$\ell_p - \ell_q$ penalty for Sparse Linear and Sparse Multiple Kernel Multi-Task Learning

Alain Rakotomamonjy*, Rémi Flamary, Gilles Gasso, Stéphane Canu

LITIS, EA 4108 - INSA / Université de Rouen
Avenue de l'Université - 76801 Saint-Etienne du Rouvray Cedex
firstname.lastname@insa-rouen.fr

***Abstract*—Recently, there has been a lot of interest around multi-task learning (MTL) problem with the constraints that tasks should share a common sparsity profile. Such a problem can be addressed through a regularization framework where the regularizer induces a joint-sparsity pattern between task decision functions. We follow this principled framework and focus on $\ell_p - \ell_q$ (with $0 \leq p \leq 1$ and $1 \leq q \leq 2$) mixed-norms as sparsity-inducing penalties. Our motivation for addressing such a larger class of penalty is to adapt the penalty to a problem at hand leading thus to better performances and better sparsity pattern. For solving the problem in the general multiple kernel case, we first derive a variational formulation of the $\ell_1 - \ell_q$ penalty which helps up in proposing an alternate optimization algorithm. Although very simple, the latter algorithm provably converges to the global minimum of the $\ell_1 - \ell_q$ penalized problem. For the linear case, we extend existing works considering accelerated proximal gradient to this penalty. Our contribution in this context is to provide an efficient scheme for computing the $\ell_1 - \ell_q$ proximal operator. Then, for the more general case when $0 < p < 1$, we solve the resulting non-convex problem through a majorization-minimization approach. The resulting algorithm is an iterative scheme which, at each iteration, solves a weighted $\ell_1 - \ell_q$ sparse MTL problem. Empirical evidences from toy dataset and real-word datasets dealing with BCI single trial EEG classification and protein subcellular localization show the benefit of the proposed approaches and algorithms.**

***Index Terms*—Multi-task learning, multiple kernel learning, sparsity, mixed-norm, Support Vector Machines**

I. Introduction

Multi-Task Learning (MTL) is a statistical learning framework which seeks at learning several models in a joint manner. The idea behind this paradigm is that, when the tasks to be learned are similar enough or are related in some sense, it may be advantageous to take into account these relations between tasks. For instance, when the number of samples for learning are small, transferring some knowledge from one task to another while learning can be advantageous in term of generalization performances. Several works have provided empirical evidence on the benefit of such a framework [9], [15], [23], [35]. Application domains that have been shown to benefit from multi-task learning are medical diagnosis [5], drug therapy prediction [6], vaccine design [21] or conjoint analysis [1].

However, the notion of relatedness between tasks is vague and depends on the problem at hand. For instance, one can consider that models resulting from related tasks should be similar to a single model [15], [23]. In other works, task's relatedness is represented through a probabilistic model [52]. Prior knowledge on tasks are then translated into an appropriate regularization term or into a hierarchical Bayesian model that can be handled by a learning algorithm [14], [50], [20].

In this work, we consider that tasks to be learned share a common subset of features or kernel representation. This means that while learning the tasks, we jointly look for features or kernels that are useful for all tasks. In this context of joint feature selection with multiple related tasks, several works have already been carried out. For instance, Jebara et al. [22] has introduced a maximum entropy discrimination for solving such a problem. Some other works cast the problem into a probabilistic framework which uses automatic relevance determination and a hierarchical Bayesian model for selecting the relevant features [5], [49]. Another trend considers a regularization principle and thus minimizes a regularized empirical risk with a regularization term that favors a common sparsity profile for all tasks. Such an approach has been investigated by Argyriou et al. [2] and Obozinski et al. [33]. In these latter works, the authors propose a $\ell_1 - \ell_2$ regularization term which can be interpreted as a convex extension of the sparsity-inducing ℓ_1 norm in single task learning.

As made clear in the sequel, our contribution in this paper lies in between multi-task and multiple kernel learning. Indeed, we provide a methodological framework for learning each task decision functions while these functions use an optimal, in some sense, linear combination of only few kernels. This point highlights the relation between our contribution and multiple kernel learning. Imposing that the few selected kernels are similar across the tasks is the point that defines task's relatedness. Following Obozinski et al. [33] and Argyriou et al. [2], we induce this sparsity in joint kernel representation through a regularization principle where the regularization term is a mixed-norm penalty.

In practice and in theory as proved by the works of Lounici et al. [31], a fixed non-adaptive penalty like the $\ell_1 - \ell_2$ mixed norm, is beneficial with respects to other penalties only under certain situations. Hence, it seems natural that different penalties may suit better to different data structures. This motivates us to investigate the use of a larger class of mixed-norm penalty that can be adapted to the data at hand. We focus here on the class of $\ell_p - \ell_q$ mixed-norm penalty where $0 \leq p \leq 1$ and $1 \leq q \leq 2$. Our objective in using $p < 1$ is to make the kernel representation across tasks sparser than using $p = 1$; such an increased sparsity profile being valuable in a

presence a large amount of noisy features or irrelevant kernels. Furthermore, the sparser representation induced by $p < 1$ is expected to enhance models interpretability and improve evaluation computational efficiency. Varying q between 1 and 2 allows the task decision functions to adapt themselves to the importance of the task relatedness. Indeed, it would be clear in the following that $q = 1$ makes the task learning independent while $q > 1$ ties them through the mixed-norm. Rationales on why we have not investigated cases where $2 < q < \infty$ will also be discussed.

Our aim in this paper is to present a simple algorithm for handling the optimization problem resulting of the use of $\ell_p - \ell_q$ mixed-norms regularizers in the multi-task framework and to provide empirical evidences that making the choice of p and q adaptive with respects to the data at hand works as good as or better than a fixed $\ell_1 - \ell_2$ penalty in various situations. Algorithmically, we first show that, for the general multiple kernel case, a variational formulation of the $\ell_1 - \ell_q$ mixed-norm can be obtained. Such a novel formulation helps us in deriving a simple alternate algorithm for solving the sparse $\ell_1 - \ell_q$ multi-task problem which provably converges towards the solution of the problem. For the linear case, as such an algorithm may not be efficient, we extend existing works [11] on accelerated proximal gradient to handle the case of $\ell_1 - \ell_q$ norm. We essentially provide a novel way for computing the proximal operator of this mixed-norm. At a second stage, we address the case of the non-convex $\ell_p - \ell_q$ $(0 < p < 1)$ regularization term. The difficulty raised by this non-convex problem is tackled via a Majorization-Minimization (MM) approach [19]. This leads to an iterative scheme which solves at each iteration, a reweighted $\ell_1 - \ell_q$ multi-task learning problem.

In the next section, we present the general formulation of the sparse MTL problem as well as a brief review of closely spirit-related works. Algorithmic developments are presented in Section III. Then, some empirical results that illustrate the behavior of our algorithms are given in Section IV while some concluding remarks are drawn in Section V. For a sake of reproducibility, the Matlab code used for this paper is available at http://asi.insa-rouen.fr/enseignants/~arakotom/code/SparseMTL.html

II. Multi-Task feature/kernel selection framework

This section introduces our sparse MTL framework and discusses related works available in the literature.

A. Framework

Suppose we are given T classification tasks to be learned from T different datasets $(x_{i,1}, y_{i,1})_{i=1}^{n_1}, \cdots, (x_{i,T}, y_{i,T})_{i=1}^{n_T}$, where any $x_{i,\cdot} \in \mathcal{X}$ and $y_{i,\cdot} \in \{+1, -1\}$ and n_i denotes the i-th dataset size. For a given task t, we are looking for a decision function of the form:

$$f_t(x) = \sum_{k=1}^{M} f_{t,k}(x) + b_t \qquad \forall t \in \{1, \cdots, T\} \tag{1}$$

where a function $f_{\cdot,k}$ belongs to a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$ of kernel K_k, b_t is a bias term and M is the number of *basis* kernels provided. Depending on the input space $\mathcal{X}$, $\mathcal{H}_k$ can take different forms. For instance, if $\mathcal{X} = \mathbb{R}^d$, $\mathcal{H}_k$ can be a subset of $\mathbb{R}^d$ built from a single or several dimensions. In some other situations, $\mathcal{H}_k$ can be also an infinite dimension space defined implicitly by its kernel (*e.g* a Gaussian kernel).

The objective of this work is to learn the decision function f_t for each task under the constraints that all these functions share a common sparsity profile of their kernel representation. Hence, the pursued hope is to build a learning algorithm able to yield many vanishing functions $f_{t,k}$ for all t.

For achieving this goal, we cast our problem as the following optimization problem:

$$\min_{f_1, \cdots, f_T} C \cdot \sum_{t,i} L(f_t(x_{i,t}), y_{i,t}) + \Omega(f_1, \cdots, f_T) \tag{2}$$

where $L(f_t(x), y)$ is a loss function, Ω a sparsity-inducing penalty term involving all functions f_t and C a trade-off parameter that balances both antagonist objectives. Hereafter, we will focus on a Hinge loss function, denoted as $H(f(x), y)$, although our algorithm can be straightforwardly applied to other losses.

B. Joint sparsity-inducing penalty

Since few years now, there has been a large interest around sparse models. While different approaches are possible for generating sparse methods [17], [26], sparsity are usually induced by a penalty function [10] or a proper type of Bayesian modeling [46], [47].

For a single task empirical minimization problem, sparse models are usually induced by the use of a ℓ_1-norm regularizer [45]. For a Multi-Task Learning problem, this approach can be properly generalized by the use of appropriate norm. For instance, Obozinski et al. and Argyriou et al. [2], [33] propose a regularizer of the form :

$$\Omega(f_1, \cdots, f_T) = \sum_{k=1}^{M} \left(\sum_{t=1}^{T} \|f_{t,k}\|_{\mathcal{H}_k}^2 \right)^{1/2}.$$

This latter regularizer is a ℓ_1 block-norm that tends to produce sparse kernel solutions. For single task problem, such a regularizer has been used for sparse kernel selection in multiple kernel learning problem [3]. For single task linear problem, this regularizer is equivalent to a ℓ_1-norm penalty.

In order to be more data-adaptive, this regularizer can be generalized as :

$$\Omega_{p,q}(f_1, \cdots, f_T) = \sum_{k=1}^{M} \left(\sum_{t=1}^{T} \|f_{t,k}\|_{\mathcal{H}_k}^q \right)^{p/q} \tag{3}$$

where typically $0 \leq p \leq 1$ and $q \geq 1$. For this regularizer, a ℓ_q norm is applied to the vector of all task norms in $\mathcal{H}_k$ and then a ℓ_p norm or pseudo-norm is applied to the resulting vector. The ℓ_q norm in the regularizer controls the weights of each task for the space $\mathcal{H}_k$ and how this kernel representation will be shared across tasks. For instance, large value of q (such as

$q = \infty$) means that as soon as $\|f_{t,k}\|_{\mathcal{H}_k}$ is non-zero, another task t' can have a non-zero norm for $f_{t',k}$ without increasing significantly the regularizer $\Omega_{p,q}$. Note that for $p = 1$ and $q = 1$, the regularizer can be decoupled and thus the learning problem boils down to be T independent problems. The ℓ_p pseudo-norm controls the sparsity of the kernel representation for all tasks. For $p < 1$, regularizer (3) is expected to produce sparser solutions than for $p = 1$, hence using such a mixed-norm penalty is expected to be more efficient in presence of many irrelevant variables or kernels. Note that this kind of mixed-norm regularizer has already been proposed for single task learning for achieving composite absolute penalization [53] or for composite kernel learning [44].

However, in the context of multi-task learning, only some particular cases of the mixed-norm $\Omega_{p,q}$ have been considered. Obozinski et al. [33] use $p = 1$ and $q = 2$ while Liu et al. [27], [28], Quattoni et al. [37], [36] and Chen et al. [11] have considered the use of $p = 1$ and $q = \infty$. For all these works, the authors have focused on convex situations since $\Omega_{p,q}$ is known to be convex whenever $p, q \geq 1$ and non-convex for $p < 1$ and $q \geq 1$. Recently, several works on learning single sparse models have stressed the need of non-convex penalties for achieving better sparsity profile. For instance, Knight et al. [25] suggested the use of the so-called Bridge penalty which simply consists in replacing the ℓ_1 norm with a ℓ_p pseudo-norm with $0 < p < 1$. In our multi-task learning framework, this can be naturally generalized by using the regularizer given in Equation (3) with $0 < p < 1$. For instance, two very recent works have focused on theoretical properties of the mixed-norm $\Omega_{p,1}$ and $\Omega_{1,q}$ for variable selection in multiple regression problems [18], [29].

As we can see, there are a lot of algorithmic works that address the case where $p = 1$ and $q \in \{2, \infty\}$. These works usually aim at developing efficient algorithms in the linear decision function. The novelty of our contribution lies in considering a larger class ($p \leq 1$ and $1 \leq q \leq 2$) of mixed-norm penalties while keeping the kernel framework. By choosing p and q in these intervals, we aim a better adaptivity of the penalty to the datasets at hand. However, in this paper, we only focus on algorithms for solving the resulting optimization problems, while the task of efficiently selecting p and q has been left for future works. Furthermore, although we have focused on the use of Hinge loss function, the algorithms we propose in the sequel are generic enough to handle different type of loss functions in the optimization problem as well as heterogeneous loss functions.

As we have stated, we do not consider in this paper cases where $2 < q < \infty$. There is a main reason for this. Indeed, we believe that as q becomes greater than 2, the $\ell_1 - \ell_q$ regularizer rapidly becomes numerically equivalent to a $\ell_{1,\infty}$ one. This point can be made clear from the relation $\left(\sum_i |a_i|^q\right)^{1/q} = |a_k| \left(\sum_i |\frac{a_i}{a_k}|^q\right)^{1/q}$ where k is the index of the largest $|a_i|$. Hence, since there already exists efficient method for $\ell_{1,\infty}$ sparse multi-task learning [36], we have not addressed this case in this present work. However, we still plan in a forthcoming work, to provide a better analysis of the use of these $\ell_1 - \ell_q$ penalties so as to understand in which situations they may perform better than the $\ell_1 - \ell_\infty$ penalty.

III. Algorithms for jointly sparse multi-task SVM

In this section, we propose some algorithms for solving the sparse multi-task SVM problem when using $\Omega_{p,q}$ as a regularizer with values $p \leq 1$ and $1 \leq q \leq 2$. At first, we consider the convex case when $p = 1$ and then we introduce an algorithm which solves the problem for $p < 1$.

A. A smooth formulation of $\ell_1 - \ell_q$ regularized problem

The algorithm we propose is based on a variational formulation of the mixed-norm $\Omega_{1,q}(\cdot)$. The following proposition extends the one of Michelli et al. [32] to mixed-norm. Similar propositions have been derived for multiple and composite kernel learning [38], [44]. Here and in what follows, u/v is defined as $u/0 = \infty$ if $u \neq 0$ and $0/0 = 0$.

Proposition 1: if $s > 0$ and $\{a_{t,k} \in \mathbb{R} : k \in [1, \cdots, M], t \in [1, \cdots, T]\}$ such that at least one $|a_{t,k}| > 0$, then the following minimization problem over elements $d_{t,k}$ admits a unique minimum

$$\min_{\{d_{t,k}\}} \left\{ \sum_{k,t} \frac{|a_{t,k}|^2}{d_{t,k}} : d_{t,k} \geq 0, \sum_k \left(\sum_t d_{t,k}^{1/s} \right)^s \leq 1 \right\} = \left(\sum_k \left(\sum_t |a_{t,k}|^q \right)^{1/q} \right)^2 \quad (4)$$

where $q = \frac{2}{s+1}$. Furthermore, at optimality, we have:

$$d_{t,k}^\star = \frac{|a_{t,k}|^{\frac{2s}{s+1}} \left(\sum_u |a_{u,k}|^{\frac{2}{s+1}} \right)^{\frac{1-s}{2}}}{\sum_v \left(\sum_u |a_{u,v}|^{\frac{2}{s+1}} \right)^{\frac{s+1}{2}}} \quad (5)$$

Proof: (Sketch) The proof proceeds by writing down the Lagrangian of the minimization problem and deriving the optimality condition wrt to $d_{t,k}$. Then, we get

$$d_{t,k}^{1/s} = \lambda^{-1/(s+1)} |a_{t,k}|^{2/(s+1)} \left(\sum_u d_{u,k}^{1/s} \right)^{\frac{1-s}{1+s}}$$

where λ is the Lagrangian multiplier associated to the mixed-norm constraint. From these optimality conditions, we derive $\sum_u d_{u,k}^{1/s} = \left(\lambda^{-1} \left(\sum_u |a_{u,k}|^{2/s+1} \right)^{s+1} \right)^{1/2s}$. Then since at optimality, the mixed-norm inequality becomes an equality, we have $\lambda = \left(\sum_k \left(\sum_t |a_{t,k}|^{2/s+1} \right)^{(s+1)/2} \right)^2$. Plugging all these equations into the optimality conditions of $d_{t,k}$ proves the above proposition. ∎

Hence, by setting $a_{t,k} = \|f_{t,k}\|_{\mathcal{H}_k}$, the above proposition gives a variational formulation of $\Omega_{1,q}(\cdot)^2$. It is interesting to note how the mixed-norm on functions $f_{t,k}$ transfers to another mixed-norm on the weights $d_{t,k}$. We can see that for $q = 1$, this latter mixed-norm decouples. When $q \to 2$, which correspond to multiple kernel learning for a single task [3], we have $s \to 0$ and the mixed-norm on the weight becomes a mixed supnorm. This means that at optimality, all the weights $\{d_{\cdot,k}\}$ associated to non-zero $\{a_{\cdot,k}\}$ should

have similar values. Indeed, suppose that for tasks t and t', $d_{t',k} < d_{t,k}$ and $a_{t,k}$, $a_{t',k}$ are non-zero, then the objective value can be decreased by setting $d_{t',k} = d_{t,k}$. For this case, it would have been preferable to consider a single weight d_k associated to each RKHS $\mathcal{H}_k$.

Now let us consider the optimization problem related to our sparse multi-task learning problem for $p = 1$ and $1 \leq q \leq 2$:

$$\min_{f_1,\cdots,f_T} C \cdot \sum_{t,i} H(f_t(x_{i,t}), y_{i,t}) + \Omega_{1,q}(f_1, \cdots, f_T)^2 \quad (6)$$

Since the penalty term is convex and the square function is strictly monotonically increasing function on $\mathbb{R}_+$, squaring the penalty term in the objective function as above, leads to an equivalent optimization problem without the squaring. Here the equivalence is understood as for any hyperparameter value C, there exists a hyperparameter C' related to the optimization problem without the squaring term (as in Equation 2) so that the solutions of both problems are equal (a more formal proof of this claim is detailed in the appendix). Then, owing to the variational formulation of $\Omega_{1,q}(f_1, \cdots, f_T)^2$, we can rewrite the optimization problem related to a sparse multi-task SVM as

$$\begin{array}{cc} \min\limits_{f_1,\cdots,f_T,\mathbf{d}} & C\sum_{t,i} H(f_t(x_{i,t}), y_{i,t}) + \sum_{t,k} \frac{\|f_{t,k}\|^2}{d_{t,k}} \\ \text{s.t} & \sum_k \left(\sum_t d_{t,k}^{1/s}\right)^s \leq 1, \quad d_{t,k} \geq 0 \quad \forall k,t \end{array} \quad (7)$$

with $s = \frac{2-q}{q}$. We can note that the objective function of this optimization problem is smooth and convex and that the feasible domain is convex if $s \leq 1$. After, re-arranging the sums, we have the following equivalent optimization problem:

$$\begin{array}{cc} \min\limits_{\mathbf{d}} & J(\mathbf{d}) = \sum_t J_t(\mathbf{d}) \\ \text{s.t} & \sum_k \left(\sum_t d_{t,k}^{1/s}\right)^s \leq 1, \quad d_{t,k} \geq 0 \quad \forall k,t \end{array} \quad (8)$$

with

$$\begin{aligned} J_t(\mathbf{d}) &= \min_{f_t} C\sum_i H(f_t(x_{i,t}), y_{i,t}) + \sum_k \frac{\|f_{t,k}\|^2}{d_{t,k}} \\ &= -\min_{0\leq\alpha_t\leq C, \alpha_t^{\mathrm{T}} y=0} \frac{1}{2}\alpha_t^{\mathrm{T}} \mathbf{G}_t(\mathbf{d})\alpha_t - \alpha_t^{\mathrm{T}} 1 \end{aligned} \quad (9)$$

where $[\mathbf{G}_t(\mathbf{d})]_{i,j} = y_{i,t}y_{j,t}\sum_k d_{t,k}K_k(x_{i,t}, x_{j,t})$ and $\{\alpha_t\}$ are the vectors of Lagrangian multipliers related to the Hinge loss in problem $J_t(\mathbf{d})$. The second equality of equation 9 is due to Lagrangian duality and the strong duality of an SVM problem. We can note from these equations that for a fixed matrix $\mathbf{d}$ (matrix with entries $d_{t,k}$), each task can be trained independently.

This latter formulation shows how our sparse multi-task SVM problem is related to Multiple Kernel Learning (MKL) problem. At first, we remark that Equations (8-9) boil down to be the MKL problem when only one single task is considered [38]. When several tasks are in play, the matrix $\mathbf{d}$ makes explicit that tasks are linked through their shared sparse kernel representation. Equations (8-9) suggest the use of similar algorithms than those proposed for solving MKL problem, for instance a reduced gradient algorithm as in SimpleMKL [38] or a Semi-Infinite programming approach as proposed by Sonnenburg et al. [42]. Instead of adapting these methods to our problem, we present in the sequel a simple approach for solving problem (8).

B. An alternate optimization algorithm for the $\ell_1 - \ell_q$ case

We introduce one of our contribution of this work which is a simple iterative algorithm based on block-coordinate descent for solving the $\ell_1 - \ell_q$ problem. We show that such a coordinate descent approach boils down to an alternate optimization scheme which provably converges to the minimizer of the problem.

At first let us define the objective function of our problem (7) as

$$R(\mathbf{d}, \mathbf{f}) = C\sum_{t,i} H(f_t(x_{i,t}), y_{i,t}) + \sum_{t,k} \frac{\|f_{t,k}\|^2}{d_{t,k}} \quad (10)$$

where $\mathbf{f}$ defines the set of all functions $\{f_{t,k}\}$ After appropriate initialization of the weight matrix $\mathbf{d}$, our block-coordinate descent algorithm consists in alternatively minimizing :

(i) problem (10) with respects to $\{\mathbf{f}\}$ while keeping the matrix $\mathbf{d}$ fixed. This step simply consists in solving T single-task SVM problems which, at step v, results in the following decision function for task t :

$$f_t^{(v)}(\cdot) = \sum_{i,k} \alpha_{i,t}^{(v)} y_{i,t} d_{t,k}^{(v-1)} K_k(x_{i,t}, \cdot) + b_t^{(v)}$$

with

$$\alpha_t^{(v)} = \left\{ \begin{array}{ll} \underset{\alpha_{i,t}}{\operatorname{argmin}} & \frac{1}{2}\sum_{i,j} \alpha_{i,t}\alpha_{j,t}G_{i,j,t} - \sum_i \alpha_{i,t} \\ \text{s.t} & \sum_i \alpha_{i,t}y_{i,t} = 0, \\ & 0 \leq \alpha_{i,t} \leq C \quad \forall i \end{array} \right.$$

where $G_{i,j,t} = y_{i,t}y_{j,t}\sum_k d_{t,k}^{(v-1)} K_k(x_{i,t}, x_{j,t})$.

(ii) problem (7) with respects to $\mathbf{d}$ with $\{\mathbf{f}\}$ being fixed. Because of the relation between the $\{\alpha_t^{(v)}\}$ and the $\{f_{t,k}^{(v)}(\cdot) = \sum_i \alpha_{i,t}^{(v)} y_{i,t} d_{t,k}^{(v-1)} K_k(x_{i,t}, \cdot)\}$, this problem is equivalent to solve (4) with, at step v

$$\begin{aligned} a_{t,k} &= \|f_{t,k}^{(v)}\|_{\mathcal{H}_k} \\ &= d_{t,k}^{(v-1)} \sqrt{\sum_{i,j} \alpha_{i,t}^{(v)}\alpha_{j,t}^{(v)} y_{i,t}y_{j,t}K_k(x_{i,t}, x_{j,t})} \\ &= d_{t,k}^{(v-1)} \sqrt{\alpha_t^{(v)\mathrm{T}} \tilde{K}_{k,t} \alpha_t^{(v)}} \end{aligned}$$

where $[\tilde{K}_{k,t}]_{i,j} = y_{i,t}y_{j,t}K_k(x_{i,t}, x_{j,t})$. According to proposition (1), we have a closed-form solution $d_{t,k}^{(v)}$ of this problem given by equation (5) which now writes as :

$$d_{t,k}^{(v)} = \frac{\|f_{t,k}^{(v)}\|_{\mathcal{H}_k}^{\frac{2s}{s+1}} \left(\sum_u \|f_{u,k}^{(v)}\|_{\mathcal{H}_k}^{\frac{2}{s+1}}\right)^{\frac{1-s}{2}}}{\sum_{u'} \left(\sum_u \|f_{u,u'}^{(v)}\|_{\mathcal{H}_k}^{\frac{2}{s+1}}\right)^{\frac{s+1}{2}}} \quad (11)$$

Owing to the convexity and the smoothness of the objective function, such an algorithm should converge towards the minimizer of problem (8). In what follows, we give more details on the descent and convergence properties of this algorithm.

Proposition 2: Suppose that all the Gram matrices $K_{k,t}$ (the matrix of general term $K_k(x_{i,t}, x_{j,t})$) for all tasks are strictly positive definite, given $\mathbf{d}^{(v-1)}$ with $\forall t, k,\ d_{t,k}^{(v-1)} \neq 0$, at each iteration $v > 1$ of the alternate scheme, if $\mathbf{d}^{(v)} \neq \mathbf{d}^{(v-1)}$ then the cost function strictly decreases

$$R(\mathbf{d}^{(v)}, \mathbf{f}^{(v)}) < \min_{\mathbf{f}} R(\mathbf{d}^{(v-1)}, \mathbf{f}) < R(\mathbf{d}^{(v-1)}, \mathbf{f}^{(v-1)}) \quad (12)$$

and we have $d_{t,k}^{(v)} > 0$.

Proof: The proof proceeds by considering that the right and the left inequalities respectively derive from the optimality of the $\alpha^{(v)}$ in step (i) and the $\mathbf{d}^{(v)}$ in step (ii) of the alternating scheme. The details are given in the appendix. ∎

The above proposition makes clear that as iteration goes, the objective value decreases if the algorithm is properly initialized to a matrix with non-zero elements Furthermore, since the objective function is bounded from below, the iterates of the objective value converge. This proposition also suggests that our algorithm can get stuck into a fixed point. However as made clear in the following proposition, the sequence of $\{\mathbf{f}^{(v)}\}$ and $\{\mathbf{d}^{(v)}\}$ also converge and eventually such a fixed point would be the minimizer of our problem.

Proposition 3: For $v \in \mathbb{N}^*$, for $1 \leq q \leq 2$, under the hypothesis that all Gram matrices $K_{k,t}$ are strictly positive definite and $\mathbf{d}^{(1)} \neq 0$, the sequence $\{\mathbf{d}^{(v)}, \mathbf{f}^{(v)}\}$ converges to the minimizer of $R(\mathbf{d}, \mathbf{f})$ subject to the constraints on $\mathbf{d}$:

$$\sum_k \left(\sum_t d_{t,k}^{1/s} \right)^s \leq 1, \quad d_{t,k} \geq 0 \quad \forall k, t$$

Proof: For a sake of clarity, the proof of this proposition has been postponed to the appendix. Globally, it follows the same lines of the convergence proof of the alternate optimization algorithm proposed by Argyriou et al. [2]. ∎

A key point for the convergence of the algorithm is that the weight matrix should be initialized to non-zero value. However, if so, as iteration goes, a given weight $d_{t,k}^{(v)}$ does not strictly vanish. This can be interpreted as a weak point for an algorithm that should provide sparse solutions. However, along the iteration, $d_{t,k}^{(v)}$ may rapidly converge towards zero and can rapidly reach a neglectable value. Details on how we have evaluated solution's sparseness are given in the experimental section.

The computational complexity of this algorithm is difficult to evaluate. However, we know that at each iteration, T SVM trainings and the weight matrix $\mathbf{d}$ computation are needed. Each SVM training scales in $O(n_{t,sv}^3)$ while computing $\mathbf{d}$ is about $O(T \cdot M \cdot n_{t,sv}^2)$ with $n_{t,sv}$ being the number of support vectors related to task t. In practice, we take advantage of warm-start techniques when solving the quadratic programming associated to each SVM task, making the algorithm very efficient even compared to gradient descent techniques coupled with warm-starting [13] similar to those used in SimpleMKL. Numerical experiments given in the sequel will support such a claim.

Many previous works on joint-sparse multi-task learning have been carried to in a linear framework thus leading to efficient algorithms. Here, by considering a kernelized framework, our algorithm still relies on a sequence of SVM trainings. This can be considered as very time-consuming. However, according to very recent works on multiple kernel learning [43], [24], using such a wrapper approach (which first solves an SVM then update the weights $\mathbf{d}$) is still competitive compared to other algorithms. Hence, although we have not carried out extensive comparisons, we believe that our approach is relatively efficient (and at least is better than gradient descent techniques as proved in the experimental section). Note that in a linear framework *i.e* each $\mathcal{H}_k$ is associated to one dimension of $\mathbb{R}^d$, the kernel matrix K_k is a rank one matrix and our algorithm wastes many computational efforts in computing $\sum_t d_{t,k} K_k$ (see step (i)). Hence, instead of computing these kernels, we can directly compute this sum through the inner product of the data. We have implemented this simple trick and named this version of our approach, in the experiments, as the linear alternate optimization. We will see that some interesting gain in computational effort can be obtained.

Although we have focused on SVM and the Hinge loss function, our approach can be applied to any convex loss function as long as the problem with fixed $\mathbf{d}$ can be easily solved. For instance, with a square-loss function, minimizing problem (10) boils down to be a weighted kernel rigde regression. Furthermore, our approach can handle situations where the loss functions for each task are heterogeneous. Indeed, in such cases, we would like to solve

$$\begin{array}{ll} \min\limits_{f_1, \cdots, f_T, \mathbf{d}} & C \sum_{t,i} L_t(f_t(x_{i,t}), y_{i,t}) + \sum_{t,k} \frac{\|f_{t,k}\|^2}{d_{t,k}} \\ \text{s.t} & \sum_k \left(\sum_t d_{t,k}^{1/s} \right)^s \leq 1, \quad d_{t,k} \geq 0 \quad \forall k, t \end{array} \quad (13)$$

where each loss function $L_t(\cdot, \cdot)$ depends on each task and can be either related to a regression or classification problem. This framework has been very recently investigated by Yang et al. and is motivated by applications in genetic association mapping [51]. Our algorithm straightforwardly applies to these problems. Indeed, in our alternate optimization algorithm, the loss functions are taken into account only in the first step, where $\mathbf{d}$ is kept fixed. Thus each task learning decouples and the heterogeneity of loss functions does not pose difficulty.

C. A proximal method for the linear $\ell_1 - \ell_q$ case

Recently, several works have proposed efficient algorithms for penalized linear multi-task learning with $\ell_1 - \ell_2$ or $\ell_1 - \ell_\infty$ norms [30], [11]. These approaches are essentially based on accelerated proximal gradient (APG) method [4]. We have extended these algorithms to the case of $\ell_1 - \ell_q$ norms. Since we have exactly followed the same steps of Chen et al. [11], we only detail how we have numerically computed the proximal operator of the $\ell_1 - \ell_q$ norm.

Let us consider each linear classifier related to a task t as $f_t(x) = w_t^T x + b_t$ and the matrix $W = [w_1, \cdots, w_T] \in \mathbb{R}^{d \times T}$. At each iteration of the AGP algorithm, one has to use the so-called proximal operator which maps a matrix $V \in \mathbb{R}^{d \times T}$ to

the unique minimizer of

$$\min_{X\in\mathbb{R}^{d\times T}} \frac{1}{2}\|X-V\|_F^2 + \lambda\sum_{k=1}^{d}\|X_{k,\cdot}\|_q$$

This problem can be decomposed in d independent problems which consist of

$$\min_{x\in\mathbb{R}^T} \frac{1}{2}\|x-v\|^2 + \lambda\|x\|_q \tag{14}$$

for each dimension of the problem. For $q=\{1,2,\infty\}$, this problem has a closed-form solution which makes the global APG algorithm very efficient. Unfortunately, for $1<q<2$, one has to numerically solve this problem. However, considering a q' such that $1/q+1/q'=1$, it can still be shown that if $\|v\|_{q'}\leq\lambda$ then the solution is 0. In the other case, we have considered subgradient descent method and iterative reweighted least-square (IRLS) [41]. Since we found out that the latter is more efficient due to the simple structure of the problem, our numerical implementation uses such an approach. It is simple to show by writing the optimality conditions of problem 14 that the IRLS algorithm consists at each iteration (z) in updating x according to the formula $x^{(z)}=[P^{(z)}]^{-1}v$ with

$$P^{(z)} = \text{diag}\left(1+\frac{\lambda}{\|x^{(z-1)}\|_q^{q-1}}|x^{(z-1)}|^{q-2}\right)$$

which is just a componentwise vector multiplication.

An important remark is that the accelerated proximal algorithm considered here present a fast convergence rate when the loss function is continuously differentiable with Lipschitz gradient (for instance logistic or square loss). In our case, the Hinge loss is non-smooth therefore, we cannot guarantee any convergence rate. However as Chen et al. [11] also noticed, considering the subgradient of the Hinge loss instead of the gradient leads to very good computational efficiency. As we will show in the experimental section, this AGP algorithm is indeed very fast for $q=2$. For $1<q<2$, although we do not have a closed-form solution of problem (14), the numerical scheme we propose is still very efficient. Regarding sparsity, unlike the alternate optimization algorithm, the solution provided by this proximal approach is exactly sparse up to numerical precision.

D. The non-convex $\ell_p-\ell_q$ case

Now that we are able to solve the sparse MTL problem using a $\ell_1-\ell_q$ mixed-norms, we propose an algorithm which solves the non-convex case where $\ell_p-\ell_q$ (with $0<p<1$ and $1\leq q\leq 2$). For this novel situation, let us rewrite the regularization term as

$$\Omega_{p,q} = \sum_{k=1}^{M} g(\|f_{\cdot,k}\|_q) \text{ with } \|f_{\cdot,k}\|_q = \left(\sum_t \|f_{t,k}\|_{\mathcal{H}_k}^q\right)^{1/q} \tag{15}$$

for the linear case, $\|f_{\cdot,k}\|_q=(\sum_t|W_{k,t}|^q)^{1/q}$, and where the upper level penalty function is $g(u)=u^p, u>0$ with $p<1$. Clearly, this function is non-convex. To address this issue, we investigate the use of majorization-minimization (MM) algorithms [19] which form a general framework for optimizing non-convex objective functions. For our multi-task problem, we propose a majorization that enables us to take advantage of the $\ell_1-\ell_q$ MTL solver that we proposed above. Indeed, since $g(u)$ is concave in its positive orthant, we consider the following linear majorization of $g(\cdot)$ at a given point u_0 :

$$\forall u>0, \qquad g(u)\leq u_0^p + pu_0^{p-1}(u-u_0)$$

Note that this linear majorization can also be obtained from the Fenchel inequality related to the Legendre-Fenchel transformation of the differentiable function $g(u)$ [40]. We could have proposed a tighter majorization of $g(\cdot)$ by using for instance a local quadratic approximation. However, the main advantage of a linear majorization is that it leads to a simple algorithm. Indeed, at iteration z, applying this linear majorization of $g(\|f_{\cdot,k}\|_q)$, around a $\|f_{\cdot,k}^{(z)}\|_q$ yields to a majorization-minimization algorithm for $\ell_p-\ell_q$ multi-task learning which consists at a given $(z+1)th$ iteration, in solving :

$$\min_{f_1,\cdots,f_T} C\sum_{t,i} H(f_t(x_{i,t}),y_{i,t}) + \sum_k p\frac{\|f_{\cdot,k}\|_q}{\|f_{\cdot,k}^{(z)}\|_q^{1-p}}$$

This latter equation shows that, in order to solve the non-convex $\ell_p-\ell_q$ problem using a MM approach, one needs to iteratively solve a weighted $\ell_1-\ell_q$ multi-task problem :

$$\min_{f_1,\cdots,f_T} C\sum_{t,i} H(f_t(x_{i,t}),y_{i,t}) + \sum_{k=1}^{M}\beta_k\|f_{\cdot,k}\|_q \tag{16}$$

where β_k are some coefficients that depend on the current functions $f_{t,k}$. They are defined at the z-th iteration as:

$$\beta_k = \frac{p}{\|f_{\cdot,k}^{(z)}\|^{1-p}}, \quad \forall\, k=1,\cdots,M \tag{17}$$

This definition of the β_k implicitly requires the strict positivity of $\|f_{\cdot,k}\|$. To ensure this condition, a small term ϵ is added to $\|f_{\cdot,k}\|$ in (15). Hence, we use $\beta_k=\frac{p}{\epsilon+\|f_{\cdot,k}^{(z)}\|^{1-p}}$. This trick suggested as well by [8] avoids numerical instabilities and overall prevents from having an infinite regularization term for $\|f_{\cdot,k}\|$. In some other context, this ϵ term can play a smoothing role if chosen adaptively [12]. However, in this work, we have kept it fixed at $\epsilon=0.001$.

Now, the equivalent optimization problem with smooth regularization term is :

$$\begin{array}{cc} \min\limits_{f_1,\cdots,f_T,\mathbf{d}} & C\sum_{t,i} H(f_t(x_{i,t}),y_{i,t}) + \sum_{t,k}\beta_k^2\frac{\|f_{t,k}\|^2}{d_{t,k}} \\ \text{s.t} & \sum_k\left(\sum_t d_{t,k}^{1/s}\right)^s \leq 1, \quad d_{t,k}\geq 0 \quad \forall k,t \end{array} \tag{18}$$

where $s=\frac{2-q}{q}$. Note that the optimality conditions of this problem with respects to $f_{t,k}$ is simply given by the expression $f_{t,k}(\cdot)=\frac{d_{t,k}}{\beta_k^2}\sum_i\alpha_{i,t}y_{i,t}K_k(x_{i,t},\cdot)$. Consequently, at each MM iteration, we have to solve a weighted sparse MTL problem, where the weights are applied to the basis kernels. Hence, problem (18) can be solved using the $\ell_1-\ell_q$ algorithm just by replacing the kernel $K_k(x,x')$ with $\frac{1}{\beta_k^2}K_k(x,x')$.

Details of the $\ell_p-\ell_q$ problem solver are given in Algorithm 1. About its complexity, we can state that since the $\ell_p-\ell_q$

Algorithm 1 $\ell_p - \ell_q$ sparse MTL solver.

$\beta_k = 1$ for $k = 1, \cdots, M$
Compute $K_{k,t}$ kernel matrices for all tasks
repeat
  $K^{\beta}_{k,t} \leftarrow \frac{K_{k,t}}{\beta_k^2}$ for all k
  Solve $\ell_1 - \ell_q$ MTL problem with kernels $K^{\beta}_{k,t}$
  Update β_k using Equation (17)
until convergence of the β's

algorithm is based on n_{iter} iterations of the $\ell_1 - \ell_q$ algorithm (after appropriate rescaling of the kernels), its complexity can be approximated as n_{iter} times the $\ell_1 - \ell_q$ algorithm complexity. However, here again, we can speed-up the convergence of $\ell_p - \ell_q$, algorithm by warm-starting the $\ell_1 - \ell_q$ with results from previous iteration. Empirical experiments have shown that n_{iter} are typically lower than 10.

The local convergence of Algorithm 1 is guaranteed. Indeed, the MM programming approach proceeds by surrogating the concave part of the objective function with its affine majorization at each iteration. Therefore, the minimized function decreases until convergence to at least a local minimum [19].

IV. Numerical experiments

In this section, we present some numerical experiments that demonstrate the utility of using a $\ell_p - \ell_q$ penalty instead of a $\ell_1 - \ell_2$ one. They have been carried out on a toy dataset and on real datasets concerning BCI electro-encephalogram signals classification and protein subcellular localization.

Before delving into the experimental details, we provide some remarks on how we have evaluated the sparsity of our algorithm's output. Let us denote the vector $\mathbf{s}$ of components $s_k = \sum_t d_{t,k}$ or $s_k = \sum_t |W_{k,t}|$ depending on the used algorithm. We define the set $\mathcal{S} = \{k \in 1, \cdots, M : s_k > \gamma\}$ where γ is a threshold that allows us to neglect non-zero components due to numerical errors (diagonal loading of kernels to as to make them positive definite has been set to $1e^{-6}$). For the toy problem, we have set $\gamma = 1e^{-5}$ which we believe is small enough so as to provide rather pessimistic estimation of the vector sparseness. We have also considered an heuristic for adaptively setting γ for our alternate optimization algorithm which provides dense although small outputs. In such a case, we have set $\gamma = 0.01 \cdot \max_k(s_k)$. The rationale behind this heuristic is that kernels or variables that have weights significantly smaller than the largest one do not influence the decision function. As a numerical criterion for sparsity evaluation, for the toy problem, since we know the true relevant variables $\mathcal{S}^\star$, we have considered the F-measure between $\mathcal{S}$ and $\mathcal{S}^\star$. For the other problems, we have evaluated the cardinality of $\mathcal{S}$ using the adaptive threshold. All performances reported are evaluated based on kernels and variables in the set $\mathcal{S}$.

A. Toy dataset

Our aim throughout this first experiment is first to analyze the convergence of our alternate optimization algorithm and then to compare an $\ell_1 - \ell_2$ and an $\ell_p - \ell_2$ (with $p < 1$) penalties in term of classification performance.

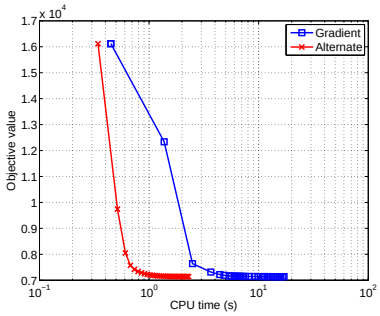
TABLE I
Comparing the computational efficiency (in seconds) of different linear approaches for $\ell_1 - \ell_q$ penalty. The experimental set-up is $d = 100$, $r = 4$, $T = 4$ and $n = 100$.

	$\ell_{1,q}$ penalty	
Methods	$q = 2$	$q = \frac{4}{3}$
Kernel Altern. Opt	1.02 ± 0.20	1.08 ± 0.20
Linear Altern. Opt	0.59 ± 0.06	0.64 ± 0.08
Proximal Descent	0.06 ± 0.01	0.10 ± 0.20

The toy problem is the same as the one used by Obozinski et al. [33]. Each task is a binary classification problem in $\mathbb{R}^d$. Among these d variables, only r of them define a subspace of $\mathbb{R}^d$ in which classes can be discriminated. For these r relevant variables, the two classes follow a Gaussian pdf with mean respectively μ and $-\mu$ and covariance matrices randomly drawn from a Wishart distribution. μ has been randomly drawn from $\{-1, +1\}^r$. The other $d-r$ non-relevant variables follow an *i.i.d* Gaussian probability distribution with zero mean and unit variance for both classes. In this experiment, we are interested in feature selection, thus, for any k, $\mathcal{H}_k$ is the finite dimension subspace built from the kth component of $\mathbb{R}^d$. We have respectively sampled n, n_v and n_t number of examples for training, validation and testing. For some experiments, n is varying, but we have always set $n_v = n$ and $n_t = 5000$. Before learning, the training set has been normalized to zero mean and unit variance and the validation and test sets have been rescaled accordingly.

1) Comparing convergence for $\ell_1 - \ell_2$ penalty: To evaluate the quality of the solution provided by our alternate optimization algorithm when considering a $\ell_1 - \ell_2$ penalty, we have compared it to the solution obtained by a reduced gradient algorithm similar to the one used for SimpleMKL [38]. Both algorithms are wrapper algorithms which in a inner loop solve several SVM problems with fixed kernel and in a outer loop optimize the weights $\mathbf{d}$. The main difference between the two approaches is the way the matrix $\mathbf{d}$ is updated. Note that in our comparison, both methods take advantage of warm-start techniques for successive SVM retrainings. The stopping criterion we have used are the following. For our alternate optimization methods, we stop when $\max(|\mathbf{d}^{(v+1)} - \mathbf{d}^{(v)}|) < 0.001$ (where the $\max$ is considered componentwisely). For the reduced gradient approach, since we can check the KKT conditions without additional computational cost [38], we also imposed that before stopping, the KKT conditions should be satisfied up to a tolerance of 0.1 for each $d_{t,k}$. Here, the comparison has been carried out for a hyperparameter $C = 100$ and for $T = 4$ tasks.

Figure 1 presents the results of this comparison. On the left, we have plotted an example of how the objective value decreases with respects to the CPU time. All the computations have been carried out on a single core of a Bi-Xeon machine with 24 Gb of memory. Source codes are in Matlab. We remark that for a given computational time, using the update equations of $\mathbf{d}$ given in Equation (5) yields to a faster convergence. Such a finding is corroborated by quantitative evaluation



	$d_{t,k}$ initialization	
	Uniform	Random
Time Grad Desc (s)	19.8 ± 7.8	23.8 ± 5.8
Time Altern. Opt. (s)	1.39 ± 0.2	1.5 ± 0.2
Diff. Obj (10^{-4})	1.43 ± 0.7	1.3 ± 0.8
$\|\Delta \mathbf{d}\|_\infty (10^{-3})$	2.1 ± 1.5	2.1 ± 1.3

Fig. 1. Comparing our block-coordinate descent algorithm in its kernel version with a gradient descent approach similar to the one used for multiple kernel learning in SimpleMKL. The left panel shows an example of how the objective value varies with respects to the CPU time. The right table summarizes the time needed for the gradient descent algorithm and for our method before convergence. Relative difference of objective value and the maximal difference between the weights returned by the two algorithms are also reported. All the criteria are averaged over 10 different training sets and with a fixed uniform and random non-zero initializations. The experimental set-up is $d = 100$, $r = 4$, $T = 4$ and $n = 100$ with $C = 100$.

performances given in the right of Figure 1. We show in that table, that for very similar objective values and matrix $\mathbf{d}$, our alternate optimization algorithm converges faster than the reduced gradient approach. The gained factor is about 15.

Table I compares the computational efficiency of several algorithms for solving a linear toy problem. We have compared the kernelized and linear versions of our alternate optimization algorithm as well as the proximal gradient algorithm which is stopped when the variation of its objective value is smaller than 0.001. Here, we have chosen algorithm hyperparameters so as to perfectly recover the sparsity pattern.

Firstly, we can see that in the linear case, the simple trick which consists in directly computing the sum $\sum_k d_{t,k} K_k$ from the examples, yields in a substantial saving of computational efforts for our algorithm (regardless of q). When $q = 2$, the proximal algorithm is very efficient with a gain factor of about 10. For $q = \frac{4}{3}$, the method we proposed for numerically computing the proximal operator still yield to efficient algorithm with gain of about 6.

2) Comparing performance: In this experiment, we aim at showing that by using a $\ell_p - \ell_2$ penalty which provides a more aggressive sparsity pattern, we are able to reduce test error compared to a $\ell_1 - \ell_2$ penalty. We also provide empirical evidence that for this toy problem, the variables that are recovered using the $\ell_p - \ell_2$ penalty are more relevant than those recovered by the $\ell_1 - \ell_2$ one. As a baseline comparison, we have also considered sparse separated SVM (each single SVM is trained according to its task data) and a sparse pooled SVM (a single SVM is trained according to all task data). Since the problem is linear, we have used the accelerated gradient algorithm but we have also checked how the linear version of our alternate optimization approach behaves.

The two penalties have been compared through different experimental situations where we have varied some parameters of the toy problem *e.g* the number of tasks, the number of training examples, the number of relevant variables. Model selection have been included into the comparison. Hence, hyperparameters have been tuned by means of a validation set and a validation error. For both $\ell_1 - \ell_2$ and $\ell_p - \ell_2$ sparse MTL, hyperparameters λ (proximal algorithm) and C (alternate optimization) have been respectively selected among 10 different values logarithmically sampled from the interval $[3, \cdots, 60]$ and $[0.01, \cdots, 100]$. For the $\ell_p - \ell_2$ penalty, sparsity parameter p has also been selected among the value $[0.2, 0.5, 0.75, 0.9]$. For each experimental situations, trials have been replicated 20 times.

Results are summarized in Figure 2. The figure shows that regardless the experimental situations considered, a $\ell_p - \ell_2$ penalty leads to better performances than the $\ell_1 - \ell_2$ one and the ℓ_1 separated SVM (results of the pooled models have not been reported since they are always worse than 0.20). The statistical significance of this claim has been evaluated using a Wilcoxon signed rank test. The test shows that the difference in performance is significant at a level of 0.05 except in few situations (*e.g* first marker of the second and fourth plots from left to right). One can also see that the alternate optimization and the proximal algorithm lead to statistically equivalent performances when using $\ell_p - \ell_2$ penalty except for few cases when the number of training example is small. We have checked that this is due to a model selection problem : the proximal algorithm seems to be more sensitive to the choice of λ. Figure 3 gives a rationale on why the $\ell_p - \ell_2$ penalty performs better. We have evaluated the F-measure of $\mathcal{S}$ compared to the true relevant variables. The $\ell_p - \ell_2$ penalty does a better job in recovering relevant variables (even when the $\ell_1 - \ell_q$ algorithm is our non-exactly sparse alternate optimization algorithm). The use of an adaptive threshold also lead to good estimation of relevant variables. Missing models in the plots have F-measures always lower than 0.3. This means for instance that an $\ell_1 - \ell_2$ penalty trained with an alternate optimization algorithm have too many weights so that $\sum_t d_{t,k} > 1e-5$. As the number r of true relevant variable increases, the gap of performance between $\ell_1 - \ell_2$ and $\ell_p - \ell_2$ penalties tends to reduce. This can be easily justified since in this case, the $\ell_p - \ell_2$ penalty becomes too aggressive and tends to discard relevant variables. We can thus conclude that the use of an $\ell_p - \ell_2$ penalty is more adapted to situations where the number of relevant variables is small compared to the problem dimensionality. This point is also illustrated in Figure 4. We can note there that the model selection procedure tends to choose larger value of p as the number of relevant variables increases. This clearly shows that if no prior knowledge on the sparsity level is available, adaptive data-driven penalty norms

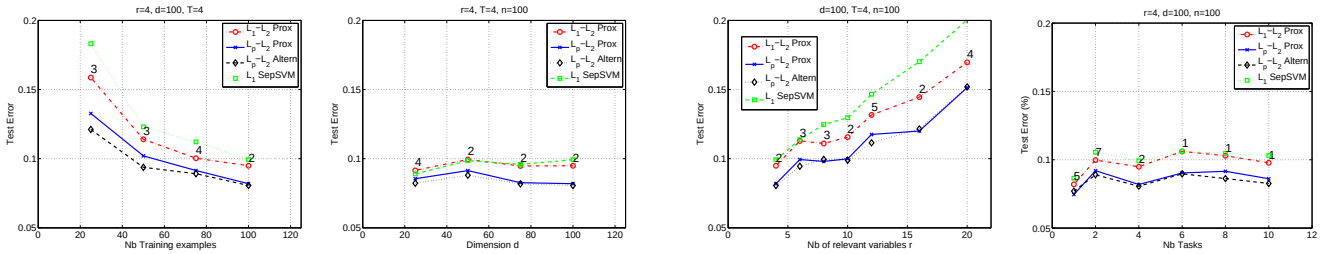


Fig. 2. Performance (test error) comparisons between $\ell_1 - \ell_2$ (red, dash-dotted), $\ell_p - \ell_2$ (blue, solid) multi-task models trained with proximal algorithms, $\ell_p - \ell_2$ trained with alternate optimization (black, dotted) and ℓ_1 separated models (green, dashed) for different experimental situations. For each experimental situation, we have kept fixed all except one of parameters : (from left to right) number of training examples n, problem dimension d, number of relevant variables r, number of tasks T. The number given next each marker represents the number of times (out of 20) the $\ell_1 - \ell_2$ penalty provides a better performance than the $\ell_p - \ell_2$ penalty both trained with proximal algorithms.

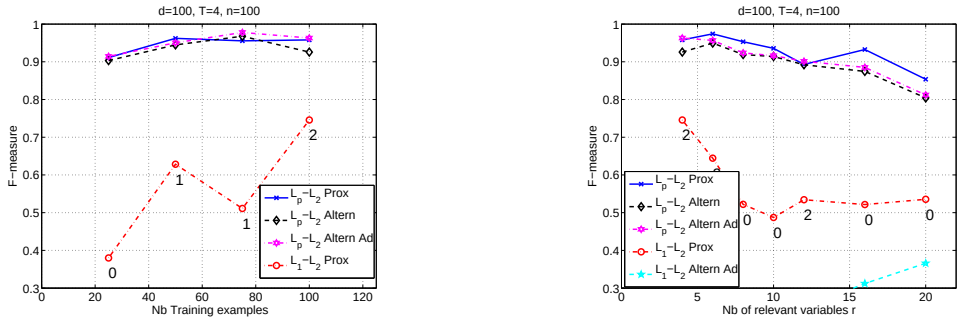


Fig. 3. Evaluation of the recovered sparsity pattern through the F-measure of retrieved variables. (left) F-measure wrt number of training examples. (right) F-measure wrt the number of relevant variables. Prox, Altern and Altern Ad respectively stand for models trained with proximal and alternate optimization and alternate optimization with adaptive thresholding for sparsity evaluation. When an algorithm exactly recovers the set of true relevant variables, F-measure should be equal to 1.

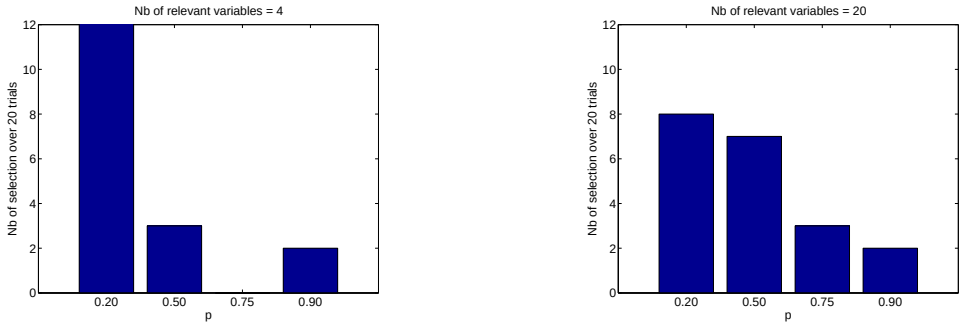


Fig. 4. Illustration of how when using an $\ell_p - \ell_2$ penalty, the learned decision functions adapt themselves to the data at hand ($d = 100$, $T = 4$ and $n = 100$). The two plots show the number of times a given p has been selected by validation. On the left, the number of relevant variables r is equal to 4 while on the right plot, we have $r = 20$. We note that as the number of relevant variables increases, the model selection procedure tends to choose a larger p.

are valuable.

B. BCI P300 single trial problem

We also illustrate the usefulness of sparse Multi-Task learning on a Brain-Computer Interface problem. Indeed, sparse MTL can be very relevant to BCI because of the need for channel/variable selection and because of the data variability with respects to different subjects. Our objective here is to show that sharing information between subjects through sparse multi-task learning can lead to improvement in performance while reducing the number of variables involved in the recognition task. The dataset we consider is the BCI P300 Speller dataset used by Hoffmann et al. [16]. For this BCI paradigm, a subject is presented a six-symbol matrix where symbols are flashed in random order. A large P300 evoked potential can be recorded in the electro-encephalogram (EEG) signals recorded from the subject's scalp in response to the intensification of the desired symbol. Each trial corresponds to the EEG signals related to the response of a given flash. Hence, the classification task is to recognize whether this trial contains or not a P300 evoked potential. The datasets involve 9 subjects including disabled ones. All preprocessing steps we used are those described by Hoffmann et al. The steps include : referencing, band-pass filtering, downsampling, single trial extraction, windsorizing, scaling and feature vector construction. In our experiments, we have restricted ourselves to a 8 channel configuration (Fz, Cz, Pz, Oz, P7, P3, P4, P8) which leads after downsampling

TABLE II
AVERAGE AUC PERFORMANCES OF 7 DIFFERENT ALGORITHMS ON THE BCI DATASET. THE NUMBER OF VARIABLES THAT HAVE BEEN KEPT IN THE DECISION FUNCTION IS ALSO GIVEN. SEPSVM AND SEP ℓ_1 SVM RESPECTIVELY DENOTE A SVM AND A SPARSE SVM CLASSIFIER TRAINED ON ISOLATED DATASETS FOR EACH SUBJECT. FULLSVM AND FULL ℓ_1 SVM ARE THE CLASSIFIERS TRAINED ON ALL EXAMPLES. TOP) $n = 300$. BOTTOM) $n = 400$.

	$\text{MTL}_{1,2}$	$\text{MTL}_{p,2}$	$\text{MTL}_{1,q}$	SepSVM	Sep ℓ_1 SVM	Full SVM	Full ℓ_1 SVM
AUC	76.5 ± 0.6	76.1 ± 0.5	76.5 ± 0.6	75.6 ± 0.8	73.4 ± 1.3	67.1 ± 0.6	67.0 ± 0.6
# Var	191 ± 26	134 ± 33	201 ± 23	256	118 ± 30	256	238 ± 7
p-val	0.00019	0.00897	0.00014	-	0.00006	0.00006	0.00006

	$\text{MTL}_{1,2}$	$\text{MTL}_{p,2}$	$\text{MTL}_{1,q}$	SepSVM	Sep ℓ_1 SVM	FullSVM	Full ℓ_1 SVM
AUC	78.2 ± 0.6	77.8 ± 0.7	78.3 ± 0.6	77.4 ± 0.9	75.1 ± 1.3	67.9 ± 0.6	67.9 ± 0.6
# Var	205 ± 18	150 ± 35	209 ± 16	256	149 ± 32	256	249 ± 2
p-val	0.00009	0.007	0.00009	-	0.00009	0.00009	0.00009

to a feature vector of size 256. The number of single trials available for each subject is about 3300. Note that the datasets and the preprocessing algorithms are available on the EPFL BCI group website (http://bci.epfl.ch/p300).

Instead of training a classifier separately on each subject as in Hoffmann et al., we have trained linear classifiers, using our alternate optimization algorithm, for all subjects all-together using our multi-task approach (one task = one subject). Three types of penalty for the multi-task learning are considered : an $\ell_1 - \ell_2$, an $\ell_p - \ell_2$ and a $\ell_1 - \ell_q$ penalty. As a comparison, we have also learned a sparse and a classical SVM trained on a single subject, and a sparse and a classical SVM trained on all subject data. Sparse SVM has been obtained using a ℓ_1 multiple kernel learning approach where each feature is related to a kernel [38]. For selecting the hyperparameter of all algorithms, we have considered a validation approach. For all subjects, we have randomly split the available examples in 3 sets : n examples for training and validation and the rest for testing. For the experiment, we have set the training set size to $n = 300$ and $n = 400$. C and p have been selected from the same sets as the previous experiments while $q \in \{\frac{4}{3}, \frac{5}{3}, \frac{20}{11}\}$. Note that using a small part of the examples for training is motivated by the use of ensemble of SVM (that we do not consider here) [39] at a later stage of the EEG classification procedure. The performance is measured by AUC, due to the post-processing that is done throughout repetitions in the P300 : as the final decision regarding letters is taken after several trials, the correct row and column should receive high scores to correctly identify the letter.

Results averaged over 10 trials are presented in Table II. The baseline performance is the one provided by a classical SVM trained on a single subject (SepSVM). When comparing performance of a given approach to that baseline, a Wilcoxon sign-rank test has been evaluated and the p-value being reported. We remark that training with all examples lead to significantly worse performances compared to the baseline. However, when learning through a multi-task approach, we achieve a slight but significant increase of performance. Interestingly, the three multitask approaches yield a significant dimensionality reduction while slightly improving performances. When comparing performances of the three different multitask penalties, we see that AUC scores are equivalent but the $\ell_p - \ell_2$ penalty need far fewer variables.

C. Protein subcellular localization

This last real-world experiment further highlights the utility of our approach in a kernel selection context. Indeed, we consider here two datasets for bacterial protein localization : the PSORT+ dataset contains four classes and 541 examples and the other, called PSORT-, has five classes with 1444 examples. For each datasets, 69 kernels have been computed and they are publicly available on http://www.fml.tuebingen.mpg.de/raetsch/suppl/protsubloc. This website also provides some information about the post-processing for performance evaluations. This classification problem is actually a multiclass problem that we address through pairwise binary classification. In order to reduce the number of kernels to compute, we are interested in joint kernel selection for all pairwise problems. Hence, we have considered a one-against-all framework where each one-against-all problem is a task. Note that we could have also considered that each task is related to a one-against-one pairwise problem, but in order to be compliant with the experimental setting of Zien and Ong and the way they evaluate performances, we have considered the one-against-all framework.

In our experiments, we have compared sparse MTL with $\ell_1 - \ell_2$ and sparse MTL with $\ell_p - \ell_2$ and $\ell_1 - \ell_q$. Due to the multiclass nature of the problem, comparisons with pooled and independent models are not possible. Data permutations as well as the $80\% - 20\%$ splitting into training and testing sets are also provided by Zien and Ong [54]. Hyperparameters C, p anq q have been selected through a validation method by randomly splitting the training set then training and validating on the resulting splits. C has been selected from 10 values logarithmically sampled from the interval $[0.01, 100]$ while p and q are respectively chosen from $\{0.5, 0.75, 0.9\}$ and $\{\frac{4}{3}, \frac{5}{3}, \frac{20}{11}\}$

Averaged over 10 trials results are given in Figure 5. We have also given the performance achieved by the multiclass MKL algorithm of Zien and Ong [54]. This MKL algorithm learns a linear combination of kernels that is jointly optimal for all winner-takes-all decision functions. Hence, their method is very similar to our sparse MTL learning with a $\ell_1 - \ell_2$ penalty. Results show that our algorithms and the different penalties yield to similar accuracy performances (according to a Wilcoxon sign-rank test) and they are competitive with the multiclass MKL of Zien and Ong. However, once again using

TABLE III
EXAMPLES OF AVERAGE TIME NEEDED FOR OUR ALTERNATE OPTIMIZATION ALGORITHM, THE GRADIENT DESCENT APPROACH AND PROXIMAL DESCENT FOR PRODUCING A SOLUTION OF A $\ell_1 - \ell_q$ MTL PROBLEM. NOTE THAT FOR PSORT PROBLEMS, LINEAR ALGORITHMS ARE NOT CONSIDERED SINCE WE ARE DEALING WITH KERNELS ON STRUCTURED DATA. FOR THE BCI PROBLEM, $n = 300$.

	Altern. Opt			
Data	Linear	Kernel	Gradient	Proximal
BCI, $q = 2$	18.8 $\pm$ 0.8	71.8 $\pm$ 4.2	309 $\pm$ 190	0.98 $\pm$ 0.17
BCI, $q = \frac{4}{3}$	14.8 $\pm$ 0.7	46.8 $\pm$ 2.3	-	4.1 $\pm$ 0.5
PSORT +, $q = 2$	-	57.4 $\pm$ 6	110 $\pm$ 35	-
PSORT +, $q = \frac{4}{3}$	-	62.2 $\pm$4	127 $\pm$ 41	-
PSORT -, $q = 2$	-	350.5 $\pm$ 18	1450 $\pm$ 400	-
PSORT -, $q = \frac{4}{3}$	-	364.5 $\pm$ 32	1480 $\pm$ 300	-

an adaptive value of p or q compared to a pre-defined choice of $p = 1$ and $q = 2$ leads to significantly fewer selected kernels (up to a level of 0.05). We can also point out that the validation approach tends to select a value of p and q respectively of 0.9 and $\frac{4}{3}$. By using a value of p slightly smaller than 1, we thus achieve a substantial reduction of the number of selected kernels (compared to $\ell_1 - \ell_2$). Right plot of Figure 5 gives an example of the resulting weights $d_{t,k}$ for different penalties for the PSORT+ problem. We remark that some kernels (*e.g the third, seventh and eighth)* have been selected by the $\ell_1 - \ell_2$ and $\ell_1 - \ell_q$ penalties but have been discarded by the $\ell_p - \ell_2$ one.

D. Computational efficiency on the real-world problems

In order to have an idea on the computational efficiency of our algorithms for $\ell_1 - \ell_q$ multi-task problems, we have reported in Table III, the time they need for converging. we also provide results for gradient descent approach (when $q = 2$) and proximal gradient descent for linear problems such as the BCI problem. Stopping criterion are the same as those used for the toy dataset problem. Note that while our alternate optimization and the gradient descent algorithms solve the same problem, the proximal algorithm solves an equivalent one. However, since it is hard to find the closed-form relation between the hyperparameters producing the same solution, for a relatively fair comparison, we have chosen these hyperparameters so as to have a similar level of sparsity. The results we obtain are on the same lines of those obtained for the toy problem. Regardless of the situation, gradient descent is the less efficient approach. When compared to the kernel version of the alternate optimization algorithm, the loss in computational effort is from 2 to 4. When comparing linear methods, proximal descent is the most efficient method with a gain in computation of the order of 3 in the worst case.

V. CONCLUSION

In this paper, we investigated the use of mixed-norms for multi-task SVM with joint sparsity constraint. We went beyond convexity and proposed a large class of mixed-norm penalty based on $\ell_p - \ell_q$ norm, (with $p \leq 1$ and $1 \leq q \leq 2$). For solving the resulting optimization problem, we first derive a general algorithm which addresses the convex case $p = 1$ and the use of multiple kernels. For the linear case, a more efficient proximal algorithm have been investigated. For the case $p < 1$, we fitted the optimization problem into the Majorization-Minimization framework, and proposed an iterative reweighted version of the $\ell_1 - \ell_q$ algorithm. Experimental results on toy data set brought evidence that $\ell_p - \ell_q$ penalties lead to enhanced performance and better sparsity pattern compared to a $\ell_1 - \ell_2$ penalty especially in situations where a large number of variables are in play. Then, results on real-world datasets from various domains have shown the potential (in terms of accuracy and variable selection) of our approach on applications where variable or kernel selections are of primary importance.

Now we plan to extend our efforts in the following directions. While this paper is essentially a proof of concept that adapting the penalty to the problem at hand is an interesting approach, up to now we have dealt with this adaptivity only through validation methods and grid search on p and q. Now, for addressing efficiently such an adaptivity, we will focus on algorithmic methods that would allow us to jointly select p and q. Notably, we project to investigate regularization path and continuation methods. Furthermore, we will also consider faster algorithms that can handle large-scale problems. Then, future works will also aim at theoretically analyzing the consistency of our $\ell_p - \ell_q$ approach for variable selection.

VI. APPENDIX

A. Equivalence between problems (2) and (6)

The equivalence between these two problems comes from two properties : i) the equivalence between constrained and regularized convex optimization problem and ii) the equivalence of optimization problems when objective functions or constraints are transformed through the composition of a monotonically increasing function.

Here we give the proof for a simple case without loss of generality. Let us consider the following optimization problems with $F(\cdot)$ and $G(\cdot)$ being two strictly convex functions of $\mathbb{R}^d$,

$$(R) : \min_{x \in \mathbb{R}^d} F(x) + \lambda G(x) \quad \text{and} \quad (C) : \min_{x \in \mathbb{R}^d, G(x) \leq \tau} F(x)$$

with λ and τ some parameters. These two problems are equivalent in the sense that for any λ, there exists a τ such that the minimizers of (R) and (C) are the same [48]. Now, according to the same notion of equivalence, problem (C) is also equivalent to

$$(C^2) : \min_{x \in \mathbb{R}^d, G(x)^2 \leq \tau^2} F(x)$$

owing to the monotonically increasing transformation of the constraints [7]. Since (C^2) is equivalent to

$$(R^2) : \min_{x \in \mathbb{R}^d} F(x) + \lambda_2 G(x)^2$$

where λ_2 is another parameter. We thus have equivalence between (R) and (R^2).

Data	$\text{MTL}_{1,2}$	$\text{MTL}_{p,2}$	$\text{MTL}_{1,q}$	MCMKL
PSORT +	93.87 ± 2.82	93.62 ± 3.04	93.88 ± 2.73	93.8
# Kernels	15.4 ± 1.17	7.4 ± 1.42	15.9 ± 1.05	18
PSORT -	95.92 ± 1.35	95.90 ± 1.12	96.02 ± 1.33	96.1
# Kernels	12.9 ± 0.31	7.5 ± 0.85	12.8 ± 0.42	14

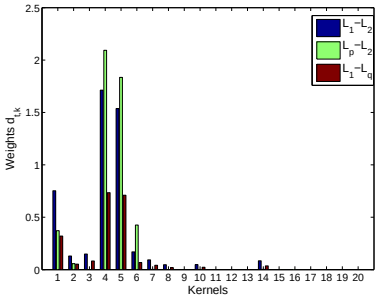


Fig. 5. (left) Average F1 score and number of selected kernels using our algorithms with different penalties on protein subcellular localization problems. Scores of the multiclass MKL of Zien and Ong. [54] have also been reported. (right) Example of kernel weights resulting from the different penalties for the PSORT+ problem. For a sake of clarity, we have restricted the plot to the 20 first kernels. We note that different penalties lead to different sets of selected variables and for some variables that have been selected by all three models, the weighting can largely differ. The number of kernels selected by Zien and Ong's method has not been explicitly reported and we have extrapolated them from one of their figure in [34].

B. Descent property of the $\ell_1 - \ell_q$ algorithm

For the right strict inequality, positive definiteness of $\mathbf{G}_t$ implies strict convexity and the solution's uniqueness of the minimization problem in f_t related to all tasks. Besides, $f_t^{(v)} = f_t^{(v-1)}$ would imply according to Equation (11) that $\mathbf{d}^{(v)} = \mathbf{d}^{(v-1)}$. Hence, since by hypothesis $\mathbf{d}^{(v)} \neq \mathbf{d}^{(v-1)}$, we have $f_t^{(v)} \neq f_t^{(v-1)}$. These points lead to the strict inequality :

$$\min_{\mathbf{f}} R(\mathbf{d}^{(v-1)}, \mathbf{f}) < R(\mathbf{d}^{(v-1)}, \mathbf{f}^{(v-1)})$$

Before showing the left strict inequality, the properties relating $\mathbf{d}^{(v)}$ and $\mathbf{d}^{(v-1)}$ are proved. We can first note that if $d_{t,k}^{(v-1)} = 0$ then $d_{t,k}^{(v)} = 0$. But if $d_{t,k}^{(v-1)} > 0$ then strict positivity of $d_{t,k}^{(v)}$ stems from the positive definiteness of matrices $K_{k,t}$. Indeed, since we have

$$\|f_{t,k}^{(v)}\|^2 = [d_{t,k}^{(v-1)}]^2 \alpha_t^{(v)\mathrm{T}} K_{k,t} \alpha_t^{(v)} > 0$$

which, according to Equation (11) yields to $d_{t,k}^{(v)} > 0$.

Now, since $d_{t,k}^{(v-1)} > 0$, the left strict inequality of equation (12) naturally comes from the strict convexity and the solution's uniqueness of problem (4).

C. Convergence of the $\ell_1 - \ell_q$ algorithm

Here we present the proof of convergence of our algorithm which follows the lines of the one of Argyriou et al. [2].

Let us define

$$\begin{aligned} S(\mathbf{f}) &:= R(\mathbf{d}(\mathbf{f}), \mathbf{f}) \\ &= C \sum_{t,i} H(f_t(x_{i,t}), y_{i,t}) + \left(\sum_t \left(\sum_k \|f_{t,k}\|_{\mathcal{H}_k}^q \right)^{1/q} \right)^2 \end{aligned}$$

The second equality in the definition of $S(\mathbf{f})$ naturally stems from Proposition 1 since $\mathbf{d}^{(1)} \neq 0$. For q so that $1 \leq q \leq 2$, the mixed-norm regularizer term $\sum_t \left(\sum_k \|f_{t,k}\|_{\mathcal{H}_k}^q \right)^{1/q}$ is convex as a sum of convex functions. The composition with a strictly increasing and strictly convex function (the square term) on $\mathbb{R}_+$ makes the overall regularizer strictly convex. Thus, even though the loss function $H(\cdot,\cdot)$ is just convex, $S(\mathbf{f})$ is still strictly convex and admits an unique minimizer.

Now let us introduce

$$g(\mathbf{f}) := \min_{\mathbf{u}} \{R(\mathbf{d}(\mathbf{f}), \mathbf{u})\} \tag{19}$$

we show in the sequel that the function $g(\mathbf{f})$ is continuous. This comes from the fact that the function :

$$G(\mathbf{d}) := \min_{\mathbf{u}} R(\mathbf{d}, \mathbf{u})$$

is continuous and differentiable. Indeed, for a given $\mathbf{d}$, the optimization problem defining $G(\mathbf{d})$ is just T independent SVM problems; each task being related to each SVM problem with a kernel $\sum_k d_{t,k} K_{t,k}$. Results from multiple kernel learning have shown that each SVM objective value, for a task t, is a continuous and differentiable function [38] with respects to $\{d_{t,k}\}$. From this point, we can conclude that $G(\mathbf{d})$ is also continuous and differentiable. Thus $g(\mathbf{f})$ is continuous as a composition of continuous functions.

Now let us show that the sequence $\{S(\mathbf{f}^{(v)}) : v \in \mathbb{N}\}$ converges. We can observe that since $S(\mathbf{f}) = R(\mathbf{d}(\mathbf{f}), \mathbf{f})$ and $\mathbf{d}(\mathbf{f})$ minimizes $R(\cdot, \mathbf{f})$, we have the following inequalities :

$$S(\mathbf{f}^{(v+1)}) \leq g(\mathbf{f}^{(v)}) \leq S(\mathbf{f}^{(v)}).$$

Hence, the sequence $\{S(\mathbf{f}^{(v)}) : v \in \mathbb{N}\}$ is not increasing and since the loss function H is bounded from below, it is bounded. Thus as v goes to ∞, the sequence $S(\mathbf{f}^{(v)})$ converges to a value $S^\star$. From the continuity and boundedness of $S(\mathbf{f}^{(v)})$, we can also deduce that the mixed-norm regularizer and the sequence $\{\mathbf{f}^{(v)}\}$ are bounded where boundedness of $\{\mathbf{f}^{(v)}\}$ is understood according to some norm (*e.g* the norm induced by the inner product $\langle \mathbf{f}, \mathbf{f} \rangle = \sum_{t,k} \langle f_{t,k}, f_{t,k} \rangle_{\mathcal{H}_k}$). As a consequence, there exists a subsequence $\{\mathbf{f}^{(v_i)} : i \in \mathbb{N}\}$ that converges towards $\mathbf{f}^\star$.

Now, we show that $\mathbf{f}^\star$ is a minimizer of $R(\cdot,\cdot)$. Consider any convergent subsequence $\{\mathbf{f}^{(v_i)} : i \in \mathbb{N}\}$ of $\{\mathbf{f}^{(v)} : v \in \mathbb{N}\}$. Since $S(\mathbf{f}^{(v_i+1)}) \leq g(\mathbf{f}^{(v_i)}) \leq S(\mathbf{f}^{(v_i)})$, $g(\mathbf{f}^{(v_i)})$ converges towards $S^\star$. By the continuity of functions $S(\mathbf{f})$ and $g(\mathbf{f})$, we thus have $g(\mathbf{f}^\star) = S(\mathbf{f}^\star)$. This implies that $\mathbf{f}^\star$ is a minimizer of $R(\mathbf{d}(\mathbf{f}^\star), \cdot)$ because $R(\mathbf{d}(\mathbf{f}^\star), \mathbf{f}^\star) = S(\mathbf{f}^\star)$. Furthermore, $\mathbf{d}(\mathbf{f}^\star)$ is the minimizer of $R(\cdot, \mathbf{f}^\star)$ subject to constraints on $\mathbf{d}$. Thus, since the objective function $R(\cdot,\cdot)$ is smooth and strictly convex, the pair $(\mathbf{d}(\mathbf{f}^\star), \mathbf{f}^\star)$ is a stationary point of $R(\cdot,\cdot)$ and thus its unique minimizer.

At this point, we have shown that any convergent subsequent of $\{\mathbf{f}^{(v)} : v \in \mathbb{N}\}$ converges to the minimizer of $R(\cdot,\cdot)$. Then since $S(\mathbf{f})$ is continuous and $\{\mathbf{f}^{(v)} : v \in \mathbb{N}\}$ is bounded, it follows that the whole sequence converges towards the minimizer of $R(\cdot,\cdot)$.

REFERENCES

[1] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," in *Advances in Neural Information Processing Systems*, 2007.

[2] ——, "Convex multi-task feature learning," *Machine Learning*, vol. 73, no. 3, pp. 243–272, 2008.

[3] F. Bach, G. Lanckriet, and M. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proceedings of the 21st International Conference on Machine Learning*, 2004, pp. 41–48.

[4] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, pp. 183–202, 2009.

[5] J. Bi, T. Xiong, S. Yi, M. Dundar, and B. Rao, "An improved multi-task learning approach with applications in medical diagnosis," in *Proceedings of the 18th European Conference on Machine Learning*, 2008.

[6] S. Bickel, J. Bogojeska, T. Lengauers, and T. Scheffer, "Multi-task learning for hiv therapy screening," in *ICML '08: Proceedings of the 25th international conference on Machine learning*. New York, NY, USA: ACM, 2008, pp. 56–63.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[8] E. Candès, M. Wakin, and S. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *J. Fourier Analysis and Applications*, vol. 14, pp. 877–905, 2008.

[9] R. Caruana, "Multi-task learning," *Machine Learning*, vol. 28, pp. 41–75, 1997.

[10] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal Scientific Comput.*, vol. 20, no. 1, pp. 33–61, 1999.

[11] X. Chen, W. Pan, J. Kwok, and J. Carbonell, "Accelerated gradient method for multi-task sparse learning problem," in *Proceedings of the International Conference on Data Mining*, 2009.

[12] I. Daubechies, R. DeVore, M. Fornasier, and S. Gunturk, "Iteratively reweighted least squares minimization for sparse recovery," *Commun. Pure Appl. Math*, vol. to appear, 2009.

[13] D. DeCoste and K. Wagstaff., "Alpha seeding for support vector machines," in *International Conference on Knowledge Discovery and Data Mining*, 2000.

[14] D. Dunson, Y. Xue, and L. Carin, "The matrix stick-breaking process : flexible bayes meta analysis," *Journal of American Statistical Association*, vol. 103, no. 481, pp. 317–327, 2008.

[15] T. Evgeniou and M. Pontil, "Regularized multi-task learning," in *Proceedings of the tenth Conference on Knowledge Discovery andData minig*, 2004.

[16] U. Hoffmann, J. Vesin, T. Ebrahimi, and K. Diserens, "An efficient p300-based brain-computer interface for disabled subjects," *Journal of Neuroscience Methods*, vol. 167, no. 1, pp. 115–125, 2008.

[17] M. Hu, Y. Chen, and J. Kwok, "Building sparse multi-kernel svm classifiers," *IEEE Trans. On Neural Networks*, vol. 20, no. 5, pp. 827–839, 2009.

[18] J. Huang, S. Ma, H. Xie, and C. Zhang, "A group bridge approach for variable selection," *Biometrika*, vol. 96, no. 2, pp. 339–355, 2009.

[19] D. Hunter and K. Lange, "A tutorial on MM algorithms," *The American Statistician*, vol. 58, pp. 30–37, 2004.

[20] H. D. III, "Bayesian multitask learning with latent hierarchies," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2009.

[21] L. Jacob, F. Bach, and J.-P. Vert, "Clustered multi-task learning," in *Advances in Neural Information Processing Systems*, 2008.

[22] T. Jebara, "Multi-task feature and kernel selection for svms," in *Proceeding of the 21st International Conference on Machine Learning*, 2004.

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

[24] M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Müller, and A. Zien, "Efficient and accurate lp-norm multiple kernel learning," in *Advances in neural information processing systems 22*, 2009.

[25] K. Knight and W. Fu, "Asymptotics for lasso-type estimators," *Annals of Statistics*, vol. 28, pp. 1356–1378, 2000.

[26] D. Lee, K.-H. jung, and J. Lee, "Constructing sparse kernel machines using attractors," *IEEE Trans. On Neural Networks*, vol. 20, no. 4, pp. 721–729, 2009.

[27] H. Liu, J. Lafferty, and L. Wasserman, "Non parametric regression and classification with joint sparsity constraints," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2009.

[28] H. Liu, M. Palatucci, and J. Zhang, "Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery," in *Proceedings of the Twenty-sixth International Conference on Machine Learning*, 2009.

[29] H. Liu and J. Zhang, "On the estimation and variable selection consistency of the sum of q-norm regularized regression," Department of Statistics, Carnegie Mellon University, Tech. Rep., 2009.

[30] J. Liu, S. Ji, and J. Ye, "Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization," in *Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence*, 2009.

[31] K. Lounici, A. Tsybakov, M. Pontil, and S. V. de Geer, "Taking advantage of sparsity in multi-task learning," in *Proceedings of Computational Learning Theory*, 2009.

[32] C. Micchelli and M. Pontil, "Learning the kernel function via regularization," *Journal of Machine Learning Research*, vol. 6, pp. 1099–1125, 2005.

[33] G. Obozinski, B. Taskar, and M. Jordan, "Joint covariate selection and joint subspace selection for multiple classification problems," *Statistics and Computing*, vol. to appear, 2009.

[34] C. Ong and A. Zien, "An automated combination of kernels for predicting protein subcellular localization," in *Proceedings of the 8th Workshop on Algorithms in Bioinformatics (WABI 2008)*, 2008, pp. 186–197.

[35] S. Ozawa, A. Roy, and D. Roussinov, "A multitask learning model for online pattern recognition," *IEEE Trans. on Neural Networks*, vol. 20, no. 3, pp. 430–445, 2009.

[36] A. Quattoni, X. Carreras, M. Collins, and T. Darrell, "An efficient projection for $\ell_{1,\infty}$ regularization," in *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[37] A. Quattoni, M. Collins, and T. Darrell, "Transfer learning for image classification with sparse prototype representations," in *Proceedings of CVPR*, 2008.

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

[39] A. Rakotomamonjy and V. Guigue, "BCI competition III: Dataset II - ensemble of SVMs for BCI P300 speller," *IEEE Trans. Biomedical Engineering*, vol. 55, no. 3, pp. 1147–1154, 2008.

[40] R. Rockafellar, *Convex Analysis*. Princeton University Press, 1996.

[41] R. Saab, R. Chartrand, and Özgür Yilmaz, "Stable sparse approximations via nonconvex optimization," in *33rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2008.

[42] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1531–1565, 2006.

[43] T. Suzuki and R. Tomioka, "Spicymkl," arXiv, Tech. Rep. 0909.5026, 2009.

[44] M. Szafranski, Y. Grandvalet, and A. Rakotomamonjy, "Composite kernel learning," in *Proceedings of the 22nd International Conference on Machine Learning*, 2008.

[45] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society*, vol. 46, pp. 267–288, 1996.

[46] M. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[47] D. Tzikas, A. Likas, and N. Galatsanos, "Sparse bayesian modeling with adaptive kernel learning," *IEEE Trans. on Neural Networks*, vol. 20, no. 6, pp. 926–937, 2009.

[48] P. Weiss, "Algorithmes rapides d'optimisation convexe. applications à la reconstruction d'images et à la détection de changements." Ph.D. dissertation, Université de Nice Sophia-Antipolis, 2008.

[49] T. Xiong, J. Bi, B. Rao, and V. Cherkassky, "Probabilistic joint feature selection for multi-task learning," in *Proceedings of SIAM International Conference on Data Mining*, 2006.

[50] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram, "Multi-task learning for classification with dirichlet process priors," *Journal of Machine Learning Research*, 2007.

[51] X. Yang, S. Kim, and E. Xing, "Heterogeneous multitask learning with joint sparsity constraints," in *Advances in neural information processing systems 22*, 2009.

[52] K. Yu, V. Tresp, and A. Schwaighofer, "Learning gaussian processes from multiple tasks," in *Proceeding of the 22nd International Conference on Machine Learning*, 2005.

[53] P. Zhao, G. Rocha, and B. Yu, "The composite absolute penalties family for grouped and hierarchical variable selection," *Annals of Statistics*, to appear.

[54] A. Zien and C. Ong, "Multiclass Multiple Kernel Learning," in *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, 2007, pp. 1191–1198.