Unveiling Groups of Related Tasks in Multi-Task Learning

Jordan Frecon and Saverio Salzo
CSML, Istituto Italiano di Tecnologia (Italy)
Email: jordan.frecon@iit.it - saverio.salzo@iit.it

Massimiliano Pontil
CSML, Istituto Italiano di Tecnologia (Italy)
Dept. Computer Science, University College London (UK)
Email: massimiliano.pontil@iit.it

***Abstract*—A common approach in multi-task learning is to encourage the tasks to share a low dimensional representation. This has led to the popular method of trace norm regularization, which has proved effective in many applications. In this paper, we extend this approach by allowing the tasks to partition into different groups, within which trace norm regularization is separately applied. We propose a continuous bilevel optimization framework to simultaneously identify groups of related tasks and learn a low dimensional representation within each group. Hinging on recent results on the derivative of generalized matrix functions, we devise a smooth approximation of the upper-level objective via a dual forward-backward algorithm with Bregman distances. This allows us to solve the bilevel problem by a gradient-based scheme. Numerical experiments on synthetic and benchmark datasets support the effectiveness of the proposed method.**

I. Introduction

Multi-task learning (MTL) addresses the problem of leveraging information across multiple related task in order to facilitate learning. There have been various ways to define task relatedness. In this paper we follow the MTL framework outlined in [1] where dependencies among a set of linear regression tasks are exploited through a trace norm regularization problem. It encourages the regression vectors to lie on a low dimensional subspace, thereby constraining the tasks to share a low dimensional representation – see also [2], [3] for a discussion. More precisely, [1] learn the tasks by minimizing the following objective function

$$\sum_{t=1}^{T} \hat{\mathcal{E}}_t(w_t) + \lambda \|W\|_{\mathrm{tr}}, \tag{1}$$

over the matrix $W = [w_1, \ldots, w_T] \in \mathbb{R}^{P \times T}$ of regression vectors of the different learning tasks, where $\hat{\mathcal{E}}_t$ is the empirical error for task t (e.g. the square error), $\lambda > 0$ is a regularization parameter and $\|W\|_{\mathrm{tr}}$ denotes the trace norm (sum of the singular values) of W.

A key insight in MTL is that leveraging similarities between the tasks reduces the amount of data needed to learn each individual task. However, when not all the tasks are related, MTL could cause negative transfer between the tasks and degrades performance (see e.g. [4]). A natural approach to overcome this problem is to weaken the task relatedness assumption, allowing the tasks to partition into groups, so that tasks are related within each group but not across. In the context of MTL methods based on trace norm regularization, this means that the regularizer in (1) would be replaced by the sum of the trace norm restricted to the sub-matrices of regression vectors belonging to the tasks in the prescribed partition. However, in practice, one does not know *a priori* the group structure and this needs to be inferred from data. Identifying the groups of related tasks may be a key step not only in order to decrease the prediction error but also to help model interpretability.

Related Work. The idea of simultaneously learning groups of related tasks and their parameters is not new and has drawn significant attention within the multitask and transfer learning literature. The closest work to our contribution probably traces back to [5], where the authors proposed an alternate minimization scheme in order to simultaneously find the regression matrix and the groups of related tasks. A related method has been devised in [6] to handle the case where all the tasks parameters within a group share the same sparsity patterns. Among other approaches, a popular technique consists in expressing the regression matrix as the product of two low rank matrices: a variable-latent matrix and a latent task matrix whose sparsity patterns will permit to *a posteriori* discover the groups [7], [8]. Yet another line of work is clustering the tasks based on similarity of tasks parameters, in the spirit of k-means clustering, see e.g. [9], [10]. The idea of formulating hyper-parameter optimization, and in particular the problem of learning task relationships, via bilevel optimization has been considered in [11] but in a different context.

Contributions and Outline. We show that the problem of learning how multiple tasks are related can be tackled by extending the smooth bilevel framework proposed in [12], [13], originally devised for vector problems, to a matrix setting. This methodology, which we review in Section II-A, is based on solving two nested optimization problems, a lower-level problem which optimizes over the matrix of task parameters, and an upper level problem, defined via the solution of the lower problem, which optimizes over hyperparameters defining the group structure. Here, we embrace the idea of [12] to replace the lower-level problem with a smooth dual forward-backward algorithm with Bregman distances and make several advances.

First, by elaborating on recent results on the Fréchet derivative of generalized matrix functions, we i) design a smooth

proximity operator for generalized matrix functions in Section II-B, ii) provide an efficient computation of upper-level gradient in Section II-C and iii) prove convergence guarantees of the bilevel scheme in Section II-D. Second, we extend in Section II-E the classical bilevel cross-validation model to k-fold cross-validation and then show empirically that this leads to better generalization. Numerical experiments illustrating the good performance of the proposed method on synthetic and real datasets are reported in Section III.

Notations. Throughout the paper, we set, for every $n \in \mathbb{N}$, $[n] = \{1, \dots, n\}$ and $\odot \colon \mathbb{R}^T \times \mathbb{R}^{P\times T} \to \mathbb{R}^{P\times T}$, $\theta \odot W = [\theta_1 w_1 \cdots \theta_T w_T]$. We define the sampling operator $\mathbf{X} \colon \mathbb{R}^{P\times T} \to \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_T}$ such that

$$\mathbf{X}W = \begin{bmatrix} X_1 w_1 \\ \vdots \\ X_T w_T \end{bmatrix}, \quad X^{(t)} = \begin{bmatrix} x_{t,1}^\top \\ \vdots \\ x_{t,n_t}^\top \end{bmatrix} \in \mathbb{R}^{n_t \times P}.$$

In addition, we denote by $\iota_{\mathcal{C}}$ the indicator function of a set $\mathcal{C}$, meaning the function taking value zero in $\mathcal{C}$ and $+\infty$ otherwise. Finally, we let $2^{\mathcal{C}}$ be the power set of $\mathcal{C}$.

II. Tasks Grouping via Bilevel Optimization

Firstly, we cast the problem of learning the groups of tasks into the bilevel optimization framework outlined in [12], [13]. Secondly, we extend the algorithmic solution to the matrix setting by designing a smooth proximity function. Finally, we report the algorithmic solution, provide convergence guarantees and propose an extension to handle multiple data splits.

A. Bilevel framework

Let $P \geq 1$ be the dimension of the feature space and $T \geq 1$ be the number of tasks. For every task $t \in [T]$, we let $(x_{t,i}, y_{t,i})_{1\leq i\leq n_t} \in (\mathbb{R}^P \times \mathbb{R})^{n_t}$ and $(x_{t,i}^{(\mathrm{val})}, y_{t,i}^{(\mathrm{val})})_{1\leq i\leq n_t} \in (\mathbb{R}^P \times \mathbb{R})^{n_t}$ be the t-th training set and validation set, respectively. We encapsulate the task-grouping structure into an hyperparameter θ, belonging to the set

$$\Theta = \left\{ [\theta_1 \cdots \theta_L] \in [0,1]^{T\times L} \;\middle|\; \sum_{l=1}^{L} \theta_l = \mathbf{1}_T \right\}$$

defining at most L groups. This parameter is a relaxation of a binary variable which is equal to one if the t-th task belongs to the l-th group, and 0 otherwise. The bilevel problem is formalized as follows.

Problem II.1 (Bilevel Problem)**.** *Solve*

$$\underset{\theta\in\Theta}{\text{minimize}}\ \mathcal{U}(\theta) \equiv C(\hat{W}(\theta)) \tag{2}$$

where $C(W) = \frac{1}{2}\|y^{(\mathrm{val})} - \mathbf{X}^{(\mathrm{val})}W\|_F^2$ *is the validation error and where* $\hat{W}(\theta)$ *solves the lower-level problem*

$$\underset{W\in\mathbb{R}^{P\times T}}{\text{minimize}}\ \mathcal{L}(W,\theta) \equiv f(W) + g(A_\theta W), \tag{3}$$

with

$$\begin{cases} f(W) = \frac{1}{2}\|y - \mathbf{X}W\|_F^2 + \frac{\epsilon}{2}\|W\|_F^2, \\ g(Z) = \lambda \sum_{l=1}^{L} \|Z_l\|_{\mathrm{tr}}, \end{cases} \tag{4}$$

where $f \colon \mathbb{R}^{P\times T} \to \mathbb{R}$ *is convex and smooth,* $g \colon \mathbb{R}^{L\times(P\times T)} \to \mathbb{R}$ *is convex and nonsmooth,* $\epsilon > 0$*, and* $A_\theta \colon W \mapsto (\theta_1 \odot W, \dots, \theta_L \odot W)$ *is a linear operator defining the group structure.*

In Problem (II.1), the lower-level problem estimates $\hat{W}(\theta)$ given some group structure θ, while the upper-level problem finds θ such that $\hat{W}(\theta)$ minimizes the validation error. Note that a penalty term $\frac{\epsilon}{2}\|W\|_F^2$ is added to the lower-level problem in order to ensure the uniqueness of $\hat{W}(\theta)$.

Now, since in Problem II.1 the solution $\hat{W}(\theta)$ is in general not available in closed form, in practice one has to design an iterative procedure converging to $\hat{W}(\theta)$ that is properly stopped, say, at the Q-th iterate $W^{(Q)}(\theta)$. It can be approached by several proximal splitting methods, e.g., the Condat-V u primal-dual algorithm [14], [15]. However, they would involve the proximity operator of the trace norm, ultimately leading to a nonsmooth updating mappings. In order to circumvent this issue, since the lower-level objective is strongly convex, we embrace the idea of [12] and solve the dual problem of (3) by a forward-backward algorithm with Bregman distances [16], [17]. Then the primal solution is recovered by the primal-dual relationship. The resulting bilevel problem is the following.

Problem II.2 (Approximate Bilevel Problem)**.** *Given a maximum number of inner iterations* $Q \in \mathbb{N}$*, some step-size* $\gamma > 0$ *and a Legendre function* Φ*, solve*

$$\underset{\theta\in\Theta}{\text{minimize}}\ \mathcal{U}^{(Q)}(\theta) \equiv C(W^{(Q)}(\theta)) \tag{5}$$

with

$$\left\lfloor \begin{array}{l} \text{for } q = 0, 1, \dots, Q-1 \\ \quad \left\lfloor \begin{array}{l} W^{(q)}(\theta) = \nabla f^*\big(-A_\theta^* U^{(q)}(\theta)\big) \\ U^{(q+1)}(\theta) = \mathrm{prox}_{\gamma g^*}^{\Phi}\big(\nabla\Phi(U^{(q)}(\theta)) + \gamma A_\theta W^{(q)}(\theta)\big) \end{array} \right. \\ W^{(Q)}(\theta) = \nabla f^*\big(-A_\theta^* U^{(Q)}(\theta)\big), \end{array} \right. \tag{6}$$

where f^* *and* g^* *are the Fenchel conjugates of* f *and* g *respectively and where the Bregman proximity operator reads*

$$\mathrm{prox}_{\gamma g^*}^{\Phi}(V) = \underset{U\in\mathbb{R}^{L\times(P\times T)}}{\mathrm{argmin}}\ \gamma g^*(U) + \Phi(U) - \langle U, V\rangle. \tag{7}$$

We highlight that the objective $\mathcal{U}^{(Q)}$ is a nonconvex function whose properties depend on the choice of the Legendre function Φ. Indeed, since C and ∇f^* are smooth, then $\mathcal{U}^{(Q)}$ is smooth provided that the proximity operator $\mathrm{prox}_{\gamma g^*}^{\Phi}$ is smooth too. Designing a smooth proximity operator is the cornerstone of the proposed method and will be investigated in the next section. Having a smooth objective $\mathcal{U}^{(Q)}$ permits to address its minimization by a projected gradient algorithm of the form

$$\begin{array}{l} \theta^{(0)} \in \Theta \text{ is chosen arbitrarily} \\ \text{for } k = 0, 1, \dots, K-1 \\ \quad \theta^{(k+1)} = \mathrm{Proj}_\Theta(\theta^{(k)} - \beta\nabla\mathcal{U}^{(Q)}(\theta^{(k)})), \end{array} \tag{8}$$

where Proj_Θ denotes the projection onto Θ and $\beta > 0$ is the step-size chosen as $\beta = 1/\mathcal{L}$ where $\mathcal{L}$ is estimated as follows.

Remark II.1 (Estimation of Lipschitz constant)**.** In practice, we estimate a Lipschitz constant $\mathcal{L}$ computationally by sampling 5 random groups $\theta^{(1)}, \dots, \theta^{(5)}$, i.e.,

$$\mathcal{L} = \max_{1 \leq i < j \leq 5} \frac{\|\nabla \mathcal{U}^{(Q)}(\theta^{(i)}) - \nabla \mathcal{U}^{(Q)}(\theta^{(j)})\|}{\|\theta^{(i)} - \theta^{(j)}\|}. \tag{9}$$

B. Smooth proximity operator of matrix function

In this section, we focus on the design of a smooth proximity function $\operatorname{prox}^{\Phi}_{\gamma g^*}$.

First, we recall that since g is the sum of L trace norms, then g^* is separable and equal to the indicator function of $\mathcal{B}_{\mathrm{sp}}(\lambda)^L$, where $\mathcal{B}_{\mathrm{sp}}(\lambda)$ is the spectral ball of $\mathbb{R}^{P\times T}$ with radius λ. Since g^* is separable, we look for a function Φ separable as well, that is, $\Phi \colon U \mapsto \sum_{l=1}^{L} \phi(U_l)$ for some Legendre function $\phi \colon \mathbb{R}^{P\times T} \to]-\infty, +\infty]$. Then it follows that

$$\operatorname{prox}^{\Phi}_{\gamma g^*}(V) = \left(\operatorname{prox}^{\phi}_{\iota_{\mathcal{B}_{\mathrm{sp}}(\lambda)}}(V_1), \dots, \operatorname{prox}^{\phi}_{\iota_{\mathcal{B}_{\mathrm{sp}}(\lambda)}}(V_L)\right).$$

One usual choice is to consider $\phi = \frac{1}{2}\|\cdot\|^2$ which leads to $\operatorname{prox}^{\phi}_{\iota_{\mathcal{B}_{\mathrm{sp}}(\lambda)}}$ being the projection onto the spectral ball $\mathcal{B}_{sp}(\lambda)$, which is thus nonsmooth.

In order to find a smooth proximity function, we follow the idea of [13], [12] and look for a Legendre function Φ such that $\operatorname{dom}\Phi = \operatorname{dom} g^*$. Here the novelty lies in the design of ϕ such that $\operatorname{dom}\phi = \mathcal{B}_{\mathrm{sp}}(\lambda)$. To do so, we consider a Legendre function acting on the singular values as follows.

Definition II.1 (Choice of Legendre function ϕ)**.** Let $U \in \mathbb{R}^{P\times T}$ with singular value decomposition

$$U = A(U)\operatorname{diag}[\sigma(U)]B(U)^\top,$$

where $A(U) \in \mathbb{R}^{P\times P}$ and $B(U) \in \mathbb{R}^{T\times T}$ are orthogonal matrices, $r = \min(P,T)$, $\sigma(U) = (\sigma_1(U), \dots, \sigma_r(U))$ are the singular values of U listed in decreasing order, and $\operatorname{diag}[\sigma(U)] \in \mathbb{R}^{P\times T}$ has the singular values of U on the principal diagonal and zero entries otherwise. Then we define

$$\phi \colon \mathbb{R}^{P\times T} \to]-\infty, +\infty], \ \phi(U) = \sum_{i=1}^{r} \kappa(\sigma_i(U)), \tag{10}$$

where $\kappa \colon \mathbb{R} \to]-\infty, +\infty]$ is such that

$$\kappa(t) = \begin{cases} -\sqrt{\lambda^2 - t^2} & \text{if } |t| \leq \lambda \\ +\infty & \text{otherwise.} \end{cases} \tag{11}$$

For such choice, it is easy to prove that $\operatorname{prox}^{\phi}_{\iota_{\mathcal{B}_{\mathrm{sp}}(\lambda)}} = \nabla\phi^*$ where ϕ^* denotes the Fenchel conjugate of ϕ. We recall from [18, Proposition 6.2], that ϕ and ϕ^* are differentiable in the interior of $\mathcal{B}_{\mathrm{sp}}(\lambda)$ and

$$\nabla\phi(U) = A(U)\operatorname{diag}[(\kappa'(\sigma_i(U)))_{1\leq i\leq r}]B(U)^\top, \tag{12}$$

$$\nabla\phi^*(U) = A(U)\operatorname{diag}[(\kappa^{*\prime}(\sigma_i(U)))_{1\leq i\leq r}]B(U)^\top, \tag{13}$$

where

$$\kappa'(t) = t/\sqrt{\lambda^2 - t^2} \text{ and } \kappa^{*\prime}(s) = \lambda s/\sqrt{1+s^2}. \tag{14}$$

Now, it only remains to prove that the mappings $\nabla\phi \colon \operatorname{int}(\mathcal{B}_{\mathrm{sp}}(\lambda)^L) \to \mathbb{R}^{P\times T}$ and $\nabla\phi^* \colon \operatorname{int}(\mathcal{B}_{\mathrm{sp}}(\lambda)^L) \to \mathbb{R}^{P\times T}$ are differentiable. However, contrary to the case considered in [12], here $\nabla\phi$ and $\nabla\phi^*$ are generalized matrix functions, that is, are defined through the singular value decomposition [19], [20]. Thus, establishing their differentiability and computing the corresponding derivatives is more challenging. While some formula has been devised in specific cases, e.g., for the SVD [21], [22], [23] and for diagonalizable matrices [24], here we rely on a general result concerning the Fréchet derivative of generalized matrix functions given in [25, Theorem 3.5 and Corollary 3.10], which we recall below.

Lemma II.1 (Generalized Daleckii-Krein formula)**.** *Let $V = A(V)\operatorname{diag}[\sigma_1(V), \dots, \sigma_r(V)]B(V)^\top$ be the singular value decomposition of a matrix $V \in \mathbb{R}^{P\times T}$, where $A(V) \in \mathbb{R}^{P\times P}$, $B(V) \in \mathbb{R}^{T\times T}$, and $r = \min(P,T)$. Let $h : [0, b[\to \mathbb{R}$ be differentiable such that $h(0) = 0$ and $\sigma(V) \subset [0, b[$. Then, the generalized matrix function $h^\diamond \colon U \mapsto A(U)\operatorname{diag}[(h(\sigma_i(U)))_{1\leq i\leq r}]B(U)^\top$ is differentiable at V and its Fréchet derivative at V, applied to the direction $E \in \mathbb{R}^{P\times T}$, is*

$$\nabla h^\diamond(V)[E] = A(V)\Big(F[h](V) \circ \hat{E}(V) + G[h](V) \circ \Upsilon(\hat{E}(V))\Big)B(V)^\top, \tag{15}$$

where the symbol $\circ$ denotes the Hadamard product of matrices; $\hat{E}(V) = A(V)^\top E B(V)$; the linear operator $\Upsilon \colon \mathbb{R}^{P\times T} \to \mathbb{R}^{P\times T}$ is a generalization of the transpose operator; $F[h](V) \in \mathbb{R}^{P\times T}$ and $G[h](V) \in \mathbb{R}^{P\times T}$ are appropriate symmetric matrices depending on the singular values of V (the precise expressions of those matrices are borrowed from [25, Theorem 3.8]).

Therefore, it follows from Lemma II.1 that the generalized matrix functions $\nabla\phi$ and $\nabla\phi^*$ are Fréchet differentiable and, for any $E \in \mathbb{R}^{P\times T}$,

$$\nabla^2\phi(V)[E] = A(V)\Big(F[\kappa'](V) \circ \hat{E}(V) + G[\kappa'](V) \circ \Upsilon(\hat{E}(V))\Big)B(V)^\top \tag{16}$$

and

$$\nabla^2\phi^*(V)[E] = A(V)\Big(F[\kappa^{*\prime}](V) \circ \hat{E}(V) + G[\kappa^{*\prime}](V) \circ \Upsilon(\hat{E}(V))\Big)B(V)^\top. \tag{17}$$

C. Hypergradient computation

Now that $\mathcal{U}^{(Q)}$ is smooth, we address the computation of its gradient $\nabla\mathcal{U}^{(Q)}$, called hypergradient in this context. The overall procedure, reported in Algorithm 1, is made of two steps. First, the output $W^{(Q)}$ of algorithm (6) is computed by using the formulas (12)–(13) and (14). Second, we embrace a reverse mode differentiation algorithm where one needs to subsume formulas (16)-(17).

Overall, the computational load of the hypergradient is mostly due to the required $4QL$ singular value decompositions of matrices of size $P \times T$. This number can be halved by storing the decompositions during the solving of the lower-level problem.

D. Convergence guarantees

We prove that the sequence generated by (6) with the Legendre function stated in Definition II.1 uniformly converges to $\hat{W}(\theta)$. In addition, by elaborating on [12, Theorem 2.1], one additionally have convergence of the approximate bilevel scheme to the exact one as the number of inner iterations grows.

Theorem II.1. *The sequence $\{W^{(Q)}(\theta)\}_{Q\in\mathbb{N}}$ generated by* (6)*, where Φ is defined through the separable Hellinger-like function* (10)-(11)*, converges to the minimizer $\hat{W}(\theta)$ of the lower-level objective in Problem II.1 for any step-size $0 < \gamma < \epsilon\lambda^{-1}\|A_\theta\|^{-2}$. In addition, if $\gamma = \epsilon\lambda^{-1}\|A_\theta\|^{-2}/2$, then for every $Q \in \mathbb{N}$*

$$\frac{1}{2}\|W^{(Q)}(\theta) - \hat{W}(\theta)\|_2^2 \leq \frac{2\lambda\epsilon^{-2}\|A_\theta\|^2}{Q} D_\Phi(\hat{U}(\theta), U^{(0)}),$$

where D_Φ is the Bregman distance associated to Φ, i.e., ($\forall U \in \operatorname{dom}\Phi$, $\forall V \in \operatorname{int}\operatorname{dom}\Phi$),

$$D_\Phi(U, V) = \Phi(U) - \Phi(V) - \langle \nabla\Phi(U), U - V\rangle. \tag{18}$$

Moreover, the approximate Problem II.2 converges to the exact Problem II.1 in the sense that

$$\begin{cases} \inf_{\theta\in\Theta} \mathcal{U}^{(Q)}(\theta) \underset{Q\to+\infty}{\longrightarrow} \inf_{\theta\in\Theta} \mathcal{U}(\theta) \\ \operatorname{argmin}_{\theta\in\Theta} \mathcal{U}^{(Q)}(\theta) \underset{Q\to+\infty}{\longrightarrow} \operatorname{argmin}_{\theta\in\Theta} \mathcal{U}(\theta), \end{cases} \tag{19}$$

where the latter is meant as set convergence, i.e., $\max\{\operatorname{dist}(\hat{\theta}, \operatorname{argmin}\mathcal{U}) \,|\, \hat{\theta} \in \operatorname{argmin}\mathcal{U}^{(Q)}\} \to 0$ *as* $Q \to +\infty$.

Proof. We follow the reasoning done in [12, Section A.2] which itself relies on the results of [16], [17]. We are only left with proving that the Legendre function ϕ is λ^{-1}-strongly convex and hence Φ too. To do so, for every $E \in \mathbb{R}^{P\times T}$ and $V \in \mathbb{R}^{P\times T}$, by successively making use of Lemma II.1, the inequality $\hat{E}_{i,j}(V)\hat{E}_{j,i}(V) \geq -\frac{1}{2}(\hat{E}_{i,j}(V)^2 + \hat{E}_{j,i}(V)^2)$, and the fact that $F[\kappa'](V)$ and $G[\kappa'](V)$ are symmetric, one can prove that

$$\begin{aligned}\langle E, \nabla^2\phi(V)[E]\rangle_{\mathrm{F}} \geq \sum_{i\neq j\leq\nu} \big(F_{i,j}[\kappa'](V) - G_{i,j}[\kappa'](V)\big)\hat{E}_{i,j}(V)^2 \\ + \sum_{(i,j)} F_{i,j}[\kappa'](V)\hat{E}_{i,j}(V)^2.\end{aligned}$$

Finally, by distinguishing multiple cases depending on the values of $F[\kappa'](V)$ and $G[\kappa'](V)$, we get $\langle E, \nabla^2\phi(V)[E]\rangle_{\mathrm{F}} \geq \lambda^{-1}\|E\|^2$ which completes the proof. □

E. Extension to multiple splits

Formally, one can consider all the possible splittings of the dataset $\mathcal{D}$, i.e.,

$$\begin{aligned} &\underset{\theta\in\Theta}{\text{minimize}} \sum_{\mathcal{I}\in 2^{\mathcal{D}}} C_{\mathcal{D}\setminus\mathcal{I}}(\hat{W}_{\mathcal{I}}(\theta)) \\ &\text{where } \begin{cases} C_{\mathcal{D}\setminus\mathcal{I}} = \frac{1}{2}\|y_{|\mathcal{D}\setminus\mathcal{I}} - \mathbf{X}_{|\mathcal{D}\setminus\mathcal{I}} \cdot \|_F^2, \\ \hat{W}_{\mathcal{I}}(\theta) \text{ solves } \underset{W\in\mathbb{R}^{P\times T}}{\text{minimize}} \mathcal{L}_{\mathcal{I}}(W, \theta), \end{cases} \end{aligned} \tag{20}$$

with $\mathcal{L}_{\mathcal{I}}(W, \theta) = \frac{1}{2}\|y_{|\mathcal{I}} - \mathbf{X}_{|\mathcal{I}}W\|_2^2 + \frac{\epsilon}{2}\|W\|_F^2 + \lambda\sum_{l=1}^L \|\theta_l \odot W\|_{\mathrm{tr}}$. Here a fraction $\mathcal{I}$ of the data is picked to form the training set while the rest of the points $\mathcal{D}\setminus\mathcal{I}$ is assigned to the validation set. Note that this formulation fits Problem II.1 by replacing $\hat{W}(\theta)$ with the concatenation of parameter matrices $[\hat{W}_{\mathcal{I}}(\theta)]_{\mathcal{I}\in 2^{\mathcal{D}}}$. Hence, Theorem II.1 also encompasses the case of multiple splittings. Interestingly, this formulation is amenable to a stochastic optimization procedure that we will exploit in Section III-A. Notice that this corresponds to k-fold cross validation when $\mathcal{I}$ is restricted to belong to one of k equal sized partitions of $\mathcal{D}$.

Algorithm 1 Hypergradient computation (HyperGrad)

Require: Training set $\mathcal{D}_{\mathrm{trn}} = (X_t, y_t)_{1\leq t\leq T}$, validation set $\mathcal{D}_{\mathrm{val}} = (X_t^{(\mathrm{val})}, y_t^{(\mathrm{val})})_{1\leq t\leq T}$, number of inner iterations Q, group structure θ, initial point $U^{(0)}$ (optional)

Initialization
If $U^{(0)}$ is not provided, then let $U^{(0)}(\theta) \equiv 0 \in \mathbb{R}^{P\times T\times L}$.

Solving the lower-level problem
Let $D_t = X_t^\top X_t + \epsilon\mathrm{Id}_P$ for $t = 1, \dots, T$.
for $q = 0$ to $Q - 1$ **do**
$W^{(q)}(\theta) = \Big(D_t^{-1}\big(X_t^\top y_t - \sum_{l=1}^L \theta_{l,t} \odot u_{t,l}^{(q)}(\theta)\big)\Big)_{1\leq t\leq T}$
$V^{(q)}(\theta) = \Big(\nabla\phi(U_l^{(q)}(\theta)) + \gamma\theta_l \odot W^{(q)}(\theta)\Big)_{1\leq l\leq L}$
$U^{(q+1)}(\theta) = \Big(\nabla\phi^*\big(V_l^{(q)}(\theta)\big)\Big)_{1\leq l\leq L}$
end for
$W^{(Q)}(\theta) = \Big(D_t^{-1}\big(X_t{}^\top y_t - \sum_{l=1}^L \theta_l \odot u_{t,l}^{(Q)}(\theta)\big)\Big)_{1\leq t\leq T}$.

Computing the hypergradient
$Z^{(Q)}(\theta) = \Big(D_t^{-1} X_t^{(\mathrm{val})\top}(X_t^{(\mathrm{val})} w_t^{(Q)}(\theta) - y_t^{(\mathrm{val})})\Big)_{1\leq t\leq T}$,
$\tilde{H}^{(Q)} = -\big(\theta_l \odot Z^{(Q)}(\theta)\big)_{1\leq l\leq L}$,
$H^{(Q)} = -\big(U_l^{(Q)}(\theta) \odot Z^{(Q)}(\theta)\big)_{1\leq l\leq L}$.
for $q = Q - 1$ to 0 **do**
$S^{(q)} = \big(\nabla^2\phi^*(V_l^{(q)}(\theta))[\tilde{H}_l^{(q+1)}]\big)_{1\leq l\leq L}$,
$Z^{(q)}(\theta) = \Big(D_t^{-1}\sum_{l=1}^L \theta_{t,l} \odot s_{t,l}^{(q)}\Big)_{1\leq t\leq T}$,
$\tilde{H}^{(q)} = \big(\nabla^2\phi(U_l^{(q)}(\theta))[S_l^{(q)}] - \gamma\theta_l \odot Z^{(q)}(\theta)\big)_{1\leq l\leq L}$,
$H^{(q)} = \gamma\big(-U_l^{(q)}(\theta) \odot Z^{(q)}(\theta) + S_l^{(q)} \odot W^{(q)}(\theta)\big)_{1\leq l\leq L} + H^{(q+1)}$.
end for

Ensure: Hypergradient $\nabla\mathcal{U}^{(Q)}(\theta) = H^{(0)}$, dual variable $U^{(Q)}$

III. Numerical Experiments

In this section, we provide numerical experiments supporting the effectiveness of the proposed method.

A. Cross-validation vs. Monte-Carlo cross-validation

We compare four variants of the proposed method reported in Algorithm 2. If not mentioned otherwise, we set $Q = 500$ inner iterations, $\epsilon = 10^{-3}$ and momentum $\beta_k = 0$ for $k \in \mathbb{N}_*$.

- (CV) corresponds to the standard case of bilevel cross-validation where the inner-level problem and outer-level are optimized over fixed training and validation set, respectively.

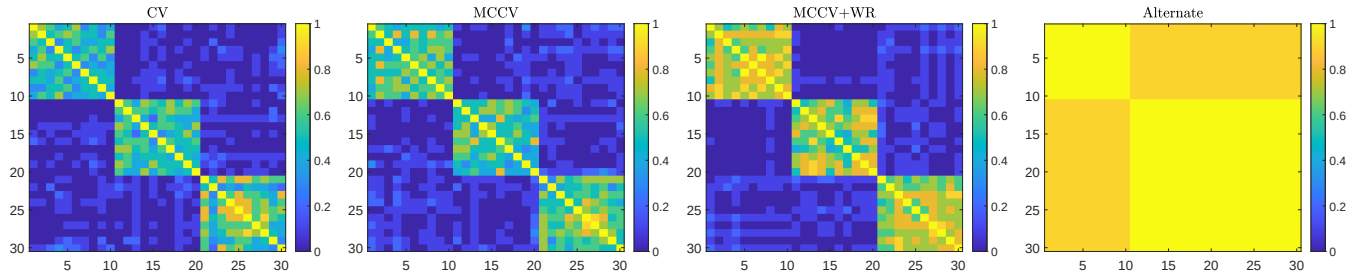

Fig. 1. Comparison of the average covariance matrix $\theta^\top\theta$ obtained by bilevel cross-validation (CV), Monte-Carlo cross-validation (MCCV), Monte-Carlo cross-validation with warm restart (MCCV+WR) and alternate optimization (Alternate). It shows that (MCCV+WR) better captures the grouping of the tasks into 3 groups while (Alternate) fails to identify the groups.

Algorithm 2 Proposed BiGMTL

Require: Dataset $\mathcal{D}$, number of groups L, momentum $[\beta_0, \dots, \beta_{K-1}] \in [0,1]^K$, number of inner iterations Q, number of outer iterations K.

Estimate $\mathcal{L}$ (see Remark II.1). Set $\alpha = 1/\mathcal{L}$.
Initialize $\theta^{(0)}$. Set $U(\theta) \equiv 0 \in \mathbb{R}^{P\times T\times L}$.

for $k = 0$ to $K-1$ **do**
Randomly split $\mathcal{D}$ into $\mathcal{D}_{\text{trn}}$ and $\mathcal{D}_{\text{val}}$ of equal size

Computation of the hypergradient
if warm-restart of the inner-level algorithm **then**
$(\nabla\mathcal{U}^{(Q)}(\theta^k), U) = \text{HyperGrad}(\mathcal{D}_{\text{trn}}, \mathcal{D}_{\text{val}}, Q, \theta^{(k)}, U)$
else
$\nabla\mathcal{U}^{(Q)}(\theta^k) = \text{HyperGrad}(\mathcal{D}_{\text{trn}}, \mathcal{D}_{\text{val}}, Q, \theta^{(k)})$
end if

Upgrade the groups
$g_{k+1} = \beta_k g_k + (1-\beta_k)\nabla\mathcal{U}^{(1)}(\theta^{(k)})$
$\theta^{(k+1)} = \mathcal{P}_\Theta(\theta^k - \alpha g_{k+1})$
end for

Ensure: Group-structure $\theta^{(K)}$

- (MCCV) is a variant in the same spirit of Monte-Carlo cross-validation where, at each outer-iteration $k \in \mathbb{N}$, a fraction of the data is randomly selected to form the training set and the rest of the points is assigned to the validation set. This method can be seen as a stochastic optimization of the problem stated in Section II-E. For the sake of simplicity, we choose sets of equal size.
- (CV+WR) and (MCCV+WR) are a computionnally efficient alternatives to (CV) and (MCCV), respectively. They differs from the fact that they only operate a single inner iteration, i.e., $Q = 1$, but perform warm-restart of the inner-level algorithm. We set the momentum parameter $\beta_k = (k-1)/k$ for $k \in \mathbb{N}_*$ in order to perform moving average gradient descent.

All four methods are compared on the following toy dataset.

Synthetic dataset We consider a synthetic dataset similar to the one in [5]. It is made of $T = 30$ tasks arranged in $L = 3$ groups of 10 tasks each. Using a randomly generated orthogonal matrix of dimension $P \times P$, we build L orthogonal subspaces $(\mathcal{H}_l)_{1\le l\le L}$ of $\mathbb{R}^P$ each one with a random dimension $r_l \ge 2$, such that $\sum_{l=1}^L r_l \le P$. In each subspace $\mathcal{H}_l$, we randomly pick 10 vectors w_t^*, so to form the oracle regression matrix $W^* = [w_1^* \cdots w_T^*] \in \mathbb{R}^{P\times T}$ with $T = \sum_{l=1}^L T_l$. The training, validation and test sets are designed as follows. For each task t, the design matrix $X_t \in \mathbb{R}^{N\times P}$, with $N = 10$ and $P = 20$, is first drawn according to a normal distribution, and then normalized column-wise. The corresponding output is defined according to $y_t = X_t w_t^* + \varepsilon_t$ with $\varepsilon_t \sim \mathcal{N}(0, \sigma_t^2)$, where $\sigma_t^2 = \sigma_0^2 \text{Var}[X_t w_t^*]$ and $\sigma_0 = 0.1$, so to have the same signal to noise ratio for each task. Notice that, contrary [5], here we allow for groups of tasks whose rank is picked at random at each realization.

In our experiment, we do not assume that the true number of groups is known. Instead, we let the methods find at most 6 groups. The regularization parameter λ is selected on a grid $\lambda \in \Lambda = [10^{-2.5}, \dots, 10^{2.5}]$ to minimize the validation error. Results are averaged over 10 realizations and are presented in Table I in terms of mean test error and relative group error $\|\cdot - \theta^*\|_F^2 / \|\theta^*\|_F$, where θ^* denotes the oracle group structure. The corresponding standard deviations are reported in between parenthesis.
Overall, we have found a slight improvement in performing Monte-Carlo cross-validation (MCCV) over standard cross-validation (CV). The (MCCV+WR) strategy yields a large gain in both test error and group recover at a much lower computational cost than (MCCV). This result is additionally supported by the visual inspection of the average task covariance matrix $\theta^\top\theta$, reported in Figure 1, showing a clearer grouping into 3 groups of 10 tasks each.

In order to quantify the computational gain of the proposed variant (CV+WR) over (CV), we have conducted the following experiment. For a fixed $\lambda = 0.25$, we have run the two variants on the synthetic experiment for an increasing number of tasks. Simulations were run on an Intel® Core™ i7-9750H Processor at 2.60 Ghz. The average time per iteration is reported in Figure 2 (left plot). Unsurprisingly, (CV+WR) is about 500 times faster than (CV). We have found empirically that both methods converge with a number of iterations of the same order of magnitude ($\approx$ factor 5), hence suggesting that the variant (CV+WR) converges 100 times faster. We have further investigated the time requirement of (CV+WR) by

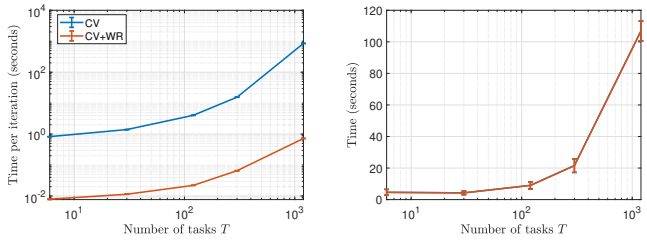

Fig. 2. Comparison between the proposed standard bilevel approach (CV) with $Q = 500$ inner iterations and its variant (CV+WR). (Left) Average execution time per iteration. (Right) Time requirement of the entire bilevel scheme.

	Test err. ($\times 10^{-2}$)	Group err. ($\times 10^{-1}$)
CV	3.923 ($\pm$ 0.663)	6.60 ($\pm$ 1.95)
CV+WR	3.769 ($\pm$ 0.488)	4.27 ($\pm$ 2.72)
MCCV	3.918 ($\pm$ 0.707)	6.20 ($\pm$ 2.90)
MCCV+WR	**3.605 ($\pm$ 0.536)**	**3.87 ($\pm$ 3.09)**
Alternate	4.417 ($\pm$ 0.458)	13.67 ($\pm$ 2.11)

TABLE I
COMPARISON BETWEEN BILEVEL CROSS-VALIDATION (CV), MONTE-CARLO CROSS-VALIDATION (MCCV), THEIR VARIANTS WITH WARM RESTART, NAMELY (CV+WR) AND (MCCV+WR), AS WELL AS AN ALTERNATE OPTIMIZATION METHOD (ALTERNATE).

setting a stopping criterion based the mean norm difference between two successive iterates θ. The tolerance is fixed to 10^{-4} and the corresponding execution times, averaged over 10 realizations, are reported in Figure 2 (right plot). These indicate that (CV+WR), and thus also (MCCV+WR), are very practical methods for solving large-scale problems. In the following we restrict ourself to that variant denoted BiGMTL for *Bilevel Grouping for Multi-Task Learning*.

B. Bilevel vs. alternate optimization

In order to further contrast our bilevel optimization procedure, we have designed an alternating scheme to minimize the lower-level objective function of Problem II.1 with respect to W and θ, i.e.,

$$\begin{aligned} &\text{for } k = 0, 1, \ldots, K-1 \\ &\left\lfloor \begin{aligned} W^{(k+1)} &= \underset{W \in \mathbb{R}^{P \times T}}{\text{minimize}}\ \mathcal{L}(W, \theta^{(k)}) \\ \theta^{(k+1)} &= \underset{\theta \in \Theta}{\text{minimize}}\ \mathcal{L}(W^{(k+1)}, \theta) \end{aligned} \right. \end{aligned} \tag{21}$$

In practice, each minimization step is replaced by an algorithm which performs Q steps. This method is in the same spirit of [5] which considered a trace norm square penalty instead. As an aside, note that the latter model do not fit our proposed bilevel framework since, to the best of our knowledge, we cannot devise a smooth solver satisfying the assumptions of [12, Theorem 2.1] when the trace norm is replaced by the trace norm square.

Experiments are conducted on the previous synthetic dataset for various values of Q, namely 1 (alternate gradient descent), 10, 50 and 500 (to mimic alternate minimization). Results are provided in Table I and Figure 1 for the best choice of Q and show that such alternating optimization poorly identify the groups of tasks. Indeed, it mostly estimate either a single group or two groups (tasks 1 to 10 and 11 to 30), thus missing the existence of a third group. The bilevel scheme is thus key to achieve both good test performance and group recovery.

C. Comparison with state-of-the art

We now compare the performance of the proposed method against the following methods whose codes are available online. All their hyperparameters are choosen thin grids to minimize the validation error.

- (Whom) [5] is the closest work to our contribution. This method is based on an alternating minimization procedure which depends on a regularization parameter λ enforcing trace norm square regularization and a smoothing parameter set to 10^{-3}.
- (RMTL) The "Robust Multi-Task Learning" model [26] involves two regularization parameters. The first controls the amounts trace norm penalty and thus plays the role of our λ while the second specifies the amount of regularization applied to detect outlier tasks.
- (GO-MTL) The "Grouping and Overlap in Multi-Task Learning" algorithm [7] requires to set 2 regularization parameters and the number of latent tasks, which we assume to be equal to our guess of number of groups of tasks.
- (MeTaG) The "Multi- Level Task Grouping" method [27] aims to learn multi-level task groups by assuming that there are H levels.

In addition, we compare to STL [1] and STL2 [3] which are the counterpart of the proposed method and of [5], respectively, when all the tasks are assumed to be related (i.e., single group). The experiments are conducted on 10 realizations of the synthetic dataset described in Section III-A as well on the following benchmark datasets.

Animals with Attributes 2 [28]. This datasets consists of 37322 images of $T = 50$ animals classes with pre-extracted ResNet features for each image. Each task consists of the binary classification of one type of animal. We reduce the feature dimensionality to 468 by using a PCA so as to retain 90% of the total variance. For each task, the data is then normalized and split into balanced training, validation, and test sets of 600 examples each such that there is an equal number of samples from the positive and negative class. We look for a partition of the animals into at most $L = 10$ groups. Results are averaged over 10 splits.

Parkinson Disease [29]. This multi-task regression dataset is obtained from biomedical voice measurements taken from 42 people with early-stage Parkinson's disease. Each task corresponds to the prediction of the symptom score of one patient (so $T = 42$). The observation collects 19 continuous variables including age, gender, time interval, and voice measurements. We split the data in order to obtain an average of 50, 50 and 68 observations per task for the training, validation and test sets, respectively. We look for a grouping of the tasks into at most $L = 10$ groups. Results are averaged over 5 random splits.

	Synthetic (mse $\times 10^{-2}$)	Animals (accuracy)	Parkinson (mse $\times 10^{-1}$)	MNIST (accuracy)
Proposed BiGMTL	**3.65** ($\pm$0.50)	**84.96% ($\pm$ 0.90)**	**4.98** ($\pm$0.20)	95.00% ($\pm$ 0.13)
STL [1]	4.41 ($\pm$ 0.36)	83.55% ($\pm$ 0.79)	5.01 ($\pm$0.15)	94.96% ($\pm$ 0.09)
Whom [5]	5.29 ($\pm$ 0.40)	84.72% ($\pm$ 0.65)	5.03 ($\pm$0.16)	**95.02% ($\pm$ 0.11)**
STL2 [3]	4.23 ($\pm$ 0.30)	84.06% ($\pm$ 0.82)	5.11 ($\pm$0.12)	94.76% ($\pm$ 0.16)
RMTL [26]	4.23 ($\pm$ 0.35)	74.04% ($\pm$ 4.28)	5.08 ($\pm$0.19)	94.72% ($\pm$ 0.10)
GO-MTL [7]	6.11 ($\pm$ 0.70)	80.51% ($\pm$ 0.73)	5.95 ($\pm$0.17)	94.96% ($\pm$ 0.13)
MeTaG [27]	7.33 ($\pm$ 0.40)	84.35% ($\pm$ 0.80)	5.03 ($\pm$0.18)	95.01% ($\pm$ 0.09)

TABLE II
RESULTS ON BENCHMARK DATA SETS. WE REPORT THE AVERAGE OVER MULTIPLE SPLITS AND THE STANDARD DEVIATION IN PARENTHESIS.

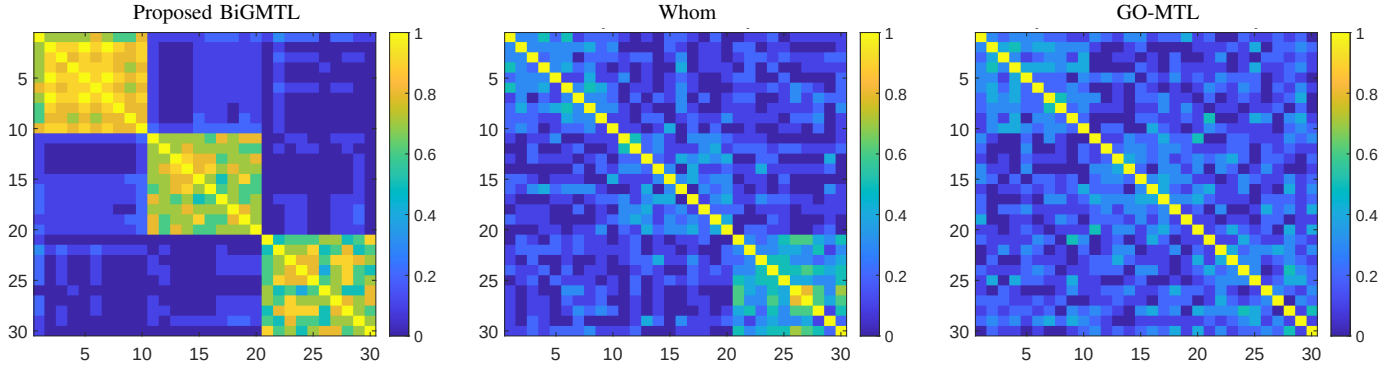

Fig. 3. Mean group covariance matrix $\theta^\top\theta$ on the synthetic experiment. Only the proposed method manages to clearly estimate the three groups of tasks.

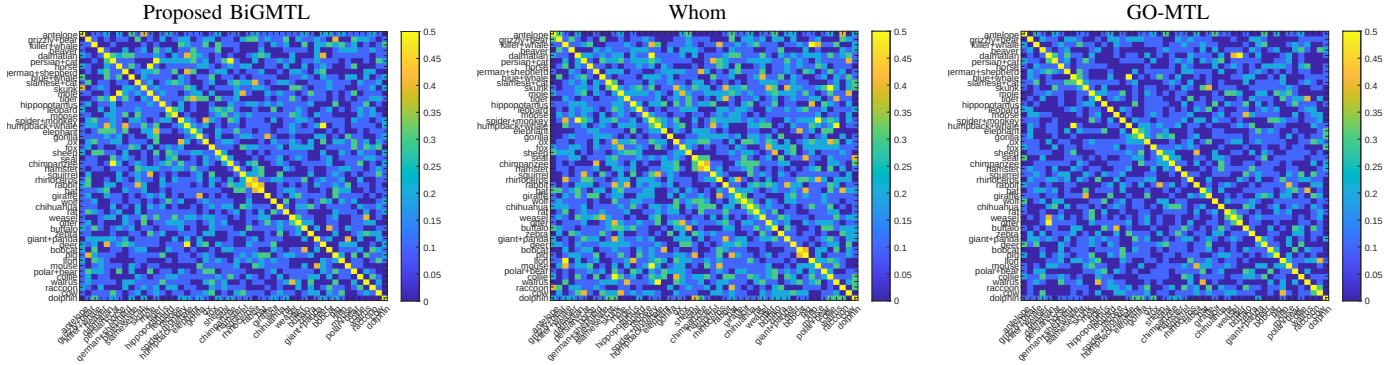

Fig. 4. Mean group covariance matrix $\theta^\top\theta$ estimated on the Animals with Attributes 2 dataset.

MNIST digits [30]. This dataset is made of handwritten digits represented by 28×28 variables. The images are preprocessed with PCA to reduce the dimensionality to 64. Each task corresponds to the binary classification of one digit. The training and validation sets are made of 500 samples respectively, while the test set is made of 1000 samples. We look for a grouping into at most $L = 10$ groups. Results are averaged over 10 splits.

Results are reported in Table II either in terms of mean test error or mean test accuracy depending on the type of problem at hand. Overall, the proposed (BiGMTL) methods yields an improvement over the state-of-the-art methods while no improvement was found on the MNIST dataset. This result has to be contrasted by the fact the MNIST dataset only involves very few tasks. We additionnally report in Figure 3 and Figure 4 the average group covariance matrix $\theta^\top\theta$ estimated by some of the methods on the synthetic dataset and Animals with Attributes 2 dataset, respectively. Notice that while (BiGMTL) and (Whom) explicitly compute θ, (GO-MTL) implicitly gets the group structure θ through the sparsity patterns of the estimated latent task matrix. A visual inspection shows that the covariance matrix of the proposed method is more contrasted thus indicating that the groups found are more robust to the choice of train-validation splitting. For the sake of illustration purposes, we have reported in Figure 5 some of the most recurrent groupings of animals estimated by the proposed method. We believe that most of them make sense.

Note that in principle it is likely that more flexible models such as [8] which yields many hyperparameters might yield better performance. However, this is beyond the scope of the paper as this would require very tedious grids searches. On the contrary the proposed method only requires validation of

Fig. 5. Illustration of some of the most recurrent groupings of animals (top and bottom) obtained by the proposed method.

a single hyperparameter, thus making it more practical for larger scale settings as long as one can efficiently perform 2 SVDs per iteration. A MATLAB toolbox will be made publicly available.

IV. Conclusion

We addressed the simultaneous learning of groups of related tasks and their regression parameters. We framed this problem within the framework of (approximate) bilevel optimization outlined in [13], [12]. A key novelty of our approach is to devise an efficient algorithm to compute the gradient of the upper level objective, exploiting recent results on the derivative of generalized matrix functions. We have provided experimental results that indicate the advantage of working with a variable number of groups over standard trace norm regularization and previous state-of-the-art approaches. In particular, we observed that the groups found by our method are robust to the choice of train-validation splitting. This added to its low computational cost suggest that it would be a promising candidate for interpretability purposes on large scale experiments. A valuable direction of future research is to provide learning bounds for the proposed method.

References

[1] T. K. Pong, P. Tseng, S. Ji, and J. Ye, "Trace norm regularization: Reformulations, algorithms, and multi-task learning," *SIAM J. on Optimization*, vol. 20, no. 6, pp. 3465–3489, Dec. 2010. [Online]. Available: http://dx.doi.org/10.1137/090763184

[2] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *Journal of Machine Learning Research*, vol. 6, pp. 1817–1853, 2005.

[3] A. Argyriou, T. Evgeniou, and M. Pontil, "Convex multi-task feature learning," *Machine Learning*, vol. 73, no. 3, pp. 243–272, Dec 2008.

[4] Y. Zhang and Q. Yang, "A survey on multi-task learning," *arXiv preprint arXiv:1707.08114*, 2017.

[5] Z. Kang, K. Grauman, and F. Sha, "Learning with whom to share in multi-task feature learning," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*. USA: Omnipress, 2011, pp. 521–528.

[6] M. Kshirsagar, E. Yang, and A. C. Lozano, "Learning task clusters via sparsity grouped multitask learning," in *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18-22, 2017, Proceedings, Part II*, 2017, pp. 673–689.

[7] A. Kumar and H. Daume III, "Learning task grouping and overlap in multi-task learning," in *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[8] J.-Y. Jeong and C.-H. Jun, "Variable selection and task grouping for multi-task learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18. New York, NY, USA: ACM, 2018, pp. 1589–1598.

[9] T. Evgeniou, C. A. Micchelli, and M. Pontil, "Learning multiple tasks with kernel methods," *Journal of Machine Learning Research*, vol. 6, pp. 615–637, 2005.

[10] L. Jacob, J.-P. Vert, and F. Bach, "Clustered multi-task learning: A convex formulation," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 745–752.

[11] R. Flamary, A. Rakotomamonjy, and G. Gasso, "Learning constrained task similarities in graphregularized multi-task learning," *Regularization, Optimization, Kernels, and Support Vector Machines*, p. 103, 2014.

[12] J. Frecon, S. Salzo, and M. Pontil, "Bilevel learning of the group lasso structure," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 8311–8321.

[13] P. Ochs, R. Ranftl, T. Brox, and T. Pock, "Techniques for gradient-based bilevel optimization with non-smooth lower level problems," *Journal of Mathematical Imaging and Vision*, vol. 56, no. 2, pp. 175–194, 2016.

[14] L. Condat, "A primal-dual splitting method for convex optimization involving lipschitzian, proximable composite terms," *J. Optim. Theory Appl.*, vol. 158, pp. 460–479, 2013.

[15] B. Vũ, "A splitting algorithm for dual monotone inclusions involving cocoercive operators," *Adv. Comput. Math.*, vol. 38, pp. 667–681, 2013.

[16] H. H. Bauschke, J. M. Borwein, and P. L. Combettes, "Bregman monotone optimization algorithms," *SIAM Journal on control and optimization*, vol. 42, no. 2, pp. 596–636, 2003.

[17] Q. Van Nguyen, "Forward-backward splitting with Bregman distances," *Vietnam Journal of Mathematics*, vol. 45, no. 3, pp. 519–539, 2017.

[18] A. Lewis and H. Sendov, "Nonsmooth analysis of singular values. part i: Theory," *Set-Valued Anal.*, vol. 13, pp. 213–241, 2005.

[19] F. Arrigo, M. Benzi, and C. Fenu, "Computation of generalized matrix functions," *SIAM J. Matrix Anal. Appl.*, vol. 37, pp. 836–860, 2016.

[20] J. Hawkins and A. Ben-Israel, "On generalized matrix functions," *Linear and Multilinear Algebra*, vol. 1, pp. 163–171, 1973.

[21] T. Papadopoulo and M. I. Lourakis, "Estimating the jacobian of the singular value decomposition: Theory and applications," in *European Conference on Computer Vision*. Springer, 2000, pp. 554–570.

[22] M. B. Giles, "Collected matrix derivative results for forward and reverse mode algorithmic differentiation," in *Advances in Automatic Differentiation*, C. H. Bischof, H. M. Bücker, P. Hovland, U. Naumann, and J. Utke, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 35–44.

[23] M. W. Seeger, A. Hetzel, Z. Dai, and N. D. Lawrence, "Auto-differentiating linear algebra," *CoRR*, vol. abs/1710.08717, 2017. [Online]. Available: http://arxiv.org/abs/1710.08717

[24] C. Ionescu, O. Vantzos, and C. Sminchisescu, "Matrix backpropagation for deep networks with structured layers," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2965–2973.

[25] V. Noferini, "A formula for the Fréchet derivative of a generalized matrix function," *SIAM Journal on Matrix Analysis and Applications*, vol. 38, no. 2, pp. 434–457, 2017.

[26] J. Chen, J. Zhou, and J. Ye, "Integrating low-rank and group-sparse structures for robust multi-task learning," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 42–50.

[27] L. Han and Y. Zhang, "Learning multi-level task groups in multi-task learning," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[28] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 9, pp. 2251–2265, Sep. 2019.

[29] A. Tsanas, M. A. Little, P. E. McSharry, and L. O. Ramig, "Accurate telemonitoring of parkinson's disease progression by noninvasive speech tests," *IEEE transactions on Biomedical Engineering*, vol. 57, no. 4, pp. 884–893, 2010.

[30] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.