Learning Output Kernels for Multi-Task Problems

Francesco Dinuzzo *

Abstract

Simultaneously solving multiple related learning tasks is beneficial under a variety of circumstances, but the prior knowledge necessary to correctly model task relationships is rarely available in practice. In this paper, we develop a novel kernel-based multi-task learning technique that automatically reveals structural inter-task relationships. Building over the framework of output kernel learning (OKL), we introduce a method that jointly learns multiple functions and a low-rank multi-task kernel by solving a non-convex regularization problem. Optimization is carried out via a block coordinate descent strategy, where each subproblem is solved using suitable conjugate gradient (CG) type iterative methods for linear operator equations. The effectiveness of the proposed approach is demonstrated on pharmacological and collaborative filtering data.

1 Introduction

Combining multiple datasets for solving related inference problems is a common and successful practice in several domains such as econometrics and marketing analysis, recommender systems on the web, bioinformatics, pharmacology, and many others. Indeed, simultaneously solving multiple inference problems can improve performances, provided that the relationships between them are correctly modeled. The themes of transfer learning, multi-task learning or "learning to learn" [9, 19, 44] have attracted considerable attention in the literature, see [30] for a recent survey. A possible way to enforce relationships between the tasks is to extract a set of shared features. This can be done by employing multi-layer neural networks where the hidden layer is shared among the tasks [9], or also within a convex regularization framework [4]. Task relationships can be also modeled within the framework of Gaussian Processes estimation [7, 8, 31, 40, 46, 49] by designing a joint covariance function. A variety of other models have been proposed to represent and exploit inter-task relationships [3, 23, 32, 48, 50, 51].

*Max Planck Institute for Intelligent Systems, Spemannstrasse 38,72076 Tübingen, Germany. E-mail: fdinuzzo@tuebingen.mpg.de

Correctly modeling the inter-task relationships for a given application is critical but not always easy. In fact, in many inference problems, the tasks are known to be related with each other, but the available prior knowledge is not sufficient to model the relationships in advance. For this reason, some recent works have been focusing on inferring inter-task relationships automatically from the data, while solving the multi-task learning problem. One possibility is to assume that the tasks can be clustered into homogeneous groups and try to learn such clustering from the data [6]. Optimization-based approaches that jointly infer the task parameters and the inter-task relationships in the form of a similarity matrix include the spectral regularization approach of [5] and the method of [52] based on convex optimization. An online method has been also proposed recently to learn multiple linear classifiers as well as a task relationship matrix [38].

Regularized kernel methods [39] have been employed successfully in a variety of single-task learning problems. Their extension to the multi-task setting [15] calls for the design of suitable operator-valued kernels that model similarities of both inputs and tasks. Some classes of operator-valued kernels have been recently reviewed in [2]. Within the framework of regularization in RKHS of vector-valued functions, the problem of inferring task relationships boils down to the problem of learning a multi-task kernel. A possible way to address this problem consists in learning a linear combination of task-specific kernels [47], within the framework of multiple kernel learning (MKL). Recently, a class of output kernel learning (OKL) methods [12, 13] has been introduced in the context of multi-output learning problems (such as vectorial regression, multi-class and multi-label classification) to automatically synthesize a decomposable matrix-valued kernel that encodes the relationships between the output components. Differently from MKL, OKL methods don't require the specification of a set of basis kernels to be combined, as the output kernel is searched into the whole cone of positive semidefinite kernels.

This paper explores the possibility of automatically learning the relationships between multiple tasks via output kernel learning. To this end, we extend a technique for learning low-rank output kernels proposed in [11] to the multi-task setting. The low-rank encouraging regularization can be shown to be useful both for computational reasons and learning performances. The extension developed in this paper is based on the use of a suitable weighted loss that allows for multiple datasets with different input sampling patterns. The resulting method can be also interpreted as a non-linear kernel based generalization of low-rank matrix completion techniques. Even in the simpler setup of linear low-rank matrix factorization, the case of incomplete observations has been recognized to be computationally challenging, see e.g. [42], since existing optimization techniques based on eigendecompositions do not carry over to the weighted case. For this reason, we have developed a new strategy to solve the optimization problem. Similarly to [52], we learn multiple tasks as well as their relationships by solving a joint optimization problem. However, differently from [52] and [38], that learn multiple linear functions by convex optimization, we learn multiple non-linear functions by solving a non-convex optimization problem. Albeit non-convex, the

problem exhibits a special structure that allows for effective optimization via a block-coordinate descent procedure based on the iterative application of the conjugate gradient (CG) algorithm for linear operator equations. Our method is also related to the technique derived in [8] in the context of Bayesian estimation of Gaussian Processes, that allows to learn a similarity (covariance) matrix between the tasks. While [8] aims at optimizing a marginal likelihood type functional, our method is based on the minimization of a functional with trace norm regularization plus rank constraint. In [8], the authors adopt a general purpose gradient-descent solver to optimize their objective functional, whereas in this paper we develop a novel optimization strategy that is specifically designed to solve the proposed OKL optimization problem. Differently from [8], our method is able to deal with the case of incomplete sampling, and automatically encourages low-rank solutions without introducing relaxations of the original problem.

2 Weighted output kernel learning

Let $\mathbf{I}$ denote the identity matrix and e the vector of all ones (of suitable dimensions). For any matrix $\mathbf{A}$, let $\mathbf{A}^T$ denote the transpose, $\mathrm{tr}(\mathbf{A})$ the trace, $\mathrm{rank}(\mathbf{A})$ the rank, $\mathrm{vec}(\mathbf{A})$ the vectorization operator, $\mathbf{A}^\dagger$ the pseudo-inverse, $\|\mathbf{A}\|_F = \left(\mathrm{tr}\left(\mathbf{A}^T\mathbf{A}\right)\right)^{1/2}$ the Frobenius norm, and $\|\mathbf{A}\|_* := \mathrm{tr}\left(\left(\mathbf{A}^T\mathbf{A}\right)^{1/2}\right)$ the nuclear norm. For any pair of matrices of the same size $\mathbf{A}, \mathbf{B}$, let $\langle \mathbf{A}, \mathbf{B} \rangle_F := \mathrm{tr}(\mathbf{A}^T\mathbf{B})$ denote the Frobenius inner product. The symbols $\otimes$ and $\odot$ denote the Kronecker product and the Hadamard (element-wise) product, respectively. Finally, let $\mathbb{S}_+^m$ denote the closed cone of positive semidefinite matrices of order m, and

$$\mathbb{S}_+^{m,p} = \left\{\mathbf{A} \in \mathbb{S}_+^m : \mathrm{rank}(\mathbf{A}) \leq p\right\} \subseteq \mathbb{S}_+^m$$

the cone of positive semidefinite matrices with rank less than or equal to p.

2.1 Output kernel learning for multi-task problems

Consider the problem of learning several functions (tasks) $g_j : \mathcal{X} \to \mathbb{R}$ $(j = 1, \ldots, m)$ from multiple datasets of input-output pairs (x_{ij}, y_{ij}). Since the input set $\mathcal{X}$ is common to all the tasks, the functions g_j can be combined into a single vector-valued function $g : \mathcal{X} \to \mathbb{R}^m$, to be learned from a single dataset of ℓ pairs (x_i, y_i), where the output data y_i are now vectors that may contain missing components. Let $\mathbf{W} \in \{0,1\}^{\ell \times m}$ denote a binary *weight matrix* specifying which output components are missing for each example. More precisely, the i-th row of the weight matrix is a binary vector w_i with zeros in correspondence with the missing components of y_i. Since the weight matrix is always available, we can assume without loss of generality that all the missing output components have been imputed with zeros.

In the following, we describe a regularization approach where the function g is searched into a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ of vector-valued functions $g : \mathcal{X} \to \mathbb{R}^m$ associated with a decomposable reproducing kernel H of the form

$$H(x_1, x_2) = K_X(x_1, x_2) \cdot \mathbf{L},$$

where K_X is positive semidefinite scalar kernel (called *input kernel*) and $\mathbf{L} \in \mathbb{S}_+^m$ is a symmetric positive semidefinite matrix (called *output kernel*). For more details about RKHS of vector-valued functions, see [27].

The function $g \in \mathcal{H}$ and the output kernel $\mathbf{L}$ are simultaneously learned by solving a joint regularization problem of the form

$$\min_{\substack{\mathbf{L} \in \mathbb{S}_+^{m,p} \\ g \in \mathcal{H}}} \left(\sum_{i=1}^{\ell} \frac{\|w_i \odot (g(x_i) - y_i)\|_2^2}{2\lambda} + \frac{\|g\|_{\mathcal{H}}^2}{2} + \frac{\mathrm{tr}(\mathbf{L})}{2} \right). \tag{1}$$

The objective functional contains a data-fitting term, taking into account the prediction error in correspondence with the observed output components, a regularization term on the unknown function g, and a trace regularization on the output kernel. Optimization of the output kernel is carried over the low-rank cone $\mathbb{S}_+^{m,p}$, thus also imposing a hard rank constraint.

In view of the representer theorem [14, 24], the minimization problem with respect to g admits a solution of the form

$$g^*(x) = \mathbf{L} \sum_{i=1}^{\ell} c_i K_X(x, x_i),$$

with suitable coefficient vectors $c_i \in \mathbb{R}^m$. Letting $\mathbf{K} \in \mathbb{S}_+^{\ell}$ be such that $\mathbf{K}_{ij} = K(x_i, x_j)$, the following finite-dimensional optimization problem is obtained

$$\min_{\substack{\mathbf{L} \in \mathbb{S}_+^{m,p} \\ \mathbf{C} \in \mathbb{R}^{\ell \times m}}} \left(\frac{\|\mathbf{Y} - \mathbf{KCL}\|_{\mathbf{W}}^2}{2\lambda} + \frac{\langle \mathbf{C}^T \mathbf{KC}, \mathbf{L} \rangle_F}{2} + \frac{\mathrm{tr}(\mathbf{L})}{2} \right), \tag{2}$$

where the matrices $\mathbf{Y}, \mathbf{C} \in \mathbb{R}^{\ell \times m}$ are defined as

$$\mathbf{Y} = (y_1, \ldots, y_\ell)^T, \qquad \mathbf{C} = (c_1, \ldots, c_\ell)^T,$$

and the (semi)-norm $\|\cdot\|_{\mathbf{W}}$ is given by

$$\|\mathbf{A}\|_{\mathbf{W}} = \|\mathbf{W} \odot \mathbf{A}\|_F.$$

Observe that $\mathbf{Y}$ is a sparse matrix (missing values have been imputed with zeros). Moreover, since $\mathbf{W}$ is a binary matrix with the same sparsity pattern of $\mathbf{Y}$, we also have

$$\mathbf{Y} = \mathbf{W} \odot \mathbf{Y}. \tag{3}$$

Within the formulation of problems (1) and (2), a first possibility is to simply set $p = m$. With such a choice, the hard constraint on the rank is not present, but the trace regularization will still encourage low-rank solutions, see e.g. [16]. It is nevertheless convenient to allow for the more general choice $p \leq m$. A first reason is that, by providing an explicit bound $p < m$ on the rank, it is possible to control the computational and memory requirements of the method before running the optimization. This holds since, within the optimization framework developed in the next section, only matrices of rank p are stored and manipulated, whereas the full matrix $\mathbf{L}$ is never formed explicitly. A second reason is that, when the regularization parameter λ is small enough, the hard rank constraint becomes active, and may yield interesting solutions that are not obtainable by using only the trace regularization (see, for example, the experiment in subsection 4.1).

The objective functional of (2) is not jointly (quasi)-convex. However, it is convex separately with respect to both $\mathbf{L}$ and $\mathbf{C}$. In addition, for $p = m$ it is an invex function [28] in the interior of the feasible set, meaning that every stationary point is a global minimizer. When all the output components are observed, i.e. $\mathbf{W} = ee^T$, problem (2) can be attacked using techniques based on eigendecompositions. Since these techniques do not apply anymore for general weight matrices, in the following we develop a new strategy to obtain a minimizer of (2).

2.2 Equivalent formulations

Before going into the details of the optimization procedure proposed to solve (2), is it useful to introduce some equivalent reformulations of the optimization problem. The proofs of the two following Lemmas are reported in the appendix.

Lemma 2.1 *If* $(\mathbf{A}, \mathbf{B})$ *is an optimal solution of the following problem:*

$$\min_{\substack{\mathbf{A}\in\mathbb{R}^{\ell\times p}\\ \mathbf{B}\in\mathbb{R}^{m\times p}}} \left(\frac{\|\mathbf{Y} - \mathbf{K}\mathbf{A}\mathbf{B}^T\|_{\mathbf{W}}^2}{2\lambda} + \frac{\langle \mathbf{A}, \mathbf{K}\mathbf{A}\rangle_F}{2} + \frac{\|\mathbf{B}\|_F^2}{2} \right), \tag{4}$$

then any pair $(\mathbf{C}, \mathbf{L})$ *such that*

$$\mathbf{L} = \mathbf{B}\mathbf{B}^T, \qquad \mathbf{A} = \mathbf{C}\mathbf{B},$$

is an optimal solution of problem (2).

Lemma 2.1 provides a new formulation of the low-rank output kernel learning problem (2) that involves only "thin" matrices $\mathbf{A}$ and $\mathbf{B}$, whose size can be controlled by selecting the parameter p. Such formulation turns out to be particularly convenient for optimization purposes. It is also insightful to observe that problem (2) can be reformulated as a reduced rank least square problem with a "kernelized" nuclear norm regularization.

Lemma 2.2 *If $\boldsymbol{\Theta}$ is an optimal solution of the following problem:*

$$\min_{\boldsymbol{\Theta}\in\mathbb{R}^{\ell\times m}} \left[\frac{\|\mathbf{Y}-\mathbf{K}\boldsymbol{\Theta}\|_{\mathbf{W}}^2}{2\lambda} + \operatorname{tr}\left(\left(\boldsymbol{\Theta}^T\mathbf{K}\boldsymbol{\Theta}\right)^{1/2}\right)\right], \quad \text{subject to} \quad \operatorname{rank}(\boldsymbol{\Theta}) \leq p,$$

then the pair

$$\mathbf{L} = \left(\boldsymbol{\Theta}^T\mathbf{K}\boldsymbol{\Theta}\right)^{1/2}, \quad \mathbf{C} = \mathbf{L}^{\dagger}\boldsymbol{\Theta},$$

is an optimal solution of problem (2).

Lemma 2.2 shows that the low-rank output kernel learning problem (2) is a non-linear kernelized generalization of least squares low-rank matrix approximation (RMF) with nuclear norm regularization. Indeed, RMF can be obtained as a particular case by choosing the input kernel as a Kronecker delta kernel

$$K(x_1, x_2) = \delta_K(x_1, x_2) = \begin{cases} 1, & x_1 = x_2 \\ 0, & \text{else} \end{cases}$$

so that

$$\mathbf{K} = \mathbf{I}, \qquad \operatorname{tr}\left(\left(\boldsymbol{\Theta}^T\mathbf{K}\boldsymbol{\Theta}\right)^{1/2}\right) = \|\boldsymbol{\Theta}\|_*.$$

The related MMMF (maximum-margin matrix factorization) technique [33, 43] would also correspond to the case $\mathbf{K} = \mathbf{I}$, but with hinge-type (SVM) losses instead of the square loss.

3 A block coordinate descent strategy

In the following, we focus on the solution of (4). If p is much smaller than m, handling the low-dimensional factor $\mathbf{B}$ is much more convenient than directly handling the full output kernel matrix. Let $J(\mathbf{A}, \mathbf{B})$ denote the objective functional of (4). Observe that, although J is non-convex, it is unconstrained and separately convex with respect to both factors. Therefore, it is natural to adopt a block coordinate descent technique that iteratively alternates between optimization with respect to the two blocks. Clearly, other optimization strategies are also possible, but the block coordinate descent approach is memory efficient, robust, and simple to implement. In addition, it turns out to behave well in practice, since very few iterations are typically sufficient to obtain a good solution, especially when combined with a warm-start regularization path procedure (see subsection 3.3). As shown in the following, the two subproblems boil down to the solution of linear equations of the form

$$\mathcal{L}_A\mathbf{A} = y_A, \tag{5}$$

$$\mathcal{L}_B\mathbf{B} = y_B \tag{6}$$

where $\mathcal{L}_A$ and $\mathcal{L}_B$ are linear operators mapping matrices into vectors. In the following subsections, we derive equations (5)-(6), and discuss the application of suitable iterative methods for solving them.

3.1 Sub-problem w.r.t. A

The sub-problem w.r.t. $\mathbf{A}$ is the most numerically challenging of the two since, in general, the linear operator $\mathcal{L}_A$ is not symmetric. For any fixed $\mathbf{B}$, a necessary and sufficient condition for $\mathbf{A}$ to be optimal is obtained by setting to zero the partial derivative of the objective functional:

$$\frac{\partial J}{\partial \mathbf{A}}(\mathbf{A}, \mathbf{B}) = \mathbf{K}\left[\frac{\mathbf{W} \odot \left(\mathbf{K}\mathbf{A}\mathbf{B}^T - \mathbf{Y}\right)\mathbf{B}}{\lambda} + \mathbf{A}\right] = \mathbf{0}.$$

A sufficient condition is given by

$$\mathbf{W} \odot \left(\mathbf{K}\mathbf{A}\mathbf{B}^T\right)\mathbf{B} + \lambda\mathbf{A} = \mathbf{Y}\mathbf{B},$$

where we have used (3) in order to simplify the right hand side. If the weight matrix is full, this last equation reduces to a discrete-time Sylvester equation, a well studied class of linear matrix equations, see e.g. [41]. However, for general $\mathbf{W}$, the optimality condition is not a Sylvester equation anymore, due to presence of the Hadamard product. Now, letting

$$\mathbf{W}_D = \operatorname{diag}(\operatorname{vec}(\mathbf{W})),$$

$$\mathcal{L}_A\mathbf{A} = \left[\left(\mathbf{B}^T \otimes \mathbf{I}\right)\mathbf{W}_D\left(\mathbf{B} \otimes \mathbf{K}\right) + \lambda\mathbf{I}\right]\operatorname{vec}(\mathbf{A}),$$

$$y_B = \operatorname{vec}(\mathbf{Y}\mathbf{B}),$$

and using the matrix identities (8) and (10), the sufficient optimality condition can be rewritten in the form (5).

If $\mathbf{W}$ is full, it is easy to verify, using the mixed-product identity (9) for the Hadamard product, that the operator $\mathcal{L}_A$ reduces to

$$\mathcal{L}_A\mathbf{A} = \left[\left(\mathbf{B}^T\mathbf{B}\right) \otimes \mathbf{K} + \lambda\mathbf{I}\right]\operatorname{vec}(\mathbf{A}),$$

and is therefore symmetric and positive definite. In such a case, one can either apply the conjugate gradient algorithm, or also use a procedure based on eigen-decompositions in order to solve (5). Unfortunately, these methods cannot be applied for general weight matrices, since the operator $\mathcal{L}_A$ is not guaranteed to be symmetric.

A first possible way to attack the problem is trying to directly obtain a solution of the non-symmetric operator equation (5) by means of iterative methods that can handle non-symmetric equations, such as the generalized minimal residual method (GMRES) [37]. However, the computational requirements of GMRES grow with the number of iterations and, for large scale problems, a restart strategy is needed to limit the memory requirements at the expense of convergence speed. The alternative is to introduce a change of variable to make the problem symmetric, and then apply a preconditioned conjugate gradient (CG) algorithm [20] on the new linear equation. Generally, CG requires less computational resources than GMRES, since the search directions are obtained via

simple recursive updates and a restart strategy is not required. For additional details about these and others iterative method for solving linear equations, see e.g. [36].

Another way to address non-symmetry is to look for a change of variable that makes the problem symmetric. A first possible change of variable derives from the observation that the optimal solution must be in the form $\mathbf{A} = \mathbf{CB}$. By using this representation, one can equivalently solve a new linear equation with respect to $\mathbf{C}$:

$$\mathbf{W} \odot \left[\mathbf{K} \left(\mathbf{W} \odot \mathbf{C}\right) \mathbf{B}\mathbf{B}^T\right] + \lambda \mathbf{C} = \mathbf{Y}.$$

Once again, the equation can be rewritten in the form

$$\mathcal{L}_C \mathbf{C} = \mathrm{vec}(\mathbf{Y}),$$

where

$$\mathcal{L}_C \mathbf{C} = \left[\mathbf{W}_D \left((\mathbf{B}\mathbf{B}^T) \otimes \mathbf{K}\right) \mathbf{W}_D + \lambda \mathbf{I}\right] \mathrm{vec}(\mathbf{C}).$$

This time, the operator $\mathcal{L}_C$ is symmetric and positive definite, so that CG applies. A disadvantage of this approach is that, for $p \ll m$, $\mathbf{C}$ is much higher-dimensional than $\mathbf{A}$. Nevertheless, when the weight matrix $\mathbf{W}$ is sparse, $\mathbf{C}$ is also a sparse matrix with the same sparsity pattern, and one can take advantage of this property.

Another approach is based on a factorization of the input kernel matrix of the form

$$\mathbf{K} = \mathbf{F}\mathbf{F}^T.$$

The factor $\mathbf{F}$ can be a Cholesky factor, a matrix square root, or also a low-rank factor. We can use the factorization to introduce a new change of variable

$$\mathbf{A}_F = \mathbf{F}^T \mathbf{A}, \tag{7}$$

so that the matrix $\mathbf{A}_F$ solves the equation

$$\mathbf{F}^T \mathbf{W} \odot \left(\mathbf{F}\mathbf{A}_F\mathbf{B}^T\right) \mathbf{B} + \lambda \mathbf{A}_F = \mathbf{F}^T \mathbf{Y}\mathbf{B}.$$

This last equation can be rewritten in vectorized form

$$\mathcal{L}_F \mathbf{A}_F = \mathrm{vec}(\mathbf{F}^T \mathbf{Y}\mathbf{B}),$$

where

$$\mathcal{L}_F \mathbf{A}_F = \left[\left(\mathbf{B}^T \otimes \mathbf{F}^T\right) \mathbf{W}_D \left(\mathbf{B} \otimes \mathbf{F}\right) + \lambda \mathbf{I}\right] \mathrm{vec}(\mathbf{A}_F).$$

Independently of which factorization of $\mathbf{K}$ has been adopted, the linear operator $\mathcal{L}_F$ is symmetric and positive definite, so that CG can be applied. Once a solution $\mathbf{A}_F$ has been obtained, the matrix $\mathbf{A}$ can be recovered as a solution of the linear system (7).

In our experiments, all of the aforementioned approaches have been tested. The last approach based on the Cholesky factorization of the input kernel matrix turned out to be generally faster than the others, therefore we selected it for our current implementation.

3.2 Sub-problem w.r.t. B

The sub-problem w.r.t $\mathbf{B}$ is essentially a multiple ridge regression problem. First of all, observe that the objective functional depends on $\mathbf{A}$ only through the product $\mathbf{E} = \mathbf{K}\mathbf{A}$. Therefore, when $\mathbf{A}$ is fixed, a necessary and sufficient condition for $\mathbf{B}$ to be optimal is

$$\frac{\partial J}{\partial \mathbf{B}}(\mathbf{A}, \mathbf{B}) = \mathbf{E}^T \frac{\mathbf{W} \odot \left(\mathbf{E}\mathbf{B}^T - \mathbf{Y}\right)}{\lambda} + \mathbf{B}^T = \mathbf{0}.$$

In this case, it turns out that an expression for the rows of $\mathbf{B}$ can be even written in closed form. Indeed, let b_j $(j = 1, \ldots, m)$ denote the rows of $\mathbf{B}$, y^j and w^j denote the columns of $\mathbf{Y}$ and $\mathbf{W}$, respectively. Then, we have

$$b_j = \left(\mathbf{E}^T \mathrm{diag}(w^j)\mathbf{E} + \lambda \mathbf{I}\right)^{-1} \mathbf{E}\mathrm{diag}(w^j)y^j.$$

Observe that the updates for the different rows can be computed in parallel. Finally, by letting

$$\mathcal{L}_B \mathbf{B} = \left[(\mathbf{E}^T \otimes \mathbf{I})\mathrm{diag}(\mathrm{vec}(\mathbf{W}^T))(\mathbf{E} \otimes \mathbf{I}) + \lambda \mathbf{I}\right] \mathrm{vec}(\mathbf{B}),$$

and

$$y_B = \mathrm{vec}(\mathbf{Y}^T \mathbf{E}),$$

the optimality condition can be also rewritten in the form (6). Here, the operator $\mathcal{L}_B$ can be seen to be symmetric and positive definite, therefore it is possible to use a conjugate gradient method to obtain a solution.

3.3 Implementation details

Several important details must be taken into account in order to guarantee a correct and efficient convergence behavior.

3.3.1 Warm-start path procedure

It is typically necessary to train the kernel machine for several different values of the regularization parameter λ. Generally, the regularization problem is better conditioned for large values of λ. On the same time, the solution is expected to depend continuously on the regularization parameter. For these reasons, it is convenient to initialize the optimization of each problem with the solution obtained with the previous value of λ, after sorting the different values of the regularization parameter in decreasing order. Such *warm-start* regularization path strategy is very effective in practice: if the grid of values of λ is sufficiently fine, one or two iterations of block coordinate descent for each value of λ are often sufficient to converge with a good accuracy.

3.3.2 Initialization

It is possible to show that the rank of $\mathbf{B}$ cannot increase between an iteration of block-coordinate descent and the next. For this reason, at the very beginning of the warm-start procedure, $\mathbf{B}$ is initialized to a full-rank matrix. Otherwise, an optimal solution of high rank may be missed by the algorithm. Another issue comes from the fact that the origin is a stationary point of the objective functional for any value of λ. Although, for very large values of the regularization parameter, the origin is actually a global minimizer, this is not the case anymore for small values. Now, in view of the warm-start procedure, the algorithm may never move away from the origin if the initial λ is very large. To safeguard against this behavior, $\mathbf{B}$ is re-initialized to a full-rank matrix every time it becomes too small (for example, w.r.t. the Frobenius norm).

4 Experiments

In this section, we analyze the performance of weighted OKL on a variety of multi-task problems, including pharmacokinetic-pharmacodynamics (PK-PD) problems, and popular collaborative filtering benchmarks (MovieLens datasets). In all the experiments on real data, the performance of weighted OKL is compared with the following methods:

- **Independent single-task learning**: corresponds to fixing the output kernel as $\mathbf{L} = \mathbf{I}$ instead of optimizing it, thus learning each task independently of the others.
- **Pooled single-task learning**: corresponds to fixing the output kernel as $\mathbf{L} = ee^T$, thus assuming that all the tasks are the same.
- **Regularized Matrix Factorization** (RMF): corresponds to fixing the input kernel as the Kronecker delta kernel, so that $\mathbf{K} = \mathbf{I}$.

All the experiments have been run in a MATLAB environment with an Intel i5 CPU 2.4 GHz, 4 GB RAM.

4.1 Reconstruction and denoising of multiple signals

In order to analyze the dependence of the computation time on the rank bound parameter p of the proposed OKL method, we conducted some controlled experiments on synthetic data. The experiments show both the computational and the potential predictive advantage induced by the hard rank constraint. First of all, we generated 50 independent realizations Z_k, $(k = 1, \ldots, 50)$ of a Gaussian Process on the interval $[-1, 1]$ of the real line with zero-mean and covariance function

$$\operatorname{cov}\left[(Z_k)_{x_1}, (Z_k)_{x_2}\right] = \exp(-10|x_1 - x_2|).$$

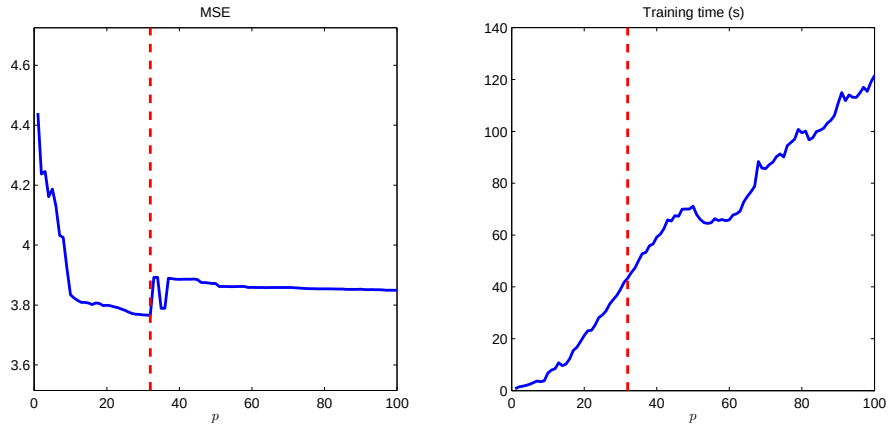

Figure 1: The continuous lines represent the MSE (left panel) and training time (right panel) of OKL, as a function of the parameter p. The vertical dashed line corresponds to the value of p that minimizes MSE.

Then, we generated $m = 200$ new processes U_j as

$$U_j = \sum_{k=1}^{50} B_{jk} Z_k,$$

where the mixing coefficients B_{jk} are independently drawn from a uniform distribution on the interval $[0, 1]$. Output data y_{ij} have been generated by sampling the processes U_j in correspondence with 100 points in the interval $[-1, 1]$ randomly drawn from a uniform grid of 200 points, and corrupting them by adding a zero-mean Gaussian noise with a signal to noise ratio of 1:1. For each process U_j, only 70 out of the 100 inputs have been used for training, while the remaining 30 are used for tuning the regularization parameter λ.

The input kernel for OKL is the same as the covariance function of the processes Z_k. Performances are measured by the reconstruction MSE (mean squared error), which is computable since we have also access to the generated non-corrupted signals. Figure 1 reports the MSE (left panel) and the computation time needed to train OKL over a whole range of values of the regularization parameter λ (right panel), both as a function of the parameter p. Observe that the training time is roughly increasing with p since matrices of higher dimension are involved in the computation.

The best performance (in terms of MSE) is observed in correspondence with an intermediate value of the rank bound parameter (the vertical line corresponds to $p = 32$). Hence, the best rank is considerably lower than the "true rank" of the model used to generate the non-corrupted data (namely 50). A mismatch between the two is to be expected for finite and noisy samples. Conditions under which the rank of the original model can be recovered will depend on the

sample size, the noise level, and the criteria used to choose the regularization parameter. It would be interesting to determine such conditions, a problem that we leave to future investigations.

4.2 Pharmacological data

Reconstructing response curves in multiple subjects from sparse sets of individual measurements is a typical problem in pharmacology. The curves are generally expected to exhibit similar shapes but, on the same time, considerable inter-subject variability.

The `Study 810` dataset [18, 26] has been obtained from a multicentric clinical trial (Study 810) for testing the efficacy of paroxetine (an antidepressant drug). The dataset contains time profiles of the so-called Hamilton Depression Rating Scale (HAMD) score for several patients suffering from major depressive disorders. The HAMD is an index obtained by processing a multiple choice questionnaire, that is used to measure the severity of a patient's major depression. The patients under study were either treated with placebo or administered paroxetine at two different doses. The HAMD score was evaluated at several visits, planned for each patient at weeks 0,1,2,3,4,6,and 8. The drug is considered effective if a significant decreasing in the HAMD over the weeks is observed in the patients under treatment. Due to frequent dropouts (patients abandoning the study before its completion), several of the scores are missing, especially in the last weeks. Patients with missing scores cannot be simply removed from the dataset, since dropouts are highly correlated with non-decreasing of the HAMD score, and their removal would bias the efficacy assessment. Reconstructing the missing scores is a multi-task learning problem where each patient corresponds to a task, inputs are the time instants, and outputs are the scores. A full-rank ($p = m$) weighted OKL method has been applied, by modeling the HAMD score time profiles as piece-wise linear functions. This can be done by simply using a linear spline input kernel:

$$K(t_1, t_2) = 1 + \min\{t_1, t_2\}.$$

The overall dataset contains 2855 scores for 494 patients. Following the setup of [13], we extracted a test set containing 1012 scores, including all the scores taken after the third week for a subset of 450 randomly chosen patients. The remaining 1843 examples are further divided into a validation set containing 553 (about 30%) examples, and a training set containing the remaining 1290 examples (about 70%). The random training/validation selection is repeated 50 times, and each time the RMSE on both the validation and the test set is computed. The regularization parameter is chosen so as to minimize the average validation error. Table 1 reports the average and standard deviation of the test RMSE obtained by using OKL, a nuclear norm regularized low-rank matrix approximation method (RMF), the pooled, and the independent baselines, showing a clear advantage of the OKL multi-task approach. The

numerical rank of the best found output kernel is 7. Training over a whole regularization path took about 36 seconds in the average.

Table 1: `Study 810` dataset: best average RMSE on test data (and their standard deviation over the 50 splits)

Pooled	Independent	RMF	OKL
6.86 (0.02)	6.72(0.16)	6.66(0.4)	**5.37**(0.2)

The `PK-PD 27` dataset [29, 34] contains xenobiotics concentration time profiles for 27 human subjects, with samples taken at $\{0.5, 1, 1.5, 2, 4, 7, 12, 24\}$ hours after a drug administration. The goal is to reconstruct the continuous-time response curves over the non-negative real time semiline, a multi-task learning problem where the tasks are the concentration time profiles for each subject, and the inputs are the time instants. We apply the weighted OKL method described in this paper with full rank ($p = m$), as well as RMF, the independent, and the pooled baselines. The input kernel is chosen as

$$K(t_1, t_2) = t_1 t_2 W(h(t_1), h(t_2)),$$

where W is the cubic spline kernel

$$W(x_1, x_2) = \frac{x_1 x_2 \min\{x_1, x_2\}}{2} - \frac{(\min\{x_1, x_2\})^3}{3},$$

and h is a simple transformation designed to obtain an asymptotic decay to zero of the response curve:

$$h(t) = (1 + t)^{-1}.$$

The kernel is designed so as to incorporate the available prior knowledge about the typical shape of a concentration response curve, which has to be smooth, with zero initial condition, and asymptotically decaying to zero. In order to simulate a realistic sparse sampling scenario, we follow the approach of [31], where only 3 measurements per subject (out of the 8 available) are randomly selected for training. The remaining samples are used for test. The selection is repeated 100 times, and the results are averaged. Table 2 reports the values of average RMSE in correspondence with the best value of λ, again showing an advantage of OKL over the other methods. Figure 2 reports the average RMSE as a function of the regularization parameter for the different method. Figure 3 reports the average training time of OKL as a function of the regularization parameter. The numerical rank of the best found output kernel is 27 (therefore, in this case we get a full rank solution). Training OKL over a whole regularization path took about 1.5 seconds in the average.

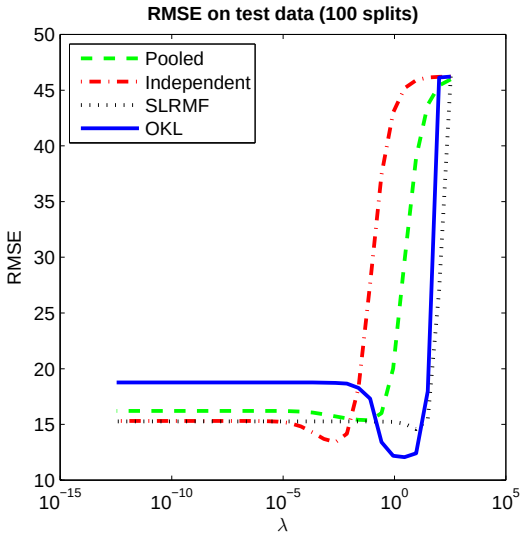

Figure 2: `PK-PD 27` dataset: average RMSE on the test data as a function of the regularization parameter λ.

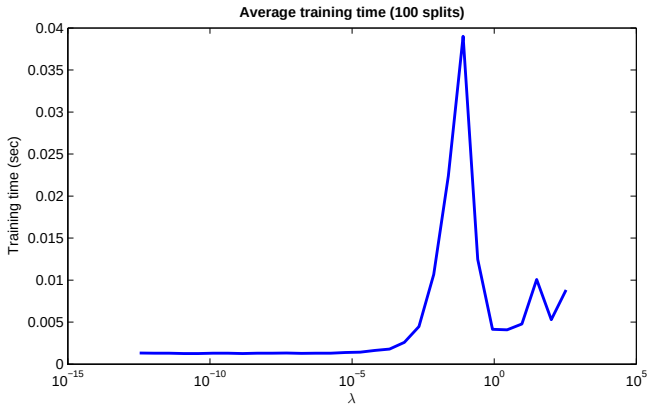

Figure 3: `PK-PD 27` dataset: average training time of OKL as a function of the regularization parameter λ. The total training time over all the values of λ (area under the curve) is about 1.5 seconds.

Table 2: `PK-PD 27` dataset: best average RMSE on test data (and their standard deviation over the 100 splits)

Pooled	Indep.	RMF	OKL
15.4(0.68)	13.4(1.55)	14.6(1.97)	**12** (0.92)

4.3 Collaborative filtering (MovieLens) data

The MovieLens datasets [1] are popular collaborative filtering benchmarks, containing collections of ratings in the range $\{1, \ldots, 5\}$ assigned by several users to a set of movies. The goal is to learn the preferences of each user for all the movies. This can be interpreted as a multi-task learning problem, where each task is the preference function of one of the users. Currently, three datasets of different sizes are available, see Table 3. In addition to the ratings, the datasets contain additional metadata associated with the movies (e.g. genre and title), the users (e.g. gender, age, occupation) or the ratings themselves (timestamp, tags).

Table 3: `MovieLens` datasets: total number of users, movies, and ratings.

Dataset	Users	Movies	Ratings
`MovieLens100K`	943	1682	10^5
`MovieLens1M`	6040	3706	10^6
`MovieLens10M`	69878	10677	10^7

The weighted OKL method described in this paper has been applied to the three MovieLens datasets. The input kernel uses the movie's metadata, while the similarity between the users is learned automatically from the data. Although the framework allows to incorporate any type of movie metadata, only the Id and the genre (present in all three datasets) have been used for simplicity. The input kernel has been designed as

$$K(x_1, x_2) = \delta_K(x_1^{id}, x_2^{id}) + \exp\left(-d_H(x_1^g, x_2^g)\right),$$

where δ_K is the Kronecker delta kernel (non-zero only when the movie Ids are equal), x_1^g, x_2^g are binary vectors encoding the genre metadata, and d_H is the normalized Hamming distance. It can be easily shown that K is a valid positive semidefinite kernel. In these problems, keeping the δ_K kernel part is important, since it allows to treat two distinct movies with identical metadata as different.

Learning performance are evaluated using both the root mean squared error (RMSE), and the normalized mean absolute error (NMAE). The latter is obtained by normalizing the mean absolute error (MAE) by a factor that depends on the range of the ratings, that is NMAE = MAE/$(r_{\max} - r_{\min})$, see

[1] http://www.grouplens.org/

e.g. [17]. For the MovieLens datasets, we have $r_{\min} = 1$ and $r_{\max} = 5$, so that NMAE $= 0.25 \cdot$ MAE. For `MovieLens100K` and `MovieLens10M`, performance are evaluated on the test sets r_a and r_b provided with the datasets. The dataset `MovieLens1M` does not come with predefined test sets, therefore we also extracted a random test set containing about the 50% of the ratings for each user, a setup adopted in [22,45]. The regularization parameter is tuned automatically by minimizing over a broad range the performance measure (RMSE or NMAE) on a validation set containing 25% of the non-test examples for each user. Only the remaining 75% are used for training. The results are summarized in Table 4. In all the experiments, we use raw data without any normalization, and the rank of the output kernel has been limited to $p = 5$. The rank constraint turned out to be active for all the optimal kernels. The multi-task OKL method systematically outperforms RMF, as well as the two single-task baselines. Other recent results on these datasets under various experimental settings can be found, for example, in [1,10,22,25,33,45].

Table 4: `MovieLens` datasets: test RMSE (above) and NMAE (below) for different dataset splittings for weighted OKL, RMF, the pooled and the independent baselines.

Test	Pooled	Independent	RMF	OKL
`MovieLens100K`				
r_a	1.0371	1.0605	1.0007	**0.9751**
	0.2087	0.2147	0.1949	**0.1906**
r_b	1.0423	1.0809	1.0296	**0.9958**
	0.2099	0.2181	0.2019	**0.1942**
50%	1.0209	1.0445	1.0300	**0.9557**
	0.2043	0.2109	0.1993	**0.1893**
`MovieLens1M`				
50%	0.9811	1.0297	0.9023	**0.8945**
	0.1961	0.2070	0.1761	**0.1752**
`MovieLens10M`				
r_a	0.9989	1.0344	1.6501	**0.9427**
	0.1970	0.2063	0.2892	**0.1806**
r_b	0.9810	1.0211	0.9296	**0.9141**
	0.1926	0.2034	0.1806	**0.1756**
50%	0.9441	0.9721	0.8627	**0.8501**
	0.1846	0.1915	0.1642	**0.1624**

5 Conclusions and future developments

Learning multiple tasks and simultaneously inferring the relationships between them is possible by using a kernel-based method that learns a decomposable multi-task kernel from multiple datasets. By employing a block coordinate descent strategy and iterative solvers for linear operator equations, it is possible to efficiently obtain a minimizer that yields good predictive performances. The method obviates the issue of manually specifying the similarities between the tasks. In addition, a systematic learning performance improvement with respect to single-task baselines and standard regularized low-rank matrix approximation can be observed on several datasets.

In the future, it would be worthwhile to extend the proposed method by using loss functions different from the the least square loss. Also, it would be interesting to extend the framework so as to exploit other types of structural knowledge about the task relationships, for instance along the lines of [4, 35].

A Proofs

Proof of Lemma 2.1

Any optimal $\mathbf{A} \in \mathbb{R}^{\ell \times p}$ for problem (4) admits a unique decomposition of the form

$$\mathbf{A} = \mathbf{CB} + \mathbf{U}, \qquad \mathbf{UB}^T = \mathbf{0}.$$

By letting $\mathbf{L} = \mathbf{BB}^T$, it follows that $\mathbf{L} \in \mathbb{S}_+^{m,p}$, and we have

$$\mathbf{KAB}^T = \mathbf{KCBB}^T = \mathbf{KCL}.$$

In addition, we have

$$\begin{aligned} \frac{\langle \mathbf{A}, \mathbf{KA} \rangle_F}{2} &= \frac{\langle \mathbf{U}, \mathbf{KU} \rangle_F}{2} + \frac{\langle \mathbf{C}^T\mathbf{KC}, \mathbf{L} \rangle_F}{2} \\ &\geq \frac{\langle \mathbf{C}^T\mathbf{KC}, \mathbf{L} \rangle_F}{2}. \end{aligned}$$

It follows that we can set $\mathbf{U} = \mathbf{0}$ and $\mathbf{A} = \mathbf{CB}$, without any loss of generality.

□

Proof of Lemma 2.2

Letting $\mathbf{\Theta} = \mathbf{CL}$, problem (2) can be rewritten as

$$\begin{aligned} &\min_{\mathbf{\Theta} \in \mathbb{R}^{\ell \times m}} \left(\frac{\|\mathbf{Y} - \mathbf{K\Theta}\|_{\mathbf{W}}^2}{2\lambda} + \min_{\mathbf{L} \in \mathbb{S}_+^{m,p}} N_{\mathbf{\Theta}}(\mathbf{L}) \right), \\ &\text{subject to} \quad \operatorname{rg}(\mathbf{\Theta}) \subseteq \operatorname{rg}(\mathbf{L}), \end{aligned}$$

where

$$N_{\boldsymbol{\Theta}}(\mathbf{L}) = \frac{\langle \boldsymbol{\Theta}^T \mathbf{K} \boldsymbol{\Theta}, \mathbf{L}^\dagger \rangle_F}{2} + \frac{\mathrm{tr}(\mathbf{L})}{2}$$

A minimizer of $N_{\boldsymbol{\Theta}}(\mathbf{L})$ can be expressed in closed-form:

$$\mathbf{L} = \left(\boldsymbol{\Theta}^T \mathbf{K} \boldsymbol{\Theta}\right)^{1/2},$$

so that

$$\min_{\mathbf{L} \in \mathbb{S}_+^{m,p}} N_{\boldsymbol{\Theta}}(\mathbf{L}) = \mathrm{tr}\left(\left(\boldsymbol{\Theta}^T \mathbf{K} \boldsymbol{\Theta}\right)^{1/2}\right).$$

All the constraints are satisfied provided that

$$\mathrm{rank}(\boldsymbol{\Theta}) = \mathrm{rank}(\mathbf{L}) \leq p.$$

□

B Matrix identities

We recall some identities involving the vectorization operator, the Kronecker and the Hadamard products, see e.g. [21]:

$$\mathrm{vec}\left(\mathbf{AXB}\right) = \left(\mathbf{B}^T \otimes \mathbf{A}\right) \mathrm{vec}(\mathbf{X}), \tag{8}$$

$$\left(\mathbf{A} \otimes \mathbf{B}\right)\left(\mathbf{C} \otimes \mathbf{D}\right) = \left(\mathbf{AC}\right) \otimes \left(\mathbf{BD}\right), \tag{9}$$

$$\mathrm{vec}\left(\mathbf{A} \odot \mathbf{B}\right) = \mathrm{diag}(\mathrm{vec}\left(\mathbf{A}\right))\mathrm{vec}\left(\mathbf{B}\right). \tag{10}$$

These identities hold whenever the sizes of the involved matrices are compatible.

References

[1] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *Journal of Machine Learning Research*, 10:803–826, June 2009.

[2] M. A. Alvarez, L. Rosasco, and N. D. Lawrence. Kernels for vector-valued functions: a review. *Foundations and Trends in Machine Learning*, (3):195–266, 2012.

[3] 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.

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

[5] A. Argyriou, C. A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 25–32. MIT Press, Cambridge, MA, USA, 2007.

[6] B. Bakker and T. Heskes. Task clustering and gating for Bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.

[7] A. Birlutiu, P. Groot, and T. Heskes. Multi-task preference learning with an application to hearing aid personalization. *Neurocomputing*, 73:1177–1185, 2010.

[8] E. Bonilla, K. M. Chai, and C. Williams. Multi-task Gaussian process prediction. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, pages 153–160. MIT Press, Cambridge, MA, 2008.

[9] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[10] D. DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 249–256, New York, NY, USA, 2006. ACM.

[11] F. Dinuzzo and K. Fukumizu. Learning low-rank output kernels. *Journal of Machine Learning Research - Proceedings Track*, 20:181–196, 2011.

[12] F. Dinuzzo, C. S. Ong, P. Gehler, and G. Pillonetto. Learning output kernels with block coordinate descent. In *Proceedings of the 28th Annual International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

[13] F. Dinuzzo, G. Pillonetto, and G. De Nicolao. Client-server multi-task learning from distributed datasets. *IEEE Transactions on Neural Networks*, 22(2):290–303, 2011.

[14] F. Dinuzzo and B. Schölkopf. The representer theorem for Hilbert spaces: a necessary and sufficient condition. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 189–196. 2012.

[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. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings of the American Control Conference*, volume 6, pages 4734–4739, Arlington, Virginia, June 2001.

[17] K. Y. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[18] R. Gomeni, A. Lavergne, and E. Merlo-Pich. Modelling placebo response in depression trials using a longitudinal model with informative dropout. *European Journal of Pharmaceutical Sciences*, 36(1):4–10, 2009.

[19] T. Heskes. Empirical Bayes for learning to learn. In *Proceedings of ICML*, pages 367–374. Morgan Kaufmann, 2000.

[20] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal Of Research Of The National Bureau Of Standards*, 49(6):409–436, 1952.

[21] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[22] M. Jaggi and M. Sulovský. A simple algorithm for nuclear norm regularized problems. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 471–478, Haifa, Israel, June 2010. Omnipress.

[23] T. Kato, H. Kashima, M. Sugiyama, and K. Asai. Conic programming for multitask learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(7):957 –968, 2010.

[24] G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, 1971.

[25] N. D. Lawrence and R. Urtasun. Non-linear matrix factorization with Gaussian processes. In Léon Bottou and Michael Littman, editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 601–608, Montreal, June 2009. Omnipress.

[26] E. Merlo-Pich and R. Gomeni. Model-based approach and signal detection theory to evaluate the performance of recruitment centers in clinical trials with antidepressant drugs. *Clinical Pharmacology and Therapeutics*, 84:378–384, September 2008.

[27] C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17:177–204, 2005.

[28] S. K. Mishra and G. Giorgi. *Invexity and optimization*. Nonconvex Optimization and Its Applications. Springer, Dordrecht, 2008.

[29] M. Neve, G. De Nicolao, and L. Marchesi. Nonparametric identification of population models via Gaussian processes. *Automatica*, 43(7):1134–1144, 2007.

[30] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345 –1359, 2010.

[31] G. Pillonetto, F. Dinuzzo, and G. De Nicolao. Bayesian online multitask learning of Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):193–205, 2010.

[32] Y. Qi, D. Liu, D. Dunson, and L. Carin. Multi-task compressive sensing with Dirichlet process priors. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 768–775. Omnipress, 2008.

[33] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *In Proceedings of the 22nd International Conference on Machine Learning*, pages 713–719. ACM, 2005.

[34] M. Rocchetti and I. Poggesi. Comparison of the Bailer and Yeh methods using real data. In L. Aarons et al., editor, *The population approach: Measuring and managing variability in response, concentration and dose*, pages 385–390, Brussels, Belgium, 1997. European Cooperation in the Field of Scientific and Technical Research, European Commission.

[35] B. Romera-Paredes, A. Argyriou, N. Berthouze, and M. Pontil. Exploiting unrelated tasks in multi-task learning. *Journal of Machine Learning Research - Proceedings Track*, 22:951–959, 2012.

[36] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.

[37] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3), July 1986.

[38] A. Saha, P. Rai, H. Daumé III, and S. Venkatasubramanian. Online learning of multiple tasks and their relationships. In *AISTATS*, Ft. Lauderdale, Florida, 2011.

[39] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. (Adaptive Computation and Machine Learning). MIT Press, 2001.

[40] A. Schwaighofer, V. Tresp, and K. Yu. Learning Gaussian process kernels via hierarchical Bayes. In *Advances in Neural Information Processing Systems*, volume 17, pages 1209–1216, 2005.

[41] V. Sima. *Algorithms for Linear-quadratic Optimization*. Marcel Dekker, New York, 1996.

[42] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *In 20th International Conference on Machine Learning*, pages 720–727. AAAI Press, 2003.

[43] N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press, Cambridge, MA, 2005.

[44] S. Thrun and L. Y. Pratt, editors. *Learning To Learn*. Kluwer Academic Publishers, Boston, MA, 1998.

[45] K Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Optimization Online*, 2009.

[46] Y. Wang, R. Khardon, and P. Protopapas. Shift-invariant grouped multi-task learning for gaussian processes. In *ECML/PKDD (3)*, pages 418–434, 2010.

[47] C. Widmer, N. Toussaint, Y. Altun, and G. Rätsch. Inferring latent task structure for multitask learning by multiple kernel learning. *BMC bioinformatics*, 11(Suppl 8):S5, 2010.

[48] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

[49] K. Yu, V. Tresp, and A. Schwaighofer. Learning Gaussian processes from multiple tasks. In *Proceedings of the 22th Annual international conference on Machine learning (ICML 2005)*, pages 1012–1019, 2005.

[50] S. Yu, V. Tresp, and K. Yu. Robust multi-task learning with t-processes. In *Proceedings of the 24th Annual international conference on Machine learning (ICML 2007)*, pages 1103–1110, New York, NY, USA, 2007. ACM.

[51] J. Zhang, Z. Ghahramani, and Y. Yang. Flexible latent variable models for multi-task learning. *Machine Learning*, 73(3):221–242, 2008.

[52] Y. Zhang and D.-Y. Yeung. A convex formulation for learning task relationships in multi-task learning. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 733–442, Catalina Island, CA, USA, 2010.