Convex Multi-Task Feature Learning

Andreas Argyriou[1], Theodoros Evgeniou[2], and Massimiliano Pontil[1]

[1] Department of Computer Science
University College London
Gower Street
London WC1E 6BT
UK
a.argyriou@cs.ucl.ac.uk
m.pontil@cs.ucl.ac.uk

[2] Technology Management and Decision Sciences
INSEAD
77300 Fontainebleau
France
theodoros.evgeniou@insead.edu

Summary. We present a method for learning sparse representations shared across multiple tasks. This method is a generalization of the well-known single-task 1-norm regularization. It is based on a novel non-convex regularizer which controls the number of learned features common across the tasks. We prove that the method is equivalent to solving a convex optimization problem for which there is an iterative algorithm which, as we prove, converges to an optimal solution. The algorithm has a simple interpretation: it alternately performs a supervised and an unsupervised step, where in the former step it learns task-specific functions and in the latter step it learns common-across-tasks sparse representations for these functions. We also provide an extension of the algorithm which learns sparse nonlinear representations using kernels. We report experiments on simulated and real data sets which demonstrate that the proposed method can both improve the performance relative to learning each task independently and lead to a few learned features common across related tasks. Our algorithm can also be used, as a special case, to simply select – not learn – a few common variables across the tasks[3].

Key words: Multi-Task Learning, Kernels, Regularization, Vector-Valued Functions.

[3] This is a journal version of the NIPS conference paper [4]. It includes new theoretical and experimental results.

1 Introduction

We study the problem of learning data representations that are common across multiple related supervised learning tasks. This is a problem of interest in many research areas. For example, in computer vision the problem of detecting a specific object in images is treated as a single supervised learning task. Images of different objects may share a number of features that are different from the pixel representation of images [21, 30, 32]. In modeling users/consumers' preferences [1, 23], there may be common product features (e.g., for cars, books, webpages, consumer electronics etc.) that are considered to be important by a number of people (we consider modeling an individual's preferences to be a single supervised learning task). These features may be different from standard, possibly many, product attributes (e.g., size, color, price) considered a priori, much like features used for perceptual maps, a technique for visualizing peoples' perception of products [1]. Learning common sparse representations across multiple tasks or datasets may also be of interest, for example, for data compression.

While the problem of learning (or selecting) sparse representations has been extensively studied either for single-task supervised learning (e.g., using 1-norm regularization) or for unsupervised learning (e.g., using principal component analysis (PCA) or independent component analysis (ICA)), there has been only limited work [3, 8, 22, 35] in the multi-task supervised learning setting. In this paper, we present a novel method for learning sparse representations common across many supervised learning tasks. In particular, we develop a novel non-convex multi-task generalization of the 1-norm regularization, known to provide sparse variable *selection* in the single-task case [14, 20, 29]. Our method *learns* a few features common across the tasks using a novel regularizer which both couples the tasks and enforces sparsity. These features are orthogonal functions in a prescribed reproducing kernel Hilbert space. The number of common features learned is controlled, as we empirically show, by a regularization parameter, much like sparsity is controlled in the case of single-task 1-norm regularization. Moreover, the method can be used, as a special case, for *variable selection*. We call "learning features" to be the estimation of new features which are functions of the input variables, like the features learned in the *un*supervised setting using methods such as PCA. We call "selecting variables" to be simply the selection of some of the input variables.

Although the novel regularized problem is non-convex, a first key result of this paper is that it is equivalent to another optimization problem which is convex. To solve the latter we use an iterative algorithm

which is similar to the one developed in [16]. The algorithm simultaneously learns *both* the features and the task functions through two alternating steps. The first step consists in independently learning the parameters of the tasks' regression or classification functions. The second step consists in learning, in an unsupervised way, a low-dimensional representation for these task parameters. A second key result of this paper is that this alternating algorithm converges to an optimal solution of the convex and the (equivalent) original non-convex problem.

Hence the main theoretical contributions of this paper are:

- We develop a novel non-convex multi-task generalization of the well-known 1-norm single task regularization that can be used to learn a few features common across multiple tasks.
- We prove that the proposed non-convex problem is equivalent to a convex one which can be solved using an iterative alternating algorithm.
- We prove that this algorithm converges to an optimal solution of the non-convex problem we initially develop.
- Finally, we develop a novel computationally efficient nonlinear generalization of the proposed method using kernels.

Furthermore, we present experiments with both simulated (where we know what the underlying features used in all tasks are) and real datasets, also using our nonlinear generalization of the proposed method. The results show that in agreement with previous work [3, 7, 8, 9, 15, 22, 27, 32, 34, 35] multi-task learning improves performance relative to single-task learning when the tasks are related. More importantly, the results confirm that when the tasks are related in the way we define in this paper, our algorithm learns a small number of features which are common across the tasks.

The paper is organized as follows. In Section 2, we develop the novel multi-task regularization method, in the spirit of 1-norm regularization for single-task learning. In Section 3, we prove that the proposed regularization method is equivalent to solving a convex optimization problem. In Section 4, we present an alternating algorithm and prove that it converges to an optimal solution. In Section 5, we extend our approach to learning features which are nonlinear functions of the input variables, using a kernel function. In Section 6, we report experiments on simulated and real data sets. Finally, in Section 7, we discuss relations of our approach with other multi-task learning methods as well as conclusions and future work.

2 Learning Sparse Multi-Task Representations

In this section, we present our formulation for multi-task feature learning. We begin by introducing our notation.

2.1 Notation

We let $\mathbb{R}$ be the set of real numbers and $\mathbb{R}_+$ ($\mathbb{R}_{++}$) the subset of non-negative (positive) ones. For every $n \in \mathbb{N}$, we let $\mathbb{N}_n := \{1, 2, \ldots, n\}$. If $w, u \in \mathbb{R}^d$, we define $\langle w, u \rangle := \sum_{i=1}^d w_i u_i$, the standard inner product in $\mathbb{R}^d$. For every $p \geq 1$, we define the p-norm of vector w as $\|w\|_p := (\sum_{i=1}^d |w_i|^p)^{\frac{1}{p}}$. In particular, $\|w\|_2 = \sqrt{\langle w, w \rangle}$. If A is a $d \times T$ matrix we denote by $a^i \in \mathbb{R}^T$ and $a_t \in \mathbb{R}^d$ the i-th row and the t-th column of A respectively. For every $r, p \geq 1$ we define the (r, p)-norm of A as $\|A\|_{r,p} := \big(\sum_{i=1}^d \|a^i\|_r^p\big)^{\frac{1}{p}}$.

We denote by $\mathbf{S}^d$ the set of $d \times d$ real symmetric matrices, by $\mathbf{S}^d_+$ ($\mathbf{S}^d_{++}$) the subset of positive semidefinite (positive definite) ones and by $\mathbf{S}^d_-$ the subset of negative semidefinite ones. If D is a $d \times d$ matrix, we define $\mathrm{trace}(D) := \sum_{i=1}^d D_{ii}$. If $w \in \mathbb{R}^d$, we denote by $\mathrm{Diag}(w)$ or $\mathrm{Diag}\,(w_i)_{i=1}^d$ the diagonal matrix having the components of vector w on the diagonal. If X is a $p \times q$ real matrix, $\mathrm{range}(X)$ denotes the set $\{x \in \mathbb{R}^p : x = Xz, \text{ for some } z \in \mathbb{R}^q\}$. Moreover, $\mathrm{null}(X)$ denotes the set $\{x \in \mathbb{R}^q : Xx = 0\}$. We let $\mathbf{O}^d$ be the set of $d \times d$ orthogonal matrices. Finally, if D is a $d \times d$ matrix we denote by D^+ its pseudoinverse. In particular, if $a \in \mathbb{R}$, $a^+ = \frac{1}{a}$ for $a \neq 0$ and $a^+ = 0$ otherwise.

2.2 Problem Formulation

We are given T supervised learning tasks. For every $t \in \mathbb{N}_T$, the corresponding task is identified by a function $f_t : \mathbb{R}^d \to \mathbb{R}$ (e.g., a regressor or margin classifier). For each task, we are given a dataset of m input/output data examples[4] $(x_{t1}, y_{t1}), \ldots, (x_{tm}, y_{tm}) \in \mathbb{R}^d \times \mathbb{R}$.

We wish to design an algorithm which, based on the data above, computes all the functions f_t, $t \in \mathbb{N}_T$. We would also like such an algorithm to be able to uncover *particular* relationships across the tasks. Specifically, we study the case that the tasks are related in the sense that they *all share a small set of features*. Formally, our hypothesis is that the functions f_t can be represented as

[4] For simplicity, we assume that each dataset contains the same number of examples; however, our discussion below can be straightforwardly extended to the case that the number of data per task varies.

$$f_t(x) = \sum_{i=1}^{I} a_{it} h_i(x), \qquad t \in \mathbb{N}_T, \tag{1}$$

where $h_i : \mathbb{R}^d \to \mathbb{R},\ i \in \mathbb{N}_I$, are the features and $a_{it} \in \mathbb{R},\ i \in \mathbb{N}_I, t \in \mathbb{N}_t$, the regression parameters.

Our goal is to learn the features h_i, the parameters a_{it} and the number of features I from the data. For simplicity, we first consider the case that the features are linear functions, that is, they are of the form $h_i(x) = \langle u_i, x \rangle$, where $u_i \in \mathbb{R}^d$. In Section 5, we will extend our formulation to the case that the h_i are elements of a reproducing kernel Hilbert space, hence in general nonlinear.

We make only one assumption about the features, namely that the vectors u_i are orthogonal. Hence, we consider only up to d of those vectors for the linear case. This assumption, which is similar in spirit to that of *unsupervised* methods such as PCA, will enable us to develop a convex learning method in the next section. We leave extensions to other cases for future research.

Thus, if we denote by $U \in \mathbf{O}_d$ the matrix whose columns are the vectors u_i, the task functions can be written as

$$f_t(x) = \sum_{i=1}^{d} a_{it} \langle u_i, x \rangle = \langle a_t, U^\top x \rangle.$$

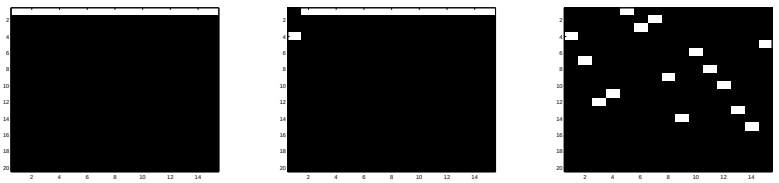
Our assumption that the tasks share a "small" set of features $I \leq d$ means that the matrix A has "many" rows which are identically equal to zero and, so, the corresponding features (columns of matrix U) will not be used by any task. Rather than learning the number of features I directly, we introduce a regularization which favors a small number of nonzero rows in the matrix A.

Specifically, we introduce the regularization error function

$$\mathcal{E}(A, U) = \sum_{t=1}^{T} \sum_{i=1}^{m} L(y_{ti}, \langle a_t, U^\top x_{ti} \rangle) + \gamma \|A\|_{2,1}^2, \tag{2}$$

where $\gamma > 0$ is a regularization parameter.[5] The first term in (2) is the average of the error across the tasks, measured according to a prescribed loss function $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ which is convex in the second argument

[5] A similar regularization function, but without matrix U, was also independently developed by [28] for the purpose of multi-task feature selection – see problem (5) below.



$\sqrt{T}$ $\qquad$ $\sqrt{T-1}+1$ $\qquad$ T

Fig. 1. Values of the $(2,1)$-norm of a matrix containing only T nonzero entries, equal to 1. When the norm increases, the level of sparsity across the rows decreases.

(for example, the square loss defined for every $y, z \in \mathbb{R}$ as $L(y,z) = (y-z)^2$). The second term is a regularization term which penalizes the $(2,1)$-norm of matrix A. It is obtained by first computing the 2-norms of the (across the tasks) rows a^i (corresponding to features i) and then the 1-norm of the vector $b(A) = (\|a^1\|_2, \ldots, \|a^d\|_2)$. The components of the vector $b(A)$ indicate how important each feature is.

The $(2,1)$-norm favors a small number of nonzero rows in the matrix A, thereby ensuring that common features will be selected across the tasks. This point is further illustrated in Figure 1, where we consider the case that the entries of matrix A take binary values and that there are only T entries which equal 1. The minimum value of the $(2,1)$-norm equals $\sqrt{T}$ and is obtained when the "1" entries are all aligned along one row. Instead, the maximum value equals T and is obtained when each "1" entry is placed in a different row (we assume here that $d \geq T$).

When the feature matrix U is prescribed and $\hat{A}$ minimizes the convex function $\mathcal{E}(\cdot, U)$ the number of nonzero components of the vector $b(\hat{A})$ will typically be nonincreasing with γ. This sparsity property can be better understood by considering the case that there is only one task, say task t. In this case, function (2) is given by

$$\sum_{i=1}^{m} L(y_{ti}, \langle a_t, U^\top x_{ti} \rangle) + \gamma \|a_t\|_1^2. \tag{3}$$

It is well known that using the 1-norm leads to sparse solutions, that is, many components of the learned vector a_t are zero, see [14] and references therein. Moreover, the number of nonzero components of a solution of problem (3) is typically a nonincreasing function of γ [26].

Since we do not simply want to select the features but also learn them, we further minimize the function $\mathcal{E}$ over U. Therefore, our ap-

proach for multi-task feature learning is to solve the optimization problem

$$\min \left\{ \mathcal{E}(A, U) : U \in \mathbf{O}^d, A \in \mathbb{R}^{d \times T} \right\}. \tag{4}$$

This method learns a low-dimensional representation which is shared across the tasks. As in the single-task case, the number of features learned will be typically nonincreasing with the regularization parameter – we will present experimental evidence of this in Section 6.

We note that solving problem (4) is challenging for two main reasons. First, it is a non-convex problem, although it is separately convex in each of the variables A and U. Secondly, the regularizer $\|A\|_{2,1}^2$ is not smooth, which makes the optimization problem more difficult. In the next two sections, we will show how to find a global optimal solution of this problem through solving an equivalent convex optimization problem. From this point on we assume that $A = 0$ does not minimize problem (4), which would be clearly a case of no practical interest.

We conclude this section by noting that when matrix U is not learned and we set $U = I_{d \times d}$, problem (4) selects a small set of variables, common across the tasks. In this case, we have the following convex optimization problem

$$\min \left\{ \sum_{t=1}^{T} \sum_{i=1}^{m} L(y_{ti}, \langle a_t, x_{ti} \rangle) + \gamma \|A\|_{2,1}^2 \ : \ A \in \mathbb{R}^{d \times T} \right\}. \tag{5}$$

We shall return to problem (5) in Sections 3 and 4 where we present an algorithm for solving it.

3 Equivalent Convex Optimization Problem

In this section, we present a central result of this paper. We show that the non-convex and nonsmooth problem (4) can be transformed into an equivalent convex problem. To this end, for every $W \in \mathbb{R}^{d \times T}$ with columns w_t and $D \in \mathbf{S}_+^d$, we define the function

$$\mathcal{R}(W, D) = \sum_{t=1}^{T} \sum_{i=1}^{m} L(y_{ti}, \langle w_t, x_{ti} \rangle) + \gamma \sum_{t=1}^{T} \langle w_t, D^+ w_t \rangle. \tag{6}$$

Under certain constraints, this objective function gives rise to a convex optimization problem, as we will show in the following. Furthermore, even though the regularizer in $\mathcal{R}$ is still nonsmooth, in Section 4 we

will show that partial minimization with respect to D has a closed-form solution and this fact leads naturally to a globally convergent optimization algorithm.

We begin with the main result of this section.

Theorem 1. *Problem (4) is equivalent to the problem*

$$\min\Big\{\mathcal{R}(W,D) \;:\; W \in \mathbb{R}^{d\times T},\; D \in \mathbf{S}_+^d,\; \operatorname{trace}(D) \le 1, \\ \operatorname{range}(W) \subseteq \operatorname{range}(D)\Big\}. \tag{7}$$

In particular, if $(\hat{A}, \hat{U})$ is an optimal solution of (4) then

$$(\hat{W}, \hat{D}) = \left(\hat{U}\hat{A}\,,\; \hat{U}\operatorname{Diag}\left(\frac{\|\hat{a}^i\|_2}{\|\hat{A}\|_{2,1}}\right)_{i=1}^{d}\hat{U}^\top\right)$$

is an optimal solution of problem (7); conversely, if $(\hat{W}, \hat{D})$ is an optimal solution of problem (7) then any $(\hat{A}, \hat{U})$, such that the columns of $\hat{U}$ form an orthonormal basis of eigenvectors of $\hat{D}$ and $\hat{A} = \hat{U}^\top\hat{W}$, is an optimal solution of problem (4).

To prove the theorem, we first introduce the following lemma which will be useful in our analysis.

Lemma 1. *For any $b = (b_1, \ldots, b_d) \in \mathbb{R}^d$ such that $b_i \neq 0, i \in \mathbb{N}_d$, we have that*

$$\min\left\{\sum_{i=1}^d \frac{b_i^2}{\lambda_i} \;:\; \lambda_i > 0,\; \sum_{i=1}^d \lambda_i \le 1\right\} = \|b\|_1^2 \tag{8}$$

and the minimizer is $\hat{\lambda}_i = \frac{|b_i|}{\|b\|_1}$, $i \in \mathbb{N}_d$.

Proof. From the Cauchy-Schwarz inequality we have that

$$\|b\|_1 = \sum_{i=1}^d \lambda_i^{\frac{1}{2}}\lambda_i^{-\frac{1}{2}}|b_i| \le \left(\sum_{i=1}^d \lambda_i\right)^{\frac{1}{2}}\left(\sum_{i=1}^d \lambda_i^{-1}b_i^2\right)^{\frac{1}{2}} \le \left(\sum_{i=1}^d \lambda_i^{-1}b_i^2\right)^{\frac{1}{2}}.$$

The minimum is attained if and only if $\frac{\lambda_i^{\frac{1}{2}}}{\lambda_i^{-\frac{1}{2}}|b_i|} = \frac{\lambda_j^{\frac{1}{2}}}{\lambda_j^{-\frac{1}{2}}|b_j|}$ for all $i, j \in \mathbb{N}_d$ and $\sum_{i=1}^d \lambda_i = 1$. Hence the minimizer satisfies $\lambda_i = \frac{|b_i|}{\|b\|_1}$. □

We can now prove Theorem 1.

Proof of Theorem 1. First suppose that (A, U) belongs to the feasible set of problem (4). Let $W = UA$ and $D = U \operatorname{Diag}\left(\frac{\|a^i\|_2}{\|A\|_{2,1}}\right)_{i=1}^d U^\top$. Then

$$\begin{aligned}\sum_{t=1}^T \langle w_t, D^+ w_t\rangle &= \operatorname{trace}(W^\top D^+ W)\\ &= \operatorname{trace}\left(A^\top U^\top U \operatorname{Diag}\left(\|A\|_{2,1}\, \|a^i\|_2^+\right)_{i=1}^d U^\top U A\right)\\ &= \|A\|_{2,1} \operatorname{trace}\left(\operatorname{Diag}\left(\|a^i\|_2^+\right)_{i=1}^d A A^\top\right)\\ &= \|A\|_{2,1} \sum_{i=1}^d \|a^i\|_2^+ \|a^i\|_2^2 \;=\; \|A\|_{2,1}^2.\end{aligned}$$

Therefore, $\mathcal{R}(W, D) = \mathcal{E}(A, U)$. Moreover, notice that W is a multiple of the submatrix of U which corresponds to the nonzero a^i and hence to the nonzero eigenvalues of D. Thus, we obtain the range constraint in problem (7). Therefore, the infimum (7) does not exceed the minimum (4). Conversely, suppose that (W, D) belongs to the feasible set of problem (7). Let $D = U \operatorname{Diag}\left(\lambda_i\right)_{i=1}^d U^\top$ be an eigendecomposition and $A = U^\top W$. Then

$$\sum_{t=1}^T \langle w_t, D^+ w_t\rangle = \operatorname{trace}\left(\operatorname{Diag}\left(\lambda_i^+\right)_{i=1}^d A A^\top\right) = \sum_{i=1}^d \lambda_i^+ \|a^i\|_2^2.$$

If $\lambda_i = 0$ for some $i \in \mathbb{N}_d$, then $u_i \in \operatorname{null}(D)$ and using the range constraint and $W = UA$ we deduce that $a^i = 0$. Consequently,

$$\sum_{i=1}^d \lambda_i^+ \|a^i\|_2^2 = \sum_{a^i \neq 0} \frac{\|a^i\|_2^2}{\lambda_i} \geq \left(\sum_{a^i \neq 0} \|a^i\|_2\right)^2 = \|A\|_{2,1}^2,$$

where we have used Lemma 1. Therefore, $\mathcal{E}(A, U) \leq \mathcal{R}(W, D)$ and the minimum (4) does not exceed the infimum (7). Because of the above application of Lemma 1, we see that the infimum (7) is attained. Finally, the condition for the minimizer in Lemma 1 yields the relationship between the optimal solutions of problems (4) and (7). □

In problem (7) we have bounded the trace of matrix D from above, because otherwise the optimal solution would be to simply set $D = \infty$ and only minimize the empirical error term in the right hand side of

equation (6). Similarly, we have imposed the range constraint to ensure that the penalty term is bounded below and away from zero. Indeed, without this constraint, it may be possible that $DW = 0$ when W does not have full rank, in which case there is a matrix D for which $\sum_{t=1}^T \langle w_t, D^+ w_t \rangle = \mathrm{trace}(W^\top D^+ W) = 0$.

In fact, the presence of the range constraint in problem (7) is due to the presence of the pseudoinverse in $\mathcal{R}$. As the following corollary shows, it is possible to eliminate this constraint and obtain the smooth regularizer $\langle w_t, D^{-1} w_t \rangle$ at the expense of not always attaining the minimum.

Corollary 1. *Problem (7) is equivalent to the problem*

$$\inf \left\{ \mathcal{R}(W, D) \ : \ W \in \mathbb{R}^{d \times T}, \ D \in \mathbf{S}^d_{++}, \ \mathrm{trace}(D) \leq 1 \right\}. \qquad (9)$$

In particular, any minimizing sequence of problem (9) converges to a minimizer of problem (7).

Proof. The theorem follows immediately from Theorem 1 and the equality of the min and inf problems in Appendix A. □

Returning to the discussion of Section 2 on the $(2,1)$-norm, we note that the rank of the optimal matrix $\hat{D}$ indicates how many common relevant features the tasks share. Indeed, it is clear from Theorem 1 that the rank of matrix $\hat{D}$ equals the number of nonzero rows of matrix $\hat{A}$.

We also note that problem (7) is similar to that in [16], where the regularizer is $\sum_{t=1}^T \langle (w_t - w_0), D^+ (w_t - w_0) \rangle$ instead of $\sum_{t=1}^T \langle w_t, D^+ w_t \rangle$ – that is, in our formulation we do not penalize deviations from a common "mean" w_0.

The next proposition establishes that problem (7) is convex.

Proposition 1. *Problem (7) is a convex optimization problem.*

Proof. Let us define the function $f : \mathbb{R}^d \times \mathbf{S}^d \to \mathbb{R} \cup \{+\infty\}$ as

$$f(w, D) := \begin{cases} w^\top D^+ w & \text{if } D \in \mathbf{S}^d_+ \text{ and } w \in \mathrm{range}(D) \\ +\infty & \text{otherwise} \end{cases}.$$

It clearly suffices to show that f is convex since L is convex in the second argument and the trace constraint on D is convex. To prove that f is convex, we show that f can be expressed as a supremum of convex functions, specifically that

$$f(w, D) = \sup\{w^\top v + \operatorname{trace}(ED) \ : \ E \in \mathbf{S}^d, v \in \mathbb{R}^d, 4E + vv^\top \in \mathbf{S}^d_-\}, \\ w \in \mathbb{R}^d, D \in \mathbf{S}^d.$$

To prove this equation, we first consider the case $D \notin \mathbf{S}^d_+$. We let u be an eigenvector of D corresponding to a negative eigenvalue and set $E = auu^\top, a \leq 0, v = 0$ to obtain that the supremum on the right equals $+\infty$. Next, we consider the case that $w \notin \operatorname{range}(D)$. We can write $w = Dz + n$, where $z, n \in \mathbb{R}^d, n \neq 0$ and $n \in \operatorname{null}(D)$. Thus,

$$w^\top v + \operatorname{trace}(ED) = z^\top Dv + n^\top v + \operatorname{trace}(ED)$$

and setting $E = -\frac{1}{4}vv^\top, v = an, a \in \mathbb{R}_+$, we obtain $+\infty$ as the supremum. Finally, we assume that $D \in \mathbf{S}^d_+$ and $w \in \operatorname{range}(D)$. Combining with $E + \frac{1}{4}vv^\top \in \mathbf{S}^d_-$ we get that $\operatorname{trace}((E + \frac{1}{4}vv^\top)D) \leq 0$. Therefore

$$w^\top v + \operatorname{trace}(ED) \leq w^\top v - \frac{1}{4}v^\top Dv$$

and the expression on the right is maximized for $w = \frac{1}{2}Dv$ and obtains the maximal value

$$\frac{1}{2}v^\top Dv - \frac{1}{4}v^\top Dv = \frac{1}{4}v^\top Dv = \frac{1}{4}v^\top DD^+Dv = w^\top D^+w.$$

□

We conclude this section by noting that when matrix D in problem (7) is additionally constrained to be diagonal, we obtain a problem equivalent to (5). Formally, we have the following corollary.

Corollary 2. *Problem (5) is equivalent to the problem*

$$\min\Bigg\{\mathcal{R}(W, \operatorname{Diag}(\lambda)) \ : \ W \in \mathbb{R}^{d\times T}, \ \lambda \in \mathbb{R}^d_+, \ \sum_{i=1}^d \lambda_i \leq 1, \\ \lambda_i \neq 0 \ \textit{whenever} \ w^i \neq 0\Bigg\} \qquad (10)$$

and the optimal $\hat{\lambda}$ is given by

$$\hat{\lambda}_i = \frac{\|\hat{w}^i\|_2}{\|\hat{W}\|_{2,1}}, \qquad i \in \mathbb{N}_d. \qquad (11)$$

4 Alternating Minimization Algorithm

In this section, we discuss an algorithm for solving the convex optimization problem (7) which, as we prove, converges to an optimal solution. The proof of convergence is a key technical result of this paper. By Theorem 1 above this algorithm will also provide us with a solution for the multi-task feature learning problem (4).

The algorithm is a technical modification of the one developed in [16], where a variation of problem (7) was solved by alternately minimizing function $\mathcal{R}$ with respect to D and W. It minimizes a perturbation of the objective function (6) with a small parameter $\varepsilon > 0$. This allows us to prove convergence to an optimal solution of problem (7) by letting $\varepsilon \to 0$ as shown below. We also have observed that, in practice, alternating minimization of the unperturbed objective function (6) converges to an optimal solution of (7), although this may not be true in theory.

The algorithm we now present minimizes the function $\mathcal{R}_\varepsilon : \mathbb{R}^{d\times T} \times \mathbf{S}^d_{++} \to \mathbb{R}$, given by

$$\mathcal{R}_\varepsilon(W, D) = \sum_{t=1}^{T}\sum_{i=1}^{m} L(y_{ti}, \langle w_t, x_{ti}\rangle) + \gamma \operatorname{trace}(D^{-1}(WW^\top + \varepsilon I_d)),$$

which keeps D nonsingular. The regularizer in this function is smooth and, as we show in Appendix B (Proposition 3), $\mathcal{R}_\varepsilon$ has a unique minimizer.

We now describe the two steps of Algorithm 1 for minimizing $\mathcal{R}_\varepsilon$. In the first step, we keep D fixed and minimize over W, that is we solve the problem

$$\min\left\{\sum_{t=1}^{T}\sum_{i=1}^{m} L(y_{ti}, \langle w_t, x_{ti}\rangle) + \gamma \sum_{t=1}^{T} \langle w_t, D^{-1}w_t\rangle \;:\; W \in \mathbb{R}^{d\times T}\right\},$$

where, recall, w_t are the columns of matrix W. This minimization can be carried out independently across the tasks since the regularizer decouples when D is fixed. More specifically, introducing new variables for $D^{-\frac{1}{2}}w_t$ yields a standard 2-norm regularization problem for each task with the same kernel $K(x, x') = x^\top Dx'$.

In the second step, we keep matrix W fixed, and minimize $\mathcal{R}_\varepsilon$ with respect to D. To this end, we solve the problem

$$\min\left\{\sum_{t=1}^{T} \langle w_t, D^{-1}w_t\rangle + \varepsilon \operatorname{trace}(D^{-1}) : D \in \mathbf{S}^d_{++}, \operatorname{trace}(D) \le 1\right\}. \quad (12)$$

Algorithm 1 (*Multi-Task Feature Learning*)

Input: training sets $\{(x_{ti}, y_{ti})\}_{i=1}^{m}, t \in \mathbb{N}_T$

Parameters: regularization parameter γ, tolerances ε, tol

Output: $d \times d$ matrix D, $d \times T$ regression matrix $W = [w_1, \ldots, w_T]$

Initialization: set $D = \frac{I_d}{d}$

while $\|W - W_{prev}\| > tol$ **do**

  for $t = 1, \ldots, T$ **do**

    compute $w_t = \operatorname{argmin} \left\{ \sum_{i=1}^{m} L(y_{ti}, \langle w, x_{ti} \rangle) + \gamma \langle w, D^{-1} w \rangle : w \in \mathbb{R}^d \right\}$

  end for

  set $D = \frac{(WW^\top + \varepsilon I_d)^{\frac{1}{2}}}{\operatorname{trace}(WW^\top + \varepsilon I_d)^{\frac{1}{2}}}$

end while

The term $\operatorname{trace}(D^{-1})$ keeps the D-iterates of the algorithm at a certain distance from the boundary of $\mathbf{S}_+^d$ and plays a role similar to that of the barrier used in interior-point methods. In Appendix A, we provide a proof that the optimal solution of problem (12) is given by

$$D_\varepsilon(W) = \frac{(WW^\top + \varepsilon I_d)^{\frac{1}{2}}}{\operatorname{trace}(WW^\top + \varepsilon I_d)^{\frac{1}{2}}} \tag{13}$$

and the optimal value equals $\left(\operatorname{trace}(WW^\top + \varepsilon I_d)^{\frac{1}{2}}\right)^2$. In the same appendix, we also show that for $\varepsilon = 0$, equation (13) gives the minimizer of the function $\mathcal{R}(W, \cdot)$ subject to the constraints in problem (7).

Algorithm 1 can be interpreted as alternately performing a supervised and an unsupervised step. In the former step we learn task-specific functions (namely the vectors w_t) using a common representation across the tasks. This is because D encapsulates the features u_i and thus the feature representation is kept fixed. In the unsupervised step, the regression functions are fixed and we learn the common representation. In effect, the $(2, 1)$-norm criterion favors the most concise representation which "models" the regression functions through $W = UA$.

We now present some convergence properties of Algorithm 1. We state here only the main results and postpone their proofs to Appendix B. Let us denote the value of W at the n-th iteration by $W^{(n)}$. First, we observe that, by construction, the values of the objective are non-increasing, that is,

$$\mathcal{R}_\varepsilon(W^{(n+1)}, D_\varepsilon(W^{(n+1)})) \leq \min\{\mathcal{R}_\varepsilon(V, D_\varepsilon(W^{(n)})) : V \in \mathbb{R}^{d\times T}\} \leq \mathcal{R}_\varepsilon(W^{(n)}, D_\varepsilon(W^{(n)}))\,.$$

These values are also bounded, since L is bounded from below, and thus the iterates of the objective function converge. Moreover, the iterates $W^{(n)}$ also converge as stated in the following theorem.

Theorem 2. *For every $\varepsilon > 0$, the sequence $\{(W^{(n)}, D_\varepsilon(W^{(n)}) : n \in \mathbb{N}\}$ converges to the minimizer of $\mathcal{R}_\varepsilon$ subject to the constraints in (12).*

Algorithm 1 minimizes the perturbed objective $\mathcal{R}_\varepsilon$. In order to obtain a minimizer of the original objective $\mathcal{R}$, we can employ a modified algorithm in which ε is reduced towards zero whenever $W^{(n)}$ has stabilized near a value. Our next theorem shows that the limiting points of such an algorithm are optimal.

Theorem 3. *Consider a sequence of functions $\{\mathcal{R}_{\varepsilon_\ell} : \ell \in \mathbb{N}\}$ such that $\varepsilon_\ell \to 0$ as $\ell \to \infty$. Any limiting point of the minimizers of this sequence (subject to the constraints in (12)) is an optimal solution to (7).*

We proceed with a few remarks on an alternative formulation for problem (7). By substituting equation (13) with $\varepsilon = 0$ in equation (6) for $\mathcal{R}$, we obtain a regularization problem in W only, which is given by

$$\min\left\{\sum_{t=1}^{T}\sum_{i=1}^{m} L(y_{ti}, \langle w_t, x_{ti}\rangle) + \gamma\|W\|_{\mathrm{tr}}^2 \;:\; W \in \mathbb{R}^{d\times T}\right\},$$

where we have defined $\|W\|_{\mathrm{tr}} := \mathrm{trace}(WW^\top)^{\frac{1}{2}}$.

The expression $\|W\|_{\mathrm{tr}}$ in the regularizer is called the *trace norm*. It can also be expressed as the sum of the singular values of W. As shown in [17], the trace norm is the convex envelope of $\mathrm{rank}(W)$ in the unit ball, which gives another interpretation of the relationship between the rank and γ in our experiments. Solving this problem directly is not easy, since the trace norm is nonsmooth. Thus, we have opted for the alternating minimization strategy of Algorithm 1, which is simple to implement and natural to interpret. We should note here that a similar problem has been studied in [31] for the particular case of an SVM loss function. It was shown there that the optimization problem can be solved through an equivalent semidefinite programming problem. We will further discuss relations with that work as well as other work in Section 7.

We conclude this section by noting by Corollary 2 that we can make a simple modification to Algorithm 1 so that it can be used to solve the

variable selection problem (5). Specifically, we modify the computation of the matrix D (penultimate line in Algorithm 1) as $D = Diag(\lambda)$, where the vector $\lambda = (\lambda_1, \ldots, \lambda_d)$ is computed using equation (11).

5 Learning Nonlinear Features

In this section, we consider the case that the features are associated to a kernel and hence they are in general nonlinear functions of the input variables. First, in Section 5.1 we use a representer theorem for an optimal solution of problem (7), in order to obtain an optimization problem of bounded dimensionality. Then, in Section 5.2 we show how to solve this problem using a novel algorithm which is a variation of Algorithm 1. This algorithm differs from the nonlinear one of [16], being simpler to implement and relying on the representer theorem of Section 5.1.

5.1 A Representer Theorem

We begin by restating our optimization problem when the functions learned belong to a *reproducing kernel Hilbert space*, see e.g. [27, 33] and references therein. Formally, we now wish to learn T regression functions $f_t, t \in \mathbb{N}_T$ of the form

$$f_t(x) = \langle a_t, U^\top \varphi(x) \rangle = \langle w_t, \varphi(x) \rangle \,, \quad x \in \mathbb{R}^d,$$

where $\varphi : \mathbb{R}^d \to \mathbb{R}^M$ denotes the feature map. This map will, in general, be nonlinear and its dimensionality M may be large. In fact, the theoretical and algorithmic results which follow apply to the case of an infinite dimensionality as well. As typical, we assume that the kernel function $K(x, x') = \langle \varphi(x), \varphi(x') \rangle$ is given. As before, in the following we will use the subscript notation for the columns of a matrix, for example w_t denotes the t-th column of matrix W.

We begin by recalling that Appendix A applied to problem (7) leads to a problem in W with the trace norm in the regularizer. Modifying slightly to account for the feature map, we obtain the problem

$$\min \left\{ \sum_{t=1}^{T} \sum_{i=1}^{m} L(y_{ti}, \langle w_t, \varphi(x_{ti}) \rangle) + \gamma \|W\|_{\mathrm{tr}}^2 \;:\; W \in \mathbb{R}^{d \times T} \right\}. \qquad (14)$$

This problem can be viewed as a generalization of the standard 2-norm regularization problem. Indeed, in the case $t = 1$ the trace norm

$\|W\|_{\rm tr}$ is simply equal to $\|w_1\|_2$. In this case, it is well known that an optimal solution $w \in \mathbb{R}^d$ of such a problem is in the span of the training data, that is

$$w = \sum_{i=1}^{m} c_i \, \varphi(x_i)$$

for some $c_i \in \mathbb{R}$, $i = 1, \ldots, m$. This result is known as the representer theorem – see e.g., [33]. We now extend this result to the more general form (14). A related representer theorem was first proved in [2]. Here, we give an alternative proof connected to the theory of operator monotone functions. We also note that this theorem is being extended to a general family of spectral norms in [6].

Theorem 4. *If W is an optimal solution of problem (14) then for every $t \in \mathbb{N}_T$ there exists a vector $c_t \in \mathbb{R}^{mT}$ such that*

$$w_t = \sum_{s=1}^{T} \sum_{i=1}^{m} (c_t)_{si} \varphi(x_{si}). \tag{15}$$

Proof. Let $\mathcal{L} = \text{span}\{\varphi(x_{si}) : s \in \mathbb{N}_T, i \in \mathbb{N}_m\}$. We can write $w_t = p_t + n_t\,,\ t \in \mathbb{N}_T$ where $p_t \in \mathcal{L}$ and $n_t \in \mathcal{L}^\perp$. Hence $W = P + N$, where P is the matrix with columns p_t and N the matrix with columns n_t. Moreover we have that $P^\top N = 0$. From Lemma 3 in Appendix C, we obtain that $\|W\|_{\rm tr} \geq \|P\|_{\rm tr}$. We also have that $\langle w_t, \varphi(x_{ti})\rangle = \langle p_t, \varphi(x_{ti})\rangle$. Thus, we conclude that whenever W is optimal, N must be zero. □

An alternative way to write (15), using matrix notation, is to express W as a multiple of the input matrix. The latter is the matrix $\Phi \in \mathbb{R}^{M \times mT}$ whose (t,i)-th column is the vector $\varphi(x_{ti}) \in \mathbb{R}^M$, $t \in \mathbb{N}_T, i \in \mathbb{N}_m$. Hence, denoting with $C \in \mathbb{R}^{mT \times T}$ the matrix with columns c_t, equation (15) becomes

$$W = \Phi \, C. \tag{16}$$

We now apply Theorem 4 to problem (14) in order to obtain an equivalent optimization problem in a number of variables independent of M. This theorem implies that we can restrict the feasible set of (14) only to matrices $W \in \mathbb{R}^{d \times T}$ satisfying (16) for some $C \in \mathbb{R}^{mT \times T}$.

Let $\mathcal{L} = \text{span}\{\varphi(x_{ti}) : t \in \mathbb{N}_T, i \in \mathbb{N}_m\}$ as above and let δ its dimensionality. In order to exploit the *unitary invariance* of the trace norm, we consider a matrix $V \in \mathbb{R}^{M \times \delta}$ whose columns form an orthogonal basis of $\mathcal{L}$. Equation (16) implies that there is a matrix $\Theta \in \mathbb{R}^{\delta \times T}$ such that

$$W = V\Theta\,. \tag{17}$$

Substituting equation (17) in the objective of (14) yields the objective function

$$\sum_{t=1}^{T}\sum_{i=1}^{m} L(y_{ti}, \langle V\vartheta_t, \varphi(x_{ti})\rangle) + \gamma\Big(\mathrm{trace}(V\Theta\Theta^\top V^\top)^{\frac{1}{2}}\Big)^2 =$$
$$\sum_{t=1}^{T}\sum_{i=1}^{m} L(y_{ti}, \langle \vartheta_t, V^\top\varphi(x_{ti})\rangle) + \gamma\Big(\mathrm{trace}(\Theta\Theta^\top)^{\frac{1}{2}}\Big)^2 =$$
$$\sum_{t=1}^{T}\sum_{i=1}^{m} L(y_{ti}, \langle \vartheta_t, V^\top\varphi(x_{ti})\rangle) + \gamma\|\Theta\|_{\mathrm{tr}}^2\,.$$

Thus, we obtain the following proposition.

Proposition 2. *Problem (14) is equivalent to*

$$\min\left\{\sum_{t=1}^{T}\sum_{i=1}^{m} L(y_{ti}, \langle \vartheta_t, z_{ti}\rangle) + \gamma\|\Theta\|_{\mathrm{tr}}^2 \;:\; \Theta \in \mathbb{R}^{\delta\times T}\right\}, \tag{18}$$

where

$$z_{ti} = V^\top\varphi(x_{ti})\,,\; t \in \mathbb{N}_T, i \in \mathbb{N}_m\,. \tag{19}$$

Moreover, there is an one-to-one correspondence between optimal solutions of (14) and those of (18), given by $W = V\Theta$.

Problem (18) is a problem in δT variables, where $\delta T \leq mT^2$, and hence it can be tractable regardless of the dimensionality M of the original features.

5.2 An Alternating Algorithm for Nonlinear Features

We now address how to solve problem (18) by applying the same strategy as in Algorithm 1. It is clear from the discussion in Section 4 that (18) can be solved with an alternating minimization algorithm, which we present as Algorithm 2.

In the initialization step, Algorithm 2 computes a matrix $R \in \mathbb{R}^{\delta\times\delta}$ which relates the orthogonal basis V of $\mathcal{L}$ with a basis $\{\varphi(x_{t_\nu i_\nu})\,,\, \nu \in \mathbb{N}_\delta, t_\nu \in \mathbb{N}_T, i_\nu \in \mathbb{N}_m\}$ from the inputs. We can write this relation as

$$V = \widetilde{\Phi}R \tag{20}$$

where $\widetilde{\Phi} \in \mathbb{R}^{M\times\delta}$ is the matrix whose ν-th column is the vector $\varphi(x_{t_\nu i_\nu})$.

Algorithm 2 (*Multi-Task Feature Learning with Kernels*)

Input: training sets $\{(x_{ti}, y_{ti})\}_{i=1}^m, t \in \mathbb{N}_T$

Parameters: regularization parameter γ, tolerances ε, tol

Output: $\delta \times T$ coefficient matrix $B = [b_1, \ldots, b_T]$, indices $\{(t_\nu, i_\nu) : \nu \in \mathbb{N}_\delta\} \subseteq \mathbb{N}_T \times \mathbb{N}_m$

Initialization: using only the kernel values, find a matrix $R \in \mathbb{R}^{\delta \times \delta}$ and indices $\{(t_\nu, i_\nu)\}$ such that $\left\{\sum_{\nu=1}^{\delta} \varphi(x_{t_\nu i_\nu}) r_{\nu\mu} : \mu \in \mathbb{N}_\delta\right\}$ form an orthogonal basis for the features on the training data

compute the modified inputs $z_{ti} = R^\top \left(K(x_{t_\nu i_\nu}, x_{ti})\right)_{\nu=1}^{\delta}, t \in \mathbb{N}_T, i \in \mathbb{N}_m$

set $\Delta = \frac{I_\delta}{\delta}$

while $\|\Theta - \Theta_{prev}\| > tol$ **do**

  for $t = 1, \ldots, T$ **do**

    compute $\vartheta_t = \operatorname{argmin}\left\{\sum_{i=1}^m L(y_{ti}, \langle \vartheta, z_{ti}\rangle) + \gamma \langle \vartheta, \Delta^{-1}\vartheta\rangle : \vartheta \in \mathbb{R}^\delta\right\}$

  end for

  set $\Delta = \frac{(\Theta\Theta^\top + \varepsilon I_\delta)^{\frac{1}{2}}}{\operatorname{trace}(\Theta\Theta^\top + \varepsilon I_\delta)^{\frac{1}{2}}}$

end while

return $B = R\Theta$ and $\{(t_\nu, i_\nu) : \nu \in \mathbb{N}_\delta\}$

To compute R using only Gram matrix entries, one approach is Gram-Schmidt orthogonalization. At each step, we consider an input x_{ti} and determine whether it enlarges the current subspace or not by computing kernel values with the inputs forming the subspace. However, Gram-Schmidt orthogonalization is sensitive to round-off errors, which can affect the accuracy of the solution ([19, Sec. 5.2.8]). A more stable but computationally less appealing approach is to compute an eigendecomposition of the $mT \times mT$ Gram matrix $\Phi^\top\Phi$. A middle strategy may be preferable, namely, randomly select a reasonably large number of inputs and compute an eigendecomposition of their Gram matrix; obtain the basis coefficients; complete the vector space with a Gram-Schmidt procedure.

After the computation of R, the algorithm computes the inputs in (19), which by (20) equal $z_{ti} = V^\top\varphi(x_{ti}) = R^\top\widetilde{\Phi}^\top\varphi(x_{ti}) = R^\top\widetilde{K}(x_{ti})$. We use $\widetilde{K}(x)$ to denote the δ-vector with entries $K(x_{t_\nu i_\nu}, x)$, $\nu \in \mathbb{N}_\delta$. In the main loop, the Θ-step solves T independent regularization problems

using the Gram entries $z_{ti}^\top \Delta z_{tj}\,, i,j \in \mathbb{N}_m, t \in \mathbb{N}_T$. The Δ-step is the computation of a $\delta \times \delta$ matrix square root.

Finally, the output of the algorithm, matrix B, satisfies that

$$W = \widetilde{\Phi} B \tag{21}$$

by combining equations (17) and (20). Thus, a prediction on a new input $x \in \mathbb{R}^d$ is computed as

$$f_t(x) = \langle w_t, \varphi(x) \rangle = \langle b_t, \widetilde{\Phi}^\top x \rangle = \langle b_t, \widetilde{K}(x) \rangle, \qquad t \in \mathbb{N}_T.$$

One can also express the learned features in terms of the input basis $\{\varphi(x_{t_\nu i_\nu}) : \nu \in \mathbb{N}_\delta\}$. To do this, we need to compute an eigendecomposition of $B^\top \widetilde{K} B$, where $\widetilde{K} = \widetilde{\Phi}^\top \widetilde{\Phi}$ is the kernel matrix on the basis points. Indeed, we know that $W = \widetilde{U} \Sigma Q^\top$, where $\widetilde{U} \in \mathbb{R}^{M \times \delta'}, \Sigma \in \mathbf{S}_{++}^{\delta'}$ diagonal, $Q \in \mathbb{R}^{T \times \delta'}$ orthogonal, $\delta' \leq \delta$, and the columns of $\widetilde{U}$ are the significant features learned. From this and (21) we obtain that

$$\widetilde{U} = \widetilde{\Phi} B Q \Sigma^{-1} \tag{22}$$

and Σ, Q can be computed from

$$Q \Sigma^2 Q^\top = W^\top W = B^\top \widetilde{\Phi}^\top \widetilde{\Phi} B\,.$$

Finally, the coefficient matrix A can be computed from $W = UA$, (21) and (22), yielding

$$A = \begin{pmatrix} \Sigma Q^\top \\ 0 \end{pmatrix}\,.$$

The computational cost of Algorithm 2 depends mainly on the dimensionality δ of $\mathcal{L}$. Note that kernel evaluations using K appear only in the initialization step. There are $O(\delta m T)$ kernel computations during the orthogonalization process and $O(\delta^2 m T)$ additional operations for computing the vectors z_{ti}. However, these costs are incurred only once. Within each iteration, the cost of computing the Gram matrices in the Θ-step is $O(\delta^2 m^2 T)$ and the cost of each learning problem depends on δ. The matrix square root computation in the Δ-step involves $O(\delta^3)$ operations. Thus, for most commonly used loss functions, it is expected that the overall cost of the algorithm is $O(\delta^2 m^2 T)$ operations. In particular, in several cases of interest, such as when all tasks share the same training inputs, δ can be small and Algorithm 2 can be particularly efficient. We would also like to note here that experimental

trials, which are reported in Section 6, showed that usually between 20 and 100 iterations were sufficient for Algorithms 1 and 2 to converge.

As a final remark, we note that an algorithm similar to Algorithm 2 would not work for variable selection. This is true because Theorem 4 does not apply to the optimization problem (10), where matrix D is constrained to be diagonal. Thus, variable selection – and in particular 1-norm regularization – with kernels remains an open problem. Nevertheless, this fact does not seem to be significant in the multi-task context of this paper. As we will discuss in Section 6, variable selection was outperformed by feature learning in our experimental trials. However, variable selection could still be important in a different setting, when a set including some "good" features is a priori given and the question is how to select exactly these features.

6 Experiments

In this section, we present numerical experiments with our methods, both the linear Algorithm 1 and the nonlinear Algorithm 2, on synthetic and real data sets. In all experiments, we used the square loss function and automatically tuned the regularization parameter γ with cross-validation.

6.1 Synthetic Data

We first used synthetic data to test the ability of the algorithms to learn the common across tasks features. This setting makes it possible to evaluate the quality of the features learned, as in this case we know what the common across tasks features are.

Linear Synthetic Data

We consider the case of regression and a number of up to $T = 200$ tasks. Each of the w_t parameters of these tasks was selected from a 5-dimensional Gaussian distribution with zero mean and covariance $Cov = \mathrm{Diag}(1, 0.64, 0.49, 0.36, 0.25)$. To these 5-dimensional w_t's we kept adding up to 20 irrelevant dimensions which are exactly zero. The training and test data were generated uniformly from $[0, 1]^d$ where d ranged from 5 to 25. The outputs y_{ti} were computed from the w_t and x_{ti} as $y_{ti} = \langle w_t, x_{ti} \rangle + \vartheta$, where ϑ is zero-mean Gaussian noise with standard deviation equal to 0.1. Thus, the true features $\langle u_i, x \rangle$ we wish to learn were in this case just the input variables. However, we did not a priori

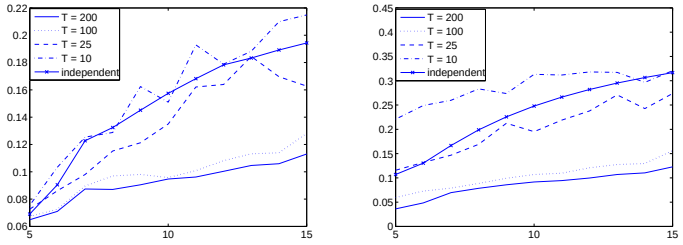


Fig. 2. Linear synthetic data. Left: test error versus the number of variables, as the number of tasks simultaneously learned changes. Right: Frobenius norm of the difference of the learned and actual matrices D versus the number of variables, as the number of tasks simultaneously learned changes. This is a measure of the quality of the learned features.

assume this and we let our algorithm learn – not select – the features. That is, we used Algorithm 1 to learn the features, not its variant which performs variable selection (see our discussion at the end of Section 4). The desired result is a feature matrix U which is close to the identity matrix (on 5 columns) and a matrix D approximately proportional to the covariance Cov used to generate the task parameters (on a 5×5 principal submatrix).

We generated 5 and 20 examples per task for training and testing, respectively. To test the effect of the number of jointly learned tasks on the test performance and (more importantly) on the quality of the features learned, we tried our methods with $T = 10, 25, 100, 200$ tasks. For $T = 10, 25$ and 100, we averaged the performance metrics over randomly selected subsets of the 200 tasks, so that our estimates have comparable variance. We also estimated each of the 200 tasks independently using standard ridge regressions.

We present, in Figure 2, the impact of the number of tasks simultaneously learned on the test performance as well as the quality of the features learned, as the number of irrelevant variables increases. First, as the left plot shows, in agreement with past empirical and theoretical evidence – see e.g., [8] – learning multiple tasks together significantly improves on learning the tasks independently, as the tasks are indeed related in this case. Moreover, performance improves as the number of tasks increases. More important, this improvement increases with the number of irrelevant variables.

The plot on the right of Figure 2 is the most relevant one for our purposes. It shows the distance of the learned features from the actual

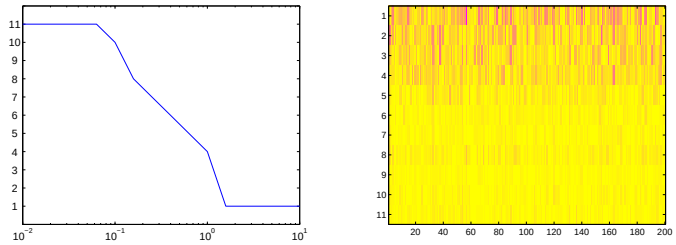


Fig. 3. Linear synthetic data. Left: number of features learned versus the regularization parameter γ (see text for description). Right: matrix A learned, indicating the importance of the learned features – the first 5 rows correspond to the true features (see text). The color scale ranges from yellow (low values) to purple (high values).

ones used to generate the data. More specifically, we depict the Frobenius norm of the difference of the learned 5×5 principal submatrix of D and the actual Cov matrix (normalized to have trace 1). We observe that adding more tasks leads to better estimates of the underlying features, a key contribution of this paper. Moreover, like for the test performance, the relative (as the number of tasks increases) quality of the features learned increases with the number of irrelevant variables. Similar results were obtained by plotting the residual of the learned U from the actual one, which is the identity matrix in this case.

We also tested the effect of the regularization parameter γ on the number of features learned (as measured by $\mathrm{rank}(D)$) for 6 irrelevant variables. We show the results on the left plot of Figure 3. As expected, the number of features learned decreases with γ. Finally, the right plot in Figure 3 shows the absolute values of the elements of matrix A learned using the parameter γ selected by leave-one-out cross-validation. This is the resulting matrix for 6 irrelevant variables and all 200 simultaneously learned tasks. This plot indicates that our algorithm learns a matrix A with the expected structure: there are only five important features. The (normalized) 2-norms of the corresponding rows are $0.31, 0.21, 0.12, 0.10$ and 0.09 respectively, while the true values (diagonal elements of Cov scaled to have trace 1) are $0.36, 0.23, 0.18, 0.13$ and 0.09 respectively.

Nonlinear Synthetic Data

Next, we tested whether our nonlinear method (Algorithm 2) can outperform the linear one when the true underlying features are nonlin-

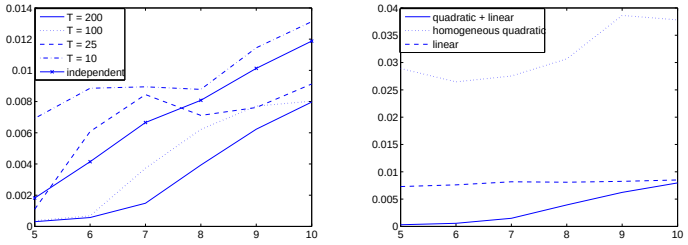


Fig. 4. Nonlinear synthetic data. Left: test error versus number of variables as the number of simultaneously learned tasks changes, using a quadratic + linear kernel. Right: test error versus number of variables for 200 tasks, using three different kernels (see text).

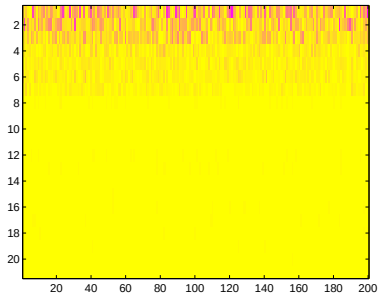


Fig. 5. Matrix A learned in the nonlinear synthetic data experiment. The first 7 rows correspond to the true features (see text).

ear. For this purpose, we created a new synthetic data set in the same way as before, but this time we used a feature map $\phi : \mathbb{R}^5 \to \mathbb{R}^7$. More specifically, we have 6 relevant linear and quadratic features and a bias term: $\varphi(x) = \left(x_1^2, x_4^2, x_1x_2, x_3x_5, x_2, x_4, 1\right)$. That is, the outputs were generated as $y_{ti} = \langle w_t, \varphi(x_{ti})\rangle + \vartheta$, with the task parameters w_t corresponding to the features above selected from a 7-dimensional Gaussian distribution with zero mean and covariance equal to $\mathrm{Diag}(0.5, 0.25, 0.1, 0.05, 0.15, 0.1, 0.15)$. All other components of each w_t were 0. The training and test sets were selected randomly from $[0, 1]^d$ with d ranging from 5 to 10 and each set contained 20 examples per task. Since there are more task parameters to learn than in the linear case, we used more data per task for training in this simulation.

We report the results in Figure 4. As for the linear case, the left plot in the figure shows the test performance versus the number of tasks simultaneously learned, as the number of irrelevant variables increases. Note that the dimensionality of the feature map scales quadratically with the input dimensionality shown on the x-axis of the plot. The kernel used for this plot was $K_{ql}(x, x') := (x^\top x' + 1)^2$. This is a "good" kernel for this data set because the corresponding feature map includes all of the monomials of φ. The results are qualitatively similar to those in the linear case. Learning multiple tasks together improves on learning the tasks independently. In this experiment, a certain number of tasks (greater than 10) is required for improvement over independent learning.

Next, we tested the effects of using the "wrong" kernel, as well as the difference between using a nonlinear kernel versus using a linear one. These are the most relevant to our purpose tests for this experiment. We used three different kernels. One is the quadratic + linear kernel defined above, the second is $K_q(x, x') := (x^\top x')^2$ and the third $K_l(x, x') := x^\top x' + 1$. The results are shown on the right plot of Figure 4. First, notice that since the underlying feature map involves both quadratic and linear features, it would be expected that the first kernel gives the best results, and this is indeed true. Second, notice that using a linear kernel (and the linear Algorithm 1) leads to poorer test performance. Thus, our nonlinear Algorithm 2 can exploit the higher approximating power of the most complex kernel in order to obtain better performance.

Finally, Figure 5 contains the plot of matrix A learned for this experiment using kernel K_{ql}, no irrelevant variables and all 200 tasks simultaneously, as we did in Figure 3 for the linear case. Similarly to the linear case, our method learns a matrix A with the desired structure: only the first 7 rows have large entries. Note that the first 7 rows correspond to the monomials of φ, while the remaining 14 rows correspond to the other monomial components of the feature map associated with the kernel.

6.2 Conjoint Analysis Experiment

Next, we tested our algorithms using a real data set from [23] about people's ratings of products.[6] The data was taken from a survey of 180 persons who rated the likelihood of purchasing one of 20 different personal computers. Here the persons correspond to tasks and the

[6] We would like to thank Peter Lenk for kindly sharing this data set with us.

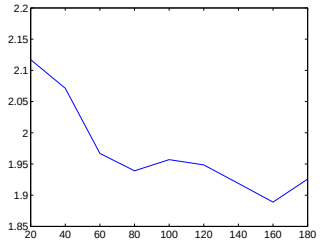


Fig. 6. Conjoint experiment with computer survey data: average root mean square error vs. number of tasks.

computer models to examples. The input is represented by the following 13 binary attributes: telephone hot line (TE), amount of memory (RAM), screen size (SC), CPU speed (CPU), hard disk (HD), CD-ROM/multimedia (CD), cache (CA), color (CO), availability (AV), warranty (WA), software (SW), guarantee (GU) and price (PR). We also added an input component accounting for the bias term. The output is an integer rating on the scale $0 - 10$. As in one of the cases in [23], for this experiment we used the first 8 examples per task as the training data and the last 4 examples per task as the test data. We measure the root mean square error of the predicted from the actual ratings for the test data, averaged across the persons.

We show results for the linear Algorithm 1 in Figure 6. In agreement with the simulations results above and past empirical and theoretical evidence – see e.g., [8] – the performance of Algorithm 1 improves as the number of tasks increases. It also performs better (for all 180 tasks) – test error is 1.93 – than independent ridge regressions, whose test error is equal to 3.88. Moreover, as shown in Figure 7, the number of features learned decreases as the regularization parameter γ increases, as expected.

This data has been used also in [16]. One of the empirical findings of [16, 23], a standard one regarding people's preferences, is that estimation improves when one also shrinks the individual w_t's towards a "mean of the tasks", for example the mean of all the w_t's. Hence, it may be more appropriate for this data set to use the regularization $\sum_{t=1}^{T} \langle (w_t - w_0), D^+(w_t - w_0) \rangle$ as in [16] instead of $\sum_{t=1}^{T} \langle w_t, D^+ w_t \rangle$ which we use here. Indeed, test performance is better with the former than the latter. The results are summarized in Table 1. We also note that the hierarchical Bayes method of [23], similar to that of [7], also shrinks the w_t's towards a mean across the tasks. Algorithm 1 performs

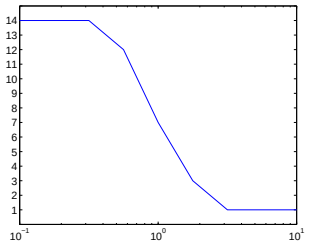


Fig. 7. Conjoint experiment with computer survey data: number of features learned (with 180 tasks) versus the regularization parameter γ.

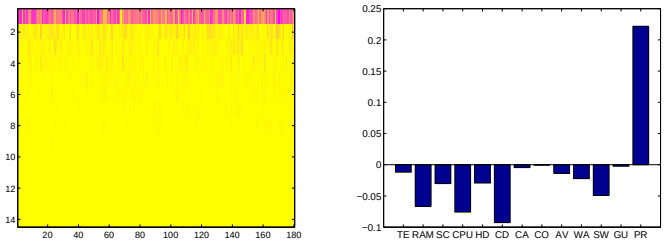


Fig. 8. Conjoint experiment with computer survey data. Left: matrix A learned, indicating the importance of features learned for all 180 tasks simultaneously. Right: the most important feature learned, common across the 180 people/tasks simultaneously learned.

similarly to hierarchical Bayes (despite not shrinking towars a mean of the tasks) but worse than the method of [16]. However, we are mainly interested here in learning the common across people/tasks features. We discuss this next.

We investigate which features are important to all consumers as well as how these features weight the 13 computer attributes. We demonstrate the results in the two adjacent plots of Figure 8, which were obtained by simultaneously learning all 180 tasks. The plot on the left shows the absolute values of matrix A of feature coefficients learned for this experiment. This matrix has only a few large rows, that is, only a few important features are learned. In addition, the coefficients in each of these rows do not vary significantly across tasks, which means that the learned feature representation is shared across the tasks. The plot on the right shows the weight of each input variable in the most important feature. This feature seems to weight the technical charac-

Table 1. Comparison of different methods for the computer survey data. MTL-FEAT is the method developed here.

Method	RMSE
Independent	3.88
Hierarchical Bayes [23]	1.90
RR-Het [16]	1.79
MTL-FEAT (linear kernel)	1.93
MTL-FEAT (Gaussian kernel)	1.85
MTL-FEAT (variable selection)	2.01

teristics of a computer (RAM, CPU and CD-ROM) against its price. Note that (as mentioned in the introduction) this is different from *selecting* the most important variables. In particular, in this case the relative "weights" of the 4 variables used in this feature (RAM, CPU, CD-ROM and price) are *fixed* across all tasks/people.

We also tested our multi-task variable *selection* method, which constrains matrix D in Algorithm 1 to be diagonal. This method led to inferior performance. Specifically, for $T = 180$, our multi-task variable selection method had test error equal to 2.01, which is worse than the 1.93 error achieved with our multi-task feature learning method. This supports the argument that "good" features should combine multiple attributes in this problem. Finally, we tested Algorithm 2 with a Gaussian kernel, achieving a slight improvement in performance – see Table 1. By considering radial kernels of the form $K(x, x') = e^{-\omega\|x-x'\|^2}$ and selecting ω through cross-validation, we obtained a test error of 1.85 for all 180 tasks. However, interpreting the features learned is more complicated in this case, because of the infinite dimensionality of the feature map for the Gaussian kernel.

6.3 School Data

We have also tested our algorithms on the data from the Inner London Education Authority, available at the web site of the Center for Multilevel Modeling[7]. This data set has been used in previous work on multi-task learning, for example in [18], [7] and [15]. It consists of examination

[7] Available at `http://www.mlwin.com/intro/datasets.html`.

scores of 15362 students from 139 secondary schools in London during the years 1985, 1986, 1987. Thus, there are 139 tasks, corresponding to predicting student performance in each school. The input consists of the year of the examination (YR), 4 school-specific and 3 student-specific attributes. Attributes which are constant in each school in a certain year are: percentage of students eligible for free school meals, percentage of students in VR band one (highest band in a verbal reasoning test), school gender (S.GN.) and school denomination (S.DN.). Student-specific attributes are: gender (GEN), VR band (can take the values 1,2 or 3) and ethnic group (ETH). Following [15], we replaced categorical attributes (that is, all attributes which are not percentages) with one binary variable for each possible attribute value. In total, we obtained 27 attributes.

We generated the training and test sets by 10 random splits of the data, so that 75% of the examples from each school (task) belong to the training set and 25% to the test set. We note that the number of examples (students) differs from task to task (school). On average, the training set includes about 80 students per school and the test set about 30 students per school. Moreover, we tuned the regularization parameter with 15-fold cross-validation. To account for different school populations, we computed the cross-validation error within each task and then normalized according to school population. The overall mean squared test error was computed by normalizing for each school in a similar way. In order to compare with previous work on this data set, we used the measure of percentage explained variance from [7]. Explained variance is defined as one minus the mean squared test error over the total variance of the data and indicates the percentage of variance explained by the prediction model.

The results for this experiment are shown in Table 2. For comparison, we have also reported the best result from [7], which was obtained with a hierarchical Bayesian multi-task method described therein, and that obtained in [15] with a multi-task regularization method using $\langle (w_t - w_0), (w_t - w_0) \rangle$ for regularization (unlike our method, no matrix D was included and estimated). We note that a number of key differences between Bayesian approaches, like the one of [7] and [23], and regularization ones, like the one discussed in this paper, have been analyzed in [16] – we refer the reader to that work for more information on this issue. As shown in the table, our multi-task feature learning algorithm has superior performance over the other methods for this data set.

Table 2. Comparison of different methods for the school data.

Method	Explained variance
Independent	$22.3 \pm 1.9\%$
Bayesian MTL [7]	$29.5 \pm 0.4\%$
Regularized MTL [15]	$34.8 \pm 0.5\%$
MTL-FEAT (linear kernel)	$37.1 \pm 1.5\%$
MTL-FEAT (Gaussian kernel)	$37.6 \pm 1.0\%$

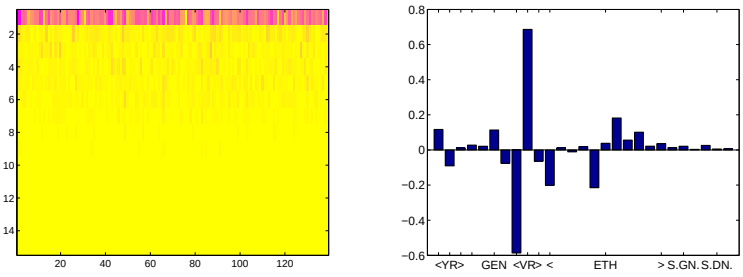


Fig. 9. School data. Left: matrix A learned for the school data set using a linear kernel. For clarity, only the 15 most important learned features/rows are shown. Right: The most important feature learned, common across all 139 schools/tasks simultaneously learned.

This data set seems well-suited to the approach we have proposed, as one may expect the learning tasks to be very related – as also discussed in [7, 15] – in the sense assumed in this paper. Indeed, one may expect that academic achievement should be influenced by the same factors across schools, if we exclude statistical variation of the student population within each school. This is confirmed in Figure 9, where the learned coefficients and the most important feature are shown. As expected, the predicted examination score depends very strongly on the student's VR band. The other factors are much less significant. Ethnic background (primarily British-born, Carribean and Indian) and gender have the next largest influence. What is most striking perhaps is that none of the school-specific attributes has any noticeable significance.

Finally, the effects of the number of tasks on the test performance and of the regularization parameter γ on the number of features learned

Table 3. Performance of the algorithms for the dermatology data.

Method	Misclassifications
Independent (linear)	16.5 ± 4.0
MTL-FEAT (linear)	16.5 ± 2.6
Independent (Gaussian)	9.8 ± 3.1
MTL-FEAT (Gaussian)	9.5 ± 3.0

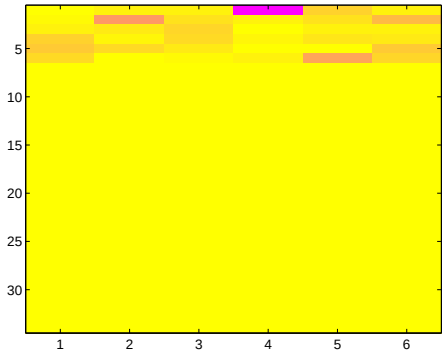


Fig. 10. Dermatology data. Feature coefficients matrix A learned, using a linear kernel.

are similar to those for the conjoint and synthetic data: as the number of tasks increases, test performance improves and as γ increases sparsity increases. These plots are similar to Figures 6 and 7 and are not shown for brevity.

6.4 Dermatology Data

Finally, we discuss a real-data experiment where it seems (as these are real data, we cannot know for sure whether indeed this is the case) that the tasks are unrelated (at least in the way we have defined in this paper). In this case, our methods find features which are different across the tasks and do not improve or decrease performance relative to learning each task independently.

We used the UCI dermatology data set[8] as in [22]. The problem is a multi-class one, namely to diagnose one of six dermatological diseases based on 33 clinical and histopathological attributes. As in the aforementioned paper, we obtained a multi-task problem from the six binary classification tasks. We divided the data set into 10 random splits of 200 training and 166 testing points and measured the average test error across these splits.

We report the misclassification test error in Table 3. Algorithm 1 gives similar performance to that obtained in [22] with joint feature *selection* and linear SVM classifiers. However, similar performance is also obtained by training 6 independent classifiers. The test error decreased when we ran Algorithm 2 with a single-parameter Gaussian kernel, but it is again similar to that obtained by training 6 independent classifiers with a Gaussian kernel. Hence one may conjecture that these tasks are weakly related to each other or unrelated in the way we define in this paper.

To further explore this point, we show the matrix A learned by Algorithm 1 in Figure 10. This figure indicates that different tasks (diseases) are explained by different features. These results reinforce our hypothesis that these tasks may be independent. They indicate that in such a case our methods do not "hurt" performance by simultaneously learning all tasks. In other words, in this problem our algorithms did learn a "sparse common representation" but did not – and probably should not – force each feature learned to be equally important across the tasks.

7 Conclusion

We have presented an algorithm which learns common sparse representations across a pool of related tasks. These representations are assumed to be orthonormal functions in a reproducing kernel Hilbert space. Our method is based on a regularization problem with a novel type of regularizer, which is a mixed $(2,1)$-norm.

We showed that this problem, which is non-convex, can be reformulated as a convex optimization problem. This result makes it possible to compute the optimal solutions using a simple alternating minimization algorithm, whose convergence we have proven. For the case of a high-dimensional feature map, we have developed a variation of the algorithm which uses kernel functions. We have also proposed a varia-

[8] Available at `http://www.ics.uci.edu/mlearn/MLSummary.html`.

tion of the first algorithm for solving the problem of multi-task feature *selection* with a linear feature map.

We have reported experiments with our method on synthetic and real data. They indicate that our algorithms learn sparse feature representations common to all the tasks whenever this helps improve performance. In this case, the performance obtained is better than that of training the tasks independently. Moreover, when applying our algorithm on a data set with weak task interdependence, performance does not deteriorate and the representation learned reflects the lack of task relatedness. As indicated in the experiments, one can also use the estimated matrix A to visualize the task relatedness. Finally, our experiments have shown that *learning* orthogonal features improves on just *selecting* input variables.

To our knowledge, our approach provides the first convex optimization formulation for multi-task feature learning. Although convex optimization methods have been derived for the simpler problem of feature selection [22], prior work on multi-task feature learning has been based on more complex optimization problems which are not convex [3, 8, 13] and, so, are at best only guaranteed to converge to a local minimum.

Our algorithm also shares some similarities with recent work in [3] where they also alternately update the task parameters and the features. Two main differences are that their formulation is not convex and that, in our formulation, the number of learned features is not fixed in advance but it is controlled by a regularization parameter.

As noted in Section 4, our work relates to that in [31], which investigates regularization with the trace norm in the context of collaborative filtering. Regularization with the trace norm for collaborative filtering is also investigated in [2]. In fact, the sparsity assumption which we have made in our work, starting with the $(2, 1)$-norm, connects to the low rank assumption in that work. Hence, it may be possible that our alternating algorithm, or some variation of it, could be used to solve the optimization problems of [31, 2]. Such an algorithm could be used with any convex loss function.

Our work may be extended in different directions. First, it would be interesting to carry out a learning theory analysis of the algorithms presented in this paper. Results in [12, 25] may be useful for this purpose. Another interesting question is to study how the solutions of our algorithm depend on the regularization parameter and investigate conditions which ensure that the number of features learned decreases with the degree of regularization, as we have experimentally observed in this paper. Results in [26] may be useful for this purpose.

Second, on the algorithmic side, it would be interesting to explore whether our formulation can be extended to the more general class of spectral norms in place of the trace norm. A special case of interest is the $(2, p)$-norm for $p \in [1, \infty)$. This question is being addressed in [6].

Finally, a promising research direction is to explore whether different assumptions about the features (other than the orthogonality one which we have made throughout this paper) can still lead to convex optimization methods for learning other types of features. More specifically, it would be interesting to study whether non-convex models for learning structures across the tasks, like those in [35] where ICA type features are learned, or hierarchical features models like in [32], can be reformulated in our framework.

Acknowledgements

We wish to thank Raphael Hauser and Ying Yiming for observations which led to Proposition 1, Zoubin Ghahramani, Mark Herbster, Andreas Maurer and Sayan Mukherjee for useful comments and Peter Lenk for sharing his data set with us. A special thanks to Charles Micchelli for many useful insights.

A Proof of Equation (13)

Proof. Consider a matrix $C \in \mathbf{S}_+^d$. We will compute $\inf\{\mathrm{trace}(D^{-1}C) : D \in \mathbf{S}_{++}^d, \mathrm{trace}(D) \leq 1\}$. We can write $D = U\mathrm{Diag}(\lambda)U^\top$, with $U \in \mathbf{O}^d$ and $\lambda \in \mathbb{R}_{++}^d$. We first minimize over λ. Applying Lemma 1, we have that

$$\inf\left\{ \mathrm{trace}\Big(C^{\frac{1}{2}}U\mathrm{Diag}(\lambda)^{-1}U^\top C^{\frac{1}{2}}\Big) : \lambda \in \mathbb{R}_{++}^d, \sum_{i=1}^d \lambda_i \leq 1 \right\} = \left(\sum_{i=1}^d \big\|C^{\frac{1}{2}}u_i\big\|_2\right)^2 = \big\|U^\top C^{\frac{1}{2}}\big\|_{2,1}^2 .$$

We now show that

$$\inf\Big\{ \big\|U^\top C^{\frac{1}{2}}\big\|_{2,1}^2 : U \in \mathbf{O}_d \Big\} = \Big(\mathrm{trace}\, C^{\frac{1}{2}}\Big)^2$$

and that a minimizing U is a system of eigenvectors of C. To see this, note that

$$\left\|C^{\frac{1}{2}}u_i\right\|_2^2 = \operatorname{trace}\left(u_i^\top C^{\frac{1}{2}} C^{\frac{1}{2}} u_i\right) =$$
$$\operatorname{trace}\left(C^{\frac{1}{2}} u_i u_i^\top u_i u_i^\top C^{\frac{1}{2}}\right) \operatorname{trace}(u_i u_i^\top u_i u_i^\top) \geq$$
$$\left(\operatorname{trace}\left(C^{\frac{1}{2}} u_i u_i^\top u_i u_i^\top\right)\right)^2 =$$
$$\left(\operatorname{trace}\left(C^{\frac{1}{2}} u_i u_i^\top\right)\right)^2 = (u_i^\top C^{\frac{1}{2}} u_i)^2$$

since $u_i u_i^\top u_i u_i^\top = u_i u_i^\top$. The equality is verified if and only if $C^{\frac{1}{2}} u_i u_i^\top = a\, u_i u_i^\top$, for some $a \in \mathbb{R}$, or equivalently $C^{\frac{1}{2}} u_i = a u_i$, that is, if and only if u_i is an eigenvector of C. Equality at the application of Lemma 1 holds if and only if $\lambda_i = \frac{\left\|C^{\frac{1}{2}} u_i\right\|_2}{\left\|U^\top C^{\frac{1}{2}}\right\|_{2,1}}$, $i \in \mathbb{N}_d$. Thus, the optimal value is $\left(\operatorname{trace} C^{\frac{1}{2}}\right)^2$ and is attained if and only if $D = \frac{C^{\frac{1}{2}}}{\operatorname{trace} C^{\frac{1}{2}}}$. □

In addition, it can be shown that $\min\{\operatorname{trace}(D^+ C) : D \in \mathbf{S}_+^d, \operatorname{trace}(D) \leq 1, \operatorname{range}(C) \subseteq \operatorname{range}(D)\}$ also equals $\left(\operatorname{trace} C^{\frac{1}{2}}\right)^2$, using similar arguments as above. The difference here is that Lemma 1 is applied only to those i such that $\left\|C^{\frac{1}{2}} u_i\right\|_2 \neq 0$ and the corresponding λ_i are guaranteed to be nonzero by the range constraint.

B Convergence of Algorithm 1

In this appendix, we present the proofs of Theorems 2 and 3. For this purpose, we substitute equation (13) in the definition of $\mathcal{R}_\varepsilon$ obtaining the objective function

$$\mathcal{S}_\varepsilon(W) := \mathcal{R}_\varepsilon(W, D_\varepsilon(W)) =$$
$$\sum_{t=1}^{T} \sum_{i=1}^{m} L(y_{ti}, \langle w_t, x_{ti} \rangle) + \gamma \left(\operatorname{trace}(WW^\top + \varepsilon I_d)^{\frac{1}{2}}\right)^2 .$$

Moreover, we define the following function which formalizes the W-step of the algorithm,

$$g_\varepsilon(W) := \min\{\mathcal{R}_\varepsilon(V, D_\varepsilon(W)) : V \in \mathbb{R}^{d \times T}\} .$$

Since $\mathcal{S}_\varepsilon(W) = \mathcal{R}_\varepsilon(W, D_\varepsilon(W))$ and $D_\varepsilon(W)$ minimizes $\mathcal{R}_\varepsilon(W, \cdot)$, we obtain that

$$\mathcal{S}_\varepsilon(W^{(n+1)}) \leq g_\varepsilon(W^{(n)}) \leq \mathcal{S}_\varepsilon(W^{(n)}) . \tag{23}$$

We begin by observing that $\mathcal{S}_\varepsilon$ has a *unique* minimizer. This is a direct consequence of the following proposition.

Proposition 3. *The function $\mathcal{S}_\varepsilon$ is strictly convex for every $\varepsilon > 0$.*

Proof. It suffices to show that the function

$$W \mapsto \left(\operatorname{trace}(WW^\top + \varepsilon I_d)^{\frac{1}{2}}\right)^2$$

is strictly convex. But this is simply a *spectral* function, that is, a function of the singular values of W. By [24, Sec. 3], strict convexity follows directly from strict convexity of the real function $\sigma \mapsto \left(\sum_i \sqrt{\sigma_i^2 + \varepsilon}\right)^2$. This function is strictly convex because it is the square of a positive strictly convex function. □

We note that when $\varepsilon = 0$, the function $\mathcal{S}_\varepsilon$ is regularized by the trace norm squared which is not a strictly convex function. Thus, in many cases of interest $\mathcal{S}_0$ may have multiple minimizers. This may happen, for instance, if the loss function L is not strictly convex, which is the case with SVMs.

Next, we show the following continuity property which underlies the convergence of Algorithm 1.

Lemma 2. *The function g_ε is continuous for every $\varepsilon > 0$.*

Proof. We first show that the function $G_\varepsilon : \mathbf{S}^d_{++} \to \mathbb{R}$ defined as

$$G_\varepsilon(D) := \min\left\{\mathcal{R}_\varepsilon(V, D) : V \in \mathbb{R}^{d\times T}\right\}$$

is convex. Indeed, $G_\varepsilon(D)$ is the minimal value of T separable regularization problems with a common kernel function determined by D. For a proof that the minimal value of a 2-norm regularization problem is convex in the kernel, see [5, Lemma 2]. Since the domain of this function is open, G_ε is also continuous (see [11, Sec. 4.1]).

In addition, the matrix-valued function $W \mapsto (WW^\top + \varepsilon I_d)^{\frac{1}{2}}$ is continuous. To see this, we recall the fact that the matrix-valued function $Z \in \mathbf{S}^d_+ \mapsto Z^{\frac{1}{2}}$ is continuous. Continuity of the matrix square root is due to the fact that the square root function on the reals, $t \mapsto t^{\frac{1}{2}}$, is *operator monotone* – see e.g., [10, Sec. X.1].

Combining, we obtain that g_ε is continuous, as the composition of continuous functions. □

Proof of Theorem 2. By inequality (23) the sequence $\{\mathcal{S}_\varepsilon(W^{(n)}) : n \in \mathbb{N}\}$ is nonincreasing and, since L is bounded from below, it is bounded. As a consequence, as $n \to \infty$, $\mathcal{S}_\varepsilon(W^{(n)})$ converges to a

number, which we denote by $\widetilde{\mathcal{S}_\varepsilon}$. We also deduce that the sequence $\left\{\text{trace}\left(W^{(n)}W^{(n)\top}+\varepsilon I_d\right)^{\frac{1}{2}} : n\in\mathbb{N}\right\}$ is bounded and hence so is the sequence $\{W^{(n)} : n\in\mathbb{N}\}$. Consequently there is a convergent subsequence $\{W^{(n_\ell)} : \ell\in\mathbb{N}\}$, whose limit we denote by $\widetilde{W}$.

Since $\mathcal{S}_\varepsilon(W^{(n_\ell+1)}) \le g_\varepsilon(W^{(n_\ell)}) \le \mathcal{S}_\varepsilon(W^{(n_\ell)})$, $g_\varepsilon(W^{(n_\ell)})$ converges to $\widetilde{\mathcal{S}_\varepsilon}$. Thus, by Lemma 2 and the continuity of $\mathcal{S}_\varepsilon$, $g_\varepsilon(\widetilde{W}) = \mathcal{S}_\varepsilon(\widetilde{W})$. This implies that $\widetilde{W}$ is the minimizer of $\mathcal{R}_\varepsilon(\cdot, D_\varepsilon(\widetilde{W}))$, because $\mathcal{R}_\varepsilon(\widetilde{W}, D_\varepsilon(\widetilde{W}))$ $= \mathcal{S}_\varepsilon(\widetilde{W})$.

Moreover, recall that $D_\varepsilon(\widetilde{W})$ is the minimizer of $\mathcal{R}_\varepsilon(\widetilde{W},\cdot)$ subject to the constraints in (12). Since the regularizer in $\mathcal{R}_\varepsilon$ is smooth, any directional derivative of $\mathcal{R}_\varepsilon$ is the sum of its directional derivatives with respect to W and D. Hence, $(\widetilde{W}, D_\varepsilon(\widetilde{W}))$ is the minimizer of $\mathcal{R}_\varepsilon$.

We have shown that any convergent subsequence of $\{W^{(n)} : n\in\mathbb{N}\}$ converges to the minimizer of $\mathcal{R}_\varepsilon$. Since the sequence $\{W^{(n)} : n\in\mathbb{N}\}$ is bounded it follows that it converges to the minimizer as a whole. □

Proof of Theorem 3. Let $\left\{\left(W_{\ell_n}, D_{\varepsilon_{\ell_n}}(W_{\ell_n})\right) : n\in\mathbb{N}\right\}$ be a limiting subsequence of the minimizers of $\{\mathcal{R}_{\varepsilon_\ell} : \ell\in\mathbb{N}\}$ and let $(\widetilde{W}, \widetilde{D})$ be its limit as $n\to\infty$. From the definition of $\mathcal{S}_\varepsilon$ it is clear that $\min\{\mathcal{S}_\varepsilon(W) : W\in\mathbb{R}^{d\times T}\}$ is a decreasing function of ε and converges to $\bar{\mathcal{S}} = \min\{\mathcal{S}_0(W) : W\in\mathbb{R}^{d\times T}\}$ as $\varepsilon\to 0$. Thus, $\mathcal{S}_{\varepsilon_{\ell_n}}(W_{\ell_n}) \to \bar{\mathcal{S}}$. Since $\mathcal{S}_\varepsilon(W)$ is continuous in both ε and W (see proof of Lemma 2), we obtain that $\mathcal{S}_0(\widetilde{W}) = \bar{\mathcal{S}}$. □

C Proof of Lemma 3 used in the proof of Theorem 4

Lemma 3. *Let $P, N\in\mathbb{R}^{d\times T}$ such that $P^\top N = 0$. Then $\|P+N\|_{\text{tr}} \ge \|P\|_{\text{tr}}$. The equality is attained if and only if $N = 0$.*

Proof. We use the fact that, for matrices $A, B\in\mathbf{S}_+^n$, $A\succeq B$ implies that $\text{trace}A^{\frac{1}{2}} \ge \text{trace}B^{\frac{1}{2}}$. This is true because the square root function on the reals, $t\mapsto t^{\frac{1}{2}}$, is *operator monotone* – see [10, Sec. V.1]. We apply this fact to the matrices $P^\top P + N^\top N$ and $N^\top N$ to obtain that

$$\|P+N\|_{\text{tr}} = \text{trace}((P+N)^\top(P+N))^{\frac{1}{2}} = \text{trace}(P^\top P + N^\top N)^{\frac{1}{2}} \ge \text{trace}(P^\top P)^{\frac{1}{2}} = \|P\|_{\text{tr}}.$$

The equality is attained if and only if the spectra of $P^\top P + N^\top N$ and $P^\top P$ are equal, whence $\text{trace}(N^\top N) = 0$, that is $N = 0$. □

References

1. D.A. Aaker, V. Kumar, and G.S. Day. *Marketing Research*. John Wiley & Sons, 2004. 8th edition.
2. J. Abernethy, F. Bach, T. Evgeniou, and J-P. Vert. Low-rank matrix factorization with attributes. Technical Report 2006/68/TOM/DS, INSEAD, 2006. Working paper.
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. Multi-task feature learning. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, 2007. In press.
5. A. Argyriou, C. A. Micchelli, and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT)*, volume 3559 of *LNAI*, pages 338–352. Springer, 2005.
6. A. Argyriou, C.A. Micchelli, M. Pontil, and Y. Ying. Representer theorems for spectral norms. Working paper, Dept. of Computer Science, University College London, 2007.
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. J. Baxter. A model for inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.
9. S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *Proceedings of the 16th Annual Conference on Learning Theory (COLT)*, volume 2777 of *LNCS*, pages 567–580. Springer, 2003.
10. R. Bhatia. *Matrix analysis*. Graduate texts in Mathematics. Springer, 1997.
11. J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. CMS Books in Mathematics. Springer, 2005.
12. A. Caponnetto and E. De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, August 2006.
13. R. Caruana. Multi–task learning. *Machine Learning*, 28:41–75, 1997.
14. D. Donoho. For most large underdetermined systems of linear equations, the minimal l1-norm near-solution approximates the sparsest near-solution. Preprint, Dept. of Statistics, Stanford University, 2004.
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. T. Evgeniou, M. Pontil, and O. Toubia. A convex optimization approach to modeling consumer heterogeneity in conjoint estimation. Technical Report, INSEAD, 2006.

17. 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.
18. H. Goldstein. Multilevel modelling of survey data. *The Statistician*, 40:235–244, 1991.
19. G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.
20. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Verlag Series in Statistics, 2001.
21. B. Heisele, T. Serre, M. Pontil, T. Vetter, and T. Poggio. Categorization by learning and combining object parts. In *Advances in Neural Information Processing Systems 14*, pages 1239–1245. MIT Press, 2002.
22. T. Jebara. Multi-task feature and kernel selection for SVMs. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.
23. P. J. Lenk, W. S. DeSarbo, P. E. Green, and M. R. Young. Hierarchical Bayes conjoint analysis: recovery of partworth heterogeneity from reduced experimental designs. *Marketing Science*, 15(2):173–191, 1996.
24. A. S. Lewis. The convex analysis of unitarily invariant matrix functions. *Journal of Convex Analysis*, 2(1):173–183, 1995.
25. 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.
26. C. A. Micchelli and A. Pinkus. Variational problems arising from balancing several error criteria. *Rendiconti di Matematica, Serie VII*, 14:37–86, 1994.
27. C. A. Micchelli and M. Pontil. On learning vector–valued functions. *Neural Computation*, 17:177–204, 2005.
28. G. Obozinski, B. Taskar, and M.I. Jordan. Multi-task feature selection. Technical report, Dept. of Statistics, UC Berkeley, June 2006.
29. T. Poggio and F. Girosi. A sparse representation for function approximation. *Neural Computation*, 10:1445–1454, 1998.
30. T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio. Theory of object recognition: computations and circuits in the feedforward path of the ventral stream in primate visual cortex. AI Memo 2005-036, Massachusetts Institute of Technology, 2005.
31. 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.
32. A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 762–769, 2004.
33. G. Wahba. *Splines Models for Observational Data*, volume 59 of *Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

34. K. Yu, V. Tresp, and A. Schwaighofer. Learning Gaussian processes from multiple tasks. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.
35. J. Zhang, Z. Ghahramani, and Y. Yang. Learning multiple related tasks using latent independent component analysis. In *Advances in Neural Information Processing Systems 18*, pages 1585–1592. MIT Press, 2006.