒ moindres carrés pénalisé : estimateur du MAP

⇒  $\lambda \sim 1/\sigma_0^2$

**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 linéaire avec perte quadratique et pénalité  $L_1$  :

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

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

Régression linéaire avec perte quadratique et pénalité  $L_1$  :

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

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

Par rapport à pénalité ridge :

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

⇒ solution parcimonieuse (sparse) : sélection de variables.

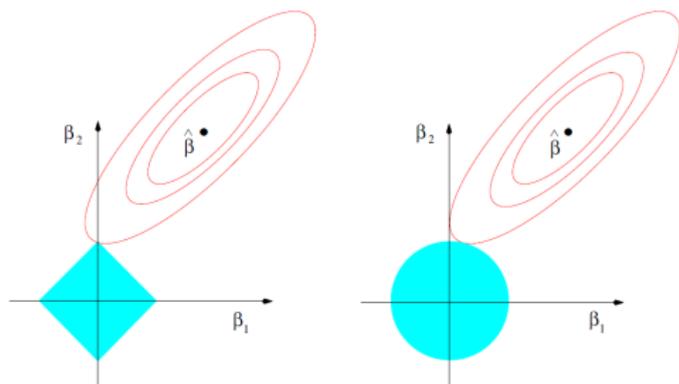
Formulation équivalente :

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

- ▶  $\forall \lambda, \exists t$  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 interpréter 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  $\|\beta\|_2^2 \leq t$

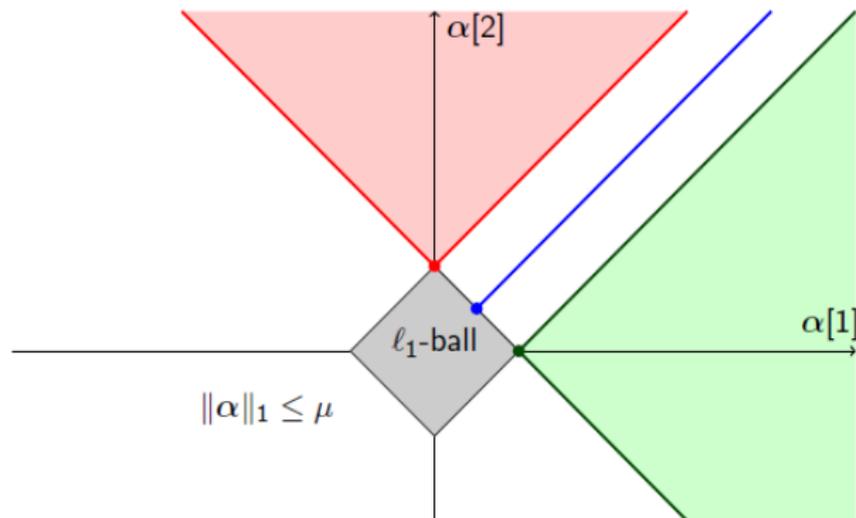
⇒ solution = projection de  $\hat{\beta}$  sur les régions admissibles

⇒ points de singularité de  $\|\cdot\|_1$  = solution parcimonieuse

# Pénalité Lasso & sélection de variables<sup>2</sup>

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

Regularizing with the  $\ell_1$ -norm

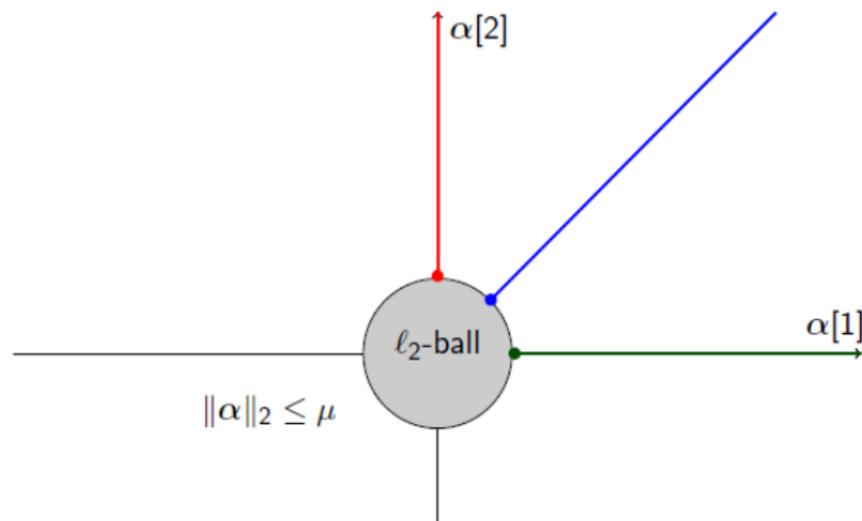


The projection onto a convex set is "biased" towards singularities.

# Pénalité Lasso & sélection de variables<sup>3</sup>

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

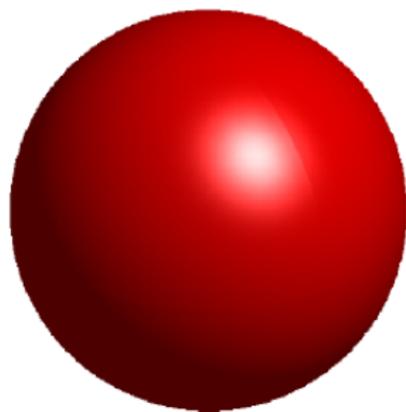
Regularizing with the  $\ell_2$ -norm



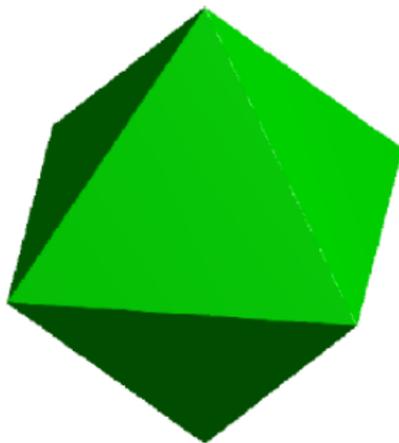
The  $\ell_2$ -norm is isotropic.

# Pénalité Lasso & sélection de variables

Boules  $L_1$  et  $L_2$  en 3D :<sup>4</sup>



(a)  $\ell_2$ -norm ball



(b)  $\ell_1$ -norm ball

⇒ sphère vs diamant.

On va considérer une **version sensiblement différente** :

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

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  $n$

On va considérer une **version sensiblement différente** :

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

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
- ⇒ ici : algorithme de "coordinate descent"
- ▶ simple, intuitif et efficace - à la base du package `glmnet`

Pour un seul prédicteur le problème s'écrit :

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

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

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

Pour un seul prédicteur le problème s'écrit :

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

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

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

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

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

**Solution** pour un prédicteur unique :

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

# Lasso - résolution

**Solution** pour un prédicteur unique :

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

⇒ **interprétation** :

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

**Solution** pour un prédicteur unique :

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

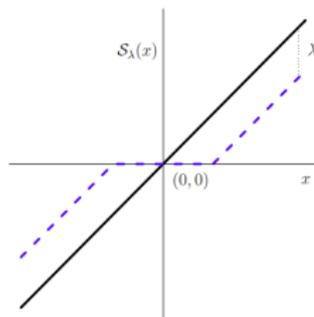
Solution pour un prédicteur unique :

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

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

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

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



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

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 :

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

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

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 :

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

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

⇒ 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

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

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

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

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

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

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

Modèles linéaires  
pénalisés

Ridge

**Lasso**

Elastic Net

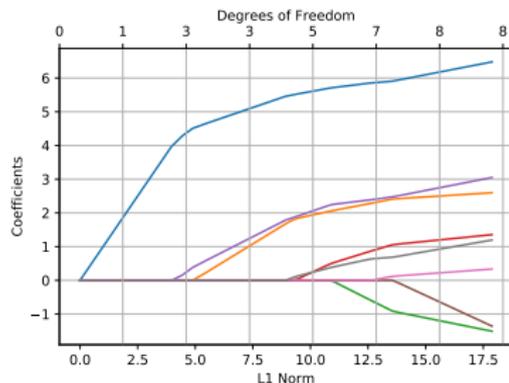
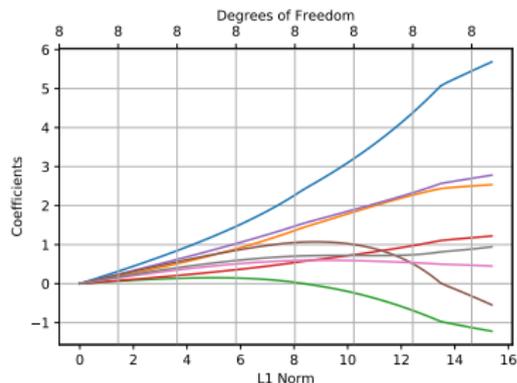
Group-Lasso

Extensions

Mise en oeuvre R

Références

## Illustration : Prostate dataset, ridge vs lasso



⇒ au bout du chemin : même solution = moindres carrés

▶ pas de pénalisation

⇒ 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**

- ▶ consistance, optimisation, ...

# Lasso - remarques

**Lasso** : méthode de sélection de variables "embedded"

- ▶ réalisée lors de la construction du modèle

Nombreux **développements théoriques**

- ▶ consistance, optimisation, ...

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

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

# Lasso - remarques

**Lasso** : méthode de sélection de variables "embedded"

- ▶ réalisée lors de la construction du modèle

Nombreux **développements théoriques**

- ▶ consistance, optimisation, ...

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

- ▶ alternatives : proximal, LARS/homotopie, ...
- ▶ adapté 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 : compromis  $L_1/L_2$

$$\Omega_\alpha(w) = \alpha \|w\|_1 + (1 - \alpha) \|w\|_2^2$$

ou :

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

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

Pénalité elastic-net : compromis  $L_1/L_2$

$$\Omega_\alpha(w) = \alpha \|w\|_1 + (1 - \alpha) \|w\|_2^2$$

ou :

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

$\Rightarrow \alpha = 1$  : 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

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

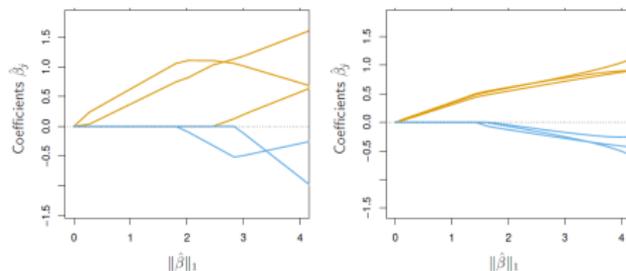
⇒ solution mal définie (ridge : répartit le poids -  $\beta/2$ )

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

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

Plus généralement : instable si variables fortement corrélées<sup>5</sup>

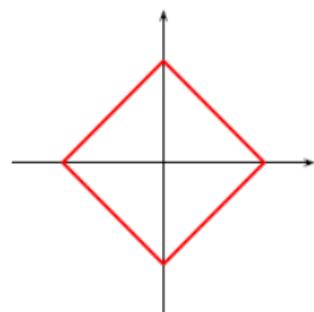


- ▶  $2 \times 3$  variables très corrélées ( $\sim 0.97$ )
- ▶ gauche : Lasso
- ▶ droite : elastic-net

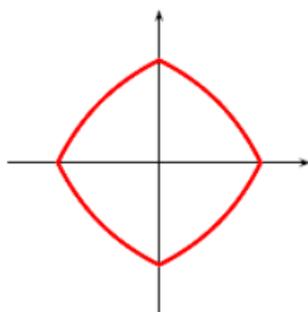
⇒ terme ridge : stabilise le lasso

⇒ tend à sélectionner des groupes de variables corrélées

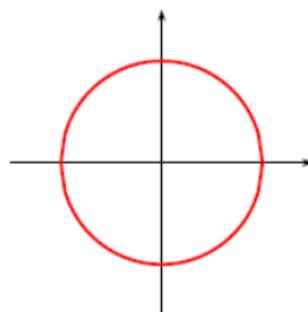
**Illustration** : boules Lasso, Enet et Ridge en 2D



(a)  $\ell_1$ -ball, 2-D



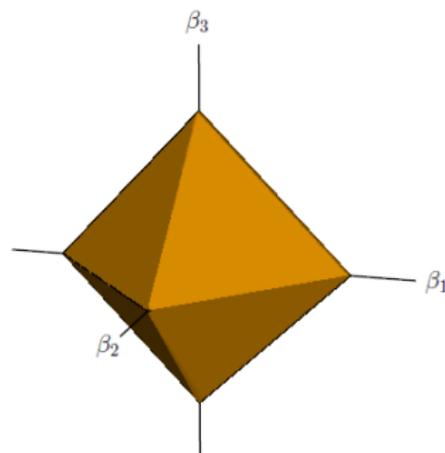
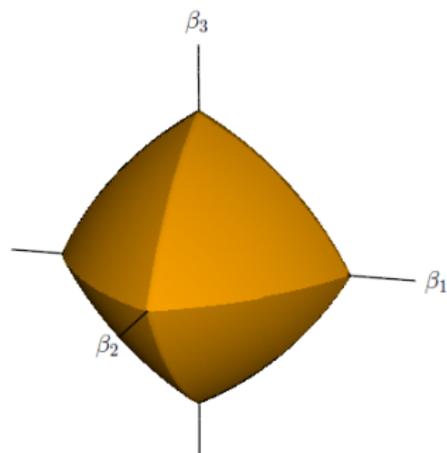
(b) elastic-net, 2-D



(c)  $\ell_2$ -ball, 2-D

(image tirée d'une présentation de Julien Mairal)

Illustration : boules Lasso et elastic-net en 3D

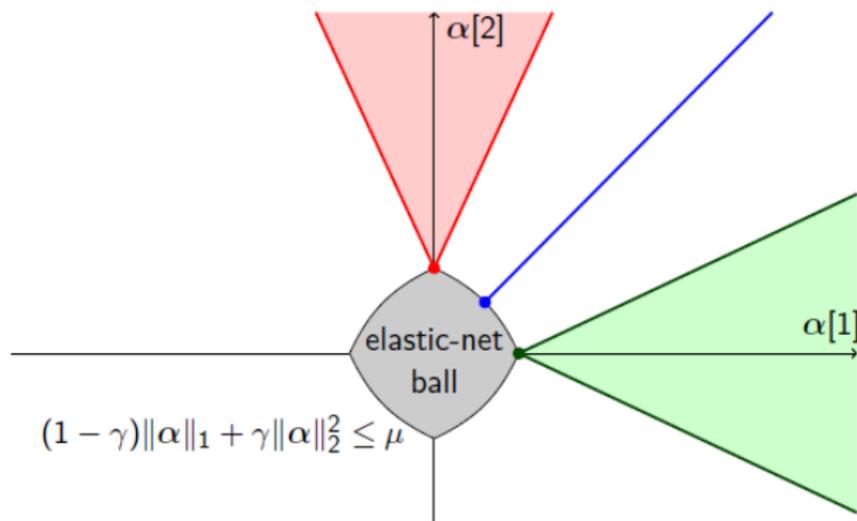


(image tirée de (Hastie et al., 2015),  $\alpha = 0.7$ )

# Pénalité Elastic-Net & sélection de variables<sup>6</sup>

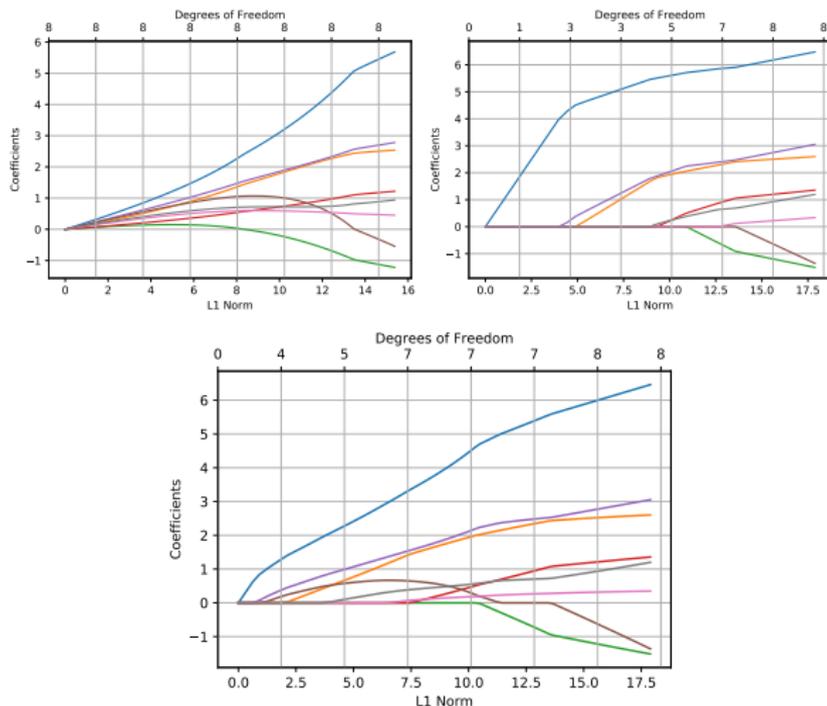
## The elastic-net

vs other penalties



# Elastic-net - illustration

Illustration : Prostate dataset, ridge & Lasso vs eNet



Outline

Apprentissage  
Statistique I

Modèles linéaires  
pénalisés

Ridge

Lasso

**Elastic Net**

Group-Lasso

Extensions

Mise en oeuvre R

Références

# 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 / interprétabilité

# Group-Lasso

Les données sont parfois **intrinsèquement groupées** :

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

⇒ intéressant / souhaitable de les **sélectionner ensemble**

- ▶ interprétabilité et pertinence du modèle

Les données sont parfois **intrinsèquement groupées** :

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

⇒ intéressant / souhaitable de les **sélectionner ensemble**

- ▶ interprétabilité et pertinence du modèle

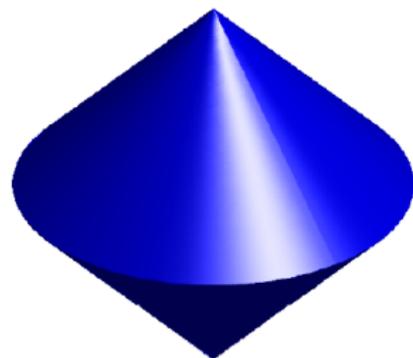
Pénalité **Group-Lasso** :

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

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

# Group-Lasso - illustration

Illustration tirée de Bach et al. (2012) :



▶ 3 variables  $x_1, x_2, x_3$

▶  $\mathcal{G} = \{[1, 2], [3]\}$

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

$\Rightarrow$  disque sur le plan  $(x_1, x_2)$  :

▶ encourage  $w_3 = 0$

▶ active  $(w_1, w_2)$  en même temps (comportement ridge)

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

▶ encourage  $(w_1 = 0 ; w_2 = 0)$

Pénalité **Group-Lasso** :

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

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
  - ▶  $\mathcal{G} \{[1], [2], \dots, [p]\}$ ,  $\left\| w_i \right\|_2 = |w_i|$
- ▶ en pratique : pondération des groupes
- ▶ par rapport à elastic-net : **groupes définis a priori**

# Group-Lasso & apprentissage multi-tâches

## Régression multivariée :

- ▶  $K$  réponses :  $y_i = [y_i^{(1)}, \dots, y_i^{(K)}] \in \mathbb{R}^K$
- ▶ 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 :  $y_i = [y_i^{(1)}, \dots, y_i^{(K)}] \in \mathbb{R}^K$
- ▶ 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 séparément

⇒ peut s'écrire comme un problème global :

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

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

⇒ problèmes découplés (résolus indépendamment)

# Group-Lasso & apprentissage multi-tâches

Régression multivariée : approche "mono-tâche"

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

⇒ chaque tâche "sélectionne" ses propres variables.

# Group-Lasso & apprentissage multi-tâches

Régression multivariée : approche "mono-tâche"

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

⇒ chaque tâche "sélectionne" ses propres variables.

Pénalisation **group-lasso** selon les tâches :

$$\sum_{k=1}^K \|W_{\cdot,k}\|_1 \Rightarrow \sum_{j=1}^p \|W_{j,\cdot}\|_2$$

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

⇒ apprentissage **multi-tâche** (résolu conjointement)

# Group-Lasso & apprentissage multi-tâches

Outline

Apprentissage  
Statistique I

Modèles linéaires  
pénalisés

Ridge

Lasso

Elastic Net

**Group-Lasso**

Extensions

Mise en oeuvre R

Références



$$\Omega(W) = \sum_{k=1}^K \left\| W_{\cdot, k} \right\|_1$$



$$\Omega(W) = \sum_{j=1}^p \left\| W_{j, \cdot} \right\|_2$$

Applications :

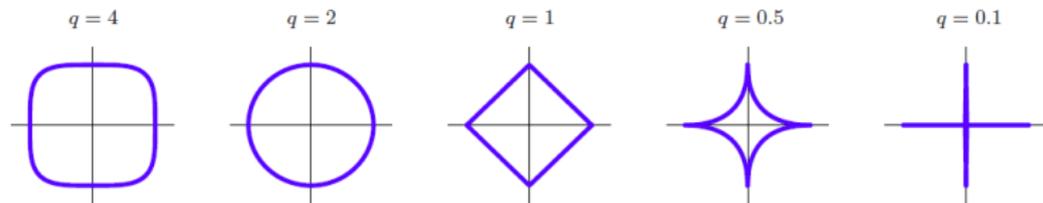
- ▶ régression multivariée
- ▶ classification multiclass par régression multinomiale

- ▶  $P(Y = k|x) = e^{\langle w^{(k)}, x \rangle + b_k} / \sum_{l=1}^K e^{\langle w^{(l)}, x \rangle + b_l}$

# Extensions

# Pénalités $L_q$ et pénalités non convexes.

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

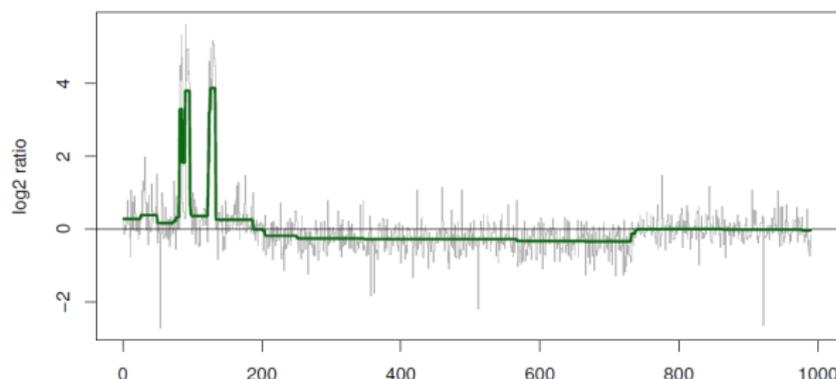


- ▶  $q = 2$  : ridge ;  $q = 1$  : Lasso
- ▶  $q = 0$  : **compte** le nombre de variables  $\neq 0$ 
  - ▶ problème combinatoire
- ▶  $q < 1$  : **non-convexes**
  - ▶ + sparse, mais plus de solution unique

⇒ **Lasso** : plus petite relaxation convexe de  $L_0$ .

Fused-Lasso<sup>8</sup> :

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



- ▶ sélectionne blocs de **variables contigues**
- ▶ applications en génomique et imagerie (extension 2D)

Modèles linéaires  
pénalisés

Ridge

Lasso

Elastic Net

Group-Lasso

**Extensions**

Mise en oeuvre R

Références

# 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 R

Package `glmnet` :

- ▶ implémenté en Fortran, interfacé en R
- ▶ 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
  - ▶ fournit un ensemble de modèles (plusieurs valeurs de  $\lambda$ )
- ▶ fonctions de `visualisation`
- ▶ optimisation de  $\lambda$  par `validation croisée`

⇒ `TP` : prise en main de `glmnet`.

Fonctions principales du package glmnet :

- ▶ `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  $\lambda$

⇒ exemple détaillé de mise en oeuvre ici : [https://web.stanford.edu/~hastie/glmnet/glmnet\\_alpha.html](https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html)

`glmnet()` & `cv.glmnet()` - point clés :

- ▶ arguments :  $X$ ,  $y$ , `family` et `alpha`
  - ▶ `family` : gaussian, binomial, multinomial, ...
  - ▶ + **type.measure** 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 une liste contenant :
  - ▶ **lambda** : grille de  $\lambda$
  - ▶ **beta** : coefficients obtenus ( $p \times 100$  pour régression)
  - ▶ en + pour `cv.glmnet()` :
    - ▶ **cvm** : performance moyenne par fold (pour les  $\lambda$ )
    - ▶ `cvstd`, `cvlo`, `cvup` : écart-type + intervalle
    - ▶ **lambda.min** :  $\lambda$  donnant meilleure performance
    - ▶ **lambda.1se** :  $\lambda > \text{lambda\_min}$  donnant meilleure performance à 1 SE près.

`glmnet()` & `cv.glmnet()` - point clés :

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

`plot.glmnet()` & `plot.cv.glmnet()` - point clés :

- ▶ `plot.glmnet()` : trace le chemin de régularisation
- ▶ `plot.cv.glmnet()` : trace les performances de validation croisée en fonction de  $\lambda$

`coef.glmnet()` & `coef.cv.glmnet()` : extraction des coefficients obtenus à partir :

1. du modèle obtenu (par `glmnet()` ou `cv.glmnet()`)
2. de la valeur de  $\lambda$  souhaitée :
  - ▶ `coef.glmnet()` : une valeur numérique
  - ▶ `coef.cv.glmnet()` : une valeur numérique ou 'lambda.min' ou 'lambda.1se'

`predict.glmnet()` & `predict.cv.glmnet()` : extraction des prédictions obtenues à partir :

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

- 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.