université PARIS-SACLAY

CentraleSupélec

Méthodes de screening pour les moindres carrés non-négatifs

MERJANI AYMANE
MOUQED MOHAMED REDA
AHMED EL BAJDALI
BADR SAISSI
SAAD AAFFOUTE

2023

CentraleSupélec | université PARIS-SACLAY

Encadrants :
CLEMENT ELVIRA

CentraleSupélec | université PARIS-SACLAY

Résumé

Ce projet s'intéresse principalement à l'étude d'une méthode d'accélération proposée par El Ghaoui pour la relaxation convexe de problèmes de représentations parcimonieuses, baptisé safe screening. Cette procédure repose sur deux éléments centraux. Le premier repose sur le fait que les solutions de problèmes convexes contiennent un très grand nombre de zéros. Le deuxième élément est la connaissance de la position des zéros de la solution va permettre de transformer le problème en un deuxième plus simple et de plus petite dimension. L'étude de ce projet permettra de réduire le temps de calcul des algorithmes pour résoudre ce type de problèmes, la mémoire utilisée, la puissance calculatoire et l'empreinte carbone de la résolution du problème.

Table des matières

1 Introduction

1.1 Contexte

Dans un contexte où la gestion de données devient centrale, il est important de trouver des solutions afin de minimiser le stockage et de réduire les calculs. C'est donc dans cette optique que la méthode de screening est mise au point, afin d'identifier et de sélectionner rapidement les données pertinentes nécessaires. L'objectif est de réduire le set de données, sans en réduire la qualité, et ainsi économiser de l'espace, du temps et de l'énergie.

1.2 Notations

1.2.1 Applications vectorielles

Dans ce document, on utilisera les applications vectorielles suivantes (toutes ces applications sont des normes vectorielles sauf la première) :

$$\forall x \in \mathbb{R}^n, n \in \mathbb{N}_+^*$$

$$\|x\|_0 = \sum_{i=1}^{n} 1_{\{x_i \neq 0\}}$$

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

$$\|x\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}}$$

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

1.2.2 Modélisation

- $y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_m \end{pmatrix}$ est un vecteur de dimension m,

- $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{pmatrix}$ est le vecteur de dimension n des paramètres inconnus du modèle,

- $A = \begin{pmatrix} a_{11} & a_{12} & \dots & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & \dots & a_{2n} \\ \vdots & & & & \\ \vdots & & & & \\ & & & & \vdots \\ & & & & \vdots \\ a_{m1} & \dots & \dots & a_{mn-1} & a_{mn} \end{pmatrix}$ est une matrice de taille $m \times n$.

1.3 Nécessité d'un terme de régularisation

1.3.1 Overfitting et underfitting

Les deux problèmes les plus courants en Machine Learning sont : l'overfitting et l'underfitting. L'overfitting se produit lorsque le modèle est trop complexe et s'ajuste trop étroitement aux données d'entraînement, conduisant à une mauvaise généralisation sur de nouvelles données. En revanche, l'underfitting se produit lorsque le modèle est trop simple et ne parvient pas à saisir les motifs des données d'entraînement, entraînant également une mauvaise performance. Trouver le bon équilibre entre la complexité du modèle et la quantité de données d'entraînement est essentiel pour obtenir de bons résultats en machine learning.

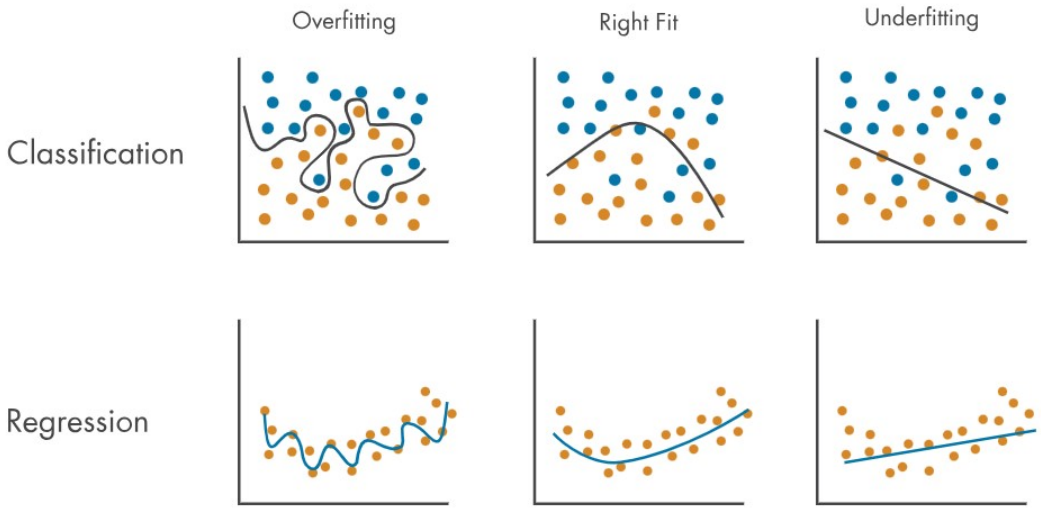


FIGURE 2 – Underfitting et Overfitting

1.3.2 Pénalisation ou régularisation

La pénalisation est une technique couramment utilisé dans les problèmes de régression pour éviter l'overfitting. Il exite plusieurs techniques de régularisation ou pénalisation :

- Pénalisation Lasso (L1) qui permet de détecter les paramètres moins importants donc le modèle est plus simple ce qui implique qu'il est moins sujet à l'overfitting.
- Pénalisation Ridge (L2) qui permet de construire un modèle plus homogène et donc moins sujet à l'overfitting.
- Pénalisation Elastic Net qui à la fois détecte les paramètres moins importants et qui les homogénéisent.

2 Estimateur Lasso

L'estimateur LASSO (Least Absolute Selection and Shrinkage Operator) est défini pour $\lambda > 0$ par :

$$\hat{x} = \underset{x \in \mathbb{R}^n}{\text{Argmin}} \left\{ \frac{1}{2}\|y - Ax\|_2^2 + \lambda \|x\|_1 \right\}.$$

Minimiser cette expression a un intérêt majeur si la matrice A est une matrice creuse (sparse) c'est à dire qu'elle contient un nombre important de valeurs nulles.

2.1 Convexité et non-différentiabilité

On pose l'application $f_\lambda : x \mapsto \frac{1}{2}\|y - Ax\|_2^2 + \lambda\|x\|_1$. L'objectif de cette partie sera donc d'étudier la fonction f_λ, qui va nous permettre de donner quelques propriétés de l'estimateur LASSO " $\widehat{x}(\lambda)$ "

La fonction f_λ est convexe, non différentiable.

On montrera également que la solution du problème de minimisation ne peut pas être unique.

Démonstration : Soient $x_1, x_2 \in \mathbb{R}^n$ et $t \in [0, 1]$,

$$f_\lambda\left(tx_1 + (1-t)x_2\right) = \frac{1}{2}\left\|y - A\left(tx_1 + (1-t)x_2\right)\right\|_2^2 + \lambda\left\|tx_1 + (1-t)x_2\right\|_1.$$

Alors par application de l'inégalité triangulaire pour le deuxième terme, on a que :

$$\left\|tx_1 + (1-t)x_2\right\|_1 \leq t\left\|x_1\right\| + (1-t)\left\|x_2\right\|_1$$

Et d'autre part,

$$\begin{aligned}
&\left\|y - A\left(tx_1 + (1-t)x_2\right)\right\|_2^2 \\
=&\left\|t\left(y - Ax_1\right) + (1-t)\left(y - Ax_2\right)\right\|_2^2 \\
=&t^2\left\|y - Ax_1\right\|_2^2 + (1-t)^2\left\|y - Ax_2\right\|_2^2 + 2t(1-t) < y - Ax_1, y - Ax_2 > \\
\leq&t^2\left\|y - Ax_1\right\|_2^2 + (1-t)^2\left\|y - Ax_2\right\|_2^2 + 2t(1-t)\left\|y - Ax_1\right\|_2\left\|y - Ax_2\right\|_2 \\
\leq&t^2\left\|y - Ax_1\right\|_2^2 + (1-t)^2\left\|y - Ax_2\right\|_2^2 + t(1-t)\left(\left\|y - Ax_1\right\|_2^2 + \left\|y - Ax_2\right\|_2^2\right) \\
=&t\left\|y - Ax_1\right\|_2^2 + (1-t)\left\|y - Ax_2\right\|_2^2
\end{aligned}$$

Dans les inégalités, nous avons utilisé respectivement l'inégalité de Cauchy-Schwarz ainsi que l'inégalité suivante : $uv \leq \frac{1}{2}\left(u^2 + v^2\right)$ pour tout $u, v \in \mathbb{R}$. Ainsi :

$$\|y - A(tx_1 + (1-t)x_2)\|_2^2 \leq t\|y - Ax_1\|_2^2 + (1-t)\|y - Ax_2\|_2^2$$

Il en vient que :

$$f_\lambda(tx_1 + (1-t)x_2) \leq tf_\lambda(x_1) + (1-t)f_\lambda(x_2).$$

$\Longrightarrow$ la fonction f_λ est bien convexe. Puisque $n > m$ alors $rg(A) < n$ (et donc la matrice A^TA est non inversible). En outre, la matrice Hessienne de f_λ qui est A^TA, n'est pas définie-positive, donc on en déduit que la fonction f_λ n'est pas strictement convexe. De plus, étant donné que pour tout $x \in \mathbb{R}^n$ la fonction norme $1, x \mapsto \|x\|_1$, n'est pas différentiable sur $\mathbb{R}^n$ alors on en déduit que f_λ ne l'est également pas.

2.2 Dualité de Fenchel

2.2.1 Problème dual

Posons : $g(z) = \frac{1}{2}\|y - z\|_2^2$ and $f(x) = \lambda\|x\|_1$. Cherchons donc :

$$\inf_{x\in\mathbb{R}^n} (f(x) + g(Ax))$$

Par Fenchel Rockafeller [3], comme g et f sont des fonctions convexes,propres, semi-continues inférieures (même continues car ils sont des normes) et $0 \in \text{int}\,(\mathbb{R}^n - A\mathbb{R}^n) = \text{int}\,(\mathbb{R}^n) = \mathbb{R}^n$

$$\inf_{x\in\mathbb{R}^n} (f(x) + g(Ax)) = -\inf_{\phi\in\mathbb{R}^n} (f^*(A^*\phi) + g^*(-\phi)) \tag{1}$$

Nous avons :

$$\begin{aligned}
g^*(u) &= \sup_x \left(\langle x \mid u\rangle - \frac{1}{2}\|y - x\|_2^2\right) \\
&= \sup_x \left(\langle x \mid u\rangle - \frac{1}{2}\|y\|_2^2 - \frac{1}{2}\|x\|_2^2 + \langle y, x\rangle\right) \\
&= \sup_x \left(\langle x \mid u + y\rangle - \frac{1}{2}\|x\|_2^2\right) - \frac{1}{2}\|y\|_2^2 \\
&= \sup_x \left(-\frac{1}{2}\|x - (u + y)\|_2^2 + \frac{1}{2}\|u + y\|_2^2\right) - \frac{1}{2}\|y\|_2^2 \\
&= \frac{1}{2}\|u + y\|_2^2 - \frac{1}{2}\|y\|_2^2 \text{ en prenant } x = u + y
\end{aligned}$$

Pour le cas de la fonction f, nous supposons d'abord que

$$f(x) = \|x\|_1$$

Nous voulons montrer que : $f^*(y) = I_{\|\cdot\|_\infty \le 1}(y) = \begin{cases} 0, & \|y\|_\infty \le 1 \\ +\infty, & \|y\|_\infty > 1 \end{cases}$

- Si $\|y\|_\infty \le 1$,

$$\langle x \mid y\rangle \le \|y\|_\infty \|x\|_1 \le \|x\|_1$$

Donc $f^*(y) = \sup_x (\langle x \mid y\rangle - \|x\|_1) = 0$

- Sinon, il existe i_0 tel que $\|y\|_\infty = |y_{i_0}| > 1$, On pose :

$$\text{y}_0 = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ y_{i_0} \\ 0 \\ \vdots \\ 0 \end{pmatrix} i_0\text{-eme position}$$

Soit $t > 0$, on a :

$$f^*(y) \geq \langle ty_0 \mid y \rangle - \|ty_0\|_1 = t(y_{i_0}^2 - |y_{i_0}|)$$

qui tend vers $+\infty$ lorsque t tend vers $+\infty$

Donc : $f^*(y) = \begin{cases} 0, & \|y\|_\infty \leq 1 \\ +\infty, & \|y\|_\infty > 1 \end{cases} = I_{\|\cdot\|_\infty \leq 1}(y)$

Pour le cas,

$$f(x) = \lambda \|x\|_1$$

Comme $\lambda > 0$, on a :

$$f^*(u) = \sup_x \left(\langle x \mid u \rangle - \lambda \|x\|_1\right) = \lambda \sup_x \left(\langle x \mid \lambda^{-1} u \rangle - \|x\|_1\right) = \lambda I_{\|\cdot\|_\infty \leq 1}\left(\lambda^{-1} u\right)$$

Donc le problème Lasso se transforme en :

$$\begin{aligned} \inf_{x \in \mathbb{R}^n} \left(f(x) + g(Ax)\right) &= - \inf_{\phi \in \mathbb{R}^n} \left(f^*\left(A^*\phi\right) + g^*(-\phi)\right) \\ &= - \inf_{\phi \in \mathbb{R}^n} \left(I_{\|\cdot\|_\infty \leq \lambda}(A^*\phi) + \frac{1}{2}\| - \phi + y\|_2^2 - \frac{1}{2}\|y\|_2^2\right) \\ &\Leftrightarrow \inf_{\phi \in \mathbb{R}^n} \frac{1}{2}\|y - \phi\|_2^2 \text{ tel que } \|A^*\phi\| \leq \lambda \end{aligned}$$

2.2.2 Relation entre $\hat{\phi}$ et $\hat{x}$

Soit $\hat{\phi}$ et $\hat{x}$ les minimiseurs. Comme la dualité est forte, nous avons alors l'égalité de Fenchel-Young pour f et pour g. On écrit le cas d'égalité Fenchel-Young pour g,

$$\begin{aligned} & g(A\hat{x}) + g^*(-\hat{\phi}) = \left\langle A\hat{x}, -\hat{\phi} \right\rangle \\ \Leftrightarrow\ & \frac{1}{2}\|y - A\hat{x}\|_2^2 + \frac{1}{2}\|y - \hat{\phi}\|_1^2 - \frac{1}{2}\|y\|_2^2 = -\langle A\hat{x}, \hat{\phi} \rangle \\ \Leftrightarrow\ & \langle y, -\hat{\phi} - A\hat{x} \rangle + \frac{1}{2}\|y\|_2^2 + \frac{1}{2}\|A\hat{x}\|_2^2 + \frac{1}{2}\|\hat{\phi}\|_2^2 = -\langle A\hat{x} | \hat{\phi} \rangle \\ \Leftrightarrow\ & \left\| -A\hat{x} + y - \hat{\phi}\right\|_2^2 = 0 \\ \Leftrightarrow\ & \hat{\phi} = y - A\hat{x} \end{aligned}$$

2.3 Règles de screening

Le but est minimiser la fonction définie par $f_\lambda : x \mapsto \frac{1}{2}\|y - Ax\|_2^2 + \lambda\|x\|_1$. Pour minimiser des fonctions usuelles, il faut passer par la dérivée de ces fonctions ou par leur différentielles, sauf que dans le cas étudié, la fonction n'est pas différentiable, d'où la nécessité de défnir une notion qui est proche de la notion de la différentiabilité.

2.3.1 Sous-différentiel

Quand une fonction convexe $f : I \subset \mathbb{R}^n \mapsto \mathbb{R}$ est différentiable en un point $x_0 \in I$ on a l'inégalité suivante :

$$f(x) \geq f(x_0) + \mathrm{d}f(x_0)(x - x_0), \forall x \in I$$

où $\mathrm{d}f(x_0) \in \mathbb{R}^n$ est la différentielle de f au point x_0.

Notons que la fonction f_λ n'est pas différentiable à cause de la norme 1, introduisons la notion de sous-différentiel.

Soit f une fonction convexe définie sur I, un ouvert de $\mathbb{R}^n$, à valeur dans $\mathbb{R}$. Un sous-gradient de $f : I \mapsto \mathbb{R}$ en un point x_0 de I est un vecteur $S \in \mathbb{R}^n$ tel que, pour tout x appartenant à I :

$$f(x) \geq f(x_0) + S(x - x_0)$$

L'ensemble des tous les sous-gradients est appelé sous-différentiel de la fonction f en x_0, noté $\partial f(x_0)$.

Voici une illustration du concept du sous-différentiel :

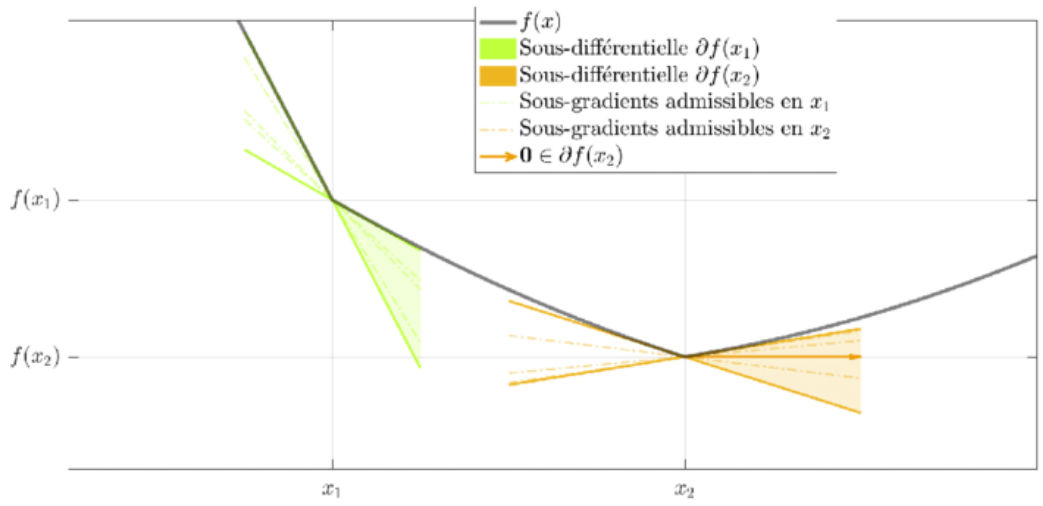


FIGURE 3 – Sous-différentiel

2.3.2 Sous-différentiel de la fonction f_λ

Il suffit de trouver le sous-différentiel de la norme 1 pour trouver le sous différentiel de la fonction f_λ vu que la norme 2 est différentiable et nous savons calculer sa différentielle.

Nous nous focalisons maintenant sur le calcul du sous-différentiel de la norme 1, pour cela nous admettons le résultat affirmant que le sous-différentiel d'une somme finie de fonctions sous-différentiables est la somme des sous-différentiels de ces fonctions.

Nous pouvons donc nous restreindre à calculer le sous-différentiel des fonctions projections sur la base canonique : $f_i : \mathbb{R}^n \longrightarrow \mathbb{R} \quad , \forall i \in \{1, 2, \dots, n\}$.

$$x \longmapsto |x_i|$$

Soit $\tilde{x} \in \mathbb{R}^n$, nous cherchons $S \in \mathbb{R}^n$ tel que :

$$|x_i| \geq |\tilde{x}_i| + S(x - \tilde{x}), \quad \forall x \in \mathbb{R}^n.$$

En particulier pour $x = \tilde{x} + e_j$ où e_j est le vecteur canonique et $j \in \{1, 2, \dots, n\}$, $j \neq i$, on a donc :

$$\begin{aligned} &|\tilde{x}_i| \geq |\tilde{x}_i| + S(x - \tilde{x}) \\ \Rightarrow &0 \geq S \times e_j \\ \Rightarrow &0 \geq S_j. \end{aligned}$$

Avec le même raisonnement pour $x := \tilde{x} - e_j$, nous avons aussi :

$$0 \leq S_j$$

Donc $S_j = 0, \forall j \in \{1, 2, \dots, n\}, j \neq i$.

Il suffit donc de calculer le sous différentiel de la fonction valeur absolue

Cette fonction est dérivable pour tout x non nul et sa dérivé vaut $\text{sign}(x)$. Calculons son sous-différentiel en zéro, nous cherchons $s \in \mathbb{R}$ tel que :

$$\begin{aligned} &|x| \geq |0| + s(x - 0), \quad \forall x \in \mathbb{R} \\ \Leftrightarrow &|x| \geq sx, \quad \forall x \in \mathbb{R} \\ \Rightarrow &\begin{cases} x \geq sx, \forall x \in \mathbb{R}^+ \\ -x \geq sx, \forall x \in \mathbb{R}^- \end{cases} \\ \Rightarrow &\begin{cases} s \leq 1, & \forall x \in \mathbb{R}^+ \\ s \geq -1, & \forall x \in \mathbb{R}^- \end{cases} \\ \Rightarrow &s \in [-1, 1]. \end{aligned}$$

Nous avons donc le sous-différentiel de $f_\lambda(x)$ est :

$$\partial f_\lambda(x) = -A^T(y - Ax) + \lambda \left\{ u \in \mathbb{R}^n : u_l = \begin{cases} [-1, 1], & \text{si } x_l = 0 \\ \text{sign}(x_l), & \text{sinon} \end{cases}, l \in \{1, 2, \dots, n\} \right\}.$$

2.3.3 Règles de screening

Soit $\hat{x}$ solution admissible, en annulant le terme $\partial f_\lambda(\hat{x})$, nous avons :
$\partial f_\lambda(\hat{x}) = 0 \Rightarrow \forall l \in [1,n], a_l^\top(y - A\hat{x}) = \lambda u_l$.
$\Rightarrow a_l^\top(y - A\hat{x}) \in \begin{cases} [-\lambda, \lambda] & \text{si } \hat{x}_l = 0 \\ \lambda \operatorname{sign}(\hat{x}_l) & \text{si } \hat{x}_l \neq 0 \end{cases}$
Nous concluons les règles de screening suivantes :

$$\text{si } \hat{x}_l = 0 : \left|a_l^\top(y - A\hat{x})\right| \leq \lambda \text{ et si } \hat{x}_l \neq 0 : \left|a_l^\top(y - A\hat{x})\right| = \lambda$$

2.4 Expression de λ_{max}

0 est un minimiseur de $f_\lambda \Leftrightarrow (\forall h > 0)(\forall v \neq 0) f_\lambda(hv) \geqslant f_\lambda(0)$.

$$(\forall h > 0)(\forall v \neq 0)$$
$$f_\lambda(hv) - f_\lambda(0) = \frac{1}{2}\|hAv - y\|_2^2 + \lambda\|hv\|_1 - \frac{1}{2}\|y\|_2^2$$

Donc,

$$\forall h > 0 \quad \forall v \neq 0$$

$$\frac{f_\lambda(hv) - f_\lambda(0)}{h} = \frac{1}{2}\left(h^2\|Av\|_2^2 - 2hv^\top A^\top y\right) + \lambda\|v\|_1$$

pour $v \neq 0$ donné,

$$(\forall h > 0)\frac{f_\lambda(hv) - f_\lambda(0)}{h} \Leftrightarrow (\forall h > 0)\frac{1}{2}\left(h\|Av\|_2^2 - 2v^\top A^\top y\right) + \lambda\|v\|_1 \geqslant 0$$
$$\Leftrightarrow \frac{1}{2}\left(0 - 2v^\top A^\top y\right) + \lambda\|v\|_1 \geqslant 0$$

(L'implication directe se justifie en faisant tendre h vers 0, l'autre implication découle de la positivité du premier terme).
Donc pour $v \neq 0$ donné,

$$(\forall h > 0 \quad f_\lambda(hv) - f_\lambda(0) \geqslant 0) \Leftrightarrow \lambda\|v\|_1 \geqslant v^\top A^\top y$$
$$\Leftrightarrow \lambda \geqslant \frac{v^\top A^\top y}{\|v\|_1}$$

Nous en déduisons donc que

$$0 \text{ minimise } f_\lambda \Leftrightarrow \forall v \neq 0 \quad \lambda \geqslant \frac{V^\top A^\top y}{\|v\|_1}$$

$$\text{On pose } \psi : v \mapsto \frac{V^\top A^\top y}{\|v\|_1}$$

$$\forall v \neq 0 \quad |\psi(v)| \leqslant \frac{\left|\sum_i v_i \left(A^\top y\right)_i\right|}{\sum_i |v_i|}$$
$$\leqslant \frac{\sum_i |v_i| \cdot \left\|A^\top y\right\|_\infty}{\Sigma_i |v_i|} \leqslant \left\|A^\top y\right\|_\infty$$

Et en prenant le vecteur $v = \text{sign}(A^\top y)$, nous obtenons que $\max_v \psi(v) = \|A^\top y\|_\infty$, et donc $\lambda_{\max} = \|A^\top y\|_\infty$.

2.5 Choix du pas de descente

Dans cette sous-section, nous discuterons du choix optimal du pas de descente en choisissant un pas de descente approprié en fonction de la constante de Lipschitz de la fonction $f(x) = \frac{1}{2}||Ax - y||_2^2$.

Commençons par établir l'expression de cette constante de Lipschitz. Le gradient de la fonction $f(x)$ est donné par :

$$\nabla f(x) = A^T(Ax - y) \tag{2}$$

Nous nous apercevons que la constante de Lipschitz du gradient est justement égale à $\sup_{||x||_2=1} ||A^T Ax||_2$, ou encore à la norme opérateur (avec des normes 2) de la hessienne de la fonction f qui est donné par :

$$\nabla^2 f(x) = A^T A \tag{3}$$

La constante de Lipschitz du gradient de la fonction est égale à la plus grande valeur propre de la matrice hessienne :

$$\sigma_f = \rho_{\max}(A^T A) \tag{4}$$

Considérons une étape de la descente de gradient à partir d'un point x vers un point z :

$$z = x - \alpha \nabla f(x) \tag{5}$$

Nous avons :

$$f(z) - f(x) \leq \langle \nabla f(x), z - x \rangle + \frac{\sigma_f}{2}||z - x||^2 \tag{6}$$

En substituant z par l'expression de l'étape de la descente de gradient, nous obtenons :

$$f(z) - f(x) \leq -\alpha||\nabla f(x)||^2 + \frac{\sigma_f}{2}\alpha^2||\nabla f(x)||^2 \tag{7}$$

La différence $f(z) - f(x)$ est négative, à condition que :

$$\frac{\sigma_f}{2}\alpha^2 - \alpha \leq 0 \tag{8}$$

Ce qui donne la condition suivante pour le pas de descente α :

$$\alpha \leq \frac{2}{\sigma_f} \tag{9}$$

Cette condition assure que le pas de descente garantit une diminution suffisante de la fonction objectif à chaque étape de l'algorithme de descente de gradient.

$$\alpha \leq \frac{2}{\sigma_f} \tag{10}$$

où σ_f est la constante de Lipschitz du gradient de la fonction.

Pour fixer un choix de pas de descente optimal, nous pouvons considérer la valeur $\alpha = \frac{2t}{\sigma_f}$, où $t \in (0, 1)$.

$$f(z) - f(x) \leq -\alpha||\nabla f(x)||^2 + \frac{\sigma_f}{2}\alpha^2||\nabla f(x)||^2 \tag{11}$$

En substituant α par $\frac{2t}{\sigma_f}$, nous avons :

$$f(z) - f(x) \leq -\frac{2t}{\sigma_f}||\nabla f(x)||^2 + \frac{\sigma_f}{2}\left(\frac{2t}{\sigma_f}\right)^2||\nabla f(x)||^2 \tag{12}$$

Ou encore :

$$f(z) - f(x) \leq -\frac{2t}{\sigma_f}||\nabla f(x)||^2 + \frac{2t^2}{\sigma_f}||\nabla f(x)||^2 \tag{13}$$

i.e :

$$f(x) - f(z) \geq \frac{2t}{\sigma_f}||\nabla f(x)||^2 - \frac{2t^2}{\sigma_f}||\nabla f(x)||^2 \tag{14}$$

NOus avons donc :

$$f(x) - f(z) \geq \frac{2}{\sigma_f}t(1-t)||\nabla f(x)||^2 \tag{15}$$

La valeur de t peut être ajustée pour contrôler la vitesse de convergence de l'algorithme.

La parabole $t(1-t)$ atteint son maximum en $t = \frac{1}{2}$, et donc un choix optimal du pas de descente consiste à prendre $\alpha_{\text{optimal}} = \frac{1}{\sigma_f}$.

2.6 Descente du gradient proximal

Nous avons implémenté un solveur Lasso en utilisant l'algorithme de descente de gradient proximal. La descente de gradient proximal est une méthode d'optimisation pour résoudre des problèmes de minimisation convexes avec des termes de régularisation non différentiables.

2.6.1 Opérateur proximal

L'opérateur proximal joue un rôle clé dans l'algorithme de descente de gradient proximal. Pour une fonction convexe f, l'opérateur proximal est défini comme suit :

$$\text{prox}_f(v) = \arg\min_x \left\{ f(x) + \frac{1}{2}\|x - v\|_2^2 \right\} \tag{16}$$

L'opérateur proximal est utilisé pour effectuer des mises à jour des coefficients dans l'algorithme de descente de gradient proximal en minimisant une combinaison linéaire de la fonction objectif et de la distance euclidienne au point courant.

Dans le cas du Lasso, la fonction de régularisation est la norme ℓ_1. L'opérateur proximal de la norme ℓ_1 est donné par :

$$\text{prox}_{\lambda\|\cdot\|_1}(v) = \text{sign}(v) \cdot \max(|v| - \lambda, 0) \tag{17}$$

2.6.2 Algorithme de descente de gradient proximal

L'algorithme de descente de gradient proximal pour le Lasso est implémenté comme suit :

1. Initialiser les coefficients x à zéro et définir les paramètres de l'algorithme (nombre d'itérations et pas).
2. Pour chaque itération :
   (a) Calculer le gradient de la fonction objectif sans terme de régularisation par rapport à x
   (b) Mettre à jour les coefficients x en utilisant la fonction proximale de la norme ℓ_1. Pour ce faire, nous calculons d'abord $x - \alpha \cdot \text{grad}$, où α est le pas de gradient et grad est le gradient de la fonction objectif (en norme 2, sans terme de régularisation). Ensuite, nous appliquons la fonction proximale à ce résultat en utilisant $\alpha \cdot \lambda$ comme paramètre de régularisation :

$$x = \text{prox}_{\alpha\lambda\|\cdot\|_1}(x - \alpha \cdot \text{grad}) \tag{18}$$

   (c) Calculer la valeur de la fonction objectif pour les coefficients x mis à jour.

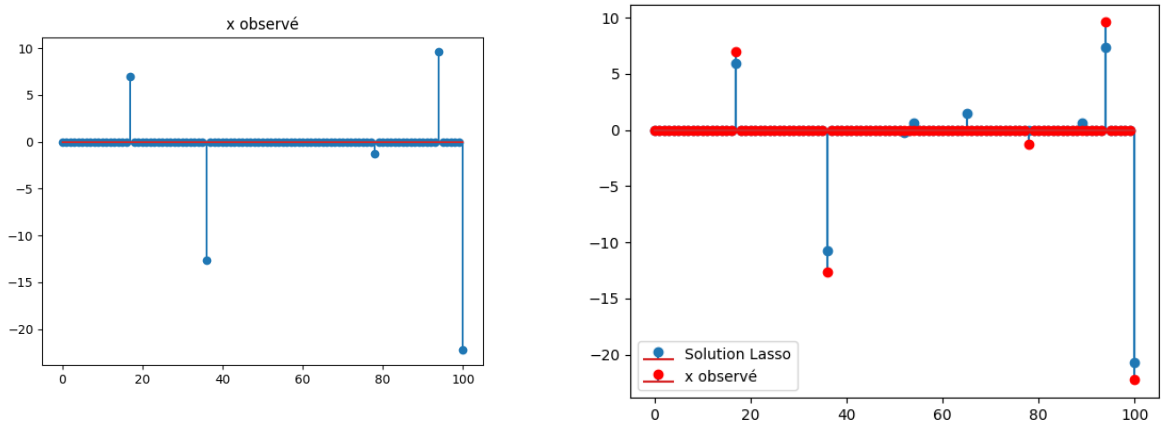
Le code Python correspondant est :

```
def proximal_Lasso(A, y, lmbda):
    nb_iter = 200
    alpha = 0.005
    x = np.zeros(n + 1)
    fct_objective = np.zeros(nb_iter)
    for i in range(nb_iter):
        grad = A.T @ (A @ x - y)
        x = proximal_l1(x - alpha * grad, alpha * lmbda)
        fct_objective[i] = 0.5 * np.linalg.norm(A @ x - y)**2 + lmbda*
                           np.sum(np.abs(x))
    return x, fct_objective
```

2.6.3 Résultats intermédiaires

Nous avons appliqué notre solveur Lasso à un ensemble de données synthétiques aléatoires et un certain nombre de composantes non nulles dans le vrai modèle. Les figures ci-dessous montrent les résultats obtenus avec cette implémentation, avec n = 100 et m = 25, et 5 composantes non nulles.

Remarque : dans ce cas, nous n'avons pas accordé beaucoup d'importance au critère portant sur le nombre de composantes non nulles (vu que la valeur de n est petite, et le coefficient de la pénalité associée aux composantes non nulles dans notre implémentation est seulement de 0,01. Mais cela peut être ajusté si besoin.



(a) Vrai modèle

(b) Solution Lasso

FIGURE 4 – Convergence de la fonction objective

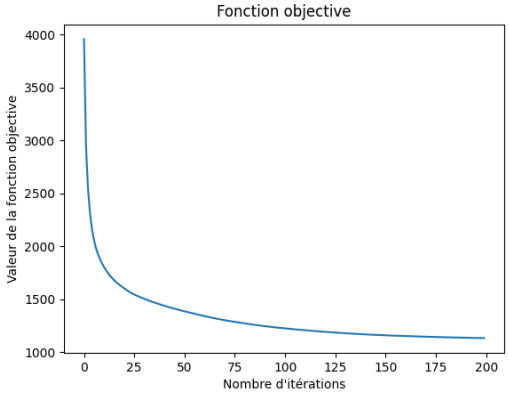


FIGURE 5 – Convergence de la fonction objective

2.7 Étude du Gap

Dans cette partie, nous allons introduire la notion de "duality gap" et en particulier la dualité forte définie dans le poly du cours théorème 15. Introduisons tout d'abord le changement de variable suivant : $\phi_t = \kappa_t(y - Ax_t)$ avec
$\kappa_t \in \mathbb{R}^+$, tel que $\lim_{t\to+\infty} \phi_t = \widehat{\phi}$, et $\lim_{t\to+\infty} x_t = \widehat{x}$.

Cette variable aura une importance algorithmique car nous avons :

$$\forall (x, \phi) \in \mathbb{R}^2 \ , f_\lambda(x) \geq -f_\lambda^*(\phi) \tag{19}$$

d'après l'inégalité de Fenchel-Young.

De plus, nous avons la condition d'admissibilité qui est la suivante :

$$0 \text{ solution} \Leftrightarrow \ \lambda \geq \ \left\|A^\top * y\right\|_\infty \tag{20}$$

On appellera par la suite : $\lambda_{\max} = \left\|A^\top * y\right\|_\infty$

Il s'agira par la suite de trouver l'expression de κ_t en utilisant la condition suivante :

$$\lambda \leq \ \kappa_t \left\|A^\top * \phi_t\right\|_\infty = \lambda_{\max} \tag{21}$$

Dès lors, nous pouvons poser : $\kappa_t = \frac{\frac{\lambda}{m}}{\left\|A^\top * \phi_t\right\|_\infty}$ avec $m \in \ \mathbb{N}$.

Il faut par ailleurs vérifier que cette expression convient.
Nous avons tout d'abord,

$$\left\|A^\top * \hat{\phi}\right\|_\infty = \max_l \left|a_l^\top * \hat{\phi}\right| = \lambda \tag{22}$$

et $\lim_{t\to+\infty}\phi_t = \widehat{\phi}$, qui doit être vérifiée lorsque $\lim_{t\to+\infty} x_t = \widehat{x}$.

Le calcul de limite donne à la fin : $\phi_t(\widehat{x}) = \frac{\widehat{\phi}}{m}$ ce qui donne $m = 1$. La condition d'amissibilté est vérifiée par construction de κ_t.

Nous voyons donc que l'expression de ϕ_t peut être simplifiée et donne donc

$$\phi_t = \frac{\lambda}{\|A^\top * \phi_t\|_\infty} * (y - Ax_t) \tag{23}$$

L'algorithme a donc pour but de trouver la solution qui nous intéresse à savoir $f_\lambda(\widehat{x})$ en minimisant l'écart entre cette valeur à savoir

$$\inf_{x\in\mathbb{R}^n} (\|f_\lambda(\widehat{x}) - f_\lambda(x)\|_\infty) \tag{24}$$

Or, nous n'avons pas accès à $\widehat{x}$ donc nous utiliserons la relation 19 afin de contourner le problème. Nous obtenons donc

$$\inf_{x\in\mathbb{R}^n} (\left\|f_\lambda^*(\widehat{\phi}) + f_\lambda(x)\right\|_\infty) \tag{25}$$

équivalent au problème suivant :

$$\inf_{x\in\mathbb{R}^n} (\left\|\lim_{t\to+\infty} f_\lambda^*(\phi_t) + f_\lambda(x)\right\|_\infty) \tag{26}$$

Sur python, nous pouvons simplifier le problème encore un peu en posons un critère ϵ aussi petit que l'on veut afin d'arrêter la boucle while ce qui se traduit donc par :

$$|f_\lambda^*(\phi_t) + f_\lambda(x)| \leq \epsilon \tag{27}$$

La condition d'admissibilité est par ailleurs vérifiée à chaque boucle grâce à la construction de κ_t.

3 Problème des moindres carrés non-négatifs

3.1 Utilité de ce problème

Le problème des moindres carrés non négatifs est une variante du problème des moindres carrés classique. Il est utilisé lorsque l'on souhaite trouver une solution qui est contrainte d'être non négative. L'objectif du problème des moindres carrés non négatifs est de minimiser la somme des carrés des écarts entre les valeurs observées et les valeurs prédites, tout en respectant la contrainte de non-négativité. Ce problème trouve des applications dans de nombreux domaines. Par exemple, en analyse de données, il peut être utilisé pour estimer les coefficients d'un modèle linéaire avec des contraintes de non-négativité. Dans le domaine de l'imagerie, il peut être utilisé pour la reconstruction d'images à partir de données incomplètes ou bruitées, en imposant que les valeurs des pixels soient non négatives.

3.2 Convexité et non-différentiabilité

Dans cette section, nous nous intéressons maintenant au problème des moindres carrés non-négatifs :

$$\hat{x} = \underset{x\in\mathbb{R}^n}{\text{Argmin}} \left\{ \frac{1}{2}\|y - Ax\|_2^2 + I_{[0,+\infty[^n}(x) \right\}.$$

Comme dans le cas du Lasso, la fonction objective est convexe ce qui permet d'appliquer les résultats de l'optimisation convexe, et elle est non-différentiable vu le deuxième terme de la fonction objective.

3.3 Dualité de Fenchel

Posons : $g(z) = \frac{1}{2}\|y - z\|_2^2$ et $f(x) = I_{[0,+\infty[^n}(x)$. Cherchons donc :

$$\inf_{x\in\mathbb{R}^n} (g(Ax) + f(x))$$

Par Fenchel Rockafeller [3], comme g et f sont des fonctions convexes,propres et semi-continues inférieurs car $\mathbb{R}^n_+$ est convexe fermé et $dom(Af) \cap dom(g) = dom(Af) \cap \mathbb{R}^n_+ \neq \emptyset$

$$\begin{aligned}\inf_{x\in\mathbb{R}^n} (f(x) + g(Ax)) &= - \inf_{\phi\in\mathbb{R}^n} \left(f^*\left(A^*\phi\right) + g^*(-\phi)\right) \\ &= - \inf_{\phi\in\mathbb{R}^n} \left(I_{]-\infty,0]^n}(A^*\phi) + \frac{1}{2}\|y - \phi\|_2^2 - \frac{1}{2}\|y\|_2^2\right)\end{aligned}$$

Comme la dualité est forte, nous avons le cas d'égalité dans l'inégalité de Fenchel -Young pour g (en utilisant la même preuve faite dans le problème Lasso) :

$$\begin{aligned} & g(A\hat{x}) + g^*(-\hat{\phi}) = \left\langle A\hat{x}, -\hat{\phi} \right\rangle \\ \Leftrightarrow\ & \hat{\phi} = y - A\hat{x}\end{aligned}$$

3.4 Sous-différentiel

Comme f n'est pas différentiable, onous calculons son sous-gradient qui est bien défini car f est convexe. Le sous-gradient S de $f(x)$ en x est défini par l'inégalité suivante pour tout $y \in \mathbb{R}^n$:

$$f(y) \geq f(x) + S(y - x)$$

Si $x_i > 0$ pour tout i, on a $f(x) = 0$. Dans ce cas, nous voulons que

$$f(y) \geq S(y - x)$$

— Si pour tout $y_i \geq 0$ (avec $y \in \mathbb{R}^n_+$), $f(y) = 0$, et nous cherchons S tel que

$$0 \geq S(y - x)$$

Choisissons donc $S_i = 0$ pour tout i.

— S'il existe i tel que $y_i < 0$, $f(y) = +\infty$, et l'inégalité est vraie pour n'importe quelle valeur de S. Nous choisissons donc $S = 0$.

S'il existe i tel que $x_i = 0$,

— Si pour tout $y_i \geq 0$ (avec $y \in \mathbb{R}^n_+$), $f(y) = 0$, donc nous cherchons S tel que

$$0 \geq S(y - x)$$

Et nous choisissons donc n'importe quelle $S_i \leq 0$.

— S'il existe i tel que $y_i < 0$, $f(y) = +\infty$, et l'inégalité est vraie pour n'importe quelle valeur de S.

Au final,

$$\partial f(x) = \left\{ u \in \mathbb{R}^n : u_l = \begin{cases} \mathbb{R}_- & \text{si } x_l = 0 \\ 0 & \text{sinon} \end{cases}, l \in \{1, 2, \ldots, n\} \right\}$$

3.5 Test de screening

Soit $\hat{x}$ solution admissible, en annulant le sous-gradient de la fonction objective :

$$\forall l \in [1, n], a_l^\top (y - A\hat{x}) \in \begin{cases} \mathbb{R}_- & \text{si } \hat{x}_l = 0 \\ 0 & \text{si } \hat{x}_l > 0 \end{cases}$$

Nous déduisons le test de screening :

$$a_l^\top (y - A\hat{x}) < 0 \Rightarrow x_l = 0$$

3.6 Sphère Gap pour les moindres carrés non-négatifs

Pour cette partie, nous avons choisi la méthode du gradient projeté au lieu du gradient proximal, dont la formule est la suivante :

$$x_{t+1} = x_t + \eta * (P_{rj}(x_t - \gamma * grad) - x_t). \tag{28}$$

Dans notre algorithme, nous avons choisi $\eta = 1$ pour le coefficient de relaxation et P_{rj} correspond à la projection de $x_t - \gamma * grad$ sur $\mathbb{R}^n_+$ (cf. Cours d'optimisation). De plus, l'étude mathématique de la sphère Gap est semblable à celle réalisée précedememnt pour l'exemple du Lasso et nous aboutissons donc au même résultat.
Ce qui devra être modifié est l'expression de ϕ_t qui rappelons-le, dans le problème du Lasso, dépendait de λ afin de la rendre admissible, d'où l'introduction dans l'algorithme du paramètre k_t.
Démonstration :
Pour que ϕ_t soit admissible, c'est-à-dire $[A^\top * \phi_t]_i \leq 0 \ \forall i \in \{1, 2, \ldots, n\}$, il faudra introduire un facteur tel que

$$\phi_t = y - k_t * A * x_t \tag{29}$$

Soit $y \in \mathbb{R}^m$ et $A \in \mathbb{R}^{m \times n}_+$,
Premier cas :
Supposons, $\forall i \in \{1, 2, \ldots, n\}$, $[A * x_t]_i \neq 0$,
Dès lors, nous avons

$$[y]_i - k_t * [A * x_t]_i \leq 0 \Leftrightarrow \frac{y_i}{[A * x_t]_i} \leq k_t \tag{30}$$

Nous pouvons donc généraliser en prenant

$$k_t = \max(\frac{\|\mathbf{y}\|_\infty}{\|\mathbf{A} * \mathbf{x_t}\|_\infty}, 1) \tag{31}$$

De plus, $\hat{\phi}$ est admissible, donc $[A^\top * \widehat{\phi}]_i \leq 0 \Leftrightarrow ||\mathbf{y}||_\infty \leq ||\mathbf{A} * \widehat{\mathbf{x}}||_\infty$,

d'où le choix de 1 dans l'expression de k_t permettant la convergence de ϕ_t vers $\widehat{\phi}$.

Deuxième cas :
$\exists i \in \{1, 2, \ldots, n\}$ tel que , $[A * x_t]_i = 0$.
L'algorithme devra donc s'arrêter car $\exists i$ tel que $[A^\top * \widehat{\phi}]_i \geq 0$ ce qui va renvoyer $+\infty$ dans l'expression de la fonction indicatrice.

4 Implémentation

4.1 Pas de descente

4.1.1 Théorie autour du pas

Dans cette sous-section, nous discuterons du choix optimal du pas de descente en choisissant un pas de descente approprié en fonction de la constante de Lipschitz de la fonction $f(x) = \frac{1}{2}||Ax - y||_2^2$.
Commencons par établir l'expression de cette constante de Lipschitz. Le gradient de la fonction $f(x)$ est donné par :

$$\nabla f(x) = A^T(Ax - y) \tag{32}$$

Nous nous apercevons que la constante de Lipschitz du gradient est justement égale à $\sup_{||x||_2=1} ||A^T Ax||_2$, ou encore à la norme opérateur (avec des normes 2) de la hessienne de la fonction $f(x)$ qui est donné par :

$$\nabla^2 f(x) = A^T A \tag{33}$$

La constante de Lipschitz du gradient de la fonction est égale à la plus grande valeur propre de la matrice hessienne :

$$\sigma_f = \lambda_{\max}(A^T A) \tag{34}$$

Considérons une étape de la descente de gradient à partir d'un point x vers un point z :

$$z = x - \alpha \nabla f(x) \tag{35}$$

Nous avons :

$$f(z) - f(x) \leq \langle \nabla f(x), z - x \rangle + \frac{\sigma_f}{2}||z - x||^2 \tag{36}$$

En substituant z par l'expression de l'étape de la descente de gradient, nous obtenons :

$$f(z) - f(x) \leq -\alpha||\nabla f(x)||^2 + \frac{\sigma_f}{2}\alpha^2||\nabla f(x)||^2 \tag{37}$$

La différence $f(z) - f(x)$ est négative, à condition que :

$$\frac{\sigma_f}{2}\alpha^2 - \alpha \leq 0 \tag{38}$$

Ce qui donne la condition suivante pour le pas de descente α :

$$\alpha \leq \frac{2}{\sigma_f} \tag{39}$$

Cette condition assure que le pas de descente garantit une diminution suffisante de la fonction objectif à chaque étape de l'algorithme de descente de gradient.

$$\alpha \leq \frac{2}{\sigma_f} \tag{40}$$

où σ_f est la constante de Lipschitz du gradient de la fonction.
Pour fixer un choix de pas de descente optimal, nous pouvons considérer la valeur $\alpha = \frac{2t}{\sigma_f}$, où $t \in (0, 1)$.

$$f(z) - f(x) \leq -\alpha||\nabla f(x)||^2 + \frac{\sigma_f}{2}\alpha^2||\nabla f(x)||^2 \tag{41}$$

En substituant α par $\frac{2t}{\sigma_f}$, nous avons :

$$f(z) - f(x) \leq -\frac{2t}{\sigma_f}||\nabla f(x)||^2 + \frac{\sigma_f}{2}\left(\frac{2t}{\sigma_f}\right)^2||\nabla f(x)||^2 \tag{42}$$

Ou encore :

$$f(z) - f(x) \leq -\frac{2t}{\sigma_f}||\nabla f(x)||^2 + \frac{2t^2}{\sigma_f}||\nabla f(x)||^2 \tag{43}$$

i.e :

$$f(x) - f(z) \geq \frac{2t}{\sigma_f}||\nabla f(x)||^2 - \frac{2t^2}{\sigma_f}||\nabla f(x)||^2 \tag{44}$$

Nous avons finalement :

$$f(x) - f(z) \geq \frac{2}{\sigma_f}t(1-t)||\nabla f(x)||^2 \tag{45}$$

La valeur de t peut être ajustée pour contrôler la vitesse de convergence de l'algorithme.
La parabole $t(1-t)$ atteint son maximum en $t = \frac{1}{2}$, et donc un choix optimal du pas de descente consiste à prendre $\alpha_{\text{optimal}} = \frac{1}{\sigma_f}$.

4.1.2 Actualisation des pas

Nous avons testé plusieurs versions de l'algorithme en tenant compte du pas : Une première version consiste à considérer un pas constant tout au long de l'exécution du code, ce pas constant était $\alpha = \frac{1}{\sigma_f}$. Les résultats n'étaient pas à la hauteur de ce que nous espérions en terme de nombre d'itérations, donc nous avons opté pour un pas variable qui varie intrinsèquement avec la dimension de la matrice hessienne de la fonction objective. Ce pas décroit naturellement vu que la dimension de la matrice décroit après chaque itération de l'algorithme. Cette modification permet d'avoir des résultats plus performants en nombre d'itérations à précision fixe. Mais nous avons

pensé que nous pouvions faire une amélioration de notre implémentation. Cette dernière consiste à varier les pas par des blocs d'itérations, nous avons choisi par exemple de les varier par blocs de 20 itérations, avec cette méthode nous obtenons une convergence plus rapide en nobmre d'itération avec une précision donnée.

4.2 Une région safe - Sphère Gap

Une fois cet algorithme de gradient proximal (sans screening) implémenté, nous nous intéressons à une implémentation avec screening, pour procéder ensuite à une comparaison entre les performances du Gradient proximal avec et sans screening. Les tests de screening faisant intervenir les solutions du primal ou du dual auxquelles nous n'avons pas accès, nous pouvons construire des régions safe contenant $\widehat{\phi}$.
En passant par la formulation duale et en notant $-D(\phi) = \frac{1}{2}\|y\|^2 - \frac{1}{2}\|y - \phi\|^2$, $-D$ est 1-fortement concave, nous avons donc pour ϕ admissible :

$$-D(\phi) \leq -D(\widehat{\phi}) + \langle \nabla(-D)(\widehat{\phi}), \phi - \widehat{\phi} \rangle - \frac{1}{2}\|\widehat{\phi} - \phi\|^2.$$

Par dualité faible, nous avons :

$$-D(\widehat{\phi}) \leq P(x).$$

De plus, par concavité et optimalité de $\widehat{\phi}$:

$$\langle \nabla - D(\widehat{\phi}), \phi - \widehat{\phi} \rangle \leq 0.$$

En combinant ces inégalités, nous obtenons :

$$-D(\phi) \leq P(x) + \langle \nabla - D(\widehat{\phi}), \phi - \widehat{\phi} \rangle - \frac{1}{2}\|\widehat{\phi} - \phi\|^2.$$

Comme $\langle \nabla - D(\widehat{\phi}), \phi - \widehat{\phi} \rangle \leq 0$, on a :

$$-D(\phi) \leq P(x) - \frac{1}{2}\|\widehat{\phi} - \phi\|^2.$$

Nous obtenons finalement :

$$2 \cdot gap(x, \phi) \geq \|\widehat{\phi} - \phi\|^2.$$

i.e :

$$\sqrt{2 \cdot gap(x, \phi)} \geq \|\widehat{\phi} - \phi\|.$$

Ce qui prouve que la sphère de centre ϕ et de rayon $R = \sqrt{2 \cdot gap(x, \phi)}$ contient $\widehat{\phi}$.
En notant $B(\phi, R)$ cette boule, nous avons la propriété suivante :

$$\max\{|\langle a_i, v\rangle| : v \in B(\phi, R)\} = |\langle a_i, \phi\rangle| + R\|a_i\|_2$$

Il suffit pour s'en apercevoir de prendre $v = \phi + R\mathbf{e}$, où $\mathbf{e}$ est un vecteur dans le disque unitaire $D(0, 1)$. Comme $v \in B(\phi, R)$, on a :

$$\|v - \phi\|_2 = R\|\mathbf{e}\|_2 \leq R.$$

Ainsi,

$$\begin{aligned} |\langle a_i, v\rangle| &= |\langle a_i, \phi + R\mathbf{e}\rangle| \\ &\leq |\langle a_i, \phi\rangle| + R|\langle a_i, \mathbf{e}\rangle|. \end{aligned}$$

le maximum de $|\langle a_i, v\rangle|$ est atteint pour $\mathbf{e} = \frac{a_i}{\|a_i\|_2}$, et donc :

$$\max\{|\langle a_i, v\rangle| : v \in B(\phi, R)\} = |\langle a_i, \phi\rangle| + R\|a_i\|_2.$$

Cette propriété va etre utile pour effectuer le test de screening, sachant que nous ne disposons pas de la solution $\widehat{\phi}$. Pour des régions safe contenant $\widehat{\phi}$, nous pouvons utiliser cette propriété pour vérifier si $\max\{|\langle a_i, v\rangle| : v \in R\} < \lambda$. Si cette condition est vérifiée, alors nous pouvons éliminer la i-ème variable.

4.3 Gradient Proximal avec screening

Suivant la discussion sur les régions safe, nous implémentons la méthode Proximale Lasso avec screening. Nous suivons une démarche similaire pour l'algorithme sans screening ;

```
def proximal_Lasso_screening(A, y, lmbda, max_iter=100, epsi=1e-4):
   m, n = A.shape
   original_n = n
   x = np.zeros(original_n)
   alpha = 1 / np.linalg.norm(A.T @ A, 2)
   final_x = np.zeros(n)
   objective_history = []
   A_screened = np.vstack([A, np.arange(original_n)])
   R = np.inf

   for iter in range(max_iter):
       res = y - A_screened[:-1, :] @ x
       grad = A_screened[:-1, :].T @ (-res)
       if iter % 20 == 0:
           alpha = alpha = 1 / np.linalg.norm(A_screened[:-1, :].T@A_screened[:-1, :], 2)
       x = proximal_l1(x - alpha * grad, alpha * lmbda)
       u = res
```

```
        Atres = A_screened[:-1, :].T @ u
        norm_Au_inf = np.linalg.norm(Atres, ord=np.inf)
        kap = (lmbda / norm_Au_inf)
        u = kap * u
        P_x = 0.5 * np.linalg.norm(res)**2 + lmbda * np.sum(np.abs(x))
        D_u = 0.5 * np.linalg.norm(y)**2 - 0.5 * np.linalg.norm(y - u)**2
        gap = P_x - D_u
        objective_history.append(P_x)
        centre = u
        R = np.sqrt(2 * gap)
        print(R)
        if np.abs(gap) < epsi:
            break

        a_reduit = A_screened[:-1, :]
        maxx_valeurs = np.abs(a_reduit.T @ centre) + R * np.linalg.norm(a_reduit, axis=0)
        to_delete = maxx_valeurs < lmbda
        A_screened = A_screened[:, ~to_delete]
        x = x[~to_delete]
        n -= np.sum(to_delete)

    final_indices = A_screened[-1, :].astype(int)
    for i, index in enumerate(final_indices):
        final_x[index] = x[i]

    screened_indices = np.setdiff1d(np.arange(original_n), final_indices)
    return final_x, centre, R, screened_indices, objective_history
```

Dans cette fonction, après chaque mise à jour de x, nous calculons u, la solution duale courante. Ensuite, nous calculons le gap entre la fonction objectif duale et la fonction objectif primale, et utilisons cela pour calculer R. Nous éliminons ensuite les variables dont la valeur maximale, calculée en utilisant la propriété présentée dans la section précédente, est inférieure à λ. Enfin, nous mettons à jour Ascreened, x et n pour refléter les variables restantes.

En utilisant ensuite la propriété présentée dans la section précédente, nous éliminons les colonnes qui passent le test, en faisant attention aux indices correspondants dans la matrice originale (moyennant une dernière ligne ajoutée qui sauvegarde l'indice initial).

4.4 Résultats de l'implémentation

La version finale de l'implémentation montre les résultats présentés dans cette partie.

4.4.1 Nombre d'itérations

La figure suivante présente une comparaison du nombre des itérations nécessaires pour atteindre la fonction objective :

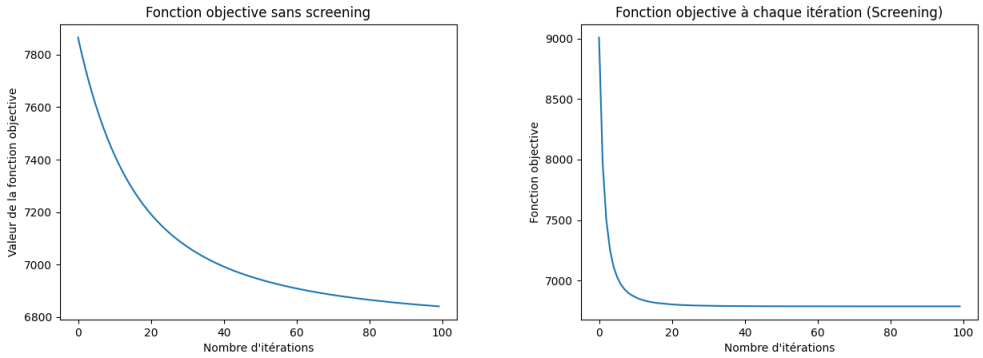


FIGURE 6 – Comparaison de la convergence de la fonction objective

Nous remarquons que la fonction objective est atteinte après plus ou moins 15 itérations dans la version avec screening, cependant, elle nécessite 80 itérations pour pouvoir atteindre la valeur minimale dans la version sans screening. Donc en raisonnant à une précision fixe, nous pouvons montrer que le screening permet de réduire drastiquement le nombre d'itération. Ce qui laisse supposer que le screening permet de réduire le temps d'exécution de l'algorithme.

4.4.2 Temps d'exécution

Comme mentionné précédemment, la diminution des nombres d'itérations nous laisse penser à la diminution du temps de l'exécution. Malheuresement, nous n'avons pas pu aboutir à cela, vu que notre code nécessite d'être optimiser. Il faut aussi préciser que la comparaison des temps de l'exécution n'est pas équitable vu que le modèle sans screening ne contient pas l'implémentation du gap et son calcul à chaque itération, cependant le modèle avec screening contient cette étape qui est très coûteuse en complexité temporelle et spatiale. Ajoutons à cela que l'implémentation avec screening contient des actualisations des pas de la descente de gradient donc des calculs du rayon spectrale. C'est pour ces raisons que le temps de l'exécution de l'algorithme avec et sans screening ne donne pas les résultats qu'on voulait.

Voilà une comparaison des temps d'exécution avec et sans screening :

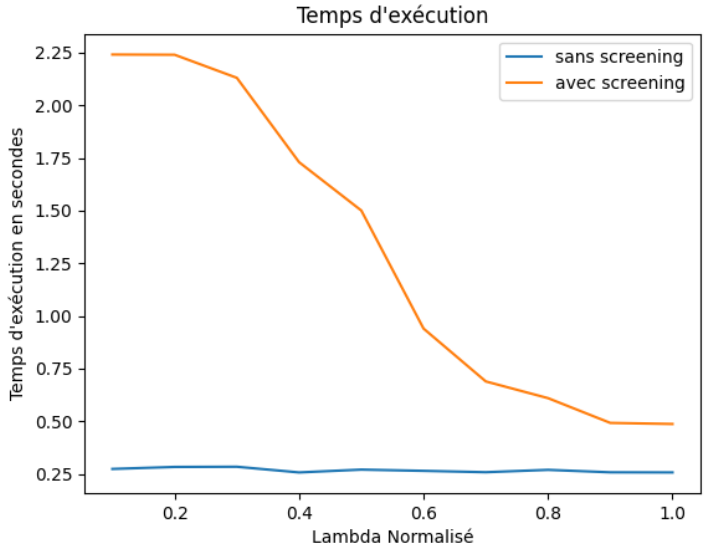


FIGURE 7 – Comparaison des temps d'exécution

4.4.3 Empreinte carbone

L'empreinte carbone est obtenu par le calcul des quantités de CO2 émises par le CPU et la RAM, qui sont calculés par un carbone tracker du module psutil. Les valeurs obtenues pour l'utilisation du CPU et de la RAM sont pondérées et additionnées pour obtenir l'empreinte carbone totale de l'exécution du code. Comme l'empreinte carbone est directement lié aux temps de l'exécution, les résultats sont toujours contradictoires à notre objectif qu'on a annoncé au début, c'est à dire que le screening causera une diminution de l'empreinte carbone. Voilà une comparaison entre les quantités de carbone en Kg émises lors de l'exécution du code :

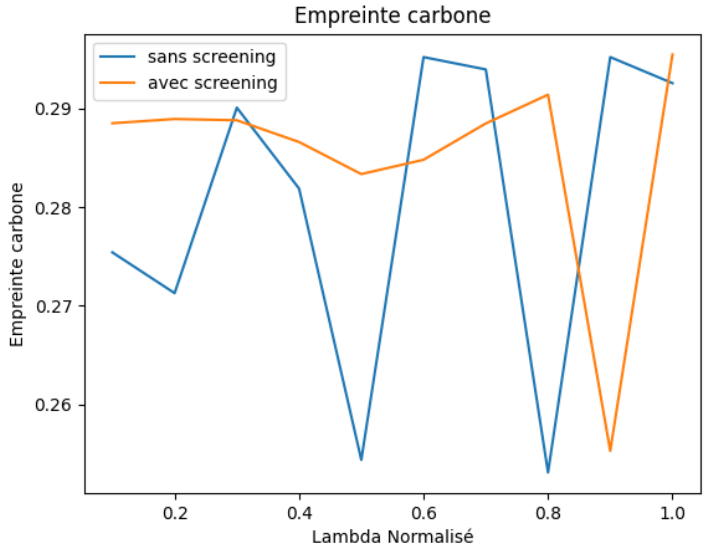


FIGURE 8 – Comparaison des empreintes carbone

4.5 Implémentation Moindres Carrés non-négatifs

Nous implémentons l'algorithme pour les moindres carrés non-négatifs en suivant une démarche analogue avec celle employée pour le gradient proximal ;

```
def mcnn_screening(A, y, max_iter=1000, epsi=1e-4):
    m, n = A.shape
    original_n = n
    alpha = 0.1 / np.linalg.norm(A.T @ A, 2)
    x = np.ones(original_n)
    objective_history = []
    A_screened = np.vstack([A, np.arange(original_n)])
    for iter in range(max_iter):
        grad = A_screened[:-1, :].T @ (A_screened[:-1, :] @ x - y)
        if iter % 20 == 0:
            alpha = alpha = 1 / np.linalg.norm(A_screened[:-1, :].T@A_screened[:-1, :], 2)
        x = np.maximum(x - alpha * grad, 0)
        Ax = A_screened[:-1, :] @ x
        if np.all(Ax > 0):
            kap = np.max([1, np.max(y / Ax)])
        else:
            break
        u = y - kap*A_screened[:-1, :] @ x
        P_x = 0.5 * np.linalg.norm(y - A_screened[:-1, :] @ x)**2
        D_u = 0.5 * np.linalg.norm(y)**2 - 0.5 * np.linalg.norm(y - u)**2
        gap = P_x - D_u
```

```
        print(gap)
        objective_history.append(P_x)
        R = np.sqrt(2 * np.abs(gap))
        if R < epsi:
            break
        to_delete= []
        for i in range(n):
            a_i = A_screened[:-1, i]
            maxx = np.dot(a_i, u) + R * np.linalg.norm(a_i)
            if maxx < 0:
                to_delete.append(i)
        A_screened = np.delete(A_screened, to_delete, axis=1)
        x = np.delete(x, to_delete)
        n -= len(to_delete)
        final_indices = A_screened[-1, :].astype(int)
    final_x = np.zeros(original_n)
    screened_indices = np.setdiff1d(np.arange(original_n), final_indices)
    final_x[final_indices] = x
    return final_x, screened_indices, objective_history
```

En se restreignant au cas ou la matrice A est positive, et en adaptant les tests de screening (la valeur absolue disparaît et lambda est remplacé par zéro, conformément aux nouveaux tests de screening obtenus, et le calcul du maximum de la quantité considérée sur les sphères gap découle d'un raisonnement similaire a celui pour le Lasso). Contrairement à ce que nous observons pour le cas du Lasso avec le Proximal avec screening, très peu de colonnes passent le test, ce qui se répercute sur la vitesse de convergence de cette implémentation avec screening par opposition à celle du Lasso ; le nombre d'itérations requises pour atteindre la convergence est très supérieur à celui qu'on avait précédemment.

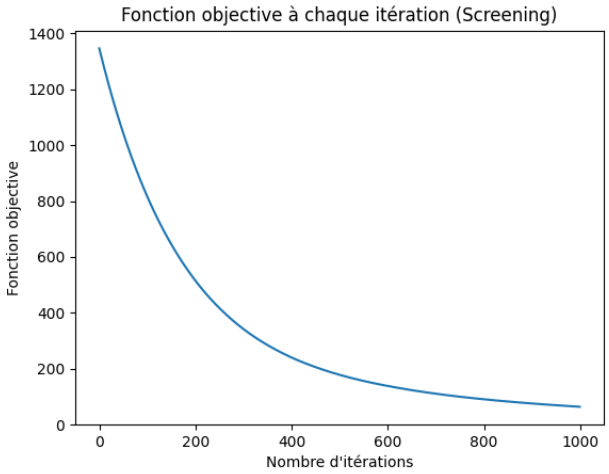


FIGURE 9 – Convergence pour l'algorithme avec screening dans le cas des Moindres Carres non negatifs

5 Conclusion

Nous avons pu obtenir des résultats concluants quant à la convergence du modèle avec screening qui s'avère être plus rapide en nombre d'itérations à précision donnée mais qui peut être plus coûteuse en temps et en énergie (et donc un impact plus grand sur l'environnement). En revanche, pour y remédier, nous pouvons dans un premier temps calculer le temps d'exécution sur un nombre d'itérations adéquats pour chaque méthode et comparer. Ensuite, régler des problèmes techniques dans les algorithmes par exemple optimiser le nombre de fois où nous calculons certaines variables mais aussi choisir peut-être une nouvelle méthode pour évaluer le pas de descente de gradient. Enfin, nous avons travaillé avec une matrice A dont les coefficients appartiennent à $\mathbb{R}_+$ ce qui n'est pas forcément une contrainte que nous pourrons trouver dans des problèmes plus concrets.

Références

[1] Elvira Clément et Herzet Cédric, Safe Rules for the Identification of zeros in the Solutions of the Slope Problem, 2020

[2] Ghaoui L. Viallon V. et Rabbani T., Safe feature elimination in sparse supervised learning, *Pacific Journal of Optimization*. 2010

[3] Pesquet Jean-Christophe, Cours d'optimisation CentraleSupélec, 2022.