# Learning the Graph of Relations Among Multiple Tasks

Andreas Argyriou, Stéphan Clémençon, Ruocong Zhang

# Learning the Graph of Relations Among Multiple Tasks

Andreas Argyriou
École Centrale Paris

Stéphan Clémençon
Telecom Paristech

Ruocong Zhang
Telecom Paristech

**Abstract**

We propose *multitask Laplacian learning*, a new method for jointly learning clusters of closely related tasks. Unlike standard multitask methodologies, the graph of relations among the tasks is not assumed to be known a priori, but is learned by the multitask Laplacian algorithm. The algorithm builds on kernel based methods and exploits an optimization approach for learning a continuously parameterized kernel. It involves solving a semidefinite program of a particular type, for which we develop an algorithm based on Douglas-Rachford splitting methods. Multitask Laplacian learning can find application in many cases in which tasks are related with each other to *varying degrees*, some strongly, others weakly. Our experiments highlight such cases in which multitask Laplacian learning outperforms independent learning of tasks and state of the art multitask learning methods. In addition, they demonstrate that our algorithm partitions the tasks into clusters each of which contains well correlated tasks.

## 1 Introduction

In recent years, *multitask learning* has been an active and growing area of interest in machine learning. The goal in such problems is to jointly learn several regression or classification tasks and in this way enhance statistical performance, compared to learning the tasks independently. The advantages of multitask learning are especially pronounced in situations lacking in sufficient samples per task. By "borrowing strength" from other tasks, it may be possible to learn better models for each task, provided that there are sufficiently strong *relations among the tasks*.

There has been a wide variety of multitask learning approaches mainly due to the

large range of possible ways in which tasks may be related. On the other side, a more generic and broadly applicable learning theoretic treatment was developed early on [10, 11, 23]. Consider an input set $\mathcal{X}$ and an output set $\mathcal{Y}$ and for simplicity that $\mathcal{X} \subseteq \mathbb{R}^d$, $\mathcal{Y} \subseteq \mathbb{R}$. Tasks can be viewed as $n$ functions $f_\ell$, $\ell = 1, \ldots, n$, to be learned from given data $\{(x_{\ell i}, y_{\ell i}) : i = 1, \ldots, m_l, \ell = 1, \ldots, n\} \subseteq \mathcal{X} \times \mathcal{Y}$. These tasks may be viewed as drawn from an unknown joint distribution of tasks, which is the source of the bias that relates the tasks.

As with many machine learning problems, regularization based approaches have been applied to multitask learning as well. In particular, each task may be represented as a linear predictive function $x \mapsto w_\ell^\top x$, or equivalently as a vector $w_\ell$ of regression parameters.[1] Thus in a regularization based setting the learning algorithm may be phrased as an optimization problem of the form

$$\min \left\{ \sum_{\ell=1}^{n} \sum_{j=1}^{m} E(w_\ell^\top x_{j\ell}, y_{j\ell}) + \gamma\, \Omega(W) : W \in \mathbf{M}_{d,n} \right\},$$

where $\mathbb{E} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a convex loss function, $\Omega$ is a penalty term, $\mathbf{M}_{d,n}$ denotes the set of real $d \times n$ matrices and $W$ is the matrix with columns $w_\ell$. The error term should favor a good fit of the data whereas the regularizer $\Omega$ should favor certain types of relations among the tasks. The positive regularization parameter $\gamma$ (which can be tuned with techniques like cross validation) determines the trade off between fitting the data well and enforcing the bias of task relatedness. Clearly, different choices of the penalty $\Omega$ may give rise to quite different multitask methods.

One popular approach has been to use $\ell_2$ type penalties on differences or various combinations of tasks, as studied in detail in [15, 17]. In such approaches, it is common to penalize distances between tasks and hence tasks are biased towards being similar to each other. Other approaches proposed in the past have relied on boosting techniques [14], neural networks [13], hierarchical Bayesian methods [7] etc. Another class of methods aims to learn tasks (that is, the columns of $W$) which lie in low-dimensional subspaces or manifolds embedded in $\mathbb{R}^d$ [1, 2, 3, 27]. Many of these methods use regularizers which involve the *trace norm*, which is defined as the sum of the singular values of $W$.

Most of the above methods rely on implicit assumptions about *all* tasks being related in a specific way. However, in many applications it is not known a priori whether all tasks are strongly related with each other. Or, in other applications it is reasonable to expect that tasks relate to each other in *varying degrees* without knowing much about the strengths of these relations. In particular, it is common that tasks *cluster in a few groups*, with weak task relations across groups, but strong task relations within each group. The main complication with such situations is that the appropriate clustering may not be known a priori. In addition, it may be important to account both for the strong intra-cluster relations and the weak inter-cluster ones. In general, using a method such as $k$-means or spectral clustering for preprocessing the tasks is not satisfactory, since usually there is insufficient prior information to obtain good clusters of tasks.

[1] Some, but not all, multitask methods can be kernelized – see [6] for conditions.

*Financial asset prediction* is one example of an application in which the tasks are dependent with an unknown underlying structure. This structure may change over time, so that clustering large historical data would generally be unsuitable for prediction. Learning in a multitask framework would enable more consistent predictions, but also a better risk assessment of the predicted assets. Another example is *recommender systems* and *preference learning*, in which the preferences of multiple consumers are to be learned. It is known from marketing research that consumers cluster into groups of similar behavior (based on demographics, geography etc.) whereas they share weaker preference behavior across groups.

Until now, there has been limited work, in the context of regularization, on learning of tasks that follow clustered distributions. For example, in [17], a convex relaxation of a task clustering problem within the $\ell_2$ regularization framework has been proposed. In [4, 18], a clustering variation on the low rank approach is used to learn multiple subspaces on which the tasks lie. Also in [7], the tasks clustering problem has been addressed in a hierarchical Bayesian framework with mixtures of distributions. In [20], a matrix factorization approach that penalizes matrix factors of $W$ with $\ell_1$ and $\ell_2$ norms has been proposed.

In this paper, we propose a new method for simultaneously learning the tasks and the relations among them. We start from a Laplacian regularization framework by [15], demonstrate its drawbacks and show how it can be suitably modified for our purposes. We then formulate our learning method as an optimization problem in a reproducing kernel Hilbert space, in which the kernel also needs to be learned. We thus derive a problem analogous to that of learning from an infinite set of kernels, with the difference that in our case the feasibility set is not convex. Despite this, we show how an alternating minimization algorithm can be used to compute good estimates of the tasks and of the graph of task relations. In addition, we propose an algorithm based on Douglas-Rachford optimization methods for solving certain semidefinite programs, which occur as subproblems. Finally we report experiments which highlight that a) our method is competitive and often outperforms state of the art multitask methods; b) our method recovers a good clustering of the tasks and their relations.

# 2 Learning the Tasks' Graph Laplacian

In our multitask learning framework, we account for the dependence structure between tasks representing them in a graph. Including the graph Laplacian as the relevant information in the optimization problem has been studied when the graph is known. We propose a joint formulation for learning both the tasks and the graph, as well as an alternating algorithm to solve the optimization problem.

## 2.1 Background

A general framework for multitask learning has been proposed in [15], based on regularization in reproducing kernel Hilbert spaces (RKHS). This framework consists of regularization problems in the joint space of tasks, with a positive definite quadratic penalty $w^\top E w$, where $w$ is the column-wise vectorization of $W$ and $E$ is a $dn \times dn$

positive definite matrix. The intuition for $E$ is that it describes relations between pairs of tasks. The authors of [15] show that such multitask learning formulations can be rephrased as regularization problems in an RKHS of functions on the augmented input-task space $\mathcal{X} \times \mathbb{N}_n$, where $\mathbb{N}_n$ denotes the set $\{1, \ldots, n\}$.

For example, the case $E = \Delta \otimes I_d$, where $\Delta$ is an $n \times n$ positive diagonal matrix and $\otimes$ denotes the Kronecker product, expresses lack of any relations among the tasks and corresponds to learning the tasks *independently*, with the diagonal entries of $D$ as regularization parameters. As another example, the main methodology studied in [15] penalizes the squared $\ell_2$ distances of the tasks $\{w_\ell, \ell \in \mathbb{N}_n\}$ *from their average*,

$$\sum_{\ell=1}^{n} \|w_\ell\|^2 + \rho \sum_{\ell=1}^{n} \|w_\ell - \frac{1}{n} \sum_{q=1}^{n} w_q\|^2 \,. \tag{1}$$

This formulation corresponds to an assumption natural in many applications, namely, that all tasks are close to each other in $\ell_2$ distance.

However, in many cases in which tasks cluster in two or more groups (like the examples of Section 1), regularizers like (1) are not appropriate. The reason is that tasks *across* different groups are *weakly* related, whereas the above type of penalty biases them towards being strongly related. This has motivated the authors of [15] to define the *graph of tasks* as a weight matrix $A$ with nonnegative entries which encodes the relatedness among the $n$ tasks. Consequently, the *tasks Laplacian* $L$ is the graph Laplacian matrix obtained from $A$,

$$L = D - A \,,$$

where $D = \mathrm{Diag}(d)$ is the diagonal matrix formed by the degrees of the vertices $d_i = \sum_{j=1}^{n} A_{ij}$. Thus in [15] a multitask methodology involving the tasks Laplacian is proposed (but not implemented or further studied). This methodology involves penalizing the tasks with the Laplacian quadratic form as follows

$$\min\left\{ \sum_{\ell=1}^{n} \sum_{j=1}^{m_\ell} E(w_\ell^\top x_{j\ell}, y_{j\ell}) + \gamma \sum_{\ell,q=1}^{n} w_\ell^\top w_q L_{\ell q} \,:\; w_\ell \in \mathbb{R}^d \;\forall \ell \in \mathbb{N}_n \right\} . \tag{2}$$

One rationale behind this regularization is that this penalty equals $\sum_{\ell,q=1}^{n} \|w_\ell - w_q\|^2 A_{\ell q}$ and hence it favors those pairs of tasks $(w_\ell, w_q)$ with large weights $A_{\ell q}$ to be similar.

The above penalty also equals $w^\top (L \otimes I_d) w$, where $w = (w_1^\top \ldots w_n^\top)^\top$ is the columnwise vectorization of $W$ and $I_d$ the $d \times d$ identity matrix. This quadratic form is positive semidefinite but not positive definite, so it is not directly equivalent to regularization in an RKHS. To address this issue, [15] suggests restricting $w$ on the range of $L \otimes I_d$, which yields an equivalent formulation in an RKHS with the multitask kernel $K((x, \ell), (t, q)) = L^+_{lq}\, x^\top t$, for every $x, t \in \mathbb{R}^d$, $\ell, q \in \mathbb{N}_n$.

We claim, however, that in multitask practice it will be necessary to optimize $W$ over the entire space. The reason is that the above restriction from [15] discards crucial information about task relations which is contained in the null space of the Laplacian. To see this, consider, as a simple example, a graph of two tasks with a positive weight,

which corresponds to the penalty $\|w_1 - w_2\|^2$. The range constraint from [15] enforces $w_1 + w_2 = 0$, since the constant vector belongs to the null space of $L$ for any Laplacian. At the same time, this constraint contradicts the information, encoded by a large weight between the two tasks, that the distance between $w_1$ and $w_2$ is small. More generally, any graph with $k$ connected components has $k$ zero eigenvalues and hence, there will be $k$ linearly independent constraints on the $w_\ell$. If tasks within each cluster were close to each other (which is our objective) then the constraints would imply that all tasks were close to zero. In practice, imposing the $k$ constraints of [15] restricts the tasks to lie on an $n - k$ dimensional subspace. This subspace depends solely on the graph topology and may be completely inconsistent to the actual tasks that have generated the data. Hence these constraints may prevent the method from learning the correct tasks, as we further demonstrate with realistic examples in Section 3.

The problem of learning the task clusters and the tasks simultaneously has also been the topic of [17]. These authors consider penalties which are combinations of three terms: an $\ell_2$ penalty on the *average* of the tasks, a measure of *within-cluster* variance similar to (1) and a measure of *between-cluster* variance which uses the cluster means. The regularization problem is a function of the matrix of ones and zeros encoding the clusters and is a nonconvex optimization problem. To deal with this in a tractable way, the authors propose a convex relaxation of the penalty, which leads to the alternative regularization

$$\begin{aligned} &\operatorname{tr}(\Pi W \Sigma_c^{-1} W^\top \Pi) \\ &\text{subject to } \Sigma_c \succeq 0, \alpha I_n \preceq \Sigma \preceq \beta I_n, \operatorname{tr} \Sigma = \gamma, \end{aligned} \tag{3}$$

where $\operatorname{tr}$ is the matrix trace, $\Sigma_c$ and $\Sigma$ relate in an affine way, $\Pi$ is a fixed projection matrix and $\alpha, \beta, \gamma$ are positive constants depending on the regularization parameters.

A very different approach [20] has been inspired by the trace norm and its formula in terms of matrix factorization. The idea is that one of the matrix factors of $W$ can be viewed as encoding the task grouping and is penalized with an $\ell_1$ sparsity penalty. At the same time, this method will favor low rank solutions since it penalizes the factors of $W$. Another approach extending trace norm regularization [4, 18] assumes that tasks lie not on one but on multiple low dimensional subspaces and simultaneously learns the tasks and these subspaces.

## 2.2 Learning the Tasks Given the Graph

The question we will address is how to learn simultaneously the $n$ tasks and the graph of tasks via its Laplacian. We will build on the proposal (2) of [15], but will follow a different path towards obtaing an RKHS formulation. We propose a small perturbation of the Laplacian penalty with a fixed constant $\varepsilon$, namely,

$$\min \left\{ \sum_{\ell=1}^{n} \sum_{j=1}^{m_\ell} E(w_\ell^\top x_{j\ell}, y_{j\ell}) + \gamma \left( \sum_{\ell,q=1}^{n} w_\ell^\top w_q L_{\ell q} + \varepsilon \sum_{\ell=1}^{n} \|w_\ell\|^2 \right) : \right.$$
$$\left. w_\ell \in \mathbb{R}^d \ \forall \ell \in \mathbb{N}_n \right\}. \tag{4}$$

This penalty equals the sum of $\sum_{\ell,q=1}^{n} \|w_\ell - w_q\|^2 A_{\ell q}$ plus the perturbation term on the tasks. When $\varepsilon$ is small, the graph term dominates the penalty and hence one should expect strong tasks similarities to conform to large weights and the opposite. Thus, formulation (4) reflects the intuition about the tasks Laplacian and at the same time has a convenient positive definite form. Throughout the paper $\varepsilon$ will be set to a fixed but very small value.

The above penalty can also be written as $w^\top((L+\varepsilon I_n)\otimes I_d)w$ or $\mathrm{tr}((L+\varepsilon I_n)W^\top W)$. Thus, the corresponding multitask kernel equals

$$K((x,\ell),(t,q)) = (L+\varepsilon I_n)^{-1}_{lq}\, x^\top t\ ,$$

for every $x, t \in \mathbb{R}^d$, $\ell, q \in \mathbb{N}_n$. Instead of the linear kernel, any scalar reproducing kernel $G$ may be used for the inputs [12] :

$$K((x,\ell),(t,q)) = (L+\varepsilon I_n)^{-1}_{lq}\, G(x,t)\ , \tag{5}$$

where $x, t$ belong to a generic input space $\mathcal{X}$. Let us call $\mathcal{H}_K$ the RKHS associated with this kernel.

It is also clear from the form $\mathrm{tr}((L+\varepsilon I_n)W^\top W)$ that a block diagonal (after simultaneous permutation of its rows and columns) Laplacian would penalize correlations of tasks within each group while ignoring any correlations across different groups. This intuition extends also to weak inter-cluster blocks of the Laplacian. Therefore, if a known or learned tasks Laplacian is given then any clustering method can be used to yield the clustering of the tasks.

Given training data $x_{j\ell} \in \mathcal{X}$, $y_{j\ell} \in \mathcal{Y}$, for $\ell \in \mathbb{N}_n$, $j \in \mathbb{N}_{m_\ell}$, training can be performed by solving the regularization problem

$$\min_{f\in\mathcal{H}_K} \left\{ \sum_{\ell=1}^{n}\sum_{j=1}^{m_\ell} E(f(x_{j\ell},\ell), y_{j\ell}) + \gamma\, \|f\|_{\mathcal{H}_K} \right\} \tag{6}$$

where $E : \mathbb{R}\times\mathcal{Y} \to \mathbb{R}$ is a prescribed convex loss function and $\gamma > 0$ is a regularization parameter to be tuned, for example, with cross validation.

Problem (6) has a unique solution, which, by the representer theorem [19], can be written in the form

$$f(t,q) = \sum_{\ell=1}^{n}\sum_{j=1}^{m_\ell} c_{j\ell} K((x_{j\ell},\ell),(t,q)) \tag{7}$$

$\forall\ t \in \mathcal{X}, q \in \mathbb{N}_n$ and for some $c \in \mathbb{R}^M$, where $M = \sum_{\ell=1}^{n} m_\ell$. Substituting this formula in (6) yields a convex optimization problem in $M$ variables

$$\min\left\{ E_y(K_{\mathbf{x}}c) + \gamma\, c^\top K_{\mathbf{x}} c\ :\ c \in \mathbb{R}^M \right\} \tag{8}$$

where $K_{\mathbf{x}}$ denotes the kernel matrix of all the input-task training data, and $E_y : \mathbb{R}^M \to \mathbb{R}$ is the convex function (parameterized by the output data) defined as $E_y(z) = \sum_{\ell=1}^{n}\sum_{j=1}^{m_\ell} E(z_{j\ell}, y_{j\ell})$, $\forall\, z \in \mathbb{R}^M$. If the task relatedness graph (and hence $K_{\mathbf{x}}$) are

given, then solving for $c$ in (8) (with a method such as SVM, ridge regression etc. in the dual) and using formula (7) yields the functions for each of the $n$ tasks.

Note that the predictive function for the $q$-th task depends only on the $q$-th column of $(L+\varepsilon I_n)^{-1}$ and not on the other columns. In fact, the entries of this column weight the contribution of the other tasks in the predictive function. Also note that, if the input samples are the same for all tasks (which is the case for many applications), the kernel matrix can be expressed as

$$K_{\mathbf{x}} = (L + \varepsilon I_n)^{-1} \otimes G_{\mathbf{x}},$$

where $G_{\mathbf{x}}$ is the kernel matrix for the input kernel $G$ on the data. In general (when input samples may differ across tasks), every $(\ell, q)$ block of $K_{\mathbf{x}}$ equals the product of $(L + \varepsilon I_n)^{-1}_{lq}$ with the $(\ell, q)$ block of $G_{\mathbf{x}}$. In the following, we introduce the matrix

$$Z = (L + \varepsilon I_n)^{-1}.$$

We also use $T_{\mathbf{x}}$ to denote the linear mapping that maps an $n \times n$ matrix $Z$ to the $M \times M$ matrix with blocks the products of $Z_{\ell q}$ with the $(\ell, q)$ blocks of $G_{\mathbf{x}}$.

## 2.3 Learning the Tasks and the Graph Jointly

Our aim is to learn the $n$ tasks without knowing the task relatedness matrix $L$ exactly. That is, we aim to learn the task functions and the graph Laplacian simultaneously. We argue that this can be done via the methodology of learning infinitely parameterized kernels as in [5]. That is, instead of learning the graph Laplacian directly, we optimize the objective function (8) with the kernel $K$ allowed to belong to a set $\mathcal{K}$,

$$\min \left\{ E_y(K_{\mathbf{x}} c) + \gamma\, c^\top K_{\mathbf{x}} c \;:\; c \in \mathbb{R}^M, K_{\mathbf{x}} \in \mathcal{K} \right\} . \tag{9}$$

The objective in (9) is convex, since the functional $K \mapsto \min\{ E_y(K_{\mathbf{x}} c) + \gamma\, c^\top K_{\mathbf{x}} c : c \in \mathbb{R}^M \}$ is convex [5, Lem. 2]. The approach of jointly learning the function and the kernel has been extensively used and justified with learning-theoretic bounds – see, for example, [21, 25, 26, 28].

To ensure that $K$ is indeed a valid kernel of the form (5) coming from a graph Laplacian, we choose

$$\mathcal{K} = \left\{ T_{\mathbf{x}}(Z) : 0 \prec Z \preceq \tfrac{1}{\varepsilon} I_n, (Z^{-1})_{\text{off}} \leq 0, Z\mathbf{1}_n = \tfrac{1}{\varepsilon}\mathbf{1}_n \right\}$$

where $A_{\text{off}}$ denotes the off diagonal entries of a matrix $A$ and $\mathbf{1}_n$ denotes the vector of $n$ ones. However, this set $\mathcal{K}$ is not convex, due to the constraint on $(Z^{-1})_{\text{off}}$.

In addition, we wish to encourage *few clusters* in the tasks graph so that, whenever possible, simpler and more structured graphs are preferred. Since we wish to have few clusters, the regularization should favor Laplacians with few small eigenvalues, which in turn implies a low effective rank for $Z$. To achieve the above objective, an appropriate penalty is the trace norm $\|Z\|_* := \sum_{i=1}^n \lambda_i(Z)$, where $\lambda_i(Z)$ denotes the

**Algorithm 1** Learning the tasks and the tasks Laplacian jointly.

**Initialization:** Select $W^0 \in \mathbf{M}_{d,n}$

**for** $k = 1, 2, \ldots$ **do**

1. $W \leftarrow \operatorname{argmin}\Big\{\sum_{\ell=1}^{n}\sum_{j=1}^{m_\ell} E(w_\ell^\top x_{j\ell}, y_{j\ell}) + \gamma\ \operatorname{tr}(WQW^\top) : W \in \mathbf{M}_{d,n}\Big\}$

2. $Q \leftarrow \operatorname{argmin}\Big\{\operatorname{tr}(WQW^\top) + \alpha \operatorname{tr}(Q^{-1}) : Q \succeq \varepsilon I_n\ ,\ Q_{\text{off}} \leq 0, Q\mathbf{1}_n = \varepsilon \mathbf{1}_n\Big\}$

**end for**

$i$-th eigenvalue of matrix $Z$. This norm is known to be the tightest convex surrogate to the rank [16] and in this case equals the trace of $Z$.

Thus, given a fixed input kernel $G$, we solve the problem

$$\min\Big\{E_y(K_{\mathbf{x}}c) + \gamma\, c^\top K_{\mathbf{x}} c + \alpha \operatorname{tr} Z : c \in \mathbb{R}^M, K_{\mathbf{x}} = T_{\mathbf{x}}(Z), \\ 0 \prec Z \preceq \tfrac{1}{\varepsilon} I_n, (Z^{-1})_{\text{off}} \leq 0, Z\mathbf{1}_n = \tfrac{1}{\varepsilon}\mathbf{1}_n\Big\},$$

where $\alpha$ is a second regularization parameter.

Applying the constraints back to the primal problem (4) and introducing the variable

$$Q = L + \varepsilon I_n\ ,$$

we obtain the optimization problem

$$\min\Bigg\{\sum_{\ell=1}^{n}\sum_{j=1}^{m_\ell} E(w_\ell^\top x_{j\ell}, y_{j\ell}) + \gamma \operatorname{tr}(WQW^\top) + \alpha \operatorname{tr}(Q^{-1}) \\ : W \in \mathbf{M}_{d,n},\ Q \succeq \varepsilon I_n,\ Q_{\text{off}} \leq 0, Q\mathbf{1}_n = \varepsilon \mathbf{1}_n\Bigg\}\ . \quad (10)$$

The objective function of this problem is non-convex whereas the feasibility set is convex.

## 2.4 Optimization

Even though problem (10) is not convex jointly in $W$ and $Q$, it is convex in one of the two matrix variables when the other remains fixed. Thus we may exploit this property to obtain an algorithm alternating between minimization of $W$ and minimization of $Q$ (Algorithm 1).

Step 1 requires solving a kernel based method such as support vector machines or kernel ridge regression in $dn$ variables. Step 2 is a semidefinite program, since a constraint of the form $\operatorname{tr}(Q^{-1}) \leq \beta$ can be rewritten as $\operatorname{tr} R \leq \beta, \left(\begin{smallmatrix} R & I_n \\ I_n & Q \end{smallmatrix}\right) \succeq 0$.

The number of variables depends only on $n$ and, up to about $100$ tasks, this SDP can be solved using interior point methods and packages such as SDPT3 or Sedumi. However, interior point methods do not scale well to larger $n$ and we had to develop a custom-made first-order method in order to handle larger numbers of tasks (Algorithm 2). Our method relies on positive semidefinite projections and hence on eigendecompositions (step 2). These can be computed exactly for a few thousand tasks and can be extended beyond that with low rank approximations combined with Lanczos or power methods.

Algorithm 2 is based on a parallel splitting algorithm of Douglas-Rachford type from [9, Prop. 27.8]. Douglas-Rachford algorithms require computation of *proximity operators*, which extend the concept of projections – see, for example, [9]. The proximity operator $\mathrm{prox}_f(x)$ of a lower semicontinuous, convex function $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ at $x \in \mathbb{R}^d$ is the unique minimizer

$$\mathrm{prox}_f(x) = \mathrm{argmin} \left\{ \tfrac{1}{2}\|y - x\|^2 + f(y) : y \in \mathbb{R}^d \right\}.$$

The first proximity operation (steps 2-4 in Algorithm 2) is that of the function

$$f_1(Q) = \begin{cases} \alpha \operatorname{tr}(Q^{-1}) & \text{if } Q \succeq \varepsilon I_n \\ +\infty & \text{otherwise} \end{cases}.$$

Since $f_1$ is a *spectral* function (a function of only the eigenvalues of $Q$), the proximity operation reduces to the proximity operation on its spectrum, by von Neumann's trace inequality [24]. This in turn separates into univariate optimization problems, which can be solved with cubic equations [8, Lem. 4.1].

The second proximity operation (step 5) is the projection on the set $\{Q \in \mathbf{S}_n : Q_{\text{off}} \leq 0\}$, where $\mathbf{S}_n$ denotes the space of $n \times n$ symmetric matrices. Finally, the third proximity operation (step 6) is that of the function

$$f_2(Q) = \begin{cases} \operatorname{tr}(WQW^\top) & \text{if } Q \in \mathbf{S}_n \text{ and } Q\mathbf{1}_n = \varepsilon \mathbf{1}_n \\ +\infty & \text{otherwise} \end{cases}.$$

This proximity operator is easy to derive with Lagrange multipliers.

# 3 Experiments

We applied our proposed multitask Laplacian learning method ("MT Laplacian") on synthetic data, where the task clustering is known, and on some real datasets. We compared our method with two benchmarks and three state of the art approaches:

- The independent ridge regression, where the tasks are learnt separately, serving as a baseline.
- The "true Laplacian" case, where we directly use the expected graph Laplacian instead of learning it (applicable only to synthetic data). Considering that the clusters have the same dependence among their tasks, and that all the tasks are generated with the same distributions, the "ground truth" graph Laplacian of our approach is expected to be proportional to this.

**Algorithm 2** Douglas-Rachford parallel splitting algorithm for SDP in step 2 of Algorithm 1.

**Initialization:** Select $X_1 = X_2 = X_3 \succeq \varepsilon I_n$

**for** $k = 1, 2, \ldots$ **do**

1. $P \leftarrow \frac{1}{3}(X_1 + X_2 + X_3)$
2. Compute eigendecomposition $X_1 = U \operatorname{Diag}(\lambda) U^\top$
3. $\mu_i \leftarrow \operatorname{argmin}\{\frac{1}{2}(\lambda_i - e)^2 + \alpha e^{-1} : e \geq \varepsilon\}\ \forall i \in \mathbb{N}_n$
4. $Y_1 \leftarrow U \operatorname{Diag}(\mu) U^\top$
5. $(Y_2)_{ij} \leftarrow \begin{cases} \min\{(X_2)_{ij}, 0\} & \text{if } i \neq j \\ (X_2)_{ij} & \text{if } i = j \end{cases} \quad \forall i, j \in \mathbb{N}_n$
6. $Y_3 \leftarrow B - BE - EB + (\frac{1}{n}\sum_{i,j=1}^n B_{ij} + \varepsilon)E$, where $B = X_3 - W^\top W$, $E = \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$
7. $K \leftarrow \frac{1}{3}(Y_1 + Y_2 + Y_3)$
8. $X_1 \leftarrow X_1 + 2K - P - Y_1$
9. $X_2 \leftarrow X_2 + 2K - P - Y_2$
10. $X_3 = X_3 + 2K - P - Y_3$

**end for**

**return** $Q \leftarrow Y_3$

- The multitask learning approach proposed in [15] for a known graph, where we use the Laplacian found by our method ("kernel with graph Laplacian"). Recall that [15] *do not learn* the Laplacian, but we report this result to demonstrate the drawbacks discussed in Section 2.1.
- Trace norm regularization, which is a state of the art method for multi-task problems [3, 27]. We used the SLEP implementation from [22].
- The clustered multitask method ("Clustered MT")[2] formulated in [17], discussed in section 2.1.

## 3.1 Simulated Data

We tested our method on an artificial dataset with two clusters of two tasks, generated as in [17], the true graph Laplacian being $\begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$. Each task $t$ in the cluster $c$ is a vector of regression coefficients in $\mathbb{R}^d$ ($d = 30$), generated as $w_t = w_c + \tilde{w}_t$, where $w_c$ is the cluster center and $\tilde{w}_t$ a task-specific vector having exactly the same non-zero features as $w_c$, drawn from $\mathcal{N}(0, \sigma_c^2)$, with $\sigma_c^2 = 16$. The cluster centers $w_c$ are orthogonal in the first $d - 2$ dimensions: the first cluster has $(d-2)/2$ features drawn from $\mathcal{N}(0, \sigma_r^2)$ ($\sigma_r^2 = 900$), and the second cluster has the other $(d-2)/2$, generated in the same way. The remaining two features are drawn from $\mathcal{N}(0, \sigma_c^2)$ for each task.

[2] The code comes from `http://cbio.ensmp.fr/~ljacob/`.

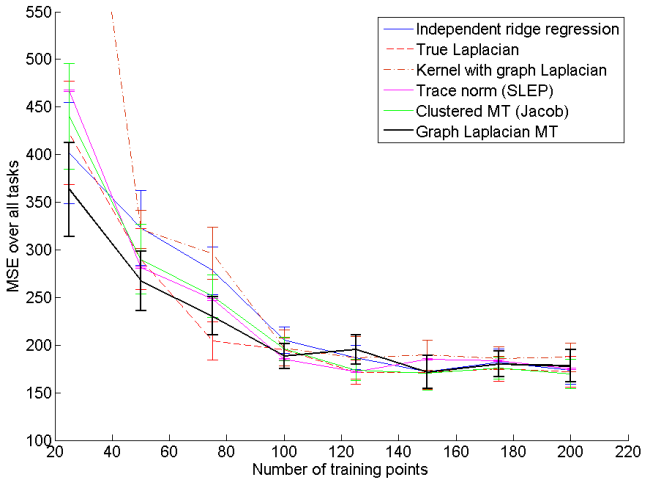


Figure 1: MSE of each tested method on simulated data

The inputs $X$ are vectors in $\mathbb{R}^d$, randomly generated from a uniform distribution on $[0, 1]$, and the corresponding output $Y_t$ for task $t$ is calculated as $Y_t = w_t^\top X + \epsilon_y$, $\epsilon_y$ being normal noise from $\mathcal{N}(0, \sigma_n^2)$ ($\sigma_n^2 = 150$). The tasks share the same input.

We first train each method on a training set of size $m$, then find the best parameters on a validation set of 2000 points, and finally test the optimal parameters on 10 test sets of 200 samples. The average test MSE across all tasks - since the task outputs have the same amplitude - and all test sets is the global error measure for each method. Results for this experiment are shown in Figure 1.

At $m = 25$, most methods fail to recover the correct tasks due to insufficient data. Starting from $m = 75$, the best error is generally obtained with the "true Laplacian" as long as there are enough observations, while the worst is the independent ridge regression. Our approach recovers the clusters and often performs better than the clustered multitask approach. All the methods have comparable MSEs starting from $m = 150$, so there is less benefit from multitask learning when the training set becomes large.

## 3.2 Real Data: Sector Indices

### 3.2.1 Financial Data Set

We applied our approach on a set of financial data composed of daily market prices of $d' = 50$ assets such as equity indices, interest rates, commodities, etc., from the year 2000 to 2012. The observed inputs are feature vectors processed from these assets' prices. More precisely, each asset provides two features, which are its recent linear trend and the variance of residuals around this trend; thus, we have $d = 100$ features per observation. The outputs to predict are future returns of sector indices, representing

European industry sectors [3]. These indices group European listed companies into 19 different categories of activity, and their values are defined as linear combinations of their constituents' share price. Sector indices may form clusters, because some groups of sectors show similar relative variation to the overall European market. For example, it is well known that the banking sector is often significantly correlated to the insurance sector, or that the food and beverage sector is correlated to the heath care one, the latter two being what practitioners call "defensive" sectors. Such behavior makes the problem a very suitable application for multitask learning, and we expect predictions that are consistent with the sectors' dependence graph to outperform those learnt independently.

#### 3.2.2 Validation and Test Method

Since our data are time series, testing must be done in a chronological way, meaning the test set must be posterior to the validation set, which is itself posterior to the training set. This is consistent with the way such models can be used in practice, excluding the use of more usual error assessment methods such as cross-validation. We propose the following test procedure, which is common among the practitioners. On the whole range of data, we define $k$ windows split into a training set made of the $m_{train}$ first points and a test set of the $m_{test}$ last points. For simplicity, the windows have $m_{train} + m_{test}$ points in total, they are equally spaced, and possibly overlapping.

We first validate learning parameters on the $k_{validation}$ first windows. On each window, the model is fitted on the training set and the error is assessed on the test set. The best parameter is chosen from the average error over all validation windows. Then, we use the best parameter to train and test the method on the $k - k_{validation}$ remaining windows, and finally get the test error as the average over all test windows.

#### 3.2.3 Results

We considered windows of 300 working days, with $m_{train} = 250$ (a year) and $m_{test} = 50$. We tested 7 sectors: Food & Beverage, Health Care, Personal & Household Goods, Retail, Financial Services, Banks, and Insurance. From both statistical and financial points of view, the first four sectors may form a cluster, while the other three often belong to another cluster. However, the dependence structure between these tasks may vary across time, so clustering on a large history beforehand generally lacks relevance for future predictions. This is also the reason why we choose a rather small training window. Table 1 shows a performance comparison between the tested methods. We see a slight improvement with our method compared to the others, even though this data set exhibits high noise and standard errors are not negligible. Validation was done with all the windows ending before January 2009. In addition, Figure 2 shows the multitask Laplacian method recovering the task clusters of the first 3 sectors and the last 3 sectors more clearly than the Clustered MT method.

[3] As defined by the ICB classification at the Supersector level: `http://www.icbenchmark.com`. The indices are defined and maintained by STOXX: `http://www.stoxx.com/indices/types/sector.html`.

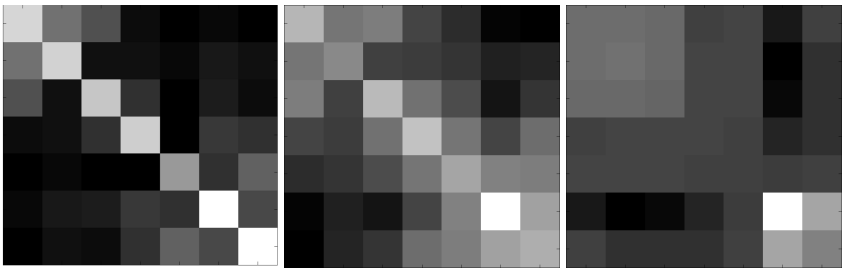

Figure 2: Recovered graph Laplacian (absolute values) on the last test window (left); recovered $W^\top W$ for MT Laplacian (middle) and Clustered MT (right).

Table 1: Prediction error (MSE) and standard error for the 7 sector indices

| Method | Test MSE (SE) $\times 10^{-3}$ |
|---|---|
| Independent ridge regression | 2.03 (0.6552) |
| Kernel with graph Laplacian | 2.31 (0.5143) |
| Trace norm | 2.10 (0.7484) |
| Clustered MT | 1.97 (0.6014) |
| MT Laplacian | 1.85 (0.5084) |

## 3.3 Real Data: Movie Ratings

Our approach also proved to be effective on MovieLens data, a well known data set used for multitask learning and recommender systems.

### 3.3.1 The MovieLens 100k Data Set

The MovieLens 100k dataset[4] contains ratings of 1682 films by 943 users. The ratings are numbers among $\{1, 2, 3, 4, 5\}$, the films are described by their title, date, and categories they belong to, among $d = 19$ different genres. Information such as age, gender, occupation and zip code are provided about users.

To apply the different multitask methods on this data set, we consider each user as a task. We want to predict the rating a user gives to a film, based on the film's features. For each film, the feature vector is $(t, g_1, \ldots, g_d)$ where $t$ is the release date and $g_i$ is 1 if the film belongs to genre number $i$, and 0 otherwise. Each film can belong to several genres. Formally, for each task $l$, we want to regress the output vector $Y_l$ of size $m_l$ on its feature matrix $X_l$ of size $m_l \times (d + 1)$.

We used the training/test arrangements provided with this data set, where 10 points were singled out as test data for each user, the rest being used for training and validation. The parameters for each method were tuned by 3-fold cross validation on the

[4]Full datasets are available at `http://www.grouplens.org/node/73`. These datasets were initially collected from the MovieLens website (`movielens.umn.edu`).

Table 2: Test error (MSE), validation error (MSE) and standard error (SE) for the MovieLens 100k dataset. Standard errors apply to the 3-fold cross-validation splits.

| Method | Test | Validation | SE |
|---|---|---|---|
| Independent ridge | 1.0896 | 1.0907 | 0.0027 |
| Trace norm | 1.0776 | 1.0875 | 0.0018 |
| Clustered MT | 1.0903 | 1.0900 | 0.0029 |
| MT Laplacian | 1.0725 | 1.0695 | 0.0029 |

training subset. Tasks have very diverse data sizes, ranging from 20 to 737 ratings in total, with a median at 65. The learning problem has on average very few data observations per task, thus we expect multitask learning approaches to outperform independent learning.

#### 3.3.2 Results

Table 2 shows test errors, validation errors and validation standard errors for this dataset. Validation performance excluded tasks with less than 30 training points, to avoid tasks with too few samples and better estimate the overall error. By taking them out, validation proved to be much closer to test on all tasks. This process did not change the validated parameters compared with the full set. Since the error is well estimated, the standard error on cross validation splits is relevant in representing error variability.

Our method significantly outperforms independent learning and the other approaches. Surprisingly, clustered multitask learning does not improve over independent learning on this problem. This is possible sometimes even though the nonconvex formulation of [17] subsumes independent learning. The reason is that the convex relaxation (3) does not subsume Frobenius regularization, due to the presense of the projection matrix $\Pi$. It is also interesting that MT Laplacian also outperforms trace norm regularization, which is a standard benchmark in recommendation problems. This fact may indicate a large significance of task clustering in problems like Movielens.

## 4 Conclusion

We have proposed a novel multitask learning method for learning tasks which exhibit clustered graphs of relations. The method is based on regularization with a penalty that involves the Laplacian of the tasks graph. Our multitask formulation is unique in allowing for learning both the tasks and the Laplacian within the same optimization problem. We have also presented a conceptually simple alternating minimization algorithm for solving this problem up to a few thousand tasks. We have proposed a first-order convex optimization algorithm for solving a semidefinite program appearing as a subproblem of our method. Finally, we reported state of the art results showing performance improvement on both simulated and real datasets, simultaneously recovering well the graph of task relations.

# References


[1] A. Agarwal, H. Daumé III, and S. Gerber. Learning multiple tasks using manifold regularization. *Advances in Neural Information Processing Systems*, 23:46–54, 2010.

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

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

[4] A. Argyriou, A. Maurer, and M. Pontil. An algorithm for transfer learning in a heterogeneous environment. In *European Conference on Machine Learning*, 2008. To appear.

[5] A. Argyriou, C. A. Micchelli, and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. In *Proceedings of the Eighteenth Conference on Learning Theory*, pages 338–352, 2005.

[6] A. Argyriou, C. A. Micchelli, and M. Pontil. When is there a representer theorem? Vector versus matrix regularizers. *Journal of Machine Learning Research*, 10:2507–2529, 2009.

[7] B. Bakker and T. Heskes. Task clustering and gating for bayesian multi–task learning. *Journal of Machine Learning Research*, 4:83–99, 2003.

[8] L. Baldassarre, J. Morales, A. Argyriou, and M. Pontil. A general framework for structured sparsity via proximal optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 82–90, 2012.

[9] H. H. Bauschke and P. L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. CMS Books in Mathematics. Springer, 2011.

[10] J. Baxter. A model for inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

[11] S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *Proceedings of the Sixteenth Annual Conference on Learning Theory*, volume 2777, pages 567–580, 2003.

[12] A. Caponnetto, C. A. Micchelli, M. Pontil, and Y. Ying. Universal multi-task kernels. *The Journal of Machine Learning Research*, 9:1615–1646, 2008.

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

[14] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. Boosted multi-task learning. *Machine learning*, 85(1-2):149–173, 2011.

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

[16] M. Fazel, H. Hindi, and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings, American Control Conference*, volume 6, pages 4734–4739, 2001.

[17] L. Jacob, F. Bach, and J.-P. Vert. Clustered multi-task learning: a convex formulation. In *Advances in Neural Information Processing Systems 21*, pages 745–752. MIT Press, 2009.

[18] 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 Machine Learning*, pages 521–528, 2011.

[19] G.S. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502, 1970.

[20] A. Kumar and H. Daumé III. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1383–1390, 2012.

[21] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[22] J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009.

[23] A. Maurer. The Rademacher complexity of linear transformation classes. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT)*, volume 4005 of *LNAI*, pages 65–78. Springer, 2006.

[24] L. Mirsky. A trace inequality of John von Neumann. *Monatshefte f ur Mathematik*, 79(4):303–306, 1975.

[25] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

[26] N. Srebro and S. Ben-David. Learning bounds for support vector machines with learned kernels. In *Proceedings of the Nineteenth Conference on Learning Theory*, volume 4005 of *LNAI*, pages 169–183. Springer, 2006.

[27] N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press, 2005.

[28] Y. Ying and C. Campbell. Generalization bounds for learning the kernel. In *Proceedings of the 22nd Annual Conference on Learning Theory*, 2009.