Chapter 1

Learning constrained task similarities in graph-regularized multi-task learning

Rémi Flamary

Laboratoire Lagrange, UMR 7293, Observatoire de la Côte d'Azur, Université de Nice Sophia-Antipolis, 06108 Nice, FRANCE

Alain Rakotomamonjy

LITIS EA 4108, Université de Rouen, 76800 Saint Etienne du Rouvray, France

Gilles Gasso

LITIS EA 4108, INSA de Rouen, 76801 Saint Etienne du Rouvray, France

This chapter addresses the problem of learning constrained task relatedness in a graph-regularized multi-task learning framework. In such a context, the weighted adjacency matrix of a graph encodes the knowledge on task similarities and each entry of this matrix can be interpreted as a hyperparameter of the learning problem. This task relation matrix is learned via a bilevel optimization procedure where the outer level optimizes a proxy of the generalization errors over all tasks with respect to the similarity matrix and the inner level estimates the parameters of the tasks knowing this similarity matrix. Constraints on task similarities are also taken into account in this optimization framework and they allow the task similarity matrix to be more interpretable for instance, by imposing a sparse similarity matrix. Since the global problem is non-convex, we propose a non-convex proximal algorithm that provably converges to a stationary point of the problem. Empirical evidence illustrates the approach is competitive compared to existing methods that also learn task relation and exhibits an enhanced interpretability of the learned task similarity matrix.

1.1 Introduction

Multi-task learning (MTL) has gained a lot of attention over the past years. Given a set of different but related tasks, the underlying idea is to jointly learn these tasks by exploiting their relationships. Such a procedure has proved useful (in terms of generalization ability) when only few samples are available for the tasks [5, 1, 26, 21]. One of the most important problems in multi-task learning is the assessment of task relatedness. Existing approaches seek task relationship either in input space, feature space [1, 27] or output space [11]. They consider relation between tasks through dedicated regularizations or based on Bayesian priors.

To be more concrete, when learning jointly the tasks, regularization approaches assume that the tasks share a common low-dimensional latent subspace or their parameters are close to the average parameter vector [10]. The links between tasks can also be enforced by imposing a joint sparsity pattern [22, 1] across tasks. However, such a similarity is globally imposed for all tasks and may hinder good generalization performance mainly when unrelated tasks are forced to borrow similar characteristics. Hence, more elaborate approaches such as pairwise similar tasks [10], clustering of tasks [10, 15, 16] or hierarchically structured tasks [28, 33] were proposed in order to cope with this issue. For instance, Widmer et al. [33] supposed the knowledge of a hierarchical tree modeling task dependency while Xing et al. [28] used agglomerative clustering to find the tree which is further applied to learn task parameters. Contrarily, clustering approaches due to [15], [16], [19] or [35] attempt to discover groups of similar tasks along with the estimation of their parameters. In the same vein, Zhang et al. [34] characterized task relatedness via a covariance matrix. Their formulation encompasses various existing methods including pairwise similarity constraints or cluster constraints. Solving for the covariance matrix turns out to be a problem of learning distance metric between tasks.

Most methods for learning task similarities provide good empirical performance, but few of them aim at enhancing the interpretability of the learned task relations. For instance, Zhang et al. [34] learn a dense task covariance matrix which depicts relation between tasks. Since this matrix is dense, it is thus difficult to interpret which task relations are the most relevant for the learning problem. In this work, we look at learning *interpretable* task similarities in multi-task learning problems. We focus on a popular multi-task framework denoted as graph-based regularized framework [10]. As formally defined in the sequel, in this framework, task relations are represented as a graph whose nodes are the tasks and the weighted edges encode some knowledge over similarities between tasks. This framework has been shown to be of practical interests [9, 32] and benefits from very efficient algorithms for solving the related multi-task optimization problems [32].

Our objective and proposal in this chapter is to learn the adjacency matrix of the task relations graph, jointly with the task decision function parameters, while making the graph the more interpretable as possible. Hence, we may accept some slight loss in generalization performance if the gain in graph interpretability is important. This interpretability of the adjacency matrix is achieved by incorporating in the global learning problem some specific constraints over the graph parameters. The constraints we consider in the sequel are usually sparsity-inducing penalties that enforce tasks to be unrelated.

Our main contribution hereafter is to propose a novel procedure for learning similarities between tasks in graph-based multi-task learning. As detailed in the sequel, since in this framework, the relation between a pair of tasks can be interpreted as a hyper-parameter of the global multi-task model, we address the problem as a hyper-parameter optimization issue ([2, 6]). Typically, when few hyper-parameters have to be optimized, a cross-validation procedure, aiming at optimizing an estimation of the generalization error is employed in conjuction with a grid-search over the hyper-parameter values [14, 4]. Since in our framework, the number of hyper-parameters (typically $\mathcal{O}(T^2)$ parameters, for T tasks) to optimize make this approach intractable, the method we advocate consists of a bilevel approach: at the outer level, a generalization criterion over all tasks is employed to measure the goodness of task relatedness parameters and this criterion is thus optimized with respect to these parameters under some sparsity-inducing constraints. The inner level is devoted to the optimization of task parameters for a fixed task relation graph. Due to the non-convexity of the generalization errors with respect to task similarity parameters, the overall problem is non-convex. Fortunately, the inner problem we design is convex and, depending on the loss functions considered, it may admit a closed-form solution. We solve this bilevel problem through non-convex proximal approach with guaranteed convergence properties. The flexibility of the approach allows the use of a broad range of generalization error proxy that can be adapted to the MTL problem at hand. It also allows easy incorporation of constraints over the task relations such as sparsity, *link* or *cannot-link* constraints in the matrix similarity learning process. As a consequence, the learned task-similarity matrix is sparse and provides improved interpretability of connections between tasks. The experimental results on synthetic and real-world problems clearly support this evidence.

The rest of the chapter is organized as follows: section 1.2 describes the graph-based multi-task learning setting we are interested in and states how the task parameters are obtained once the task similarity matrix is fixed. The learning of this matrix is explained in section 1.3 where we formulate the bilevel optimization problem and the non-convex proximal algorithm used to solve it. Finally, empirical comparisons illustrate the compelling performance of the approach. Especially, these experiments show the ability of our algorithm to unravel the underlying structure (groups or manifold) of the tasks and emphasize on the interpretability of the results.

1.2 Similarity based multi-task learning

Before describing how task relatedness is learned, we first present the general multi-task learning framework we are dealing with, as well as the regularizer we have considered for inducing transfer between tasks.

1.2.1 Multi-task learning framework

Assume we are given T learning tasks to be learned from T different datasets $(\mathbf{x}_{i,1}, y_{i,1})_{i=1}^{n_1}, \cdots, (\mathbf{x}_{i,T}, y_{i,T})_{i=1}^{n_T}$, where any $\mathbf{x}_{i,\cdot} \in \mathbb{R}^d$ and $y_{i,\cdot} \in \mathbb{R}$ and n_t denotes the t-th dataset size. In the sequel, we will represent the training examples $\{\mathbf{x}_{i,t}\}_{i=1}^{n_t}$ in a matrix form as $\mathbf{X}_t \in \mathbb{R}^{n_t \times d}$ and the corresponding labels gathered in vector $\mathbf{y}_t \in \mathbb{R}^{n_t}$. For a given task t, we are looking for a linear prediction function $f_t(\mathbf{x})$ of the form

$$f_t(\mathbf{x}) = \mathbf{w}_t^\top \mathbf{x} + b_t \tag{1.1}$$

with $\mathbf{w}_t \in \mathbb{R}^d$ and $b_t \in \mathbb{R}$ being the linear function parameters. Basically, $f_t(\mathbf{x})$ depicts the presumable dependencies between a given example $\mathbf{x}$ and its associated label y.

Multi-task methods aim at learning all T decision functions in a simultaneous way while imposing some constraints that induce relatedness between tasks. Hence, most multi-task learning problems can be cast into the following optimization setup:

$$\min_{\{\mathbf{w}_t\},\{b_t\}} \sum_{t,i} L(f_t(\mathbf{x}_{i,t}), y_{i,t}) + \Omega(\mathbf{w}_1, \cdots, \mathbf{w}_T) \tag{1.2}$$

where $L(f_t(\mathbf{x}), y)$ is a loss function measuring discrepancy between the actual and predicted output related to an example $\mathbf{x}$, Ω a regularizer inducing task relatedness involving thus all vectors $\{\mathbf{w}_t\}$.

1.2.2 Similarity-based regularization

One typical issue in multi-task learning is the choice of a regularization term Ω that efficiently helps in improving the generalization performances of the prediction functions. Indeed, most of the existing MTL regularization terms are based on a strong *prior knowledge* about the task relatedness. We can for instance mention regularizers that enforce similarity of task parameters $\{\mathbf{w}_t\}$ to the average parameter vector $\frac{1}{T}\sum_t \mathbf{w}_t$ [9, 10], that make classifiers belong to a low dimensional linear subspace [1]. Other regularizers induce the classifiers to be agglomerated into clusters [15], or compel tasks to share a common subset of discriminative kernels [23] or even impose tasks to be similar according to a pre-defined task networks [17].

Because it encompasses several forms of the above-mentioned regularizers, we focus on the graph-based regularization term proposed by Evgeniou et al. [10] that induces pairwise similarity between tasks. This regularizer is defined as

$$\Omega(\{\mathbf{w}_t\}_{t=1}^T, \{\lambda_t\}, \mathbf{P}) = \sum_t \lambda_t \|\mathbf{w}_t\|_2^2 + \sum_{t,s} \rho_{t,s} \|\mathbf{w}_t - \mathbf{w}_s\|_2^2 \tag{1.3}$$

where $\lambda_t \in \mathbb{R}^+ \setminus \{0\}$ and $\mathbf{P} \in (\mathbb{R}^+)^{T\times T}$ a matrix of general term $\rho_{t,s}$ (i.e. $\mathbf{P} = [\rho_{t,s}]_{t,s=1}^T$), are the regularization hyper-parameters. The first term of this regularizer corresponds to the classical ℓ_2-norm regularization (ridge) while the second one promotes a pairwise task similarity imposed by the $\rho_{t,s}$ parameters. From the graph point of view, the matrix $\mathbf{P}$ is the weighted graph adjacency matrix and it reflects the relationship between tasks as obviously, a large $\rho_{t,s}$ value enforces tasks s and t to be similar while if $\rho_{t,s} = 0$ then these tasks will likely be unrelated (in the ℓ_2-norm sense). We have imposed $\mathbf{P}(t,t) = 0\ \forall t$ as these diagonal terms of the matrix have no impact on the cost function. Furthermore, in order to reduce the number of hyper-parameters in the model and because it intuitively makes sense, we also have considered $\mathbf{P}$ to be a symmetric matrix. Graph-regularized multi-task learning problems are denoted as such because the regularizer given in Equation 1.3 can be also interpreted as the following. Indeed, by defining matrix $\mathbf{W} = [\mathbf{w}_1 \cdots \mathbf{w}_T]$, one can notice that Equation (1.3) equivalently writes

$$\Omega(\mathbf{W}, \{\lambda_t\}, \mathbf{P}) = \text{trace}\left(\mathbf{W}\Gamma\mathbf{W}^\top\right) \tag{1.4}$$

where $\Gamma = \Lambda + \mathbf{L}$ and Λ is a diagonal matrix with entries λ_t and $\mathbf{L}$ the Laplacian of the graph with vertices the tasks. Assuming the edges are parameters $\rho_{t,s}$, the Laplacian matrix writes $\mathbf{L} = \mathbf{D} + \mathbf{P}$ where $\mathbf{D}$ is a diagonal matrix with elements $\mathbf{D}(t,t) = \sum_{s=1}^{T} \rho_{t,s}$.

Our main contribution is to propose a framework for learning this matrix $\mathbf{P}$ of task relatedness and to make this matrix as interpretable as possible by imposing some constraints on its entries. Because the matrix $\mathbf{P}$ can also be considered as a matrix of hyper-parameters, we introduce here a problem where these hyper-parameters are learned with respect to a proxy of the generalization error. This contribution is of importance for obtaining prediction functions with good generalization capabilities as well as a task similarity matrix that is interpretable. Indeed, using our novel formulation of the problem, it becomes easy to impose single or group sparsity-inducing constraints over entries of $\mathbf{P}$.

Before providing the details of how these task relations are learned, we show in the next paragraph how problem (1.2) can be solved for fixed λ_t and matrix $\mathbf{P}$.

1.2.3 Solving the graph-regularized multi-task learning problem

We focus now on solving problem (1.2) with the regularization term defined as in Equation (1.3). For the sake of clarity, we restrict ourselves to the squared loss function, denoted as $L(f(\mathbf{x}), y) = \frac{1}{2}(f(\mathbf{x}) - y)^2$, although our algorithm can be applied to other loss functions such as the Hinge loss. We discuss this point in the sequel.

Using a quadratic loss function and matrix notation, and based on the regularization given in Equation (1.3), problem (1.2) reads

$$\min_{\{\mathbf{w}_t\},\{b_t\}} \quad J(\{\mathbf{w}_t\}, \{b_t\}, \mathbf{P}, \{\lambda_t\}) \tag{1.5}$$

where the objective function is

$$J(\cdot) = \frac{1}{2}\sum_t \|\mathbf{y}_t - \mathbf{X}_t\mathbf{w}_t - b_t\mathbb{I}_t\|_2^2 + \sum_t \lambda_t\|\mathbf{w}_t\|_2^2 + \sum_{t,s} \rho_{t,s}\|\mathbf{w}_t - \mathbf{w}_s\|_2^2 \tag{1.6}$$

with $\mathbb{I}_t \in \mathbb{R}^{n_t}$ being a vector of ones. Note that for $\lambda_t > 0, \forall t$ and $\rho_{t,s} \geq 0, \forall t, s$, this problem is strictly convex and admits a unique solution. For fixed parameters $\{\lambda_t\}_{t=1}^T$ and $\mathbf{P}$, a closed-form solution of this problem can be obtained by solving the linear system related to the normal equations. The gradient of $J(\cdot)$ w.r.t. the prediction function parameters of task k is given by:

$$\nabla_{\mathbf{w}_k} J = \mathbf{Q}_k\mathbf{w}_k + b_k\mathbf{X}_k^\top\mathbb{I}_k - 4\sum_t \rho_{t,k}\mathbf{w}_t - \mathbf{c}_k \tag{1.7}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{Q}_k = \mathbf{X}_k^\top\mathbf{X}_k + (2\lambda_k + 4p_k)\,\mathbf{I} \in \mathbb{R}^{d\times d}$,

$\mathbf{c}_k = \mathbf{X}_k^\top \mathbf{y}_k \in \mathbb{R}^d$ and finally $p_k = \sum_t \rho_{t,k}$. Similarly, the gradient of $J(\cdot)$ w.r.t. the bias term b_k takes the form

$$\nabla_{b_k} J = \mathbb{1}_k^\top \mathbf{X}_k \mathbf{w}_k + n_k b_k - \mathbb{1}_k^\top \mathbf{y}_k \tag{1.8}$$

From the gradients (1.7) and (1.8) and the resulting optimality conditions, solution of problem (1.5) is obtained by solving the system :

$$\mathbf{A}\boldsymbol{\beta} = \mathbf{c} \tag{1.9}$$

where $\boldsymbol{\beta} = \begin{bmatrix} \tilde{\mathbf{w}}_1^\top & \cdots & \tilde{\mathbf{w}}_T^\top \end{bmatrix}^\top \in \mathbb{R}^{(d+1)\cdot T}$ is the vector containing all the prediction function parameters with $\tilde{\mathbf{w}}_k = \begin{bmatrix} \mathbf{w}_k^\top & b_k \end{bmatrix}^\top \in \mathbb{R}^{d+1}$, $\mathbf{c} = \begin{bmatrix}\tilde{\mathbf{c}}_1^\top \cdots \tilde{\mathbf{c}}_T^\top\end{bmatrix}^\top \in \mathbb{R}^{(d+1)\cdot T}$ with $\tilde{\mathbf{c}}_k = \begin{bmatrix}\mathbf{c}_k^\top & \mathbb{1}_k^\top \mathbf{y}_k\end{bmatrix}^\top$ and $\mathbf{A} \in \mathbb{R}^{(d+1)\cdot T \times (d+1)\cdot T}$ is a matrix of the form:

$$\mathbf{A} = \begin{bmatrix} \tilde{\mathbf{Q}}_1 & -4\rho_{1,2}\tilde{\mathbf{I}} & \dots & -4\rho_{1,T}\tilde{\mathbf{I}} \\ -4\rho_{2,1}\tilde{\mathbf{I}} & \tilde{\mathbf{Q}}_2 & \dots & -4\rho_{2,T}\tilde{\mathbf{I}} \\ \vdots & \vdots & \ddots & \vdots \\ -4\rho_{T,1}\tilde{\mathbf{I}} & -4\rho_{T,2}\tilde{\mathbf{I}} & \dots & \tilde{\mathbf{Q}}_T \end{bmatrix} \tag{1.10}$$

involving the matrices

$$\tilde{\mathbf{Q}}_k = \begin{bmatrix} \mathbf{Q}_k & \mathbf{X}_k^\top \mathbb{1} \\ \mathbb{1}^\top \mathbf{X}_k & n_k \end{bmatrix}, \text{ and } \tilde{\mathbf{I}} = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & 0 \end{bmatrix}$$

Notice that for $\lambda_t > 0\, \forall t$, the matrix $\mathbf{A}$ is full-rank regardless of the matrices $\{\mathbf{X}_t\}$ and thus the linear system (1.9) has a unique solution which provides the optimal parameters of all prediction functions $\{f_t\}_{t=1}^T$. Note that depending on the number of tasks and the dimensionality of the training examples this matrix $\mathbf{A}$ can be pretty large. In such a case, it can be beneficial to take advantage of the sparse structure of $\mathbf{A}$ for the linear system (1.9) resolution. For instance, a Gauss-Seidel procedure, which consists in optimizing alternatingly over the parameters of a given task, can be considered.

For optimizing the task similarity matrix $\mathbf{P}$, the linear system $\mathbf{A}\boldsymbol{\beta} = \mathbf{c}$ will be of paramount importance since it defines an implicit function that relates the optimal task parameters $\{\mathbf{w}_t, b_t\}_{t=1}^T$ to the entries of $\mathbf{P}$. Indeed, the bilevel approach we apply to determine $\mathbf{P}$ (see equations 1.12 and 1.13) in the next section requires the gradient of the estimated generalization error measure in function of $\mathbf{P}$. Owing to this equation we will be able to compute this gradient via the explicit expression of the gradient of $\boldsymbol{\beta}$ with respect to $\mathbf{P}$. Consequently, while we have stated that other loss functions can be considered in our approach, the solution of the graph-regularizer multi-task learning has to satisfy a linear system of the form $\mathbf{A}\boldsymbol{\beta} = \mathbf{c}$. For instance, the Hinge loss function satisfies this property if the learned problem is solved in the dual [18, 31, 9].

1.3 Non-convex proximal algorithm for learning similarities

The multi-task approach with fixed hyperparameters described in the above section is interesting in itself. However, it may be limited by the large number of regularization parameters to be chosen, namely all the $\{\lambda_t\}_{t=1}^T$ and the matrix of task similarities $\mathbf{P}$. When there exists a strong *prior knowledge* concerning task similarities, the $\mathbf{P}$ matrix might be pre-defined beforehand. When no prior information is available, $\mathbf{P}$ can be learned from training data as done by Zhang et al. [34]. In addition, when one's objective is also to gain some insights over the structure of the tasks and how they are related, then constraints on task similarities have to be imposed. In what follows, we describe our algorithm for learning the matrix $\mathbf{P}$ as well as the regularization parameters $\{\lambda_t\}$ in the context of graph-regularized multi-task learning.

1.3.1 Bilevel optimization framework

We learn the matrix task similarity $\mathbf{P}$ as well as the regularization parameters $\{\lambda_t\}_{t=1}^T$ by considering them as hyper-parameters and by minimizing an estimate of a generalization error denoted as $E(\cdot)$. This estimate E is naturally a function of all decision function parameters $\boldsymbol{\beta}$. For addressing this problem, we consider a bilevel optimization problem similar to the one of Bennett et al. [3]: the outer level of the problem consists in minimizing E with respect to $\mathbf{P}$ and all $\{\lambda_t\}_{t=1}^T$ and the inner level aims at learning all decision function parameters $\boldsymbol{\beta}$.

While several choices of E can be considered, for the sake of clarity, we have set $E(\cdot)$ to be a validation error of the form :

$$E(\boldsymbol{\beta}^\star) = \sum_{t=1}^{T} \sum_i L_v(\tilde{y}_{i,t}, \tilde{\mathbf{x}}_{i,t}^\top \mathbf{w}_t^\star + b_t^\star) \tag{1.11}$$

where $\boldsymbol{\beta}^\star$ is a vector including the optimal decision function parameters $\mathbf{w}_t^\star$ and $b_t^\star$ for all tasks $t = 1, \ldots, T$. The sets $\{\tilde{\mathbf{x}}_{i,t}, \tilde{y}_{i,t}\}_{t=1}^T$ refer to some validation examples and $L_v(\cdot,\cdot)$ is a twice differentiable loss function that measures the discrepancy between the real and predicted output associated to an input example. Two kinds of loss have been considered in this work, one more adapted to regression tasks $L_v(y,\hat{y}) = (y-\hat{y})^2$ and another one, more suited to classification tasks which is a non-convex sigmoid function that smoothly approximates the $0-1$ loss function $L_v(y,\hat{y}) = \frac{1}{1+e^{\kappa y \hat{y}}}$ with $\kappa > 0$. Note that, at the expense of introducing some cumbersome notations, it is straightforward to modify equation (1.11), so that generalization error estimate is a leave-one-out error or a k-fold cross-validation error.

Now that $E(\cdot)$ has been formally defined, we are in position to state the bilevel optimization we are interested in. Indeed, since the vector $\boldsymbol{\beta}$ depends on the matrix $\mathbf{P}$ and the hyper-parameters λ_t, the bilevel optimization problem can be expressed as:

$$\min_{\boldsymbol{\theta}} \quad E(\boldsymbol{\beta}^*(\boldsymbol{\theta})) + \Omega_\theta(\boldsymbol{\theta}) \tag{1.12}$$

$$\text{with } \boldsymbol{\beta}^*(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\beta}} \quad J(\boldsymbol{\theta}, \boldsymbol{\beta}) \tag{1.13}$$

with $\boldsymbol{\theta} = \left[\lambda_1, \dots, \lambda_T, \{\rho_{i,j}\}_{i=1,j>i}^T\right]^\top$, a vector of size $D = T + \frac{T(T-1)}{2}$ (considering symmetry of $\mathbf{P}$ and $\mathbf{P}(t,t) = 0, \forall t$), $J(\cdot)$ the objective function defined in Equation (1.6) and Ω_θ being a regularizer over the parameters $\boldsymbol{\theta}$. Typically, Ω_θ is related to the projection onto some convex and closed subset Θ of $\mathbb{R}^D$ that defines some constraints over the vector $\boldsymbol{\theta}$. The bilevel problem (1.12) has a particular structure in that the inner problem (1.13), which is actually problem (1.5), is strictly convex for $\lambda_t > 0$, condition that is guaranteed by some specific choice of Θ. This strict convexity is of primary importance since it allows us to compute the unique $\boldsymbol{\beta}^*$ for a given $\boldsymbol{\theta}$. In general cases, this problem (1.12) is non-convex, but its structure suggests that a non-convex proximal splitting can be of interest. Indeed, since $E(\cdot)$ is supposed to be twice differentiable and Ω_θ a non-smooth function, a non-convex forward-backward splitting algorithm, such as the one proposed by [29], can be successfully lifted to our purpose especially if the proximal operator of Ω_θ can be simply computed. For a proper convex function, this proximal operator is defined as [8]:

$$P_{\Omega_{\boldsymbol{\theta}}}(\hat{\boldsymbol{\theta}}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|_2^2 + \Omega_{\boldsymbol{\theta}}(\boldsymbol{\theta})$$

Hence, if Ω_θ is defined as the indicator over a convex set Θ, the proximal operator boils down to be a projection onto the set Θ.

According to Sra [29], the non-convex proximal algorithm we use for solving (1.12) is based on the following simple iterative scheme:

$$\boldsymbol{\theta}^{k+1} = P_{\Omega_\theta}\left(\boldsymbol{\theta}^k - \eta_k \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\beta}^*(\boldsymbol{\theta}^k))\right) \tag{1.14}$$

where P_{Ω_θ} is the proximal operator of Ω_θ, η_k a step size which should satisfy $\eta_k \leq \frac{1}{L}$, L being the Lipschitz constant of $\nabla_{\boldsymbol{\theta}} E$ and $\boldsymbol{\beta}^*(\boldsymbol{\theta}^k)$ denotes the vector of all optimal decision function parameters given the fixed matrix $\mathbf{P}$ and $\{\lambda_t\}$ as defined by $\boldsymbol{\theta}^k$.

Convergence of this iterative scheme to a stationary point of problem (1.12) can be formally stated according to the following proposition:

Proposition 1 *For compact sets Θ that guarantee $\lambda_t > 0$, $\forall t$ and for any loss functions L_v that are continuous and twice differentiable on Θ, $E(\boldsymbol{\beta}(\boldsymbol{\theta}))$ is continuous and gradient Lipschitz on Θ. Hence, the sequence of $\{\theta^{(k)}\}$ obtained using iteration given by Equation (1.14) converges towards a stationary point of problem (1.12).*

Proof 1 *(Sketch) The proof proceeds by showing that $E(\cdot)$ is itself continuous and gradient Lipschitz and then by directly applying Theorem 2 in Sra [29]. For showing smoothness of $E(\cdot)$ with respect to $\boldsymbol{\theta}$, we compute $\frac{\partial^2 E}{\partial\theta_k \partial\theta_s}$ and show that each of these components of the Hessian is continuous with respect to $\boldsymbol{\theta}$. Once continuity has been shown, we use arguments on compactness of Θ to prove that absolute value of all these components are bounded. This implies that the Frobenius norm of the Hessian is bounded and so is the largest eigenvalue of the Hessian. Hence, we can state that $E(\cdot)$ is indeed gradient Lipschitz and this concludes the proof. Details about continuity and differentiability of $E(\cdot)$ as well as Hessian entries computation are given in the appendix.*

1.3.2 Gradient computation

Like all gradient proximal splitting algorithms, our approach needs the gradient of the objective function $E(\cdot)$. Next paragraphs explain how it can be efficiently computed.

At first, we apply the chain rule of differentiation [7] in order to obtain the gradient of $E(\cdot)$ with respect to $\boldsymbol{\theta}$. This leads to the general expression of the partial derivatives of $E(\cdot)$:

$$\frac{\partial E(\boldsymbol{\beta}(\boldsymbol{\theta}))}{\partial\theta_k} = \sum_{s=1}^{(d+1)\times T} \frac{\partial E(\boldsymbol{\beta}(\boldsymbol{\theta}))}{\partial\beta_s(\boldsymbol{\theta})}\frac{\partial\beta_s(\boldsymbol{\theta})}{\partial\theta_k} = \nabla_{\boldsymbol{\beta}} E(\boldsymbol{\beta})^\top \dot{\boldsymbol{\beta}}_k \tag{1.15}$$

with $\dot{\boldsymbol{\beta}}_k$ a vector containing the partial derivatives $\{\frac{\partial\beta_s}{\partial\theta_k}\}$. These latter partial derivatives can be obtained through the implicit function defined by Equation(1.9) relating the optimal values of $\boldsymbol{\beta}$ to the parameters $\boldsymbol{\theta}$. Differentiating (1.9) with respect to θ_k leads to

$$\mathbf{A}\dot{\boldsymbol{\beta}}_k + \dot{\mathbf{A}}_k\boldsymbol{\beta} = \dot{\mathbf{c}}_k$$

with $\dot{\mathbf{A}}_k$ and $\dot{\mathbf{c}}_k$ being respectively the matrix and vector of component-wise derivative of $\mathbf{A}$ and $\mathbf{c}$ with respect to θ_k (detailed expression of the matrix $\dot{\mathbf{A}}_k$ is given in the appendix). By rearranging the equation and taking into account the fact that $\dot{\mathbf{c}}_k = 0$, we have

$$\dot{\boldsymbol{\beta}}_k = -\mathbf{A}^{-1}(\dot{\mathbf{A}}_k\boldsymbol{\beta}) \tag{1.16}$$

Note that for small-size problems, computing this gradient can be relatively cheap since the inverse matrix $\mathbf{A}^{-1}$ may be obtained as a by-product of the resolution of problem (1.9). However, if $\mathbf{A}^{-1}$ has not been pre-computed, obtaining the complete gradient of $E(\cdot)$ requires to solve $D = \frac{T^2+T}{2}$ linear systems of size $(d+1)\cdot T$ and this can rapidly become intractable. In order to render the problem tractable, a simple trick proposed in Keerthi et al. [18] can also be used here. Indeed, by plugging back Equation (1.16) in (1.15),

$\frac{\partial E}{\partial \theta_k}$ can be reformulated as:

$$\begin{aligned}\frac{\partial E(\boldsymbol{\beta}(\boldsymbol{\theta}))}{\partial \theta_k} &= \nabla_{\boldsymbol{\beta}} E(\boldsymbol{\beta})^\top \mathbf{A}^{-1}(-\dot{\mathbf{A}}_k \boldsymbol{\beta}) \\ &= \mathbf{d}^\top(-\dot{\mathbf{A}}_k \boldsymbol{\beta})\end{aligned} \tag{1.17}$$

with $\mathbf{d}$ being the solution of the linear system:

$$\mathbf{A}^\top \mathbf{d} = \nabla_{\boldsymbol{\beta}} E(\boldsymbol{\beta}) \tag{1.18}$$

that does not depend on the variable θ_k used for differentiation. Hence, according to this formulation of the partial derivative $\frac{\partial E(\boldsymbol{\beta}(\boldsymbol{\theta}))}{\partial \theta_k}$, only a single linear system has to be solved for computing the full gradient of $E(\cdot)$ with respect to $\boldsymbol{\theta}$.

1.3.3 Constraints on P and λ_t

Let us now discuss the choice of the regularizer Ω_θ. Typically, Ω_θ is defined as the indicator function over a set Θ, formally

$$\Omega_\theta(\theta) = \mathbf{I}_\Theta(\boldsymbol{\theta}) = \begin{cases} 0 & \text{if } \theta \in \Theta \\ \infty & \text{otherwise} \end{cases}$$

where the set Θ defines some constraints we want to impose on the matrix similarity $\mathbf{P}$ and the hyper-parameters $\{\lambda_t\}_{t=1}^T$.

This set Θ can be defined as the intersection of several constraints and it typically translates some prior knowledge we have over the task relatedness. If no knowledge on task similarities are given, the simplest set one may choose is

$$\Theta = \begin{cases} \rho_{t,s} & : 0 \le \rho_{t,s} \le M, \forall t, s \\ \lambda_t & : m_\lambda \le \lambda_t \le M, \forall t \end{cases} \tag{1.19}$$

with $0 < m_\lambda$ and M being some lower and upper bounds. The small quantity m_λ ensures that a minimal smoothness constraint is enforced on all task parameters $\{\mathbf{w}_t\}_{t=1}^T$. The set defined by (1.19) imposes the λ_t to be strictly positive as required for convergence of the algorithm and it lets the algorithm fix all task similarity parameters. While rather simple, this choice already proves to provide good multi-task performance as well as excellent interpretability of the task similarity since it induces sparsity of the matrix $\mathbf{P}$ because negative correlations of pairwise tasks are ignored by our regularizer defined in Equation (1.3). Ignoring these negative correlations can surely induce a lack of information transfer between tasks and thus may induce slight loss of generalization performance. However, this is inherently due to the graph-based regularization framework and cannot be alleviated by our learning algorithm.

If some tasks are known to be respectively unrelated, strongly related and with unknown relatedness, the following set Θ can be considered instead

$$\Theta = \begin{cases} \rho_{t,s} & : \rho_{t,s} = 0, \text{ for non-similar tasks} \\ \rho_{t,s} & : m_\rho \le \rho_{t,s} \le M, \text{for must-be-similar tasks} \\ \rho_{t,s} & : 0 \le \rho_{t,s} \le M, \text{for all other pairwise tasks} \\ \lambda_t & : m_\lambda \le \lambda \le M, \forall t \end{cases} \tag{1.20}$$

Note that in this set, we have lower-bounded some task similarities with $m_\rho \gg 0$ for tasks that are known to be related since this will indeed force the parameters of these related tasks to be close.

In order to enhance interpretability, sparsity of matrix $\mathbf{P}$ can be further increased by considering in the regularization term an ℓ_1 regularizer in addition to the projection on the set Θ. In such a case, we may have

$$\Omega_{\Theta-\ell_1}(\boldsymbol{\theta}) = \lambda_\theta \sum_{k=1}^{\frac{T(T-1)}{2}} |\theta_k| + \mathbf{I}_\Theta(\boldsymbol{\theta}) \tag{1.21}$$

where $\mathbf{I}_\Theta(\boldsymbol{\theta})$ stands for the indicator function of the set Θ. From simple algebras, one can show that for the above-given convex sets Θ, the proximal operator of $\Omega_{\Theta-\ell_1}$ consists in a component-wise application of a soft-thresholding operator $S(\theta) = \text{sign}(\theta)(|\theta| - \lambda_\theta)_+$ followed by a projection on the set Θ with the function $(z)_+ = \max(0, z)$.

According to the iterative scheme, it is easy to consider other kinds of constraints on the matrix task similarities as long as their proximal operators are simple to obtain. For instance, we could have dropped the positivity constraints on $\rho_{t,s}$ and instead impose positive definiteness constraint on the task covariance matrix Γ defined in Equation 1.4. In this context, the proximal operator on the set of positive definite matrices would have been in play. We could have also combined sparsity constraints on components of $\mathbf{P}$ in addition to the positive definiteness of Γ, resulting in a more involved but computable proximal operator. However, these constraints would considerably increase the computational burden of the overall optimization scheme as the related proximals involve spectral decomposition of Γ and we have not considered them in this work.

1.3.4 Computational complexity

The global algorithm for learning task similarities and regularization parameters is presented in Algorithm 1. It is difficult to evaluate the number of iterations needed before convergence. We can note however that for each iteration of the algorithm, the main computational bulk resides in solving the MTL problem for fixed task-similarity matrix $\mathbf{P}$. In our case, this consists in solving (1.9) and the linear system needed for obtaining $\mathbf{d}$. A plain implementation of these two linear systems would lead to a global complexity of the order of $\mathcal{O}\left((d+1)^3T^3\right)$. Nonetheless, we believe that the specific structure of $\mathbf{A}$ can be exploited for achieving better complexity or specific efficient algorithms for graph-based regularized multi-task learning can be developed. Such an algorithm already exists for Hinge loss function and it can be adapted to the square loss. From another perspective block iterative methods as Gauss-Seidel methods can be implemented for solving the linear system (1.9). But this implementation study is left to future works.

Algorithm 1 Non-convex Proximal Splitting for Learning Task Similarities

```
1: $k \leftarrow 0$
2: initialize $\boldsymbol{\theta}^0 \in \Theta$
3: choose step size $\eta \leq \frac{1}{L}$ (or do backtracking)
4: repeat
5:     % steps for computing $\nabla_{\boldsymbol{\theta}} E$
6:     compute $\boldsymbol{\beta}(\boldsymbol{\theta}^k)$ by solving (1.9)
7:     compute $\nabla_{\boldsymbol{\beta}} E$
8:     $\mathbf{d} \leftarrow$ solution of $\mathbf{A}^\top \mathbf{d} = \nabla_{\boldsymbol{\beta}} E$
9:     for all $k$ do
10:        compute $\dot{\mathbf{A}}_k$
11:        $\frac{\partial E}{\partial \theta_k} \leftarrow -\mathbf{d}^\top \dot{\mathbf{A}}_k \boldsymbol{\beta}$
12:    end for
13:    % proximal step
14:    $\boldsymbol{\theta}^{k+1} \leftarrow P_\Theta(\boldsymbol{\theta}^k - \eta \nabla_{\boldsymbol{\theta}} E)$
15: until convergence criterion is met
```

1.4 Numerical experiments

The approach we propose for learning the task similarity matrix $\mathbf{P}$ as well as the model hyper-parameters λ_t has been tested on several numerical problems including toy examples and three real-world problems.

Besides reporting regression and classification performance, we also provide results on the interpretability of the task relations learned by our algorithm. Indeed, the similarity matrix $\mathbf{P}$ can be understood as an adjacency matrix of the task relation graph. Hence, it can be nicely plotted and its sparsity pattern analyzed.

Note that for all experiments we have considered the loss function of the inner level is the quadratic loss function, thus $J(\cdot)$ is exactly the one given in Equation (1.6). While one may argue that such a loss function is inadequate for classification problems, Rifkin et al [25] and Suykens and co-authors[30, 13, 12] however stated that it is still competitive in many of these problems.

1.4.1 Toy problems

The toy problems we consider here aim at only proving that our algorithm can learn the intrinsic structure of the tasks and how they are related. We show that even for the simple constraints Θ (1.19) we have imposed on $\boldsymbol{\theta}$, our approach is able to learn different task-relation structures such as clusters of tasks or tasks living in a non-linear manifold. The problem is built as follows: given a vector $\bar{\mathbf{w}}^\top = [1, 2]$, a rotation of angle γ_t is applied to $\bar{\mathbf{w}}$ so as to obtain the actual linear model parameters $\bar{\mathbf{w}}_t$ for task t. Examples $\{\mathbf{x}_{i,t}\}$ are

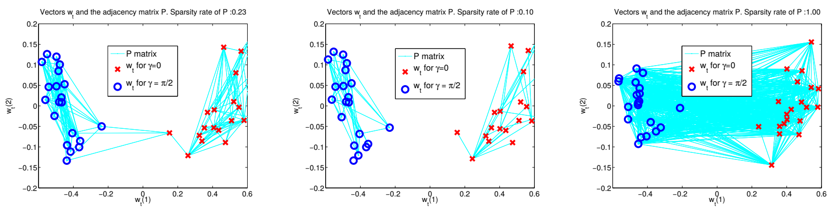

FIGURE 1.1: Learned weight vectors $\{\mathbf{w}_t\}$ and adjacency graph $\mathbf{P}$ for the *clustered tasks* toy problem. The graphs plot these 2D vectors learned for each tasks and the adjacency graph inferred by our approach is materialized as lines between the vectors. The task parameters are theoretically split in two clusters with either $\gamma_t = 0$ or $\gamma_t = \pi/2$. Results obtained by our algorithm using (left) Ω_Θ (middle) $\Omega_{\Theta-\ell_1}$ and (right) Metric-MTL (method due to Zhang et al. [34]). For the latter method, the links depict non-zero entries in the inverse covariance task matrix. Sparsity rate refers to the proportion of non-zeros entries of the learned similarity matrix.

drawn from a 2-dimensional zero-mean and unit variance normal distribution and the corresponding $y_{i,t}$ are obtained according to the equation

$$y_{i,t} = \mathbf{x}_{i,t}^\top \bar{\mathbf{w}}_t + \epsilon_{i,t}$$

where $\epsilon_{i,t} \sim \mathcal{N}(0, 0.5)$ is some additive noise added to the output. .

Two specific synthetic problems illustrate our points. In the first one, tasks are structured in two clusters by randomly applying a rotation of $\gamma_t = 0$ or $\gamma_t = \pi/2$. Hence, our algorithm should be able to recover this clustered structure. For the other example, we apply a rotation whose angle γ_t is uniformly drawn from $[0, \pi]$. Hence, tasks are supposed to be similar only to few neighbours and the adjacency matrix should reflect these local similarities of tasks. For each of these problems, 40 tasks were built and 20 examples for the learning set and 20 examples for the validation set are randomly drawn. We have reported example of the qualitative results obtained using Ω_Θ as the indicator of the set given in Equation (1.19) as well as $\Omega_{\Theta-\ell_1}$ with $m_\lambda = 0.1$, $M = 1000$ and $\lambda_\theta = 0.05$. We also report the obtained results while applying the approach of Zhang et al. [34], named hereafter Metric-MTL and based on the estimation of a dense task covariance matrix.

Figure 1.1 provides an example of results that can be obtained by our approach as well as the competitor's one on the *clustered tasks* toy problem. We note that our algorithm using Ω_Θ is able to nicely infer the tasks relation

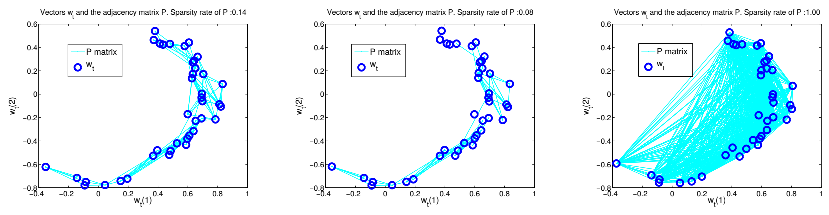

FIGURE 1.2: Learned weight vectors $\{\mathbf{w}_t\}$ and adjacency graph $\mathbf{P}$ for the *manifold-based tasks* toy problem. The graphs plot these 2D vectors learned for each tasks and the adjacency graph inferred by our approach is materialized as lines between the vectors. The task parameters theoretically lie on a manifold. Results obtained by our algorithm using (left) Ω_Θ (middle) $\Omega_{\Theta-\ell_1}$ and (right) Metric-MTL. For the latter method, the links depict non-zero entries in the inverse covariance task matrix. The sparsity rate is the proportion of non-zeros entries of the similarity matrix.

since only 23% of the adjacency matrix $\mathbf{P}$ entries are non-zero and among those coefficients, 98% corresponds to links between tasks from the same cluster. When a sparsity-inducing regularizer is further added to the constraints, the adjacency matrix is more sparse and links between tasks from different clusters disappear.

For the *manifold-based tasks* problem, results are depicted in Figure 1.2. Again, we can clearly see that our approach using both types of constraints is able to learn the underlying task structure. Indeed, we can note that, when using only Ω_Θ, ratio of non-zero coefficients in the adjacency matrix is about 14% which shows that pairwise relationships were found. In addition, as desired, links between tasks mainly exist between neighbour tasks, providing thus evidence that the manifold structure of the task has been recovered. Using $\Omega_{\Theta-\ell_1}$ as a regularizer gives a similar result although with a more aggressive sparsity pattern.

These two examples illustrate that our approach is able to learn complex relationship between tasks such as non-linear manifold and that the proposed constraints induce sparsity in the similarity matrix and thus enhance interpretability on the task relations. In comparison, looking at the rightmost plots of Figures 1.1 and 1.2, we can see that Metric-MTL is also able to recover the complex geometrical relationship between tasks but these relationships are completely hidden by the dense structure of the learned similarity matrix. Note that for this problem involving 40 tasks and a total number of 1600 training/validation examples, the optimization is of the order of 10 seconds on a recent Intel processor with non-optimized Matlab source code. Our approach can then estimate an optimal $\mathbf{P}$ matrix in a reasonable amount of time. This could not have been performed using a classical cross-validation procedure on the corresponding 580 $\rho_{t,s}$ parameters.

1.4.2 Real-world datasets

The approach we proposed has also been experimented on several real-world datasets. The results we have achieved are presented hereafter detailing the experimental set-up.

1.4.2.1 Experimental set-up

Several multi-task learning algorithms have been compared in terms of performance as well as in term of interpretability of the learned task relationships if the latter is applicable.

The baseline approach, denoted as “Ridge Indep.” is an ensemble of ridge regression problems trained independently on each task. MTL approaches that learn task relations have also been considered. This includes the Metric-MTL of Zhang et al. [34] and clustered multi-task learning (Cluster-MTL in the remainder) an approach proposed by Jacob et al. [15] where task similarities are also learned through the inference of the underlying metric between tasks.

For a fair comparison between our approach, named “CoGraph-MTL” (for Constrained Graph-regularized MTL) and the other methods, we selected competitor hyper-parameters by maximizing their performance on the validation set. Note that our bilevel approach also uses the validation set for selecting hyper-parameters but they are optimized through our non-convex proximal method in the outer level.

Depending on the datasets, a squared function $E_{\ell_2}(\cdot,\cdot)$ or a sigmoid function $E_{sig}(\cdot,\cdot)$ with $\kappa = 1$ is used as the outer loss function $L_v(\cdot,\cdot)$. In addition, for all problems, unless specified, we have used Ω_Θ as defined in Equation (1.19) with $m_\lambda = 1$ and $M = 1000$ as well as $\Omega_{\Theta-\ell_1}$ (see Equation 1.21) with $\lambda_\theta = 0.05$ for constraining our graph-regularized MTL method to be sparser.

For each dataset, 10 random splits have been generated and averaged performance measure, Mean Square Error or Area Under the ROC curve (AUC), on the test set was reported. We also performed a signed rank Wilcoxon test to evaluated the statistical difference in performance between the two variants of our method and the best performing competitor.

1.4.2.2 School Dataset

We have also tested our approach on the well known school dataset that is available online and consists in predicting the examination score of students from different schools in London. This problem can be addressed as a multi-

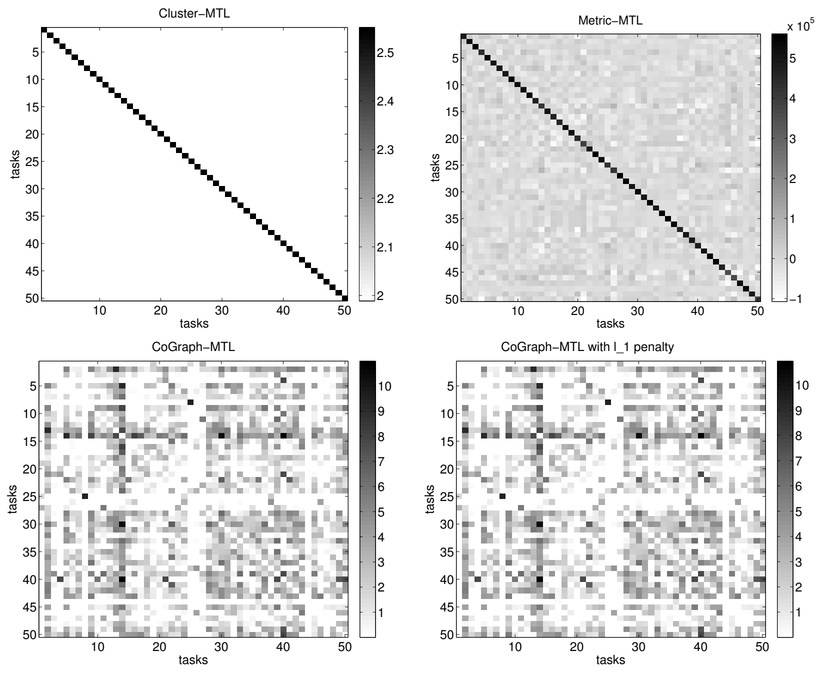

FIGURE 1.3: Example of task similarity matrices on the *School* dataset. (top-left) Clustered MTL. (top-right) Metric-MTL inverse covariance task matrix. (bottom-left) our CoGraph MTL with Ω_Θ (bottom-right) our CoGraph MTL with $\Omega_{\Theta-\ell_1}$.

task problem since differences between schools have to be taken into account for instance by learning a prediction function per school. We refer the reader to Argyriou et al. [1] for a more complete description of the data and focus instead on the feature extraction we used. In their work, they have shown that the tasks might share a common linear subspace. Hence, we took this knowledge into account and performed a PCA on the whole dataset and kept the 10 principal components out of 27 features. We learned the prediction function of the 50 tasks having the largest number of examples. For each task, we have an average number of ≈ 170 samples and we randomly selected 50 of them for the training set, 50 for the validation set and the remaining ones for the test set.

Mean Square Error (MSE) obtained for all the described methods are reported in Table 1.1, as well as the sparsity of resulting task similarity matrices. We can note that the two variants of our approach achieve the lowest prediction error and they are statistically better than the competitor according to a Wilcoxon sign-rank test.

Learned similarity matrices are depicted in Figure 1.3. We can remark that

TABLE 1.1: Mean square error on the *School* dataset. p-value of a Wilcoxon signrank test with respect to the performance of the best competitor as well as the sparsity of the resulting task relation matrix are also reported.

Method	MSE	p-value	Sparsity (%)
Ridge Indep	118.27±2.97	-	-
Cluster MTL	110.78±2.62	-	100.0
Metric MTL	108.27±2.51	-	100.0
CoGraphMTL	107.31±2.24	0.002	56.6
CoGraphMTL $\Omega_{\Theta-l_1}$	107.32±2.24	0.002	55.8

the matrices retrieved by the Cluster-MTL and Metric-MTL are rather dense and present a very similar structure: large diagonal terms and nearly-constant off diagonal components. According to these matrices, we may conclude that examination scores of one given school are related to those of all other schools. The matrices learned by our method have also a similar structure but their entries are more sparse. Hence, we can understand that a given school is related only to few other ones. A deeper understanding of these similarities may then be carried out if some more information, like geographical or social one, was available about all the schools.

1.4.2.3 BCI Dataset

In this BCI problem, our objective is to recognize the presence of an Event-Related Potential (ERP) in a recorded signal during the use of a virtual keyboard. The dataset has been recorded by the Neuroimaging Laboratory of Universidad Autnoma Metropolitana (UAM, Mexico) [20] on 30 subjects performing P300 spelling tasks on a 6×6 virtual keyboard. We consider each subject as a task and learn all classifiers simultaneously. For each subject, we approximately have 4000 single trial samples, that have been pre-processed according to the following steps: first a low pass filtering is applied to the 10 channels signal followed by a decimation, we kept a 1s time window (6 temporal samples) following the stimulus as features leading to trials containing 60 features [24]. Finally for each data split, we selected randomly for each task 100 trials for the training set, 100 trials for the validation set and the remaining samples for the test set. Note that for P300 classification, since the datasets are highly imbalanced, we decided to use the Area Under the ROC Curve as a measure of performance. For this classification task, we have used a sigmoid function as the outer loss function.

We note in Table 1.2 that all multi-task learning approaches provided statistically similar performance measures and they all perform far better than a method which learns each task independently to all others. Interestingly, the two variants of our method output similarity matrices that are considerably sparse. Hence, for each subject, instead of considering that all other

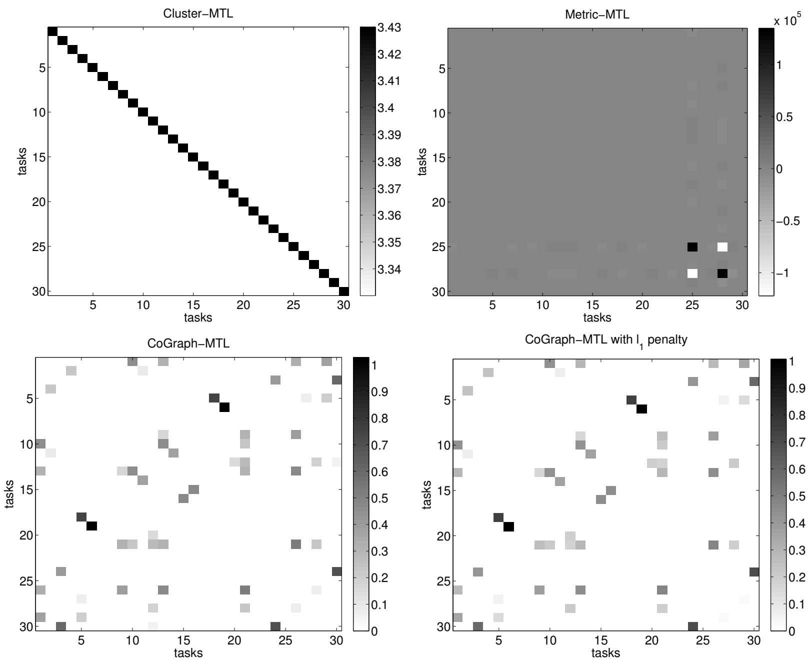

FIGURE 1.4: Example of task similarity matrices on the *BCI* dataset. (top-left) Clustered MTL. (top-right) Metric-MTL inverse covariance task matrix. (bottom-left) our CoGraph MTL with Ω_{Θ} (bottom-right) our CoGraph MTL with $\Omega_{\Theta-\ell_1}$.

subjects were similar to that one, they were able to retrieve some few others that provide sufficient information for transfer learning. Figure 1.4 provides some examples of learned task relation matrix for the different algorithms. We can see there how our method is able to extract relevant and interpretable information from the data. Indeed, the task relation matrices retrieved by our two variants are very sparse. It is thus possible to find which BCI users are related.

1.4.2.4 OCR Dataset

Finally we evaluate our approach on an OCR classification problem. We used the same OCR dataset as in the works of [22]. Here, the aim is to learn a binary classifier for each writer so as to take into account writer variability. The main difficulty in this dataset is that we have only a few examples but still want to learn robust classifiers

We focus here on two binary classification problems : “e” vs “c” and “a” vs “g” for twenty different writers. We want to learn simultaneously these

TABLE 1.2: Area under the ROC Curve (AUC) on the *BCI* dataset. p-value of a Wilcoxon signrank test with respect to the performance of the best competitor as well as the sparsity of the resulting task relation matrix are also reported.

Method	AUC	p-value	Sparsity (%)
Ridge Indep	0.65±0.01	-	-
Cluster MTL	0.78±0.00	-	100.0
Metric MTL	0.78±0.01	-	100.0
CoGraphMTL	0.78±0.00	0.625	6.9
CoGraphMTL $\Omega_{\Theta-l_1}$	0.77±0.00	0.232	6.4

TABLE 1.3: Area under the ROC Curve (AUC) on the *OCR* dataset. p-value of a Wilcoxon signrank test with respect to the performance of the best competitor as well as the sparsity of the resulting task relation matrix are also reported.

Method	AUC	p-value	Sparsity
Ridge Indep	0.94±0.01	-	-
Cluster MTL	0.96±0.01	-	100.0
Metric MTL	0.98±0.01	-	100.0
CoGraphMTL	0.96±0.01	0.002	11.1
CoGraphMTL $\Omega_{\Theta-l_1}$	0.95±0.01	0.002	9.8
CoGraphMTL Ω_{Θ_G}	0.97±0.01	0.375	51.7

40 classification tasks. The data is a raw bitmap of size 8×16 that was preprocessed as follows. We performed PCA on the raw data and kept 20 principal components. We randomly select 8 samples for the training set, 4 samples for the validation set and the remaining for the test set. Moreover, in order to prove how easy it is to add prior information in our approach, we integrated group knowledge in the learning problem by imposing specific constraints on $\boldsymbol{\theta}$, denoted as Ω_{Θ_G}. Indeed, we force a link between tasks from the same binary classification problem by imposing $0.01 \leq \rho_{t,k} \leq 1000$ (which is rather a weak constraint).

Performance of the different methods can be seen in Table 1.3. We remark that our method using Ω_{Θ_G} achieves equivalent performance than Metric-MTL whereas the two other variants of our method are slightly but significantly worse than the best performing competitor. Indeed, adding prior knowledge helps in learning robust classifiers.

We should however emphasize that the Metric-MTL approach is not able to retrieve the correct relation between tasks as shown in Figure 1.5. Indeed, the learned similarity matrix tells us that for Metric-MTL all the tasks are related. Cluster-MTL is able to infer that they form two specific clusters of tasks in the problem. Our methods are also capable of detecting these clusters and in addition yield sparse similarity matrices as very few relations between tasks from two different clusters were uncovered. Remark that even though some *must-link* constraints have been imposed when using Ω_{Θ_G} as a regularizer, some hyper-parameters have been still optimized by our algorithm.

1.5 Conclusion

We have proposed a novel framework for learning task similarities in multi-task learning problems. Unlike other previous works on this topic, we learn these similarities so as to optimize a proxy on the generalization errors of all tasks. For this purpose, we introduce a bilevel optimization problem which involves both the minimization of the generalization error and the optimization of the task parameters. The global optimization is solved by means of non-convex proximal splitting algorithm which is simple to implement and can easily be extended to situations where different constraints on task similarities have to be imposed.

Experimental results on toy problems clearly show that the method we propose can help in learning complex structures of task similarities. On real-world problems, our method clearly achieves performances similar to other multi-task learning algorithms which learn tasks similarities while providing task similarity matrices that are far more interpretable.

Acknowledgments

This work has been partly supported by the French ANR Agency Grant ANR-11-JS02-0010 Lemon.

Appendix

Continuity and differentiability of E with respect to θ_k

First, note that $\boldsymbol{\beta}^*$ is implicitly defined by the linear system $\mathbf{A}\boldsymbol{\beta} = \mathbf{c}$. Hence, we have

$$\boldsymbol{\beta} = \mathbf{A}^{-1}\mathbf{c}$$

Existence and unicity of $\boldsymbol{\beta}$ is guaranteed by invertibility of $\mathbf{A}$ since we have imposed that $\lambda_t > 0, \forall t$. Hence, since $\mathbf{A}$ and $\mathbf{c}$ are both continuous and differentiable with respect to θ_k, so is $\mathbf{A}^{-1}$ and each component of $\boldsymbol{\beta}$. Using similar arguments, we can show that each component of $\dot{\boldsymbol{\beta}}_k = -\mathbf{A}^{-1}(\dot{\mathbf{A}}_k \boldsymbol{\beta})$ is continuous.

Continuity and differentiability of $E(\cdot)$ is easily obtained since E is built from differentiability-preserving operations over differentiable functions.

Hessian computation In order to prove the gradient Lipschitz property of E, we show that the Hessian of the function $E(\boldsymbol{\beta}(\theta))$ is bounded. The Hessian is a matrix of general term:

$$\begin{aligned}
\frac{\partial E(\boldsymbol{\beta}(\boldsymbol{\theta}))}{\partial \theta_k \partial \theta_s} &= \frac{\partial}{\partial \theta_s}\Big(\mathbf{d}^\top(\dot{\mathbf{c}}_k - \dot{\mathbf{A}}_k \boldsymbol{\beta})\Big) && (1.22)\\
&= -\left[\frac{\partial \mathbf{d}^\top}{\partial \theta_s}(\dot{\mathbf{A}}_k \boldsymbol{\beta}) + \mathbf{d}^\top \dot{\mathbf{A}}_k \frac{\partial \boldsymbol{\beta}}{\partial \theta_s}\right] && (1.23)
\end{aligned}$$

where we used the fact that $\mathbf{c}$ does not depend on $\boldsymbol{\theta}$ so $\dot{\mathbf{c}}_k = \mathbf{0}$ and $\dot{\mathbf{A}}_k$ is a constant matrix whose component are equal to 0 when differentiated. By using the definition of $\mathbf{d}$ in Equation 1.18, it is easy to see that

$$\frac{\partial \mathbf{d}}{\partial \theta_s} = (\mathbf{A}^\top)^{-1}\left[\frac{\partial}{\partial \theta_s}\nabla_{\boldsymbol{\beta}} E(\boldsymbol{\beta}) - \left(\frac{\partial \mathbf{A}}{\partial \theta_s}\right)^\top \mathbf{d}\right] \qquad (1.24)$$

Now, we can note that all components of the Hessian general term (1.23) are continuous with respect to $\boldsymbol{\theta}$. For instance, $\mathbf{d}$ is continuous since it is the product of a continuous matrix $\mathbf{A}^{-1}$ and continuous vector $\nabla_{\boldsymbol{\beta}} E$ (by hypothesis on the loss function). Similar arguments can be employed for showing continuity of all other terms.

Expression of $\dot{\mathbf{A}}_k$

The expression of this gradient takes different forms according to the type of hyper-parameter. Using the definition (1.10), we get the expression

$$\dot{\mathbf{A}}_k = \begin{bmatrix}
\mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0}\\
\vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots\\
\mathbf{0} & \dots & 4\tilde{\mathbf{I}} & & -4\tilde{\mathbf{I}} & & \mathbf{0}\\
\vdots & \dots & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0}\\
\mathbf{0} & \dots & -4\tilde{\mathbf{I}} & & 4\tilde{\mathbf{I}} & \dots & \mathbf{0}\\
\vdots & \dots & \mathbf{0} & \dots & \mathbf{0} & \dots & \vdots\\
\mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0}
\end{bmatrix}, \quad \text{for} \quad \theta_k = \rho_{i,j}, \text{with } j > i$$

and

$$\dot{\mathbf{A}}_k = \begin{bmatrix} \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \dots & \vdots & \dots & \vdots \\ \mathbf{0} & \dots & 2\tilde{\mathbf{I}} & \mathbf{0} & \vdots \\ \vdots & \dots & \mathbf{0} & \dots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}, \quad \text{for} \quad \theta_k = \lambda_t, \text{with } t = 1, \dots, T$$

bibliography../../biblio/biblioAR,../../biblio/biblioRF,../biblio/biblioGG

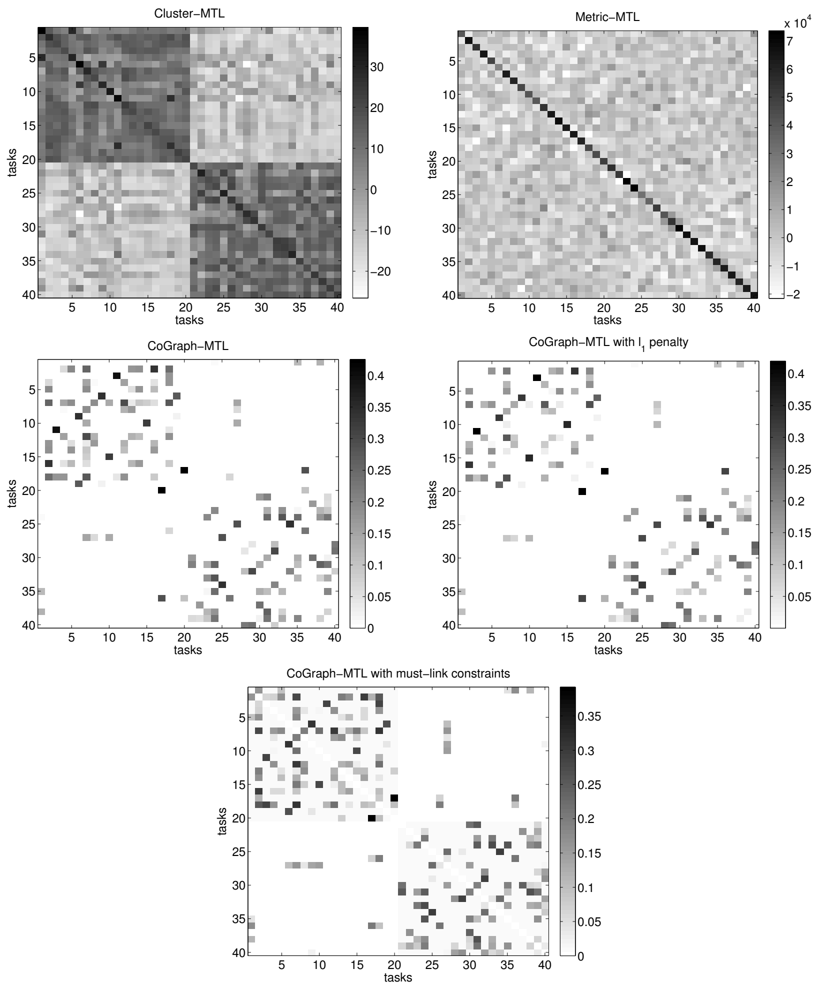

FIGURE 1.5: Examples of learned task similarity matrices on the *OCR* dataset. (top-left) Clustered MTL. (top-right) Metric-MTL inverse covariance task matrix. (middle-left) our CoGraph MTL with Ω_Θ (middle-right) our CoGraph MTL with $\Omega_{\Theta-\ell_1}$. (bottom) our CoGraph MTL with Ω_{Θ_G}

Bibliography

[1] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[2] Y. Bengio. Gradient-based optimization of hyperparameters. *Neural Computation*, 12:1889–1900, 2000.

[3] K.P. Bennett, J. Hu, X. Ji, G. Kunapuli, and J.S. Pang. Model selection via bilevel optimization. In *Neural Networks, International Joint Conference on*, pages 1922–1929, 2006.

[4] J. Bergstra and Y. Bengio. Random search for hyper-parameters optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[5] R. Caruana. Multi-task learning. *Machine Learning*, 28:41–75, 1997.

[6] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukerjhee. Choosing multiple parameters for SVM. *Machine Learning*, 46(1-3):131–159, 2002.

[7] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

[8] P.L. Combettes and J.C. Pesquet. Proximal splitting methods in signal processing. *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212, 2011.

[9] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Proceedings of the tenth Conference on Knowledge Discovery and Data mining*, 2004.

[10] Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

[11] S. Feldman, B. A. Frigyk, M. R. Gupta L. Cazzanti, and P. Sadowski. Multi-task output space regularization. *arXiv*, 2011.

[12] T. Van Gestel, J. Suykens, B. Baesens, S. Viaene, J. Vanthienen, and G. Dedene. Benchmarking least squares support vector machine classifiers. *Machine Learning*, 54(1):5–32, 2004.

[13] Tony Van Gestel, Johan A. K. Suykens, Gert R. G. Lanckriet, Annemie Lambrechts, Bart De Moor, and Joos Vandewalle. Bayesian framework for least-squares support vector machine classifiers, gaussian processes, and kernel fisher discriminant analysis. *Neural Computation*, 14(5):1115–1147, 2002.

[14] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[15] L. Jacob, F. Bach, and J.-P. Vert. Clustered multi-task learning: A convex formulation. In *Advances in Neural Information Processing Systems, NIPS*, 2008.

[16] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *Proceedings of the 28th ICML*, pages 521–528. ACM, June 2011.

[17] T. Kato, H. Kashima, M. Sugiyama, and K. Asai. Multi-task learning via conic programming. In *Advances in Neural Information Processing Systems*, 2008.

[18] S. Sathiya Keerthi, Vikas Sindhwani, and Olivier Chapelle. An efficient method for gradient-based adaptation of hyperparameters in svm models. In *Advances in Neural Information Processing Systems 19*, pages 673–680. MIT Press, 2007.

[19] A. Kumar and H. Daum III. Learning task grouping and overlap in multi-task learning. In *Proceedings of the International Conference on Machine Learning*, 2012.

[20] Claudia Ledesma-Ramirez, Erik Bojorges Valdez, Oscar Yáñez Suarez, Carolina Saavedra, Laurent Bougrain, and Gerardo Gabriel Gentiletti. An Open-Access P300 Speller Database. In *Fourth International Brain-Computer Interface Meeting*, 2010.

[21] A. Maurer, M. Pontil, and B. Romera-Paredes. Sparse coding for multi-task and transfer learning. In *Proceedings of the International Conference on Machine Learning*, 2013.

[22] Guillaume Obozinski, Ben Taskar, and Michael I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20:231–252, April 2010.

[23] A Rakotomamonjy, R Flamary, G Gasso, and S Canu. lp-lq penalty for sparse linear and sparse multiple kernel multi-task learning,. *IEEE Transactions on Neural Networks*, 22(8):1307–1320, 2011.

[24] A. Rakotomamonjy and V. Guigue. BCI competition III: Dataset II - ensemble of SVMs for BCI P300 speller. *IEEE Trans. Biomedical Engineering*, 55(3):1147–1154, 2008.

[25] R. Rifkin, G. Yeo, and T. Poggio. Regularized least squares classification. In *Advances in Learning Theory : Methods, Model and Applications*, pages 131–153. IOS Press, 2003.

[26] B. Romera-Paredes, A. Argyriou, N. Bianchi-Berthouze, and M. Pontil. Exploiting unrelated tasks in multi-task learning. In *JMLR Proceeding track*, volume 22, pages 951–959, 2012.

[27] B. Romera-Paredes, M. Hane Aung, N. Bianchi-Berthouze, and M. Pontil. Multilinear multitask learning. In *Proceedings of the International Conference on Machine Learning*, 2013.

[28] Kim Seyoung and Eric P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*, pages 543–550, 2010.

[29] S. Sra. nonconvex proximal splitting : batch and incremental algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[30] J. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *least squares support vector machine classifiers.* World Scientific, 2002.

[31] S. V. N. Vishwanathan, A. J. Smola, and M. Murty. SimpleSVM. In *International Conference on Machine Learning*, 2003.

[32] C. Widmer, M. Kloft, N. Goernitz, and G. Raetsch. Efficient training of graph-regularizer multi-task svm. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2012.

[33] Christian Widmer, Nora C Toussain, Yasemin Altun, and Rätsch Gunnar. Inferring latent task structure for multitask learning by multiple kernel learning. *BMC Bioinformatics*, 11(Suppl 8):S5, 2010.

[34] Y. Zhang and D.Y. Yeung. A convex formulation for learning task relationships in multiple task learning. In *Proceedings of Uncertainty and Artificial Intelligence*, 2010.

[35] L. Zhong and J. Kwok. Convex multitask learning with flexible task clusters. In *Proceedings of the 29th International Conference on Machine Learning (ICML), 2012*, 2012.