Implicit differentiation for hyperparameter tuning the weighted Graphical Lasso

Can POULIQUEN Paulo GONÇALVES Mathurin MASSIAS Titouan VAYER

Univ Lyon, Ens Lyon, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France.

Résumé – Nous dérivons les résultats mathématiques nécessaires à l'implémentation d'une procédure de calibration d'hyperparamètres pour le Graphical Lasso via un problème d'optimisation bi-niveau résolu par méthode du premier-ordre. En particulier, nous dérivons la Jacobienne de la solution du Graphical Lasso par rapport à ses hyperparamètres de régularisation.

Abstract – We provide a framework and algorithm for tuning the hyperparameters of the Graphical Lasso via a bilevel optimization problem solved with a first-order method. In particular, we derive the Jacobian of the Graphical Lasso solution with respect to its regularization hyperparameters.

1 Introduction

The Graphical Lasso estimator (GLASSO) [1] is a commonly employed and established method for estimating sparse precision matrices. It models conditional dependencies between variables by finding a precision matrix that maximizes the ℓ_1-penalized log-likelihood of the data under a Gaussian assumption. More precisely, the GLASSO is defined as [1]

$$\hat{\boldsymbol{\Theta}}(\lambda) = \underset{\boldsymbol{\Theta} \succ 0}{\operatorname{argmin}} \underbrace{-\operatorname{logdet}(\boldsymbol{\Theta}) + \langle \mathbf{S}, \boldsymbol{\Theta} \rangle + \lambda \|\boldsymbol{\Theta}\|_1}_{=\Phi(\boldsymbol{\Theta}, \lambda)}, \quad (1)$$

where $\mathbf{S} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top \in \mathbb{R}^{p \times p}$ is the empirical covariance matrix of the data $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$. There exist many first- and second-order algorithms to solve this problem [11, 12, 15]. These approaches all require choosing the right regularization hyperparameter λ that controls the sparsity of $\hat{\boldsymbol{\Theta}}(\lambda)$. This is a challenging task that typically involves identifying the value of λ for which the estimate $\hat{\boldsymbol{\Theta}}(\lambda)$ minimizes a certain performance criterion $\mathcal{C}$. This problem can be framed as the bilevel optimization problem

$$\begin{aligned} \lambda^{\text{opt}} &= \underset{\lambda}{\operatorname{argmin}} \, \{\mathcal{L}(\lambda) \triangleq \mathcal{C}(\hat{\boldsymbol{\Theta}}(\lambda))\} \\ &\text{s.t. } \hat{\boldsymbol{\Theta}}(\lambda) = \underset{\boldsymbol{\Theta} \succ 0}{\operatorname{argmin}} \; \Phi(\boldsymbol{\Theta}, \lambda), \end{aligned} \quad (2)$$

where the minimizations over λ and $\boldsymbol{\Theta}$ are called respectively the *outer* and *inner* problems. The standard approach to tune the hyperparameter λ is *grid-search*: for a predefined list of values for λ, the solutions of (1) are computed and the one minimizing $\mathcal{C}$ is chosen, which corresponds to solving (2) with a zero-order method.

In this paper, we propose instead a first-order method for (2), relying on the computation of the so-called *hypergradient* and the Jacobian of the GLASSO objective with respect to λ. Despite the non-smoothness of the inner problem, we derive a closed-form formula for the Jacobian.

Our main contributions are the derivations of the equations of implicit differentiation for the GLASSO: first in the single parameter regularization case for ease of exposure in Section 2, and then for matrix regularization in Section 3. Our work paves the way for a scalable approach to hyperparameter tuning for the GLASSO and its variants, and could naturally apply to more complex extensions of the GLASSO such as [3]. We provide open-source code for the reproducibility of our experiments which are treated in Section 4.

Related work Although not strictly considering the GLASSO problem, some other alternatives to grid-search have been considered in the literature, including random search [5] or Bayesian optimization [17]. While we compute the hypergradient by implicit differentiation [4], automatic differentiation in forward and backward modes have also been proposed [10].

Notation The set of integers from 1 to k is $[k]$. For a set $\mathcal{S} \subset [p]$ and a matrix $\mathbf{A} \in \mathbb{R}^{p \times p}$, $\mathbf{A}_{:,\mathcal{S}}$ (*resp.* $\mathbf{A}_{\mathcal{S},:}$) is the restriction of $\mathbf{A}$ to columns (*resp.* rows) in $\mathcal{S}$. The Kronecker and Hadamard products between two matrices are denoted by $\otimes$ and $\odot$ respectively. The column-wise vectorization operation, transforming matrices into vectors, is denoted by $\operatorname{vec}(\cdot)$ and $\operatorname{vec}^{-1}(\cdot)$ denotes the inverse operation. For a differentiable function F of two variables, $\mathbf{J}_1 F$ and $\mathbf{J}_2 F$ denote the Jacobians of F with respect to its first and second variable respectively. A fourth-order tensor $\mathbf{A}$ applied to a matrix $\mathbf{B}$ corresponds to a contraction according to the last two indices: $(\mathbf{A} : \mathbf{B})_{ij} = \sum_{k,l} A_{ijkl} B_{kl}$. The relative interior of a set $\mathcal{S}$ is denoted by $\operatorname{relint}(\mathcal{S})$.

2 The scalar case

If the solution of the inner problem $\hat{\boldsymbol{\Theta}}(\lambda)$ is differentiable with respect to λ, the gradient of the outer objective function $\mathcal{L}$, called *hypergradient*, can be computed by the chain rule:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\lambda}(\lambda) = \sum_{i,j=1}^{p} \frac{\partial \mathcal{C}}{\partial \Theta_{ij}}(\hat{\boldsymbol{\Theta}}(\lambda)) \frac{\partial \hat{\Theta}_{ij}}{\partial \lambda}(\lambda). \quad (3)$$

1. in variants, the diagonal entries of $\boldsymbol{\Theta}$ are not penalized. This is handled by the framework of Section 3

This work was partially funded by the AllegroAssai ANR-19-CHIA-0009 project.

The hypergradient can then be used to solve the bilevel problem with a first-order approach such as gradient descent: $\lambda_{k+1} = \lambda_k - \rho \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\lambda}(\lambda_k)$. The main challenge in the hypergradient evaluation is the computation of $\frac{\partial \hat{\Theta}_{ij}}{\partial \lambda}(\lambda)$, that we summarize in a $p \times p$ matrix [2] $\hat{\mathbf{J}} = (\frac{\partial \hat{\Theta}_{ij}}{\partial \lambda}(\lambda))_{ij}$. When the inner objective Φ is smooth, $\hat{\mathbf{J}}$ can be computed by differentiating the *optimality condition* of the inner problem, $\nabla_{\mathbf{\Theta}} \Phi(\mathbf{\Theta}, \lambda) = 0$, with respect to λ, as in [4].

Unfortunately, in our case the inner problem is not smooth. We however show in the following how to compute $\hat{\mathbf{J}}$ by differentiating a fixed point equation instead of differentiating the optimality condition as performed in [6] for the Lasso. The main difficulty in our case stems from our optimization variable being a matrix instead of a vector, which induces the manipulation of tensors in the computation of $\hat{\mathbf{J}}$. Let

$$\begin{aligned} F : \mathbb{R}^{p \times p} \times \mathbb{R}_+ &\to \mathbb{R}^{p \times p} \\ (\mathbf{Z}, \lambda) &\mapsto \operatorname{prox}_{\gamma \lambda \|\cdot\|_1}(\mathbf{Z}) \end{aligned}, \tag{4}$$

which is equal to the soft-thresholding operator

$$F(\mathbf{Z}, \lambda) = \operatorname{sign}(\mathbf{Z}) \odot (|\mathbf{Z}| - \lambda\gamma)_+ \,, \tag{5}$$

where all functions apply entry-wise to $\mathbf{Z}$. When $\hat{\mathbf{\Theta}}(\lambda)$ solves the inner problem (1), it fulfills a fixed-point equation related to proximal gradient descent. Valid for any $\gamma > 0$ [8, Prop. 3.1.iii], this equation is as follows:

$$\hat{\mathbf{\Theta}}(\lambda) = F(\hat{\mathbf{\Theta}}(\lambda) - \gamma(\mathbf{S} - \hat{\mathbf{\Theta}}(\lambda)^{-1}), \lambda) \,. \tag{6}$$

To compute $\hat{\mathbf{J}}$, the objective is now to differentiate (6) with respect to λ. By defining $\hat{\mathbf{Z}} \triangleq \hat{\mathbf{\Theta}}(\lambda) - \gamma(\mathbf{S} - \hat{\mathbf{\Theta}}(\lambda)^{-1})$ we will show that F is differentiable at $(\hat{\mathbf{Z}}, \lambda)$. Since F performs entry-wise soft-thresholding, each of its coordinates is weakly differentiable [9, Prop. 1, Eq. 32] and the only non-differentiable points are when $|\hat{Z}_{ij}| = \lambda\gamma$. To ensure that none of the entries of $\hat{\mathbf{Z}}$ take the value $\pm\lambda\gamma$, we will use the first-order optimality condition for the inner problem (1).

Proposition 1. *Let $\hat{\mathbf{\Theta}}(\lambda)$ be a solution of (1). Then, using Fermat's rule and the expression of the subdifferential of the ℓ_1-norm [2, Thm. 3.63, Ex. 3.41],*

$$[\hat{\mathbf{\Theta}}(\lambda)^{-1}]_{ij} - S_{ij} \in \begin{cases} \{\lambda \operatorname{sign} \hat{\Theta}(\lambda)_{ij}\}, & \text{if } \hat{\Theta}(\lambda)_{ij} \neq 0 \,, \\ [-\lambda, \lambda] & \text{otherwise.} \end{cases} \tag{7}$$

We also require the following assumption which is classical (see *e.g.* [14, Thm 3.1] and references therein).

Assumption 2 (Non degeneracy)**.** *We assume that the inner problem is non-degenerated, meaning that it satisfies a slightly stronger condition than (7):*

$$\hat{\mathbf{\Theta}}(\lambda)^{-1} - \mathbf{S} \in \operatorname{relint} \lambda \partial \| \cdot \|_1 \,. \tag{8}$$

This implies that in (7), the interval $[-\lambda, \lambda]$ in the second case becomes $(-\lambda, \lambda)$.

Using Proposition 1 under Assumption 2, we conclude that $|\hat{Z}_{ij}|$ never takes the value $\lambda\gamma$ so that (6) is differentiable. Indeed when $\hat{\Theta}(\lambda)_{ij} = 0, |[\hat{\mathbf{\Theta}}(\lambda)^{-1}]_{ij} - S_{ij}| < \lambda$ which implies $|\hat{Z}_{ij}| < \lambda\gamma$. Conversely, when $\hat{\Theta}(\lambda)_{ij} \neq 0$, $\hat{Z}_{ij} = \hat{\Theta}(\lambda)_{ij} + \gamma\lambda \operatorname{sign}(\hat{\Theta}(\lambda)_{ij})$ which implies $|\hat{Z}_{ij}| > \lambda\gamma$. Consequently, we can differentiate Equation (6) *w.r.t.* λ, yielding

$$\hat{\mathbf{J}} = \mathbf{J}_1 F(\hat{\mathbf{Z}}, \lambda) \left(\hat{\mathbf{J}} - \gamma \hat{\mathbf{\Theta}}(\lambda)^{-1} \hat{\mathbf{J}} \hat{\mathbf{\Theta}}(\lambda)^{-1} \right) + \mathbf{J}_2 F(\hat{\mathbf{Z}}, \lambda) \,. \tag{9}$$

The goal is now to solve (9) in $\hat{\mathbf{J}}$. We define $\mathbf{D} \triangleq \mathbf{J}_1 F(\hat{\mathbf{Z}}, \lambda)$ the Jacobian of F with respect to its first variable at $(\hat{\mathbf{Z}}, \lambda)$ which is represented by a fourth-order tensor in $\mathbb{R}^{p \times p \times p \times p}$:

$$D_{ijkl} = \left[\frac{\partial F}{\partial Z_{kl}}(\hat{\mathbf{Z}}, \lambda) \right]_{ij} \,. \tag{10}$$

We also note $\mathbf{E} \triangleq \mathbf{J}_2 F(\hat{\mathbf{Z}}, \lambda)$ viewed as a $p \times p$ matrix.

Jacobian with respect to Z Because the soft-thresholding operator acts independently on entries, one has $D_{ijkl} = 0$ when $(i, j) \neq (k, l)$. From Equation (5), the remaining entries are given by

$$D_{ijij} = \begin{cases} 0 \,, & \text{if } |\hat{Z}_{ij}| < \lambda\gamma \,, \\ 1 \,, & \text{otherwise.} \end{cases} \tag{11}$$

Jacobian with respect to λ Similarly to $\mathbf{D}$, $\mathbf{E}$ is given by

$$E_{ij} = \left[\frac{\partial F}{\partial \lambda}(\hat{\mathbf{Z}}, \lambda) \right]_{ij} = \begin{cases} 0 \,, & \text{if } |\hat{Z}_{ij}| < \lambda\gamma \,, \\ -\gamma \operatorname{sign}(\hat{Z}_{ij}) \,, & \text{otherwise.} \end{cases} \tag{12}$$

We can now find the expression of $\hat{\mathbf{J}}$ as described in the next proposition.

Proposition 3. *Let $\mathcal{S} \subset [p^2]$ be the set of indices i such that $\operatorname{vec}(|\hat{\mathbf{Z}}|)_i > \lambda\gamma$. The Jacobian $\hat{\mathbf{J}}$ is given by*

$$\begin{aligned} &\operatorname{vec}\left(\hat{\mathbf{J}}\right)_{\mathcal{S}} = \left[\left(\hat{\mathbf{\Theta}}(\lambda)^{-1} \otimes \hat{\mathbf{\Theta}}(\lambda)^{-1} \right)_{\mathcal{S},\mathcal{S}} \right]^{-1} \operatorname{vec}(\mathbf{E})_{\mathcal{S}} / \gamma \,, \\ &\operatorname{vec}\left(\hat{\mathbf{J}}\right)_{\mathcal{S}^c} = 0 \,. \end{aligned}$$

Proof. From (11), one has that $\mathbf{D}$ applied to $\mathbf{X} \in \mathbb{R}^{p \times p}$ is simply a masking operator $\mathbf{D} : \mathbf{X} = \mathbf{M} \odot \mathbf{X}$, where $M_{ij} = \mathbb{1}\{|\hat{Z}_{ij}| < \lambda\gamma\}$. Thus (9) reads

$$\hat{\mathbf{J}} = \mathbf{M} \odot \left(\hat{\mathbf{J}} - \gamma \hat{\mathbf{\Theta}}(\lambda)^{-1} \hat{\mathbf{J}} \hat{\mathbf{\Theta}}(\lambda)^{-1} \right) + \mathbf{E} \,. \tag{13}$$

Now by the expression of $\mathbf{E}$ (12), $\mathbf{E}$ has the same support as $\mathbf{M}$, so $\mathbf{E} = \mathbf{M} \odot \mathbf{E}$, so $\hat{\mathbf{J}} = \mathbf{M} \odot \hat{\mathbf{J}}$, and (13) simplifies to

$$\mathbf{M} \odot (\hat{\mathbf{\Theta}}(\lambda)^{-1} \hat{\mathbf{J}} \hat{\mathbf{\Theta}}(\lambda)^{-1}) = \mathbf{E}/\gamma \,. \tag{14}$$

Using the mixed Kronecker matrix-vector product property $\operatorname{vec}(\mathbf{ACB}^\top) = (\mathbf{A} \otimes \mathbf{B}) \operatorname{vec}(\mathbf{C})$, by vectorizing (14), we get

$$\operatorname{vec}(\mathbf{M}) \odot (\hat{\mathbf{\Theta}}(\lambda)^{-1} \otimes \hat{\mathbf{\Theta}}(\lambda)^{-1}) \operatorname{vec}(\hat{\mathbf{J}}) = \operatorname{vec}(\mathbf{E})/\gamma \,. \tag{15}$$

Writing $\mathbf{K} = \hat{\mathbf{\Theta}}(\lambda)^{-1} \otimes \hat{\mathbf{\Theta}}(\lambda)^{-1}$, we have $\mathbf{K} \operatorname{vec} \hat{\mathbf{J}} = \mathbf{K}_{:,\mathcal{S}} (\operatorname{vec} \hat{\mathbf{J}})_{\mathcal{S}}$ because $\operatorname{vec}(\hat{\mathbf{J}})$ is 0 outside of $\mathcal{S}$. Then, (15) can be restricted to entries in $\mathcal{S}$, yielding $\mathbf{K}_{\mathcal{S},\mathcal{S}} (\operatorname{vec} \hat{\mathbf{J}})_{\mathcal{S}} = (\operatorname{vec} \mathbf{E})_{\mathcal{S}}$, which concludes the proof. □

2. $\hat{\mathbf{J}}$ is the image by vec^{-1} of the Jacobian of $\lambda \mapsto \operatorname{vec}(\hat{\mathbf{\Theta}}(\lambda))$

3 Matrix of hyperparameters

In the vein of [18], we now consider the *weighted* GLASSO where the penalty is controlled by a matrix of hyperparameters $\mathbf{\Lambda} \in \mathbb{R}^{p\times p}$. In the weighted GLASSO, $\lambda\|\mathbf{\Theta}\|_1$ is replaced by

$$\|\mathbf{\Lambda} \odot \mathbf{\Theta}\|_1 = \sum_{k,l} \Lambda_{kl}|\Theta_{kl}|\,, \tag{16}$$

with $\mathbf{\Lambda} = (\Lambda_{kl})_{k,l\in[p]}$. Due to its exponential cost in the number of hyperparameters, grid search can no longer be envisioned. In this setting, a notable difference with the scalar hyperparameter case is the dimensionality of the terms. Indeed, the hypergradient $\nabla\mathcal{L}(\mathbf{\Lambda})$ is now represented by a $p\times p$ matrix, while $\mathbf{D}$, $\mathbf{E}$ and $\hat{\mathbf{J}}$ will be represented by fourth-order tensors in $\mathbb{R}^{p\times p\times p\times p}$. For simplicity, we compute each element of the matrix $\nabla\mathcal{L}(\mathbf{\Lambda})$ individually as

$$[\nabla\mathcal{L}(\mathbf{\Lambda})]_{kl} = \sum_{i,j=1}^{p} \frac{\partial\mathcal{C}}{\partial\Theta_{ij}}(\hat{\mathbf{\Theta}}(\mathbf{\Lambda}))\frac{\partial\hat{\Theta}_{ij}}{\partial\Lambda_{kl}}(\Lambda_{kl}) \in \mathbb{R}\,. \tag{17}$$

In the matrix case, the function F becomes $F(\mathbf{Z},\mathbf{\Lambda}) = \operatorname{sign}(\mathbf{Z}) \odot (|\mathbf{Z}| - \gamma\mathbf{\Lambda})_+$. By differentiating the fixed point equation of proximal gradient descent,

$$\hat{\mathbf{\Theta}}(\mathbf{\Lambda}) = F(\underbrace{\hat{\mathbf{\Theta}}(\mathbf{\Lambda}) - \gamma(\mathbf{S} - \hat{\mathbf{\Theta}}(\mathbf{\Lambda})^{-1})}_{\hat{\mathbf{Z}}}, \mathbf{\Lambda})\,, \tag{18}$$

with respect to Λ_{kl}, we obtain a Jacobian that can be expressed by a $p\times p$ matrix $[\hat{\mathbf{J}}_{(\Lambda_{kl})}]_{ij} = \frac{\partial\hat{\Theta}_{ij}}{\partial\Lambda_{kl}}(\mathbf{\Lambda})$. It satisfies

$$\hat{\mathbf{J}}_{(\Lambda_{kl})} = \mathbf{D} : \left(\hat{\mathbf{J}}_{(\Lambda_{kl})} - \gamma\hat{\mathbf{\Theta}}(\mathbf{\Lambda})^{-1}\hat{\mathbf{J}}_{(\Lambda_{kl})}\hat{\mathbf{\Theta}}(\mathbf{\Lambda})^{-1}\right) + \mathbf{E}_{(\Lambda_{kl})}\,.$$

Similarly to the scalar case $D_{ijkl} = \mathbb{1}_{(i,j)=(k,l)}\mathbb{1}_{|\hat{Z}_{ij}|>\gamma\Lambda_{kl}}$ and $[\mathbf{E}_{(\Lambda_{kl})}]_{ij} = -\operatorname{sign}(\hat{Z}_{kl})\mathbb{1}_{(i,j)=(k,l)}\mathbb{1}_{|\hat{Z}_{ij}|>\gamma\Lambda_{kl}}$. The following proposition thus gives the formula for $\hat{\mathbf{J}}_{(\Lambda_{kl})}$.

Proposition 4. *Let $\mathcal{S} \subset [p^2]$ be the set of indices i such that $\operatorname{vec}(|\hat{\mathbf{Z}}|)_i > \gamma\operatorname{vec}(\mathbf{\Lambda})_i$. The Jacobian $\hat{\mathbf{J}}_{(\Lambda_{kl})}$ is given by*

$$\operatorname{vec}\left(\hat{\mathbf{J}}_{(\Lambda_{kl})}\right)_{\mathcal{S}} = \left[\left(\hat{\mathbf{\Theta}}(\mathbf{\Lambda})^{-1}\otimes\hat{\mathbf{\Theta}}(\mathbf{\Lambda})^{-1}\right)_{\mathcal{S},\mathcal{S}}\right]^{-1}\operatorname{vec}\left(\mathbf{E}_{(\Lambda_{kl})}\right)_{\mathcal{S}}/\gamma,$$
$$\operatorname{vec}\left(\hat{\mathbf{J}}_{(\Lambda_{kl})}\right)_{\mathcal{S}^c} = 0\,.$$

The Jacobian of $\hat{\mathbf{\Theta}}(\mathbf{\Lambda})$ with respect to $\mathbf{\Lambda}$ can be represented by the $\mathbb{R}^{p\times p\times p\times p}$ tensor $\hat{\mathbf{J}}$ where $\hat{J}_{ijkl} = \left([\hat{\mathbf{J}}_{(\Lambda_{kl})}]_{ij}\right)_{i,j,k,l}$

We notice that the inverse of the Kronecker product, the bottleneck in the computation of $\hat{\mathbf{J}}$, only has to be computed once for all $(\Lambda_{kl})_{k,l\in[p]}$. By its expression, $\mathbf{E}_{(\Lambda_{kl})}$ is a matrix with a single ±1 element at index $(i,j) = (k,l)$. Therefore $\hat{\mathbf{J}}_{(\Lambda_{kl})}$ is obtained by extracting the only column of $\left[\left(\hat{\mathbf{\Theta}}(\mathbf{\Lambda})^{-1}\otimes\hat{\mathbf{\Theta}}(\mathbf{\Lambda})^{-1}\right)_{\mathcal{S},\mathcal{S}}\right]^{-1}$ indexed by that non-zero element.

4 Experiments

In this section, we present our proposed methodology for tuning the hyperparameter(s) of the GLASSO, and we aim to address the following three questions through our experiments: 1) How does our approach compare to grid-search? 2) What level of improvement can be achieved by extending to matrix regularization? 3) What are the limitations of our method in its current state?

To answer these questions, we generated synthetic data using the `make_sparse_spd_matrix` function of `scikit-learn`, which created a random 100×100 sparse and positive definite matrix $\mathbf{\Theta}_{\text{true}}$ by imposing sparsity on its Cholesky factor. We then sample 2000 points following a Normal distribution $\mathbf{x}_i \sim \mathcal{N}\left(0, \mathbf{\Theta}_{\text{true}}^{-1}\right), i \in [n]$ *i.i.d.*

The criterion and its gradient Selecting the appropriate criterion $\mathcal{C}$ to minimize is not an easy task without strong prior knowledge of the true matrix $\mathbf{\Theta}_{\text{true}}$ to be estimated. In our numerical validation, we use the unpenalized negative likelihood on left-out data. More precisely, we split the data into a training and testing set with a $50-50$ ratio $(\mathbf{x}_i)_{i\in[n]} = (\mathbf{x}_i)_{i\in I_{\text{train}}} \cup (\mathbf{x}_i)_{i\in I_{\text{test}}}$ and we consider the hold-out criterion $\mathcal{C}(\mathbf{\Theta}) = -\operatorname{logdet}(\mathbf{\Theta}) + \langle\mathbf{S}_{\text{test}},\mathbf{\Theta}\rangle$ where $\mathbf{S}_{\text{test}} = \frac{1}{|I_{\text{test}}|}\sum_{i\in I_{\text{test}}}\mathbf{x}_i\mathbf{x}_i^\top$ is the empirical covariance of the test samples (respectively $\mathbf{S}_{\text{train}}$ for the train set). This corresponds to the negative log-likelihood of the test data under the Gaussian assumption $\forall i \in I_{\text{test}},\ \mathbf{x}_i \sim \mathcal{N}(0,\mathbf{\Theta}^{-1})$ *i.i.d* [11]. The intuition behind the use of this criterion is that $\hat{\mathbf{\Theta}}(\lambda)$ should solve the GLASSO problem on the training set while remaining plausible on the test set. Other possible choices include reconstruction errors such as $\mathcal{C}(\mathbf{\Theta}) = \|\mathbf{\Theta}\mathbf{S}_{\text{test}} - \operatorname{Id}\|_F$, but a comparison of the effect of the criterion on the solution is beyond the scope of this paper. In our case, the criterion's gradient $\nabla\mathcal{C}(\mathbf{\Theta})$ is then equal to $-\mathbf{\Theta}^{-1} + \mathbf{S}_{\text{test}}$ [7, §A.4.1].

Computing the Jacobian Based on the previous results we have all the elements at hand to compute the hypergradient for scalar and matrix hyperparameters. In the first case it reads $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\lambda}(\lambda) = \langle\hat{\mathbf{J}}, \nabla\mathcal{C}(\hat{\mathbf{\Theta}}(\lambda))\rangle$ with $\hat{\mathbf{J}}$ as in Proposition 3, while in the latter case it can be computed with the double contraction $\nabla\mathcal{L}(\mathbf{\Lambda}) = \hat{\mathbf{J}} : \nabla\mathcal{C}(\hat{\mathbf{\Theta}}(\mathbf{\Lambda}))$ with $\hat{\mathbf{J}}$ as in Proposition 4. In the code, we use the parametrization $\lambda = \exp(\alpha)$ and $\Lambda_{kl} = \exp(\alpha_{kl})$ respectively for the scalar and matrix regularization, and optimize over α in order to impose the positivity constraint on λ, as in [6]. We rely on the GLASSO solver [13] for computing $\hat{\mathbf{\Theta}}(\cdot)$. For solving (2), we use simple gradient descent with fixed step-size $\rho = 0.1$.

Comparison with grid-search As a sanity check, we first compare our method with a single hyperparameter (scalar case) to grid search. The initial regularization parameter λ^{init} is chosen such that the estimated precision matrix $\widehat{\mathbf{\Theta}}(\lambda^{\text{init}})$ is a diagonal matrix: $\lambda^{\text{init}} = \log(\|\mathbf{S}_{\text{train}}\|_\infty)$. Figure 1 demonstrates that both methods find the same optimal λ, which we refer to as $\lambda_{\text{id}}^{\text{opt}}$, and that a first-order method that is suitably tuned can swiftly converge to this optimum. We also compute in the same Figure the relative error (RE) $\frac{\|\mathbf{\Theta}_{\text{true}} - \widehat{\mathbf{\Theta}}(\lambda)\|}{\|\mathbf{\Theta}_{\text{true}}\|}$ between the estimation and the true matrix (in blue). We notice that $\hat{\mathbf{\Theta}}(\lambda_{\text{id}}^{\text{opt}})$ results in a slightly worse RE than the optimal one. This highlights the importance of the choice of $\mathcal{C}$, which may not necessarily reflect the ability to precisely reconstruct the true precision matrix $\mathbf{\Theta}_{\text{true}}$. Nonetheless, it is important to note that the RE represents an oracle error since, in practical scenarios, we do not have access to $\mathbf{\Theta}_{\text{true}}$. This raises the

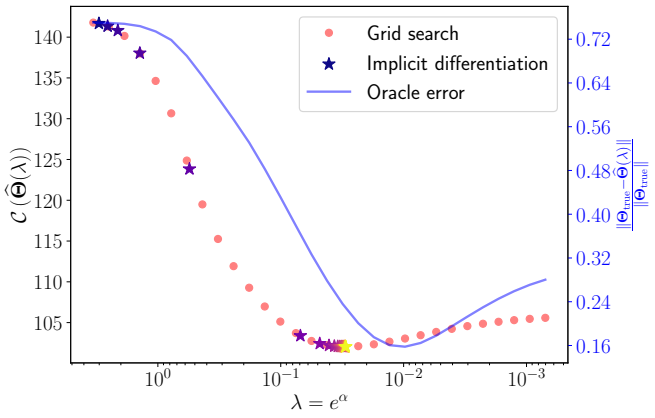

Figure 1 – Value of the criterion $\mathcal{C}$ w.r.t. λ for grid-search and our method, along with the oracle RE.

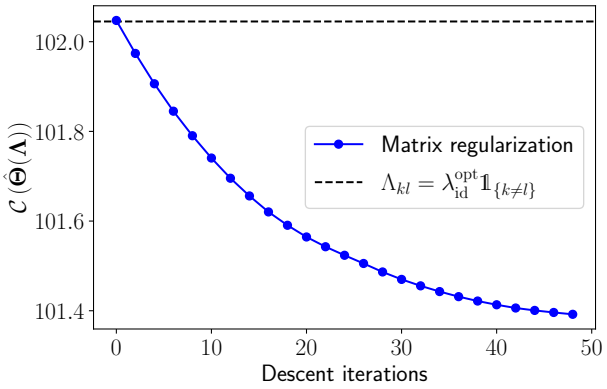

Figure 2 – Outer objective value for the bilevel problem along iterations of hypergradient descent.

essential question of criterion selection, which we defer to future research.

Matrix regularization Our approach demonstrates its value in the context of matrix regularization, where grid search is incapable of identifying the optimal solution within a reasonable amount of time. As depicted in Figure 2, leveraging matrix regularization with appropriately tuned parameters enhances the value of the bilevel optimization problem. Furthermore, as demonstrated in Figure 3, our method successfully modifies each entry Λ_{kl} of the regularization matrix, resulting in an estimated matrix $\hat{\Theta}(\Lambda^{\text{opt}})$ that aligns visually with the oracle Θ_{true}. The edge brought by this improvement remains to be further investigated with respect to the computational cost of the method. While tuning the step-size, we observed that the non-convexity in this case appears to be more severe. We speculate that utilizing more sophisticated first-order descent algorithms from the non-convex optimization literature could be more robust than plain gradient descent.

Conclusion

In this work, we have proposed a first-order hyperparameter optimization scheme based on implicit differentiation for automatically tuning the GLASSO estimator. We exploited the sparse structure of the estimated precision matrix for an

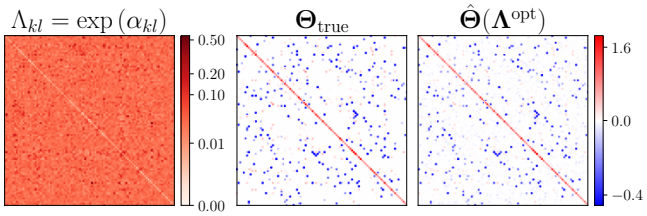

Figure 3 – Visualization of the matrices Λ^{opt}, Θ_{true} and $\hat{\Theta}(\Lambda^{\text{opt}})$.

efficient computation of the Jacobian of the function mapping the hyperparameter to the solution of the GLASSO. We then proposed an extension of the single regularization parameter case to element-wise (matrix) regularization. As future directions of research, we plan on studying the influence of the criterion $\mathcal{C}$ on the sparsity of the recovered matrix, as well as clever stepsize tuning strategies for the hypergradient descent. In the broader sense, we will also benchmark our method against data-based approaches to hyperparameter optimization such as deep unrolling [16]. Finally, we provide high-quality code available freely on GitHub [3] for the reproducibility of our experiments.

References

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

[2] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[3] A. Benfenati, E. Chouzenoux, and J-C. Pesquet. Proximal approaches for matrix optimization problems: Application to robust precision matrix estimation. *Signal Processing*, 2020.

[4] Y. Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 2000.

[5] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *JMLR*, 2012.

[6] Q. Bertrand, Q. Klopfenstein, M. Massias, M. Blondel, S. Vaiter, A. Gramfort, and J. Salmon. Implicit differentiation for fast hyperparameter selection in non-smooth convex learning. *JMLR*, 2022.

[7] S. Boyd and L. Vandenberghe. *Convex optimization*. 2004.

[8] P. Combettes and V. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale modeling & simulation*, 2005.

[9] C.-A. Deledalle, S. Vaiter, J. Fadili, and G. Peyré. Stein Unbiased GrAdient estimator of the Risk (SUGAR) for multiple parameter selection. *SIAM Journal on Imaging Sciences*, 2014.

[10] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil. Forward and reverse gradient-based hyperparameter optimization. *ICML*, 2017.

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

[12] C.-J. Hsieh, M. Sustik, I. Dhillon, P. Ravikumar, et al. QUIC: quadratic approximation for sparse inverse covariance estimation. *JMLR*, 2014.

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

[14] J. Liang, J. Fadili, and G. Peyré. Local linear convergence of forward–backward under partial smoothness. In *NeuRIPS*, 2014.

[15] F. Oztoprak, J. Nocedal, S. Rennie, and P. A. Olsen. Newton-like methods for sparse inverse covariance estimation. *NeurIPS*, 2012.

[16] H. Shrivastava, X. Chen, B. Chen, G. Lan, S. Aluru, H. Liu, and L. Song. GLAD: Learning sparse graph recovery. *ICLR*, 2020.

[17] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. *NeurIPS*, 2012.

[18] W. N. van Wieringen. The generalized ridge estimator of the inverse covariance matrix. *Journal of Computational and Graphical Statistics*, 2019.

3. https://github.com/Perceptronium/glasso-ho