En quête de précision : Un benchmark open-source et un solveur polyvalent pour le Graphical Lasso

Can POULIQUEN Paulo GONÇALVES Titouan VAYER Mathurin MASSIAS

ENS de Lyon, Inria, CNRS, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France

Résumé – Nous proposons un benchmark open source des principaux algorithmes d'optimisation et leurs implémentations pour la résolution du Graphical Lasso, afin d'aider les praticiens à choisir la meilleure option en fonction de leur cas d'usage. Notre benchmark est public, customisable et intégré à l'écosystème `benchopt` [Moreau et al., 2022] afin de favoriser la recherche ouverte et reproductible. Nous publions également une implémentation efficace d'un solveur par descente de blocs de coordonnées, avec des fonctionnalités avancées comme les pénalités pondérées, ainsi que diverses pénalités. Il permet ainsi d'utiliser la régularisation non-convexe, dont nous démontrons empiriquement la supériorité. Afin d'assurer une large diffusion et un impact pratique significatif, ce solveur est intégré à l'écosystème `scikit-learn` via le package `skglm` [Bertrand et al., 2022].

Abstract – We provide an open source and extensive benchmark of the major optimization algorithms and available implementations for solving the Graphical Lasso, to help practitioners choose the best one depending on their use case. Our benchmark is public, customizable and integrated into the `benchopt` ecosystem [Moreau et al., 2022] to foster open-research. We also release an efficient open source implementation of a block-coordinate descent solver, which we make available with all the desirable features. In particular, it supports weights and diverse penalties, unlocking the power of nonconvex regularization, whose superiority we demonstrate empirically. To ensure wide dissemination and a significant practical impact, this solver is integrated into the `scikit-learn` ecosystem via the `skglm` package [Bertrand et al., 2022].

1 Introduction

Révéler les structures sous-jacentes dans les données complexes, donc les relations cachées entre les variables d'un jeu de données, revêt une importance cruciale pour de nombreuses applications. Ces relations structurelles peuvent commodément être représentées par des graphes, qui offrent un moyen naturel de modéliser la distribution jointe de données multivariées, où les relations d'indépendance conditionnelle peuvent être exprimées à travers la structure du graphe. Si l'ensemble des sommets du graphe représente les variables du jeu de données, l'indépendance conditionnelle de deux variables est modélisée par l'absence d'arête entre les deux sommets correspondants [Koller, 2009]. Pour les données gaussiennes, la structure d'indépendance conditionnelle peut être caractérisée par l'inverse de la matrice de covariance de la distribution sous-jacente, appelée matrice de précision, qui peut être vue comme la matrice d'adjacence du graphe relationnel. L'apprentissage du graphe peut donc être reformulé comme l'estimation de la matrice de précision sous contrainte de parcimonie. Le Graphical Lasso [Banerjee et al., 2008, Friedman et al., 2008] est sans doute la méthode la plus populaire pour estimer les matrices de précision parcimonieuses dans ce cadre. Il est obtenu comme la solution d'un problème d'optimisation convexe non lisse, pour lequel de nombreux algorithmes dédiés ont été conçus. Pour atténuer le biais de la pénalité ℓ_1, des modifications basées sur des pénalités non-convexes ont montré qu'elles amélioraient considérablement la qualité de la matrice estimée. L'intégration de la non-convexité au problème initial apporte son lot de défis qui peuvent être abordés de diverses manières, souvent au prix de l'efficacité computationnelle. Bien que de nombreux algorithmes aient été proposés dans la littérature, les logiciels open-source actuellement disponibles manquent d'une implémentation d'un solveur pour le Graphical Lasso qui soit à la fois computationnellement efficace et capable de gérer adéquatement la non-convexité.

Ce travail est partiellement soutenu par le projet ANR-19-CHIA-0009.

Contributions (1) Nous fournissons une explication détaillée de différentes approches d'optimisation pour résoudre le problème du Graphical Lasso et éclairons les points clés pour des implémentations pratiques efficaces. (2) Nous réalisons un benchmark des principaux algorithmes d'optimisation pour résoudre le Graphical Lasso. Nous mettons en lumière qu'il n'existe pas de solveur universel et que selon le contexte, l'un peut être plus pertinent que l'autre. Ce benchmark est publié sous le cadre `benchopt`, et peut être reproduit ou étendu sans difficulté. (3) Nous publions une implémentation moderne en Python d'un solveur compétitif, basé sur une descente de blocs de coordonnées, primale ou duale. Notre implémentation est au moins aussi rapide que celle de `scikit-learn` [Pedregosa et al., 2011] tout en offrant bien plus de fonctionnalités. En particulier, elle gère les pénalités non-convexes, soit par repondération itérative, soit par optimisation directe lorsque cela est possible. Nous montrons numériquement que de telles pénalités conduisent à des améliorations substantielles sur la reconstruction, et l'identification du support.

Notation Nous utilisons la majuscule en gras $\boldsymbol{\Theta}$ pour les matrices, la minuscule en gras $\boldsymbol{\theta}$ pour les vecteurs et la minuscule standard θ pour les scalaires. Pour $\boldsymbol{\Theta} \in \mathbb{R}^{p \times p}$, $\|\boldsymbol{\Theta}\|_{1,\text{off}} = \sum_{i \neq j} |\theta_{ij}|$.

2 Le Graphical Lasso

Nous nous intéressons à l'estimation du graphe d'indépendance conditionnelle des variables d'un jeu de données gaussien i.i.d. La matrice d'adjacence du graphe est alors donnée par la matrice de précision $\boldsymbol{\Theta}_{\text{true}} \succ 0$, l'inverse de la matrice de covariance $\boldsymbol{\Sigma}_{\text{true}}$ de la distribution gaussienne. Soit un jeu de données de n observations i.i.d. gaussiennes $\boldsymbol{x}_i \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\text{true}})$, $\boldsymbol{x}_i \in \mathbb{R}^p$, $\forall i \in [n]$. L'objectif du Graphical Lasso est de fournir une estimation *parcimonieuse* de $\boldsymbol{\Theta}_{\text{true}} = \boldsymbol{\Sigma}_{\text{true}}^{-1}$ à partir de la matrice de covariance empirique $\boldsymbol{S} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i \boldsymbol{x}_i^\top$:

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta} \succ 0}{\operatorname{argmin}} \underbrace{-\log\det(\boldsymbol{\Theta}) + \langle \boldsymbol{S}, \boldsymbol{\Theta} \rangle}_{\mathcal{F}(\boldsymbol{\Theta})} + \underbrace{\lambda \|\boldsymbol{\Theta}\|_{1,\text{off}}}_{\mathcal{R}_\lambda(\boldsymbol{\Theta})}, \quad (1)$$

où l'attache aux données $\mathcal{F}(\boldsymbol{\Theta})$ est la log-vraisemblance négative sous hypothèse de normalité des $\boldsymbol{x}_i$ et $\mathcal{R}_\lambda(\boldsymbol{\Theta})$ est une pénalité ℓ_1 pour imposer de la parcimonie sur $\boldsymbol{\Theta}^*$, les coefficients diagonaux n'étant pas pénalisés. (1) est un problème d'optimisation convexe contraint au cône des matrices symétriques définies positives, qui peut être résolu de différentes manières. De nombreuses approches mettent à jour une paire ligne-colonne de manière itérative, et il sera pratique d'adopter le partitionnement suivant de $\boldsymbol{\Theta}$:

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\Theta}_{11} & \boldsymbol{\theta}_{12} \\ \boldsymbol{\theta}_{12}^\top & \theta_{22} \end{bmatrix} \quad (2)$$

avec $\boldsymbol{\Theta}_{11} \in \mathbb{R}^{(p-1)\times(p-1)}, \boldsymbol{\theta}_{12} \in \mathbb{R}^{p-1}, \theta_{22} \in \mathbb{R}$. Notez que la paire ligne-colonne à mettre à jour n'a pas besoin d'être la dernière ; nous illustrons Figure 1 comment les sous-vecteurs $\boldsymbol{\theta}_{12}$ et les matrices $\boldsymbol{\Theta}_{11}$ évoluent selon la paire considérée.

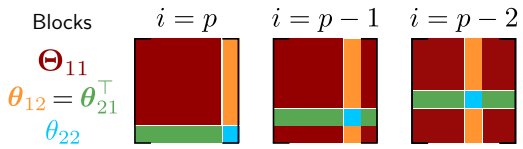

Figure 1 – Différentes instances du partitionnement par blocs (2) en fonction de la paire ligne-colonne i considérée.

3 Solveurs pour le Graphical Lasso

Nous détaillons d'abord deux des algorithmes les plus populaires pour résoudre (1), basés respectivement sur des approches par blocs de coordonnées duales et primal.

3.1 Ascension de blocs de coordonnées duale

Banerjee et al. [2008] proposent un algorithme efficace d'optimisation par blocs pour résoudre le *dual* du Graphical Lasso, qui est

$$\max_{\boldsymbol{W} \succ 0} \log\det(\boldsymbol{W}) \text{ s.t. } \|\boldsymbol{W} - \boldsymbol{S}\|_\infty \leq \lambda \,, w_{ii} = s_{ii} \; \forall i \in [d] \,.$$

De manière séquentielle, une paire ligne-colonne i de $\boldsymbol{W}$ est sélectionnée (notée $\boldsymbol{w}_{12}$ comme dans Figure 1), et mise à jour selon :

$$\underset{\boldsymbol{w} \in \mathbb{R}^{p-1}}{\operatorname{argmin}} \; \boldsymbol{w}^\top [\boldsymbol{W}_{11}]^{-1} \boldsymbol{w} \text{ s.t. } \|\boldsymbol{w} - \boldsymbol{s}_{12}\|_\infty \leq \lambda \,, \quad (3)$$

tandis que w_{22} est fixé à s_{22} lors de l'initialisation, et n'est jamais mis à jour. Bien que cela ne soit pas mentionné dans Banerjee et al. [2008], pour une implémentation plus efficace, $\boldsymbol{\Theta}$ peut être initialisé comme $\boldsymbol{W}_{\text{init}}^{-1}$ et mis à jour après chaque mise à jour de $\boldsymbol{w}_{12}$ en utilisant la formule d'inversion de Banachiewicz. Lorsque $\boldsymbol{\Theta}$ est disponible, $[\boldsymbol{W}_{11}]^{-1}$ peut être obtenu en $\mathcal{O}(p^2)$ à partir de $\boldsymbol{\Theta}$ au lieu de $\mathcal{O}(p^3)$ (voir par exemple Pouliquen et al. [2025, Prop 3.2]).

Pour éviter les inversions de matrices et tirer parti des solveurs Lasso rapides, au lieu de résoudre (3), Friedman et al. [2008] propose de résoudre son dual

$$\min_{\tilde{\boldsymbol{w}} \in \mathbb{R}^{p-1}} \tilde{\boldsymbol{w}}^\top \boldsymbol{W}_{11} \tilde{\boldsymbol{w}} + \boldsymbol{s}_{12}^\top \tilde{\boldsymbol{w}} + \lambda \|\tilde{\boldsymbol{w}}\|_1 \,, \quad (4)$$

et calculer $\boldsymbol{w}_{12}$ comme $\boldsymbol{W}_{11}\tilde{\boldsymbol{w}}$. Cet algorithme est couramment appelé GLasso, et est celui implémenté dans le très populaire `GraphicalLasso` de `scikit-learn`.

Cependant, l'algorithme de Friedman et al. [2008] présente certaines limitations : la valeur de l'objectif n'est pas décroissante et le caractère défini-positif de $\boldsymbol{\Theta}$ et $\boldsymbol{W}$ n'est pas strictement garanti à chaque itération.

3.2 Descente de blocs de coordonnées primale

Plus tard, Mazumder and Hastie [2012] proposent une alternative *primale* afin de corriger ces problèmes, appelée P-GLasso. Dans l'esprit, l'algorithme P-GLasso offre les mêmes garanties – à savoir, la décroissance de la valeur de l'objectif et des garanties strictes de SPD – que l'approche de Banerjee et al. [2008], tout en étant compatible avec les solveurs rapides de Lasso comme Friedman et al. [2008]. Parmi les nombreuses façons de concevoir l'algorithme, la plus intuitive consiste à reformuler (1) comme une minimisation alternée : à chaque itération, une ligne-colonne i est choisie, et la fonction objectif est minimisée par rapport à θ_{22} et $\boldsymbol{\theta}_{12}$ simultanément, tout en gardant $\boldsymbol{\Theta}_{11}$ fixe. D'après la formule de Schur, $\det(\boldsymbol{\Theta}) = \det(\boldsymbol{\Theta}_{11})(\theta_{22} - \boldsymbol{\theta}_{12}^\top \boldsymbol{\Theta}_{11}^{-1} \boldsymbol{\theta}_{12})$, et donc la dépendance de l'objectif par rapport à ces blocs est donnée par

$$\begin{aligned} &-\log\det(\boldsymbol{\Theta}_{11}) - \log(\theta_{22} - \boldsymbol{\theta}_{12}^\top \boldsymbol{\Theta}_{11}^{-1} \boldsymbol{\theta}_{12}) \\ &\quad + \langle \boldsymbol{\Theta}_{11}, \boldsymbol{S}_{11} \rangle + 2\boldsymbol{\theta}_{12}^\top \boldsymbol{s}_{12} + s_{22}\theta_{22} \\ &\quad + \lambda \|\boldsymbol{\Theta}_{11}\|_{1,\text{off}} + 2\lambda \|\boldsymbol{\theta}_{12}\|_1 \quad (5) \\ &\triangleq F(\boldsymbol{\theta}_{12}, \theta_{22}) \quad (6) \end{aligned}$$

P-Glasso est obtenu en supprimant d'abord la variable θ_{22} du problème d'optimisation, en remarquant que pour un $\boldsymbol{\theta}_{12}$ fixe, on a

$$\theta_{22}^*(\boldsymbol{\theta}_{12}) \triangleq \underset{\theta_{22}}{\operatorname{argmin}} \; F(\boldsymbol{\theta}_{12}, \theta_{22}) = \tfrac{1}{s_{22}} + \boldsymbol{\theta}_{12}^\top [\boldsymbol{\Theta}_{11}]^{-1} \boldsymbol{\theta}_{12} \,.$$

En remplaçant cette expression dans F, on obtient un problème de minimisation sur $\boldsymbol{\theta}_{12}$ uniquement :

$$\begin{aligned} \boldsymbol{\theta}_{12}^* &= \underset{\boldsymbol{\theta}_{12}}{\operatorname{argmin}} \; F(\boldsymbol{\theta}_{12}, \theta_{22}^*(\boldsymbol{\theta}_{12})) \quad (7) \\ &= \frac{1}{s_{22}} \underset{\tilde{\boldsymbol{\theta}} \in \mathbb{R}^{p-1}}{\operatorname{argmin}} \tilde{\boldsymbol{\theta}}^\top [\boldsymbol{\Theta}_{11}]^{-1} \tilde{\boldsymbol{\theta}} + \boldsymbol{s}_{12}^\top \tilde{\boldsymbol{\theta}} + \lambda \|\tilde{\boldsymbol{\theta}}\|_1 \quad (8) \end{aligned}$$

qui, comme (4), est un problème d'optimisation quadratique + ℓ_1 qui peut être résolu par des approches itératives standards telles que la descente de coordonnées. De manière similaire

à la formulation de Banerjee et al. [2008], l'inversion de $\boldsymbol{\Theta}_{11}$ peut être évitée en maintenant $\boldsymbol{W}$ à jour après chaque mise à jour ligne-colonne de $\boldsymbol{\Theta}$, et en utilisant la formule d'inversion de Banachiewicz pour calculer $[\boldsymbol{\Theta}_{11}]^{-1} = \boldsymbol{W}_{11} - \frac{1}{w_{22}}\boldsymbol{w}_{12}\boldsymbol{w}_{12}^\top$ à partir de $\boldsymbol{W}$. Une fois que $\boldsymbol{\theta}_{12}^*$ a été trouvé, $\boldsymbol{\theta}_{12}$ est mis à jour par $\boldsymbol{\theta}_{12}^*$ et θ_{22} est mis à jour par $\theta_{22}^*(\boldsymbol{\theta}_{12}^*)$. Cette mise à jour contraint naturellement $\boldsymbol{\Theta}$ au cône SPD grâce à la condition de Schur : $\boldsymbol{\Theta} \succ 0 \iff \boldsymbol{\Theta}_{11} \succ 0$ et $\theta_{22} - \boldsymbol{\theta}_{12}^\top[\boldsymbol{\Theta}_{11}]^{-1}\boldsymbol{\theta}_{12} > 0$. Ceci a également été récemment exploité dans Pouliquen et al. [2025] qui utilisent ce résultat pour garantir le caractère définipositif de $\boldsymbol{\Theta}$ tout en mettant à jour les blocs $\boldsymbol{\theta}_{12}$ par des réseaux de neurones.

4 De meilleures matrices de précision grâce à la non-convexité

Plusieurs améliorations peuvent être apportées à (1) pour améliorer l'estimateur du Graphical Lasso. La première est le *Weighted* Graphical Lasso, qui introduit différents niveaux de pénalisation $\gamma_{ij} = \gamma_{ji} \geq 0\ \forall i, j \in [p]$ dans la fonction de pénalité. La pénalité $\sum_{i \neq j} |\theta_{ij}|$ dans (1) est remplacée par $\sum_{i \neq j} \gamma_{ij}|\theta_{ij}|$; tous les algorithmes pour résoudre le Graphical Lasso peuvent être facilement adaptés pour résoudre sa version pondérée — bien que, comme nous le verrons dans Section 5, toutes les implémentations *ne gèrent pas forcément* les poids. Les problèmes à résoudre dans (4) et (8) deviennent des Lassos *pondérés*. Un choix adéquat des poids, par exemple par optimisation bi-niveaux, peut conduire à de meilleures estimations de la matrice de précision [Pouliquen et al., 2023]. La deuxième amélioration possible consiste à utiliser une régularisation non-convexe [Sun et al., 2018], et à résoudre

$$\min_{\boldsymbol{\Theta} \succ 0} \ -\log\det(\boldsymbol{\Theta}) + \langle \boldsymbol{S}, \boldsymbol{\Theta} \rangle + \sum_{i \neq j} \rho(|\theta_{ij}|) \tag{9}$$

avec $\rho : \mathbb{R}_+ \to \mathbb{R}$ une pénalité non-convexe induisant la parcimonie, par exemple MCP [Zhang, 2010], SCAD [Fan and Li, 2001], ℓ_q^q avec $0 < q < 1$ [Grasmair et al., 2008], log [Candes et al., 2008], etc. Il y a deux principales manières de résoudre (9) :

(i) si l'opérateur proximal de ρ est disponible (ce qui est le cas pour MCP, SCAD, $\ell_{0.5}$ parmi d'autres), l'approche primale de Mazumder and Hastie [2012] peut être utilisée telle quelle, en remplaçant la pénalité ℓ_1 dans (8) par ρ et en utilisant des méthodes proximales pour résoudre le problème de minimisation correspondant.

(ii) si $\rho : \mathbb{R}_+ \to \mathbb{R}$ est concave et différentiable (ce qui est le cas pour tous les choix mentionnés précédemment), on peut recourir à l'approche de majoration-minimisation de Candes et al. [2008], et résoudre (9) via une séquence de Graphical Lassos repondérés. En effet, si ρ est concave, pour tout choix de $\boldsymbol{\Theta}^{(0)}$, $\rho(|\theta_{ij}|) \leq \rho(|\theta_{ij}^{(0)}|) + \rho'(|\theta_{ij}^{(0)}|)(|\theta_{ij}| - |\theta_{ij}^{(0)}|)$ et donc l'objectif dans (9) peut être majoré par $-\log\det(\boldsymbol{\Theta}) + \langle \boldsymbol{S}, \boldsymbol{\Theta} \rangle + \sum_{i \neq j} \rho'(|\theta_{ij}^{(0)}|)|\theta_{ij}| + \text{cst}$. Cette borne supérieure peut être minimisée avec un solveur de Graphical Lasso pondéré avec $\gamma_{ij} = \rho'(|\theta_{ij}^{(0)}|)$, ce qui donne une nouvelle itération $\boldsymbol{\Theta}^{(1)}$, pour laquelle une nouvelle borne supérieure peut être construite et minimisée avec de nouveaux poids $\gamma_{ij} = \rho'(|\theta_{ij}^{(1)}|)$, etc.

Solveur	Package Python	Poids	Non-convexe
`GLasso`	`skglm` (le nôtre)	✓	✓
`P-GLasso`	`skglm` (le nôtre)	✓	✓
`GLasso`	`scikit-learn`	✗	✗
`G-ISTA`	✗	✗	✗
`OBN`	✗	✗	✗
`ADMM`	`GGLasso`[Schaipp et al., 2021]	✗	✗
`QUIC`	`skggm` [Laska and Narayan, 2017]	✓	✗

Table 1 – Solveurs actuellement implémentés dans le Benchmark du Graphical Lasso.

Solveur implémenté : Nous listons dans Table 1 les différents solveurs et implémentations open-source existants. Comme illustré, les fonctionnalités critiques mentionnées ci-dessus (gestion des poids et non-convexité) sont actuellement absentes de la plupart des implémentations open-source du Graphical Lasso. Nous y remédions en fournissant une implémentation efficace des approches de descente de blocs de coordonnées primale et duale (P-GLasso et GLasso), qui gèrent de plus les poids et les pénalités non-convexes, soit via l'approche directe (**i**) soit via l'approche itérative repondérée (**ii**). En utilisant l'API modulaire de `skglm`, la pénalité peut être facilement modifiée, et de nouvelles pénalités non-convexes peuvent être gérées en moins de 30 lignes de code. Comme le montre la section suivante, ce solveur moderne offre une vitesse compétitive parmi les solutions existantes, tandis que sa gestion de la non-convexité lui permet de mieux estimer la véritable matrice de précision.

5 Expériences

Dans cette section : (1) Nous mettons en évidence que, pour résoudre le mêm problème, notre solveur est au moins aussi rapide que l'implémentation Python de `scikit-learn` [Pedregosa et al., 2011] et est compétitif face aux autres implémentations. (2) Nous montrons l'avantage clair de l'utilisation de fonctions de régularisation non convexes à la fois en termes de reconstruction de matrice que d'identification de support, par rapport à la norme ℓ_1.

5.1 Un benchmark pour le Graphical Lasso

Pour évaluer les performances des principaux solveurs du Graphical Lasso en termes de temps, nous utilisons `benchopt`. Notre benchmark est disponible publiquement sur https://github.com/Perceptronium/benchmark_graphical_lasso, et peut être réexécuté en une ligne de code. Grâce à la structure modulaire de `benchopt`, de nouveaux solveurs, données et paramètres expérimentaux peuvent être ajoutés avec un effort minimal. Nous évaluons tous les solveurs sur la résolution de (1). Nous générons des données synthétiques comme suit : nous créons $\boldsymbol{\Theta}_{\text{true}}$ en utilisant la fonction `make_sparse_spd_matrix` de `sklearn`, à laquelle nous ajoutons $0.1 \cdot \boldsymbol{I}_p$ pour éviter un conditionnement trop extrême. À partir de $\boldsymbol{\Theta}_{\text{true}}$, nous échantillonnons $n = 1000$ vecteurs aléatoires gaussiens centrés i.i.d. $\boldsymbol{x}_j \sim \mathcal{N}(0, \boldsymbol{\Theta}_{\text{true}}^{-1})$ qui sont utilisés pour calculer la matrice de covariance empirique $\boldsymbol{S}$. Nous lançons ensuite tous les solveurs étudiés

https://github.com/scikit-learn-contrib/skglm

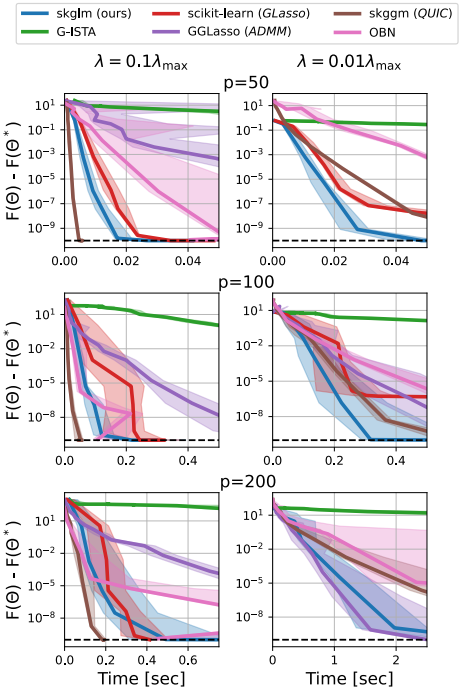

Figure 2 – Les performances de divers solveurs du Graphical Lasso sous différentes configurations. La colonne de gauche correspond à des matrices de précision plus parcimonieuses que celle de droite.

pour résoudre (1). Nous monitorons la sous-optimalité $F(\boldsymbol{\Theta}) - F(\boldsymbol{\Theta}^*)$, où F est la valeur objective de (1), en fonction du temps brut CPU. Les résultats pour différentes dimensions et degrés de parcimonie sont visible en Figure 2. Notre implémentation se place constamment parmi les meilleures dans tous les scénarios testés. Dans l'ensemble, nos résultats mettent en évidence qu'il n'existe pas de solution universelle, et que les praticiens doivent choisir une solution adéquate au problème qu'ils souhaitent résoudre. Certaines implémentations ne parviennent pas à converger vers une solution dans un temps approprié sous certains conditions (par exemple `scikit-learn` dans la configuration moins parcimonieuse à $p = 200$).

5.2 Reconstruction & récupération du support avec des pénalités non-convexes

Nous mettons ensuite en évidence les avantages clairs de remplacer la pénalité ℓ_1 par des régularisations non-convexes. Nous considérons trois pénalités différentes : la pénalité log $\rho(x) = \log(x + \epsilon)$, la pénalité $\ell_{0.5}$ $\rho(x) = \sqrt{x + \epsilon}$ et le MCP [Zhang, 2010]. Nous résolvons $l = 20$ itérations repondérées ℓ_1 comme décrit dans Section 4, approche **(ii)**, avec des poids mis à jour respectivement par $\gamma_{ij}^{(k+1)} = 1/(|\theta_{ij}^{(k)}| + \epsilon)$ pour le log, $\gamma_{ij}^{(k+1)} = 1/(2\sqrt{|\theta_{ij}^{(k)}| + \epsilon})$ pour $\ell_{0.5}$ et la dérivée du MCP prise en $|\theta_{ij}^{(k)}|$ pour le MCP, à savoir $\max(0, \lambda_{\text{MCP}} - |\theta_{ij}^{(k)}|/\gamma_{\text{MCP}})$ où λ_{MCP} et γ_{MCP} sont les paramètres de régularisation du MCP (nous prenons $\gamma_{\text{MCP}} = 3$). Nous évaluons les performances des pénalités en termes de récupération du support (mesurée par le score F1 entre les matrices bianires représentant les supports respectifs de $\boldsymbol{\Theta}_{\text{true}}$ et le $\boldsymbol{\Theta}$ estimé), et le NMSE $= \|\boldsymbol{\Theta}^* - \boldsymbol{\Theta}_{\text{true}}\|^2 / \|\boldsymbol{\Theta}_{\text{true}}\|^2$. Le cas $p = 100, n = 1000$ est illustré dans Figure 3. L'avantage de l'utilisation des variations de pénalités non-convexes en termes de reconstruction et surtout d'identification de support est clair : tandis que le score F1 du Graphical Lasso standard ℓ_1 atteint moins de 60%, nous parvenons à dépasser 80% avec la pénalité log. De plus, l'hyperparamètre λ donnant le meilleur score F1, et celui donnant le plus bas NMSE, sont assez différentes pour la norme ℓ_1, et bien plus proches pour les pénalités non-convexes, montrant qu'elles atténuent bien le célèbre dilemme "estimation-prédiction" de la norme ℓ_1.

La non-monotonicité de certaines courbes est expliquée dans [Bertrand et al., 2022, App. E6].

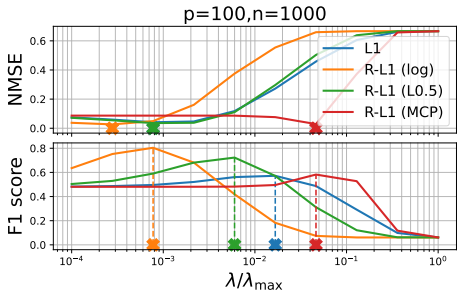

Figure 3 – Chemin de régularisation pour le Graphical Lasso repondéré avec différentes pénalités non convexes. λ est la force de régularisation pour chaque pénalité, et $\lambda_{\max}$ est le plus petit λ pour lequel la solution du Graphical Lasso est diagonale. Les croix indiquent le meilleur λ pour chaque pénalité et métrique.

References

T. Moreau, M. Massias, A. Gramfort, P. Ablin, P-A. Bannier, , et al. Benchopt: Reproducible, efficient and collaborative optimization benchmarks. *NeurIPS*, 35, 2022.

Q. Bertrand, Q. Klopfenstein, P-A. Bannier, G. Gidel, and M. Massias. Beyond l1: Faster and better sparse models with skglm. *NeurIPS*, 35, 2022.

D. Koller. Probabilistic graphical models: Principles and techniques, 2009.

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *JMLR*, 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 2008.

F. Pedregosa, G. Varoquaux, A. Gramfort, et al. Scikit-learn: Machine learning in python. *JMLR*, 2011.

C. Pouliquen, M. Massias, and T. Vayer. Schur's positive-definite network: Deep learning in the spd cone with structure. *ICLR*, 2025.

R. Mazumder and T. Hastie. The graphical lasso: New insights and alternatives. *Electronic Journal of Statistics*, 2012.

C. Pouliquen, P. Gonçalves, M. Massias, and T. Vayer. Implicit differentiation for hyperparameter tuning the weighted graphical lasso. *GRETSI*, 2023.

Q. Sun, K. M. Tan, H. Liu, and T. Zhang. Graphical nonconvex optimization via an adaptive convex relaxation. In *ICML*, 2018.

C-H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 2010.

J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 2001.

M. Grasmair, M. Haltmeier, and O. Scherzer. Sparse regularization with lq penalty term. *Inverse Problems*, 2008.

E. J. Candes, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier analysis and applications*, 2008.

F. Schaipp, O. Vlasovets, and C. L. Müller. Gglasso - a python package for general graphical lasso computation. *Journal of Open Source Software*, 2021.

J. Laska and M. Narayan. skggm 0.2.7: A scikit-learn compatible package for Gaussian and related Graphical Models, 2017.