# Learning with infinitely many features

Alain Rakotomamonjy, Rémi Flamary, Florian Yger

**Noname manuscript No.**
(will be inserted by the editor)

# Learning with infinitely many features

**A. Rakotomamonjy · R. Flamary and F. Yger**

the date of receipt and acceptance should be inserted later

**Abstract** We propose a principled framework for learning with infinitely many features, situations that are usually induced by continuously parametrized feature extraction methods. Such cases occur for instance when considering Gabor-based features in computer vision problems or when dealing with Fourier features for kernel approximations. We cast the problem as the one of finding a finite subset of features that minimizes a regularized empirical risk. After having analyzed the optimality conditions of such a problem, we propose a simple algorithm which has the flavour of a column-generation technique. We also show that using Fourier-based features, it is possible to perform approximate infinite kernel learning. Our experimental results on several datasets show the benefits of the proposed approach in several situations including texture classification and large-scale kernelized problems (involving about 100 thousand examples). *Keywords : infinite features, column generation, Gabor features, kernels.*

## 1 Introduction

While several recent works address the problem of automatic feature generation [8], most of the time these features are still manually crafted based on specific domain knowledge. For instance, in several machine vision applications, features are extracted by appropriate preprocessing of images. One widely used feature extraction method is the Gabor filter [34, 14]. Similarly, wavelet decompositions or time-frequency representations are frequent feature extraction methods for signal classification problems [24, 39, 27] . One major drawback of these feature generation methods is that they come along with several continuous parameters and

This work was supported in part by the IST Program of the European Community, under the PASCAL2 Network of Excellence, IST-216886. This publication only reflects the authors views. This work was partly supported by grants from the ASAP ANR-09-EMER-001, ANR JCJC-12 Lemon and ANR Blanc-12 Greta.

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

thus they can potentially produce infinitely many features. In most applications, one deals with these large amount of features by selecting beforehand few parameter values of the feature extraction method and from these values, only few features are generated. Our objective in this work is to propose a principled learning framework that is able to deal with infinitely many features, usually induced by continuously parametrized feature extractors, without the need of deciding in advance some set of values for all the parameters. Hence, by adapting the value of these parameters to the classification task at hand, we aim at extracting better discriminative features.

Another motivation for dealing with continuously parametrized features comes from the recent development of random Fourier features for kernel approximation [31]. This work of Rahimi et al. makes tractable large-scale kernel machines by approximating kernel using explicit feature maps. These mappings are built by first projecting data on a random direction and then by passing the resulting scalar value to a sine function. Instead of randomly building these projection directions before learning, owing to the framework we propose, we are able to learn these directions so that they better suit the data.

In this paper, we introduce a principled framework and an algorithm for dealing with such continuously parametrized features, in a supervised learning setting. The framework we propose is generic as it can deal with several settings, including single and multiple task learning, as well as with different convex and differentiable loss functions. The key point of the algorithm is the use of sparsity-inducing regularizers which allows us to select few features among the possibly infinite ones. The choice of these regularizers is not limited to the classical $\ell_1$ norm but can also be structured sparsity-inducing ones including the $\ell_1 - \ell_2$ or the $\ell_1 - \ell_\infty$ mixed-norms. After having reviewed related works in Section 2, we introduce in Section 3, the problem of learning with continuously parametrized features that we cast as the problem of finding a finite subset of features, minimizing a regularized empirical risk. Owing to the optimality conditions of this problem, we propose, in Section 4 a simple algorithm for its resolution, which iteratively selects novel features among all possible ones by means of randomization. In Section 5, we discuss about two families of parametrized feature maps that are of practical interest : Gabor filtering feature maps and explicit kernel feature maps. For this latter case, we uncover simple relations that show how approximate infinite kernel learning can also be achieved. Our experimental results in Section 6 shows, in several situations, the benefit of learning the feature parameters compared to state of the art approaches. Conclusion and perspectives are discussed in section 7. For a sake of reproducibility, the code used for producing the results in this paper will be made available on the author's webpage.

## 2 Related works

The problem of jointly learning feature parameters and a classifier is a problem that has already been addressed but essentially in the kernel learning framework [23, 37, 38]. To our knowledge the first works investigating this issue are the ones of Argyriou et al. [2, 4]. Indeed, in order to overcome the need of pre-defining bases kernel in multiple kernel learning (MKL) problem, they propose to learn a convex combination of continuously parametrized kernels. They solved the problem

through a greedy algorithm that they further refined by means of a DC algorithm for selecting the novel kernel. In the same spirit, Gehler et al. [20, 21] introduced the so-called Infinite Kernel Learning (IKL) framework which basically aims at solving a MKL problem with an infinite number of base kernels. In practice, they use this IKL framework for learning the parameters of a Gaussian kernel. Algorithmically, the works of Argyriou et al. and Gehler et al. follow a similar trend which consists in iteratively solving a learning problem with a fixed number of already selected kernels and then in selecting a novel kernel. The algorithm we propose for dealing with our learning problem also follows this column generation approach. Independently to Gehler et al. [20], Ozogur-Akyuz et. al. [30] have also investigated a theoretical framework for learning with infinitely many kernels.

Methods that alternate between a regularized empirical risk minimization and a feature (or kernel) generation step have already attracted attention especially in the boosting literature [25]. For instance, Demiriz et al. [17] has proposed a similar algorithm for linear programming boosting with $\ell_1$ penalization. Very recently, Shen et al. [35] also derived a resembling algorithm by carefully analyzing the dual formulation related to boosting problems. However, most of them only consider the binary classification case and the use of the $\ell_1$ penalty.

Still in the boosting framework, Chapelle et al. [12] have recently introduced a boosted multi-task framework that bears resemblance to the one we propose here. As a matter of fact, their work is a particular case of ours. Indeed, their framework only considers the use of the $\ell_1$ penalty omitting thus some feature-based relatedness between the tasks that can be handled by mixed-norms [29]. From an algorithmic point of view, they solved the boosted multi-task problem by means of an elegant path-following algorithm. While they can also handle infinite number of weak learners, they did not discuss the case of infinite number of features which is somewhat more general and can have broader applicability.

Since the framework we present also deals with multi-task learning, it has strong connections with other works of the literature. In particular, we can mention those of Argyriou et al. [3, 1, 5] which address the problem of learning shared low-dimensional representations in multi-task learning. Similarly to our approach, they aim at learning some features that are jointly of interest for several tasks. In the same trend, the work of Obozinski et al. [29] and Rakotomamonjy et al. [32] are also contributions that are worth mentioning as they aim at feature selection in a multi-task learning problem. Globally, we can consider our work as an extension of ideas presented in the above ones to the case of infinite number of features.

A piece of work that addresses a problem similar to ours is the one of Rosset et al. [33] which generalizes the $\ell_1$ penalized regression problem to infinite dimensional feature spaces by means of some probability measures. Algorithmically, they proposed a path-following method for computing a regularization path. While their theoretical framework is very appealing, the resulting algorithm is not practical as the path-following algorithm applies only to some specific feature maps. Furthermore, as stated by the authors themselves, the algorithm proves to be rather numerically unstable.

While the problem we investigate seems similar to those studied in other works and in other contexts, the learning framework and algorithm we propose, contribute to the state of the art by several aspects. Compared to the work of Rosset et al., we propose a novel framework which leads to a simple, stable and tractable algorithm. Furthermore, our framework can handle different convex and differen-

tiable loss functions as well as single and multiple task learning. Hence, it can deal with regression and multi-class problems. It is not limited to the use of $\ell_1$ regularizer as the randomization approach can cope with more complex sparsity-inducing regularizers. The algorithm we derive from this framework is very simple and can deal with large-scale problems. Then, when considering kernel learning, our approach using mixed-norm sparsity-inducing regularizers and explicit feature maps can be proved to be equivalent to an approximate infinite kernel learning. Compared to the genuine IKL algorithm, our method is more efficient and is able to deal with large-scale problems (e.g datasets with about 100 thousands of training examples).

## 3 Learning features maps

We now introduce our model for learning with infinitely many features in a single and multiple task supervised learning framework. We also provide the optimality conditions of the resulting optimization problem and derive a simple alternating optimization algorithm for solving this problem.

### 3.1 Framework

Suppose we are given $T$ tasks to be learned from $T$ different datasets $(\mathbf{x}_{i,1}, \mathbf{y}_{i,1})_{i=1}^{n_1}$, $\cdots$, $(\mathbf{x}_{i,T}, \mathbf{y}_{i,T})_{i=1}^{n_T}$, where any $\mathbf{x}_{i,t} \in \mathcal{X}$ and $\mathbf{y}_{i,t} \in \mathcal{Y}$ and $n_t$ denotes the $t$-th dataset size. In the sequel, we note $\mathbf{y}^{(t)}$ the vector of components $\mathbf{y}_{i,t}$. For these tasks, we have at our disposal an uncountable set of possible feature maps and we want to use only few of these features in the decision functions of all the tasks, some features being shared by all tasks, some other being specific to some given tasks. This framework which seeks a compromise between specific and shared features corresponds to the one proposed Evgeniou et al. [19] and recently named as dirty model multi-task learning by Jalali et al. [22]. We denote by $\varphi$ any finite set of $d$ feature maps $\varphi = \{\phi_{\theta_m}(\cdot)\}_{m=1}^{d}$ with $d \in \mathbb{N}$ and $\theta_m$ being some parameters related to the feature maps. Formally, we thus have $\varphi \in \mathcal{F}$ with

$$\mathcal{F} = \left\{ \{\phi_{\theta_m}(\cdot)\}_{m=1}^{d} : d \in \mathbb{N}, \theta_m \in \mathcal{P} \ \forall m \right\}$$

$\mathcal{P}$ being the parameter space. Similarly to Evgeniou et al. [19], we define the decision function of each task, as

$$f_t(\mathbf{x}) = \sum_{j=1}^{d} (w_{j,t} + \tilde{w}_j)\phi_{\theta_j}(\mathbf{x}) + b_t$$

where $\{b_t\}$ are bias parameters, $\{w_{i,t}\}$ and $\{\tilde{w}_i\}$ are respectively the components of the weight vector $\mathbf{w}_t$ specific to the task $t$ and $\tilde{\mathbf{w}}$ a weight vector shared by all tasks. Note that in our model, the feature parameters $\{\theta_j\}$ are learnt from the data. We also define $\boldsymbol{\Phi}^{(t)} \in \mathbb{R}^{n \times d}$ as the matrix with elements $\boldsymbol{\Phi}_{i,j}^{(t)} = \phi_j(\mathbf{x}_{i,t})$ . Columns of $\boldsymbol{\Phi}^{(t)}$ correspond to feature maps $\phi_j$ applied to the training examples of task $t$ and are denoted as $\boldsymbol{\Phi}_{j}^{(t)}$. Rows of $\boldsymbol{\Phi}^{(t)}$ are noted as $\boldsymbol{\Phi}_{i,\cdot}^{(t)}$. We also suppose that all

$\boldsymbol{\Phi}_j^{(t)}$ are normalized to unit-norm. Our objective is to find the best finite subset of features that leads to the lowest regularized empirical risk. This translates to the following optimization problem :

$$\min_{\varphi \in \mathcal{F}} \min_{\{\mathbf{w}_t\}_t, \tilde{\mathbf{w}}, \{b_t\}_t} \sum_{t=1}^{T} \sum_{i=1}^{n_t} L_t(\mathbf{y}_{i,t}, \boldsymbol{\Phi}_{i,\cdot}^{(t)}(\mathbf{w}_t + \tilde{\mathbf{w}}) + b_t) + \lambda_s \Omega_s(\tilde{\mathbf{w}}) + \lambda_p \Omega_p(\mathbf{w}_1, \cdots, \mathbf{w}_T) \tag{1}$$

where for any $t$, $L_t(\cdot, \cdot)$ is a convex and differentiable loss function that measures the discrepancy between the true and predicted label of an example for task $t$, $\Omega_s$ and $\Omega_p$ being some sparsity-inducing regularization terms on the weight vectors $\tilde{\mathbf{w}}$ and $\mathbf{w}_1, \cdots, \mathbf{w}_t$ and $\lambda_s$, $\lambda_p$ being some hyperparameters that balance between the loss function and the regularization terms. Typical possible loss functions are the square, the logistic or the squared-hinge loss functions. Note that for $T = 1$, by setting $\lambda_p = \infty$, this optimization problem boils down to be the usual single task learning problem. Since we want to select only few features among all possible ones, we have chosen the regularization terms to be sparsity-inducing ones like a $\ell_1$ or a mixed-norm regularizers, for instance, $\Omega_s(\tilde{\mathbf{w}}) = \sum_{j=1}^{d} |\tilde{w}_j|$ or

$$\Omega_p(\mathbf{w}_1, \cdots, \mathbf{w}_T) = \sum_{j=1}^{d} \left( \sum_{t=1}^{T} |w_{j,t}|^q \right)^{1/q} = \sum_j \|\mathbf{W}_{j,\cdot}\|_q$$

with $\mathbf{W} \in \mathbb{R}^{d \times T}$ defined as $\mathbf{W} = [\mathbf{w}_1\, \mathbf{w}_2\, \cdots\, \mathbf{w}_T]$ and $\mathbf{W}_{j,\cdot}$ being row $j$ of $\mathbf{W}$. Note that if the $j$-th row of $\mathbf{W}$ is equal to zero, then the feature $\boldsymbol{\Phi}_j^{(t)}$ is not involved in the decision function of any task.

### 3.2 Optimality conditions

The key point of the algorithmic derivation comes from the optimality conditions induced by the sparsity-inducing regularizers. Our derivation is applicable for several forms of sparsity-inducing regularizers but for a sake of clarity, we will restrict ourselves to $\Omega_p$ as $\ell_1 - \ell_q$ mixed-norms (with $1 < q < \infty$) either applied to group of features for single task learning or to group of tasks for multiple task learning. Furthermore, for the same reason, we assume that $\Omega_s = \|\tilde{\mathbf{w}}\|_1$, and omitted the bias terms $\{b_t\}$ in the decision functions.

For a fixed set of features $\varphi$, the inner optimization problem in Equation (1), denoted as the *restricted master* problem, boils down to be a simple sparse multi-task learning problem. Sparsity is induced by the non-differentiable $\ell_1 - \ell_q$ regularization term, and plays a central role in the derivation of the optimality conditions of the full problem (1). We now provide some necessary and sufficient conditions for a couple of feature matrix and weight vectors to be optimal first for the *restricted master* problem then for the full problem.

*3.2.1 Restricted master and first-order optimality conditions*

When we have a fixed number of features, we want to solve the so-called *restricted master* problem

$$\min_{\mathbf{w}_1,\cdots,\mathbf{w}_T,\tilde{\mathbf{w}}} \underbrace{\sum_{t=1}^{T}\sum_{i=1}^{n_t} L_t(\mathbf{y}_{i,t}, \boldsymbol{\Phi}_{i,\cdot}^{(t)}(\mathbf{w}_t+\tilde{\mathbf{w}}))}_{\mathrm{J}} + \lambda_s\Omega_s(\tilde{\mathbf{w}}) + \lambda_p\Omega_p(\mathbf{w}_1,\cdots,\mathbf{w}_T) \quad (2)$$

We have supposed that the loss functions are convex and differentiable while the sparsity-inducing regularizers are typically convex and non-differentiable. For simplicity, we supposed that

$$\Omega_s(\tilde{\mathbf{w}}) = \sum_j |\tilde{w}_j|$$

and

$$\Omega_p(\mathbf{w}_1,\cdots,\mathbf{w}_T) = \sum_{j=1}^{d}\left(\sum_{t=1}^{T} w_{j,t}^q\right)^{1/q} = \sum_i \|\mathbf{W}_{j,\cdot}\|_q$$

with $\mathbf{W} = [\mathbf{w}_1\, \mathbf{w}_2\, \cdots\, \mathbf{w}_T]$. Partial derivatives of $J$ with respects to $w_{j,m}$ is

$$\frac{\partial J}{\partial w_{j,m}} = \sum_i {\boldsymbol{\Phi}_{i,j}^{(m)}}^\top L_m'(\mathbf{y}_{i,m}, \boldsymbol{\Phi}_{i,\cdot}^{(m)}(\mathbf{w}_m+\tilde{\mathbf{w}}))$$

$L_m'(\cdot,\cdot)$ being the derivative of $L_m$ with respects to its second parameter. Hence, the $t-$th component of the gradient vector $\nabla_{\mathbf{W}_{j,\cdot}} J$ is (with a slight abuse of notation)

$$[\nabla_{\mathbf{W}_{j,\cdot}} J]_t = {\boldsymbol{\Phi}_j^{(t)}}^\top L_t'(\mathbf{y}^{(t)}, \boldsymbol{\Phi}^{(t)}(\mathbf{w}_t+\tilde{\mathbf{w}}))$$

Now we can state that a set of vectors $\mathbf{w}_1^\star,\cdots,\mathbf{w}_T^\star$ and $\tilde{\mathbf{w}}^\star$ is optimal for our objective function given in Equation (2) with the above regularizers if and only if there exists some subgradients $\mathbf{g}_j \in \partial\|\mathbf{W}_{j,\cdot}^\star\|_q$. so that

$$\begin{array}{ll} \|\nabla_{\mathbf{W}_{j,\cdot}} J\|_{q'} \leq \lambda_p & \text{if } \mathbf{W}_{j,\cdot}^\star = 0 \\ \nabla_{\mathbf{W}_{j,\cdot}} J + \lambda_p \mathbf{g}_j = 0 & \text{if } \mathbf{W}_{j,\cdot}^\star \neq 0 \end{array} \quad (3)$$

with $q'$ so that $\frac{1}{q}+\frac{1}{q'}=1$ and if and only if

$$\begin{array}{ll} |[\nabla_{\tilde{w}} J]_j| \leq \lambda_s & \text{if } \tilde{w}_j^\star = 0 \\ [\nabla_{\tilde{w}} J]_j + \lambda_s \text{ sign } (\tilde{w}_j) = 0 & \text{if } \tilde{w}_j^\star \neq 0 \end{array} \quad (4)$$

with $[\nabla_{\tilde{w}} J]_j = \sum_t {\boldsymbol{\Phi}_j^{(t)}}^\top L_t'(\mathbf{y}^{(t)}, \boldsymbol{\Phi}^{(t)}(\mathbf{w}_t+\tilde{\mathbf{w}}))$.

*3.2.2 Optimality conditions of the full problem*

Now, we are in position for deriving the optimality conditions of Problem (1).

*Proposition 1:* Suppose that $1 < q < \infty$, and $q'$ is so that $\frac{1}{q}+\frac{1}{q'} = 1$ and consider the feature matrices $\{\boldsymbol{\Phi}^{(t)}\}_t$ and the weight vectors $\{\mathbf{w}_t^\star\}_t$, $\tilde{\mathbf{w}}^\star$ optimizing the related *restricted master* problem. These vectors and matrices are also optimal for the full problem given in Equation (1) if and only if, $\forall j = \{1, \cdots, d\}$, there exists a vector $\mathbf{g}_j$ subgradient of $\partial\|\mathbf{W}_{j,\cdot}\|_q$ so that

$$\nabla_{\mathbf{W}_{j,\cdot}} J(\tilde{\mathbf{w}}^\star, \{\mathbf{w}_t^\star\}) + \lambda_p \mathbf{g}_j = 0 \qquad \text{for } j \text{ so that } \exists t, w_{j,t} \neq 0 \tag{5}$$

$$\|\nabla_{\mathbf{W}_{j,\cdot}} J(\tilde{\mathbf{w}}^\star, \{\mathbf{w}_t^\star\})\|_{q'} \leq \lambda_p \qquad \text{for } j \text{ so that } \forall t, w_{j,t} = 0 \tag{6}$$

$$\|\mathbf{h}\|_{q'} \leq \lambda_p \qquad \text{for any} \quad \phi \notin \varphi \tag{7}$$

$$[\nabla_{\tilde{\mathbf{w}}} J]_j + \lambda_s \text{sign}(\tilde{w}_j^\star) = 0 \qquad \text{for } j \text{ so that } \tilde{w}_j \neq 0 \tag{8}$$

$$\left|[\nabla_{\tilde{\mathbf{w}}} J]_j\right| \leq \lambda_s \qquad \text{for } j \text{ so that } \tilde{w}_j = 0 \tag{9}$$

$$\left|\sum_{t=1}^{T} \phi^\top L_t'(\mathbf{y}_{i,t}, \boldsymbol{\Phi}_{i,\cdot}^{(t)}(\tilde{\mathbf{w}}^\star + \mathbf{w}_t^\star))\right| \leq \lambda_s \qquad \text{for any} \quad \phi \notin \varphi \tag{10}$$

where $J(\tilde{\mathbf{w}}^\star, \{\mathbf{w}_t^\star\}) = \sum_{t=1}^{T}\sum_{i=1}^{n_t} L_t(\mathbf{y}_{i,t}, \boldsymbol{\Phi}_{i,\cdot}^{(t)}(\mathbf{w}_t^\star + \tilde{\mathbf{w}}^\star))$, the $t-$th component of the gradient vector $\nabla_{\mathbf{W}_{j,\cdot}} J$ is (with a slight abuse of notation)

$$[\nabla_{\mathbf{W}_{j,\cdot}} J(\tilde{\mathbf{w}}^\star, \{\mathbf{w}_t^\star\})]_t = {\boldsymbol{\Phi}_j^{(t)}}^\top L_t'(\mathbf{y}^{(t)}, \boldsymbol{\Phi}^{(t)}(\mathbf{w}_t^\star + \tilde{\mathbf{w}}^\star))$$

the $j$-th component of $\nabla_{\tilde{\mathbf{w}}} J$ being $[\nabla_{\tilde{\mathbf{w}}} J]_j = \sum_t {\boldsymbol{\Phi}_j^{(t)}}^\top L_t'(\mathbf{y}^{(t)}, \boldsymbol{\Phi}^{(t)}(\mathbf{w}_t^\star + \tilde{\mathbf{w}}^\star))$ and the vector $\mathbf{h} \in \mathbb{R}^T$ being the vector which $t$-th component is

$$h_t = \phi^\top L_t'(\mathbf{y}^{(t)}, \boldsymbol{\Phi}^{(t)}(\mathbf{w}_t^\star + \tilde{\mathbf{w}}^\star)).$$

*Proof* Let us first show that if $\{\boldsymbol{\Phi}^{(t)}\}_t$ and the weight vectors $\{\mathbf{w}_t^\star\}_t$, $\tilde{\mathbf{w}}^\star$ are optimal for the full problem then the above conditions hold. Note that in the following, we only develop the arguments for the optimality conditions related to the vectors $\{\mathbf{w}_t^\star\}_t$ as those related to $\tilde{\mathbf{w}}$ are similar. The first two inequalities directly come from the optimality conditions of the *restricted master* problem. These conditions are easily derived by writing that 0 belongs to the subdifferential of the non-smooth objective function. In order to prove the third inequality, we rather show that if there exists a $\boldsymbol{\phi}$ for which this inequality (7) is not verified then $\{\boldsymbol{\Phi}^{(t)}\}_t$, $\{\mathbf{w}_t^\star\}_t$ and $\tilde{\mathbf{w}}^\star$ are not optimal. Indeed, if such a $\boldsymbol{\phi}$ exists, we can build $\hat{\boldsymbol{\Phi}}^{(t)} = [\boldsymbol{\Phi}^{(t)}\ \boldsymbol{\phi}]$ and $\hat{\mathbf{w}}_t = [\mathbf{w}_t^{\star T}\ 0]^T$ for all $t$. According to the optimality conditions of the *restricted master* problem associated to $\hat{\boldsymbol{\Phi}}^{(t)}$, $\hat{\mathbf{w}}_t$ are not optimal. Hence, re-optimizing this problem leads to a decrease of the objective value. Thus, $\{\boldsymbol{\Phi}^{(t)}\}_t$, $\{\mathbf{w}_t^\star\}_t$ and $\tilde{\mathbf{w}}^\star$ were not optimal for the full problem since $\hat{\boldsymbol{\Phi}}^{(t)}$ can induce a lower objective value.

Let us now prove that conversely if these conditions hold then, $\{\boldsymbol{\Phi}^{(t)}\}_t$, $\{\mathbf{w}_t^\star\}_t$ and $\tilde{\mathbf{w}}^\star$ are optimal for problem (1). Again owing to the optimality conditions of the *restricted master* problem, satisfying the inequalities 5, 6, 8 and 9 means that $\{\boldsymbol{\Phi}^{(t)}\}_t$, $\{\mathbf{w}_t^\star\}_t$ are optimal for this *restricted master* problem. Thus, it suffices to show that when adding any of the feature $\boldsymbol{\phi}$ for which the third and sixth inequalities hold into the active feature set, no decrease in objective value can be obtained. This latter point naturally stands since building $\hat{\boldsymbol{\Phi}}^{(t)} = [\boldsymbol{\Phi}^{(t)}\ \boldsymbol{\phi}]$ and re-optimizing the associated *restricted master* problem will lead to the optimal solutions with $\{[\mathbf{w}_t^{\star T}\ 0]^T\}_t$ and $[\tilde{\mathbf{w}}^{\star T}\ 0]^T$ which means that the objective value has not decreased.

**Algorithm 1** Feature generation approach for solving problem (1).

set $\varepsilon > 0$
$k \leftarrow 0$
Set $\boldsymbol{\Phi}^{(t)} = [\,]$ for every task; % empty matrix
Set $\mathbf{w}_t = 0$, $\tilde{\mathbf{w}} = 0$, $b_t = 0$ for every task
**repeat**
  $\boldsymbol{\phi} \leftarrow$ feature for which constraints $\|\nabla_{\mathbf{W}_{j,\cdot}} J(\tilde{\mathbf{w}}, \{\mathbf{w}_t\}, \{b_t\})\|_{q'} \geq \lambda_p + \varepsilon$ or $|\sum_{t=1}^{T} \phi^\top L'_t(\mathbf{y}_{i,t}, \boldsymbol{\Phi}_{i,\cdot}^{(t)\star}(\tilde{\mathbf{w}} + \mathbf{w}_t) + b_t)| > \lambda_s + \varepsilon$
  **if** $\boldsymbol{\phi} = \emptyset$ **then**
    Break
  **end if**
  for every task $t$, $\boldsymbol{\Phi}^{(t)} \leftarrow [\boldsymbol{\Phi}^{(t)} \;\; \boldsymbol{\phi}]$ ; % feature concatenation
  $\{\mathbf{w}_t\}, \{b_t\}, \tilde{\mathbf{w}} \leftarrow$ result of the *restricted master* problem with feature matrix $\{\boldsymbol{\Phi}^{(t)}\}$
  **if** stopping criterion reached **then**
    Break
  **end if**
  $k \leftarrow k + 1$
**until** maximal number of iterations $k$ is reached

### 3.3 Feature generation algorithm

The optimality conditions of problem (1) suggest a strategy based on feature generation for its resolution. Indeed, it seems reasonable to consider an iterative approach where at each iteration, one solves the *restricted master* problem with the current feature matrices $\{\boldsymbol{\Phi}^{(t)}\}$, looks for a feature that violates either constraint (7) or (10), updates the feature matrices with that feature and re-optimizes. Our algorithm will be based on this iterative scheme which involves (i) a procedure that solves the *restricted master* problem (ii) a procedure that returns a violating constraint feature. Before delving into the implementation details of these two steps, we provide more insights into this algorithm. We first show under which hypotheses the proposed algorithm is able to solve problem given in Equation (1).

*Proposition 2:* Suppose that the *restricted master* problem is solved exactly, then if we are also able to solve the problems

$$\max_{\phi \notin \varphi, \|\phi\|=1} \|\mathbf{h}\|_{q'} \quad \text{and} \quad \max_{\phi \notin \varphi, \|\phi\|=1} \left| \sum_{t=1}^{T} \phi^\top L'_t(\mathbf{y}_{i,t}, \boldsymbol{\Phi}_{i,\cdot}^{(t)}(\tilde{\mathbf{w}}^\star + \mathbf{w}_t^\star)) \right| \tag{11}$$

exactly then, our algorithm solves our learning problem with a pre-defined tolerance $\varepsilon > 0$ on the non-active feature constraints (7) or (10), which means that we have

$$\|\mathbf{h}\|_{q'} \leq \lambda_p + \varepsilon \quad \text{and} \quad \left| \sum_{t=1}^{T} \phi^\top L'_t(\mathbf{y}_{i,t}, \boldsymbol{\Phi}_{i,\cdot}^{(t)}(\tilde{\mathbf{w}}^\star + \mathbf{w}_t^\star)) \right| \leq \lambda_s + \varepsilon$$

$\mathbf{h}$ being the vector of components $h_t = \phi^\top L'_t(\mathbf{y}^{(t)}, \boldsymbol{\Phi}^{(t)}(\mathbf{w}_t^\star + \tilde{\mathbf{w}}^\star))$. Furthermore that $\varepsilon$-approximate solution provided by the algorithm involves a finite set of features.

*Proof* For proving this lemma, we just need to show that our algorithm decreases the problem's objective value at each iteration and thus, since the objective value is lower bounded, in the limit the algorithm's output will satisfy the problem's

optimality conditions. Consider a current iteration with active feature set $\{\boldsymbol{\Phi}^{(t)}\}$, with $d$ features, and optimal related weights $\{\mathbf{w}_t^\star\}$ and $\tilde{\mathbf{w}}^\star$. Now, if we optimize conditions (11) and get objective values respectively lower than $\lambda_p + \varepsilon$ and $\lambda_s + \varepsilon$, then $\{\boldsymbol{\Phi}^{(t)}\}_t$, $\{\mathbf{w}_t^\star\}_t$ and $\tilde{\mathbf{w}}^\star$ are optimal for the full problem with a tolerance $\varepsilon$. If one of the objective value of conditions (11) is greater than $\lambda_p + \varepsilon$ or $\lambda_s + \varepsilon$, then adding $\phi$, the feature that maximizes violated constraints, to the active feature set leads to a strict decrease in the objective function.

Remark that this proposition tells us that our algorithm provides us an approximate solution and as the iterations go, under the assumption of proposition 2, resulting $\{\mathbf{w}_t\}$ and $\tilde{\mathbf{w}}$ converge towards the solution of the problem. However, it does not give us any guarantee about whether the solution can be attained with a finite number of features. This problem is more intricate and we have left it for future works. We can however highlight that in the context of infinite kernel learning, under some mild assumptions on the kernels, it is possible to show that the problem solution only involves a finite number of kernels. We thus conjecture that similar conditions can be brought out for our learning problem (1).

The next proposition tells us that the proposed algorithm improves the objective value of problem (1) at each iteration as long as the feature $\phi$ added to the active set violates constraints (7) or (10).

*Proposition 3:* Supposing that the *restricted master* problem is solved exactly, then at a given iteration $k$, adding to the the feature matrices $\{\boldsymbol{\Phi}^{(t)}\}$ any feature that violates constraints (7) or (10) leads to a decrease in objective value of the full problem (1).

*Proof* Suppose that at a given iteration, we have solved the minimization over $\mathbf{w}$ for $\{\boldsymbol{\Phi}^{((t)}\}_t$ and obtained the optimal weight vector $\{\mathbf{w}_t^\star\}$ and $\mathbf{w}^\star$. Consider $\phi \notin \varphi$ so that $\phi$ violates either its constraints (7) or (10) and define for the next iteration, $\{\hat{\boldsymbol{\Phi}}^{(t)} = [\boldsymbol{\Phi}^{(t)}\ \phi]\}$, $\{\mathbf{w}_t^\dagger = [\mathbf{w}_t^\star\ 0]\}_t$ and $\tilde{\mathbf{w}}^\dagger = [\tilde{\mathbf{w}}^\star\ 0]$. Note that these weight vectors $\{\mathbf{w}_t^\dagger\}$ and $\tilde{\mathbf{w}}^\dagger$ are not optimal for $\{\hat{\boldsymbol{\Phi}}^{(t)}\}$. Indeed $\phi$ also violates these constraints 7 or 10 since for any task $t$, we have $L_t'(\mathbf{y}^{(t)}, \boldsymbol{\Phi}^{(t)}(\mathbf{w}_t^\star + \tilde{\mathbf{w}}^\star)) = L_t'(\mathbf{y}^{(t)}, \hat{\boldsymbol{\Phi}}^{(t)}(\mathbf{w}_t^\dagger + \hat{\mathbf{w}}_s^\dagger))$ thus re-optimizing over the $\mathbf{w}$ would lead to a decrease in the objective value.

Note that the hypothesis regarding the *restricted master* problem seems not to be necessary for the proof. However, it plays a central role since it guarantees that the optimality conditions given in Proposition 1 are satisfied and thus they guarantee that $\{\mathbf{w}_t^\star\}$ and $\tilde{\mathbf{w}}^\star$ are optimal for the current $\{\boldsymbol{\Phi}^{(t)}\}$. In practice, depending on how loose the optimality conditions of the *restricted master* problem are, it may occur that the objective value of the full problem does not decrease. In such situations, it is possible to re-optimize the *restricted master* problem with tighter optimality conditions.

## 4 On the algorithmic implementation

After having presented the learning framework, the algorithm and analyzed its properties, we discuss in this section the implementation of the *restricted master* problem and explain how violating constraints can be found.

### 4.1 ADMM algorithm for the restricted master problem

The *restricted master* problem can be considered as a classical multi-task learning problem with sparsity-inducing regularizers. Several algorithms for solving such a problem have been recently proposed in the literature, including proximal methods [13] or path-following approach [29]. While a proximal method suits to our problem, we have chosen to consider another algorithm based on alternating direction method of multipliers (ADMM) [11]. This choice is essentially motivated by two reasons. First, ADMM approaches integrate some second-order information in the alternate minimization approach, while taking advantage of the simple proximal closed-form solutions resulting from the regularizers. In practice, ADMM proves to be significantly faster than FISTA especially when the number of features is small, which is a frequent situation for our *restricted master* problem. Secondly, ADMM provably converges even for proximal operators which do not have a closed-form solutions, which is the case for $1 < q < \infty$ and $q \neq 2$.

In the sequel we detail the different steps of the ADMM algorithm used for solving our *restricted master* problem. However, we first provide a brief review on ADMM approach.

#### *4.1.1 Generality*

The ADMM approach [11, 18] is an algorithm that solves a general problem of the form :

$$\min_{\mathbf{x},\mathbf{y}} \; f(\mathbf{x}) + g(\mathbf{y}) \tag{12}$$

$$st. \;\; \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{c} \tag{13}$$

where the variable is $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n_1 \times n_2}$, $f : \mathbb{R}^{n_1} \to \mathbb{R}$ and $g : \mathbb{R}^{n_2} \to \mathbb{R}$ are given convex functions, $\mathbf{A} \in \mathbb{R}^{m \times n_1}$ and $\mathbf{B} \in \mathbb{R}^{m \times n_2}$ and $\mathbf{c} \in \mathbb{R}^m$. ADMM is based on a augmented Lagrangian approach where the Lagrangian is :

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f(\mathbf{x}) + g(\mathbf{y}) + \mathbf{z}^{\mathrm{T}}(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - c) + \frac{\nu}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c}\|^2,$$

$\mathbf{z} \in \mathbb{R}^m$ being the multiplier associated to the linear constraints and $\nu > 0$ a predefined parameter balancing the violation of the linear constraints. Augmented Lagrangian method [28] suggests to iteratively minimize the Lagrangian function with respects to $\mathbf{x}$ and $\mathbf{y}$ and then to update the Lagrangian multiplier $\mathbf{z}$ with a dual ascent. Instead, ADMM proposes to take into account the special structure of the problem and to minimize the Lagrangian serially with respects to $\mathbf{x}$ and $\mathbf{y}$. In practice, this leads to an algorithm which general scheme is of the form

$$\mathbf{x}^{(k+1)} \in \arg\min_{\mathbf{x}} L(\mathbf{x}, \mathbf{y}^{(k)}, \mathbf{z}^{(k)}) \tag{14}$$

$$\mathbf{y}^{(k+1)} \in \arg\min_{\mathbf{y}} L(\mathbf{x}^{(k+1)}, \mathbf{y}, \mathbf{z}^{(k)}) \tag{15}$$

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + \nu(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c}) \tag{16}$$

When $\mathbf{B}$ is the identity matrix, it has been proved [18] that under very mild conditions, this algorithm converges to a solution of problem 12. A more general convergence result, involving less restrictive conditions on $\mathbf{B}$ has been recently provided in the work of Mota et al [26].

*4.1.2 Application of the ADMM to the* restricted master *problem*

When the feature matrices are fixed, we look at the solution of the following problem :

$$\min_{\mathbf{w}_1,\cdots,\mathbf{w}_T,\tilde{\mathbf{w}}} \sum_{t=1}^{T}\sum_{i=1}^{n_t} L_t(\mathbf{y}_{i,t}, \boldsymbol{\Phi}^{(t)}_{i,\cdot}(\mathbf{w}_t+\tilde{\mathbf{w}})) + \lambda_s\Omega_s(\tilde{\mathbf{w}}) + \lambda_p\Omega_p(\mathbf{w}_1,\cdots,\mathbf{w}_T) \quad (17)$$

Before applying the above-described ADMM algorithm, we perform some variable splitting in order to decouple the impact of the different losses and regularizers on the variables to optimize.

Supposing that all loss functions $L_t$ are a squared Hinge loss, then problem 17 is equivalent to

$$\begin{array}{ll} \min_{\{\mathbf{w}_t\},\{\mathbf{u}_t\},\{\mathbf{a}_t\},\tilde{\mathbf{w}},\tilde{\mathbf{u}},\{b_t\}} & \sum_{t=1}^{T} L_t(\mathbf{a}_t) + \lambda_s\Omega_s(\tilde{\mathbf{u}}) + \lambda_p\Omega_p(\mathbf{u}_1,\cdots,\mathbf{u}_T) \\ \text{st.} & \mathbf{a}_t = 1 - \mathbf{Y}_t(\boldsymbol{\Phi}^{(t)}(\mathbf{w}_t+\tilde{\mathbf{w}}) + b_t) \quad \forall t \\ & \tilde{\mathbf{u}} = \tilde{\mathbf{w}} \\ & \tilde{\mathbf{u}}_t = \tilde{\mathbf{w}}_t \quad \forall t \end{array} \quad (18)$$

with the overload notation $L_t(\mathbf{a}) = \max(0,\mathbf{a})^\top \max(0,\mathbf{a})$ with the max being applied component-wise to the vector $\mathbf{a}$ and each $\mathbf{a}_t$ being a vector of $\mathbb{R}^{n_t}$ and $\mathbf{Y}_t = \mathrm{diag}(\mathbf{y}_t)$. Discussions about the use of other loss functions are reported in the sequel.

Now, we can proceed with the augmented Lagrangian which writes :

$$\begin{aligned} \mathcal{L} = & \sum_{t=1}^{T} L_t(\mathbf{a}_t) + \lambda_s\Omega_s(\tilde{\mathbf{u}}) + \lambda_p\Omega_p(\mathbf{u}_1,\cdots,\mathbf{u}_T) \\ & + \sum_t \mu_t^\top(1 - \mathbf{Y}_t(\boldsymbol{\Phi}^{(t)}(\mathbf{w}_t+\tilde{\mathbf{w}}) + b_t) - \mathbf{a}_t) \\ & + \sum_t \frac{\xi_t}{2}\|1 - \mathbf{Y}_t(\boldsymbol{\Phi}^{(t)}(\mathbf{w}_t+\tilde{\mathbf{w}}) + b_t) - \mathbf{a}_t\|^2 \\ & + \sum_t \nu_t^\top(\mathbf{w}_t - \mathbf{u}_t) + \sum_t \frac{\eta_t}{2}\|\mathbf{w}_t - \mathbf{u}_t\|^2 + \tilde{\nu}^\top(\tilde{\mathbf{w}} - \tilde{\mathbf{u}}) + \frac{\tilde{\eta}}{2}\|\tilde{\mathbf{w}} - \tilde{\mathbf{u}}\|^2 \end{aligned} \quad (19)$$

where $\mu_t \in \mathbb{R}^{n_t}$ and $\nu_t \in \mathbb{R}^d$. Now that the Lagrangian has been formed, the ADMM algorithm consists in optimizing this Lagrangian wrt to each primal variable and then in updating the dual variables.

*Optimizing* $\mathbf{a}_t$ One of the advantage of the ADMM approach and the variable splitting we choose is that the problem nicely decouples. Hence, optimizing the Lagrangian wrt to $\mathbf{a}_t$ simply boils down to

$$\min_{\mathbf{a}_t} L_t(\mathbf{a}_t) + \mu_t^\mathrm{T}(\mathbf{z}_t - \mathbf{a}_t) + \frac{\xi_t}{2}\|\mathbf{z}_t - \mathbf{a}_t\|^2 \quad (20)$$

with $\mathbf{z}_t = 1 - \mathbf{Y}_t(\boldsymbol{\Phi}^{(t)}_{i,\cdot}(\mathbf{w}_t+\tilde{\mathbf{w}}) + \mathbf{b}_t$. After some simple "gather the terms" algebras, this problem is actually equivalent to :

$$\min_{\mathbf{a}_t} \frac{\xi_t}{2}\|\mathbf{a}_t - (\mathbf{z}_t + \frac{1}{\xi_t}\mu_t)\|^2 + L_t(\mathbf{a}_t) \quad (21)$$

| Regularizer $\Omega_p$ | $\text{prox}_{\lambda\Omega}\mathbf{V}$ |
|---|---|
| $\Omega_p(\mathbf{W}) = \sum_t \|\mathbf{w}_t\|_1$ | $\text{sign}(\mathbf{v}_t)(\|\mathbf{v}_t\| - \lambda)_+$ |
| $\Omega_p(\mathbf{W}) = \sum_t \|\mathbf{w}_t\|_2$ | $(1 - \frac{\lambda}{\|\mathbf{v}_t\|})_+\mathbf{v}_t$ |
| $\Omega_p(\mathbf{W}) = \sum_{j=1}^d \|\mathbf{W}_{j,\cdot}\|_1$ | $\text{sign}(\mathbf{W}_{j,\cdot})(\|\mathbf{W}_{j,\cdot}\| - \lambda)_+$ |
| $\Omega_p(\mathbf{W}) = \sum_{j=1}^d \|\mathbf{W}_{j,\cdot}\|_2$ | $(1 - \frac{\lambda}{\|\mathbf{W}_{j,\cdot}\|})_+\mathbf{W}_{j,\cdot}$ |

**Table 1** Example of regularizers and their proximal operators

which is the proximal operator of $L_t$ applied to the vector $\mathbf{z}_t + \frac{1}{\xi_t}\mu_t$. Hence, at optimality we have

$$\mathbf{a}_t = \text{prox}_{\frac{1}{\xi_t}L_t(\cdot)}(\mathbf{z}_t + \frac{1}{\xi_t}\mu_t) \tag{22}$$

which can be simple to obtain if the loss function admits a closed-form proximal operator.

*Optimizing* $\mathbf{u}_t$ *and* $\tilde{\mathbf{u}}$ By proceeding in the same way, the solutions of the Lagrangian minimization with respects to $\mathbf{u}_t$ and $\tilde{\mathbf{u}}$, are:

$$\mathbf{u}_t = \text{prox}_{\frac{\lambda_p}{\eta_t}\Omega_p}(\mathbf{w}_t + \frac{1}{\eta_t}\nu_t) \quad \text{and} \quad \tilde{\mathbf{u}} = \text{prox}_{\frac{\lambda_s}{\tilde{\eta}}\Omega_s}(\tilde{\mathbf{w}} + \frac{1}{\tilde{\eta}}\tilde{\nu}) \tag{23}$$

where we made the hypothesis that the proximal operator of $\Omega_p$ decouples with respect to the vectors $\{\mathbf{u}_t\}_t$. Note that if the regularizer $\Omega_p$ does not decouple then the proximal operator would have involved matrix $\mathbf{U}$ as a variable duplicate and $\mathbf{W}$ and $\nu_t$ would have been matrices. Table 1 presents some usual proximal operators that are of interest for our works.

*Optimizing* $\mathbf{w}_t$ *and* $b_t$ This minimization problem can be written as:

$$\begin{aligned}\min_{\mathbf{w}_t, b_t}\ & \mu_t^{\mathrm{T}}(1 - \mathbf{Y}_t(\boldsymbol{\Phi}^{(t)}(\mathbf{w}_t + \tilde{\mathbf{w}}) + b_t) - \mathbf{a}_t) + \\ & \tfrac{\xi_t}{2}\|1 - \mathbf{Y}_t(\boldsymbol{\Phi}^{(t)}(\mathbf{w}_t + \tilde{\mathbf{w}}) + b_t) - \mathbf{a}_t\|^2 + \nu_t^T(\mathbf{w}_t - \mathbf{u}_t) + \tfrac{\eta_t}{2}\|\mathbf{w}_t - \mathbf{u}_t\|^2\end{aligned} \tag{24}$$

For the subsequent derivations, we drop the task-related subscript $t$. First-order optimality conditions with respects to $w_t$ and $b_t$ are :

$$\begin{aligned}-\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mu + \nu - \xi\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mathbf{1} + \xi\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}(\mathbf{w} + \tilde{\mathbf{w}}) + \xi b\boldsymbol{\Phi}^T\mathbf{1} + \xi\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mathbf{a}_t - \eta\mathbf{u} + \eta\mathbf{w} = 0 \\ -\mu^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mathbf{1} - \xi\mathbf{1}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}(1 - \mathbf{a}) + \xi\mathbf{1}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{w} + \xi\mathbf{1}^{\mathrm{T}}\mathbf{1}b = 0\end{aligned} \tag{25}$$

which can be satisfied by solving the following set of linear equations

$$\begin{bmatrix}\xi\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \eta\mathbf{I} & \xi\boldsymbol{\Phi}^T\mathbf{1} \\ \xi\mathbf{1}^{\mathrm{T}}\boldsymbol{\Phi} & \xi\mathbf{1}^{\mathrm{T}}\mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{w} \\ b\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mu - \nu + \xi\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}(\mathbf{1} - \mathbf{a}) + \eta\mathbf{u} - \xi\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\tilde{\mathbf{w}} \\ \mu^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mathbf{1} + \xi\mathbf{1}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}(1 - \mathbf{a})\end{bmatrix} \tag{26}$$

We can remark that these linear equations have to be solved several times during the process of the ADMM iterations. However, only the second member changes through $\mathbf{a}$ and $\mathbf{u}$ along these ADMM iterations. Hence, one can compute the inverse of the left-hand side only a single time so that subsequent system resolutions only reduce to one matrix multiplication.

| Loss function | $L_t(\mathbf{a}_t)$ | $\mathbf{a}_t$ |
|---|---|---|
| square loss | $\mathbf{a}_t^T\mathbf{a}_t$ | $\mathbf{a}_t = \mathbf{Y}_t - \boldsymbol{\Phi}^{(t)}(\mathbf{w}_t + \tilde{\mathbf{w}}) - \mathbf{b}_t$ |
| square hinge loss | $(\mathbf{a}_t)_+^T(\mathbf{a}_t)_+$ | $\mathbf{a}_t = 1 - \mathbf{Y}_t(\boldsymbol{\Phi}^{(t)}(\mathbf{w}_t + \tilde{\mathbf{w}}) - \mathbf{b}_t)$ |
| hinge loss | $\mathbf{1}^T(\mathbf{a}_t)_+$ | $\mathbf{a}_t = 1 - \mathbf{Y}_t(\boldsymbol{\Phi}^{(t)}(\mathbf{w}_t + \tilde{\mathbf{w}}) - \mathbf{b}_t)$ |

| Loss function | $L_t(\mathbf{a}_t)$ | $\text{prox}_{\frac{1}{\xi}L_t}\mathbf{z}$ |
|---|---|---|
| square loss | $\mathbf{a}_t^T\mathbf{a}_t$ | $\frac{1}{1+2/\xi}\mathbf{z}$ |
| square hinge loss | $(\mathbf{a}_t)_+^T(\mathbf{a}_t)_+$ | $\begin{cases} \frac{1}{1+2/\xi}\mathbf{z} & \text{if } \mathbf{z} \geq 0 \\ \mathbf{z} & \text{otherwise} \end{cases}$ |
| hinge loss | $\mathbf{1}^T(\mathbf{a}_t)_+$ | $\begin{cases} \mathbf{z} - \frac{1}{\xi}\mathbf{1} & \text{if } \mathbf{z} \geq \frac{1}{\xi}\mathbf{1} \\ \mathbf{z} & \text{if } \mathbf{z} \leq 0 \\ 0 & \text{otherwise} \end{cases}$ |

**Table 2** Loss functions $L_t(\cdot)$, corresponding form of $\mathbf{a}_t$ and proximal operator of the losses. Note that the Hinge loss does not apply to our framework since it is not differentiable.

*Optimizing* $\tilde{\mathbf{w}}$ Minimizing the Lagrangian with respect to $\tilde{\mathbf{w}}$ is achieved in a similar way as above. Indeed, the minimization problem is

$$\begin{array}{l} \min_{\tilde{\mathbf{w}}} \sum_t \mu_t^{\mathrm{T}}(1 - \mathbf{Y}_t(\boldsymbol{\Phi}^{(t)}(\mathbf{w}_t + \tilde{\mathbf{w}}) + \mathbf{b}_t) - \mathbf{a}_t) + \\ \quad \sum_t \frac{\xi_t}{2}\|1 - \mathbf{Y}_t(\boldsymbol{\Phi}^{(t)}(\mathbf{w}_t + \tilde{\mathbf{w}}) + \mathbf{b}_t) - \mathbf{a}_t\|^2 + \tilde{\nu}^T(\tilde{\mathbf{w}} - \tilde{\mathbf{u}}) + \frac{\tilde{\eta}}{2}\|\tilde{\mathbf{w}} - \tilde{\mathbf{u}}\|^2 \end{array} \tag{27}$$

Deriving the objective function and equating the derivative to 0 leads to the following equation

$$\begin{array}{l} \left[\sum_t \xi_t {\boldsymbol{\Phi}^{(t)}}^{\mathrm{T}}\boldsymbol{\Phi}^{(t)} + \tilde{\eta}\mathbf{I}\right]\tilde{\mathbf{w}} = \sum_t \left({\boldsymbol{\Phi}^{(t)}}^{\mathrm{T}}\mathbf{Y}_t^{\mathrm{T}}\mu_t + \xi_t {\boldsymbol{\Phi}^{(t)}}^{\mathrm{T}}\mathbf{Y}_t^{\mathrm{T}}(\mathbf{1} - \mathbf{a}_t)\right) \\ \qquad\qquad - \sum_t \xi_t {\boldsymbol{\Phi}^{(t)}}^{\mathrm{T}}\boldsymbol{\Phi}^{(t)}\mathbf{w}_t - \sum_t \xi_t {\boldsymbol{\Phi}^{(t)}}^{T}\mathbf{1}b_t - \tilde{\nu} + \tilde{\eta}\tilde{\mathbf{u}} \end{array} \tag{28}$$

which is a simple linear system to solve.

*What if we change the loss function?* The above derivations focus on squared Hinge loss and binary classification problem. A change in the loss function would impact two things in Equation (18) : the form of $L_t(\cdot)$ and the equality constraints giving the relation between $\mathbf{a}_t$, $\mathbf{w}_t$, $\tilde{\mathbf{w}}$ and $b_t$. The former would impact the resolution of $\{\mathbf{a}_t\}$ in the proximal operator defined in Equation 22. This means that for loss functions which proximal operators are simple, the algorithm follows the same flow except that the closed-form proximal computation changes. For some other loss functions, typically the square loss, the equality constraints defining $\mathbf{a}_t$ take the following form :

$$\mathbf{a}_t = \mathbf{Y}_t - (\boldsymbol{\Phi}^{(t)}(\mathbf{w}_t + \tilde{\mathbf{w}}) + \mathbf{b}_t) \quad \forall t$$

Consequently, the linear systems given in Equations (27) and (28) involving $\mathbf{w}_t$, $\tilde{\mathbf{w}}$, $\mathbf{b}_t$ also take slightly different forms that we have not clearly specified but that can be easily derived from simple algebras. Table 2 recaps some typical loss functions $L_t(\cdot)$, the form of the related $\mathbf{a}_t$ and the resulting proximal operators.

**Algorithm 2** ADMM for our restricted master problem

Set $\{\xi_t\}$, $\{\eta_t\}$ and $\tilde{\eta}$ to positive values
Initialize variables $\{\mathbf{w}_t\}$, $\tilde{\mathbf{w}}$, $\{\mathbf{u}_t\}$, $\tilde{\mathbf{u}}$, $\{\mathbf{b}_t\}$, $\{\mathbf{a}_t\}$.
Initialize Lagrangian multipliers $\{\mu_t\}$, $\{\nu_t\}$, $\tilde{\nu}$.
**repeat**
  % Updating primal variables
  update $\{\mathbf{w}_t\}$ and $\mathbf{b}_t$ by solving the $T$ linear systems defined by Equation (26)
  update $\tilde{\mathbf{w}}$ by solving the linear system defined by Equation (28)
  update $\{\mathbf{a}_t\}$ using the proximal operators defined in Equation (22).
  update $\{\mathbf{u}_t\}$ and $\tilde{\mathbf{u}}$ using the proximal operators defined in Equation (23).
  % Updating Lagrangian multipliers
  $\mu_t \leftarrow \mu_t + \xi_t(\mathbf{z}_t - \mathbf{a}_t) \quad \forall t$
  $\nu_t \leftarrow \nu_t + \eta_t(\mathbf{w}_t - \mathbf{u}_t) \quad \forall t$
  $\tilde{\nu} \leftarrow \tilde{\nu} + \tilde{\eta}(\tilde{w} - \tilde{u}) \quad \forall t$
**until** stopping criterion is met

#### *4.1.3 Discussions on complexity*

All the steps needed for solving the *restricted master* problem with $d$ features are summarized in Algorithm 2. We can note that the algorithm is rather simple and involves essentially two computationally heavy steps : the two linear systems that have to be solved at each iteration. In a ADMM approach, it is difficult to evaluate the number of iterations needed before convergence as this number depends on the choice of $\xi_t$ and $\{\eta_t\}$. From experience, we can state that about few hundred iterations are typically needed before convergence, for the values of $\xi_t$ and $\{\eta_t\}$ used in the experiments.

Now, let us have a closer look at the computational complexity needed for each iteration. We first remark that for the linear equations (26) and (28), the involved matrices do not change over the iterations and thus their inverses can be pre-computed at the cost of $\mathcal{O}(d^3)$. Once this is done, solving the systems at each iteration consists only in a matrix-vector multiplication which takes $\mathcal{O}(d^2)$. The other steps make use of proximal operators. If we consider proximal operators that are simple enough as those considered in Tables 1 and 2, we can suppose that each component-wise operation in these operator can be computed in $\mathcal{O}(1)$. Accordingly, updating $\{\mathbf{a}_t\}$, $\{\mathbf{u}_t\}$ respectively takes $\mathcal{O}(\sum_t n_t)$ and $\mathcal{O}(dT + d)$. Hence, the major computational burden of this ADMM algorithm for solving our *restricted master* problem is the matrix inversion needed for the linear systems. Note however, that by construction, as we add one feature at a time in the global Algorithm (1), $d$ typically remains small. We can also highlight that this algorithm is linear in the number of examples and this allows us to tackle large-scale problems.

### 4.2 Finding a violating constraint

For finding the feature which violates the most its constraints, we need to solve the two optimization problems given in Equation (11). For some specific cases, these problems can be simple and may have a closed-form solutions. This is the case for the first problem when $q = 2$ and a square loss function is considered. However, in a general situations, these problems may be non-convex and difficult to optimize notably because of the mixed-norm structure and the non-linearities induced by

the feature maps. Fortunately, Proposition 3 suggests us that any feature that violates one of the two constraints in (11) can be integrated to the active feature set $\{\boldsymbol{\Phi}^{(t)}\}$ and re-optimizing the *restricted master* problem using these novel feature matrices would lead to a decrease in objective value. Hence, instead of optimizing problems (11), it suffices to find an approximate maximum of these problems which can be simply done by randomizing over a small number of feature parameters. For this purpose, our strategy consists in sampling $k_r$ random feature parameters and then in looking for the best feature, according to Equations (11), among those $k_r$ features. This conceptually simple heuristic allows us to avoid a difficult optimization problem and besides it is rather cheap. Indeed, it can be easily shown that the computational cost of evaluating $k_r$ features according to Equations (11) is about $\mathcal{O}(k_r nT)$ where $n = \sum_t n_t$.

## 5 Some useful feature maps and discussions

In this section, we will describe some examples of feature maps that are continuously parametrized and that are of practical interest.

### 5.1 Gabor features

Because they provide a simple and good model of the cortical simple cell receptive fields of vision system [34], Gabor filters are frequently used in computer vision and pattern recognition problems as a feature extractor, especially for texture recognition problems [9, 15]. Typically, a Gabor filter is of the form

$$F(z, y) = \exp\left(-\frac{z_0^2}{2\sigma_z^2} - \frac{y_0^2}{2\sigma_y^2}\right)\cos(2\pi f_0 z_0)$$

with $z_0 = z\cos\theta + y\sin\theta$ and $y_0 = -z\sin\theta + y\cos\theta$, and $\sigma_z$, $\sigma_y$, $f_0$ and $\theta$ being some parameters that control the Gabor filter scale, frequency and orientation. For producing features from an image, one first computes the convolution of the image with a set of these Gabor filters (with varying parameters), then obtains the features as the collection of the filtering response on all image points or on pre-specified landmarks of the image, or even as the average or the maximum of the filter responses over the image or part of the image. In most applications dealing with Gabor filters, in order to keep the feature extraction step tractable; these filter parameters are restricted to few values : for instance typically, 4 or 8 orientations $\theta$ are considered. Such an approach would naturally lead to a lack of optimality in the Gabor filter response as soon as the image characteristic does not fit to the filter banks. Hence, being able to learn with all possible Gabor filters should overcome this issue and should result in improved performances of Gabor feature based recognition systems.

### 5.2 Explicit kernel feature maps

Recent works have proposed methods for approximating well-known and frequently used kernels such as the intersection kernel [36] or the Gaussian kernel [31]. For instance, Rahimi et al. have shown that under some mild conditions, a shift-invariant

kernel like the Gaussian one can be arbitrarily well approximated by a kernel of the form

$$\hat{K}(\mathbf{x}, \mathbf{x}') = \frac{1}{D} \sum_{j=1}^{D} \mathbf{z}_{\mathbf{v}_j}(\mathbf{x})^T \mathbf{z}_{\mathbf{v}_j}(\mathbf{x}')$$

with $\{\mathbf{z}_{\mathbf{v}_j}(\cdot)\}_{j=1}^{D}$ being some Fourier feature maps of the form

$$\mathbf{z}_{\mathbf{v}}(\mathbf{x}) = [\cos(\mathbf{v}^T\mathbf{x}) \;\; \sin(\mathbf{v}^T\mathbf{x})]$$

where $\mathbf{v}$ is a direction vector that has been sampled from the Fourier transform of that shift-invariant kernel, and $D$ being the number of involved feature maps.

Instead of pre-defining some vectors $\{\mathbf{v}_j\}$ before learning as suggested in [31], we propose to apply our framework for selecting better random directions $\{\mathbf{v}_j\}$. In practice, we sample $k_R$ number of vectors $\mathbf{v}$ according to the probability distribution related to the target kernel (*e.g* the Gaussian kernel), compute the features $\mathbf{z}_{\mathbf{v}}$ from these vectors and select among them the one that violates the most its constraints. Note that since $\mathbf{z}_{\mathbf{v}}$ is itself composed of two related features, these two features should form a group of features in the mixed-norm regularization term of our optimization problem.

**Low-rank kernel approximation interpretation:** Because, we are using a sparsity-inducing norm in our learning framework, we actually select a subset of feature maps. For large values of the regularization parameters $\lambda_p$ and $\lambda_s$, only few features $\{\mathbf{z}_{v_j}\}_j$ will be kept in the final decision functions. Hence, if the number of selected features is lower than the number of training examples, then, we can interpret the resulting kernel approximation as a low-rank kernel approximation, since the Gram matrix of entries:

$$[\hat{K}(\mathbf{x}_n, \mathbf{x}_m)]_{n,m} = \frac{1}{N_{\mathbf{z}}} \sum_{j=1}^{N_z} \mathbf{z}_{\mathbf{v}_j}(\mathbf{x}_n)^T \mathbf{z}_{\mathbf{v}_j}(\mathbf{x}_m) \qquad \forall n, m$$

will be of rank $N_z$, $N_z$ being the number of features kept by our algorithm.

This low-rank kernel approximation is of interest in several ways : it makes large-scale kernel machines tractable and the resulting model becomes cheap to evaluate since only few features are in play. Our experimental results will support these statements.

**Approximate infinite kernel learning:** the work of Rahimi et al. [31] also shows that approximating different families of shift-invariant kernels can be easily done by only changing the probability distribution from which the vectors $\mathbf{v}$ are sampled. Hence, it becomes possible to perform approximate multiple kernel learning. Indeed, since according to Rahimi et al., a shift-invariant kernel can be approximated by

$$\frac{1}{D} \sum_{j=1}^{D} \mathbf{z}_{\mathbf{v}_j}(\mathbf{x})^T \mathbf{z}_{\mathbf{v}_j}(\mathbf{x}')$$

it is easy to show (owing to linearity of the Fourier transform) that a convex combination of pre-defined shift-invariant kernels $\sum_{m=1}^{M} d_m k_m(\mathbf{x}, \mathbf{x}')$ can be approximated by

$$\sum_m \sum_{j=1}^{D} \frac{d_m}{D} \mathbf{z}^m_{\mathbf{v}_j}(\mathbf{x})^T \mathbf{z}^m_{\mathbf{v}_j}(\mathbf{x}')$$

with $\mathbf{z}^m_{\mathbf{v}_j}(\cdot)$ being the explicit feature map of direction $\mathbf{v}_j$ related to kernel $k_m(\cdot,\cdot)$. Furthermore, the uniform convergence property of the Fourier features presented by Rahimi et al. still applies to this sum of kernels.

According to these simple findings, it becomes easy to perform approximate infinite kernel learning. Indeed, in practice, we can do this approximation by randomly selecting a Gaussian kernel bandwidth, and by randomly generating groups of features $\mathbf{z}_\mathbf{v}(\cdot)$ related to that kernel. Then, among the groups of features that have been generated, we keep the one that maximizes the constraint violations (7) or (10) and re-optimize the *restricted master* problem. This procedure is then re-iterated until a stopping criterion is met.

Formally, for this approximate IKL, in a single task learning case, an $\ell_1 - \ell_\infty$ mixed-norm penalty has to be considered in order for the features from same group to have the same weighting coefficients. In practice, the choice of the mixed norm has few impacts.

### 5.3 One pass mini-batch feature selection interpretation

When dealing with continuously parametrized features, instead of using our infinite set of features framework, it would have been possible to generate a huge but finite number of features and then perform a classical $\ell_1$-like feature selection. From this point of view of very high-dimensional feature selection, we can understand our algorithm as a method performing one-pass mini-batch feature selection; mini-batch because we process a chunk of features at a time and select the one that maximizes constraint violation and one-pass because, due to the randomization and the continuous parametrization of the features, with high probability, a given feature will be visited only a single time during the randomization, although a very similar feature can be randomly drawn. As in the work of Bordes et al. [10] which also considers one-pass optimization, our experiments show that when using an infinite or a very large number of features, a one-pass feature selection optimization approach does not induce any loss of performance while being computationally more efficient.

## 6 Numerical experiments

We have conducted some experiments which emphasize the effectiveness of our method, denoted as ***GrFL*** for Group Feature Learning, in two contexts : large-scale kernel approximation and automated parameter selection. We show that compared to baseline method such as classical $\ell_1$ feature selection with pre-defined features, denoted as *fixed feat* or compared to competing algorithms such as IKL and low-rank kernel approximations, our approach is able to provide decision functions with equivalent or significantly better accuracy and for similar performance, i) that use less features and thus are cheaper to evaluate and ii) that need less time for training. Note that for all the following experiments, kernel and regularization parameters of all methods have been set by cross-validation and we have set $\xi_t = \max(10/n_t, 0.1)$ and $\eta_t = \max(10, d)$.

| | | FISTA | | ADMM | |
|---|---|---|---|---|---|
| dataset | $\lambda$ | Time | Obj. Val | Time | Obj. Val |
| mfeat | 5 | 785.47 ± 68.3 | 208.0 ± 1.60 | 207.37 ± 24.00 | 205.9 ± 1.51 |
| mfeat | 1 | 1238.85 ± 107.9 | 54.30 ± 0.50 | 1378.9 ± 169.5 | 52.23 ± 0.5 |
| wines | 5 | 0.33 ± 0.06 | 43.99 ± 0.6 | 0.09 ± 0.0 | 43.98 ± 0.6 |
| wines | 0.1 | 13.08 ± 2.3 | 2.14 ± 0.0 | 0.50 ± 0.0 | 2.14 ± 0.0 |

**Table 3** Comparing the running time and objective values achieved by ADMM and FISTA for two UCI datasets and different regularization parameter values.

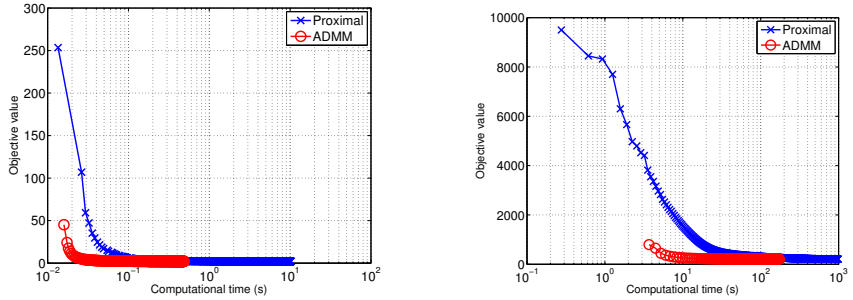


**Fig. 1** Examples of objective value evolution wrt to time for the proximal algorithm FISTA and the ADMM approach. (left) *Wines* dataset. (right) *multi feat* dataset.

### 6.1 Efficiency comparison on the restricted master problem

Our first experiment aims at evaluating the efficiency of our ADMM algorithm for solving the *restricted master* compared to an algorithm like FISTA [7] which is based on gradient descent and proximal operator.

For this purpose, we have considered two UCI multiclass problems (*Wines* and *multiple features*) that we handle through our multi-task framework by defining each binary *one-against-all* problem as a task. The algorithm runs on the original features which numbers are respectively 13 and 649. As we only want to evaluate the training time of these algorithms we have used 95% of the datasets as training examples. We have used an $\ell_1 - \ell_1$ regularizer for $\Omega_p(\cdot)$ and considered the same stopping criterion for both algorithms, which is based on approximated KKT conditions or a maximal number of iterations (3000). Running time and objective value achieved by both algorithms after convergence are reported in Table 3. They clearly show that for similar objective values, ADMM yields faster convergence. Note that for the *multi feat* problem, when $\lambda = 1$, FISTA stops because of the maximal number of iterations and thus has a lower running time, but provides significantly worse objective values. Figure 1 gives an example of how the objective value varies across time for the two considered datasets.

### 6.2 Gabor filter based texture classification

In this other experiment, we provide empirical evidences that substantial performance gain can be achieved by automatically learning the feature parameters compared to the use of fixed pre-defined ones jointly with an embedded feature selection method. Results of an approach considering the *restricted master* prob-

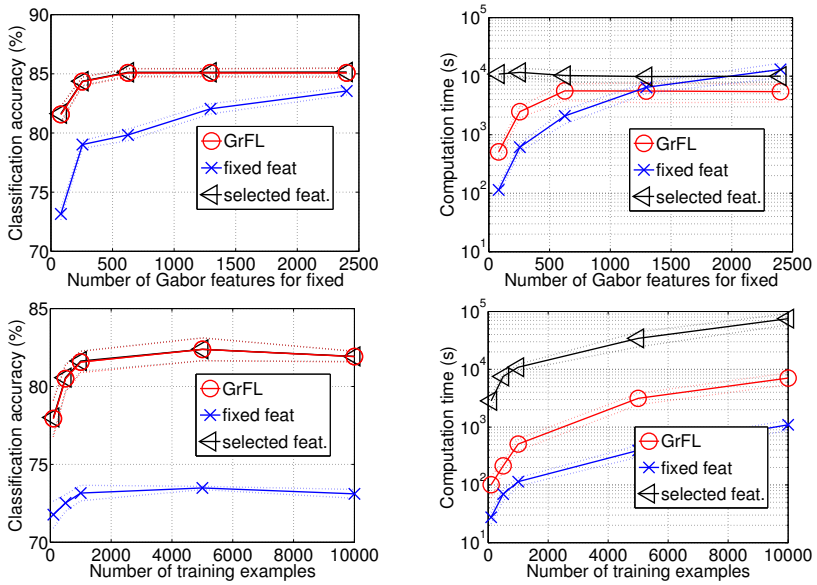


**Fig. 2** Classification performance on three Brodatz textures classification in a multiclass one-against-all context using squared Hinge loss context. Performances of our algorithm, a *fixed feature* and a *selected feature* approach are reported. Top plots respectively show the accuracy and the running time vs the number of pre-computed Gabor features with 1000 training examples per class. Bottom plots depict the same measures but with varying number of training examples per class. For these plots, the *fixed feat* algorithm considers only $k_S = 81$ features.

lem with a random subset of size $k_{sel}$ of the features visited by our algorithm, denoted as *selected feat*, have also been reported for a sake of evaluating the computational gain of using "one-pass" feature selection. For this latter approach, in order to have results in a reasonable amount of time we have fixed $k_{sel}$ to 3000 features. The regularization parameters of all algorithms have been selected among $\lambda = [1, 2, 5, 10]$.

This texture recognition problem consists in classifying $16 \times 16$ patches extracted from three Brodatz textures D2, D29 and D92. For each texture, we have extracted varying number (from 100 to 10000) of overlapping patches from the left side of the image for training and 5000 patches from the right side for testing. The multiclass problem is handled in the same way as for the previous experiment. Regarding the feature generation, we have proceeded as follows. For a given Gabor function, a feature is obtained by computing the inner product of all translated version of the Gabor function located at some pre-defined positions with the image patch and then by summing the absolute value of the responses over the positions. When Gabor parameters are pre-defined in advance, we generate a number $k_S$ of features that depends on the number of samples used for each parameter $\theta$, $f_0$ , $\sigma_z$ and $\sigma_y$ . For our group feature learning case, the number $k_R$ of sampled features used for checking constraint violations is fixed to $k_R = \min(k_S, 500)$.

Results are summarized in Figure 2 where we can note the drastic gain in performance achieved by our algorithm. We also note that as the number of predefined parameters used for the Gabor feature increases, the performance gap

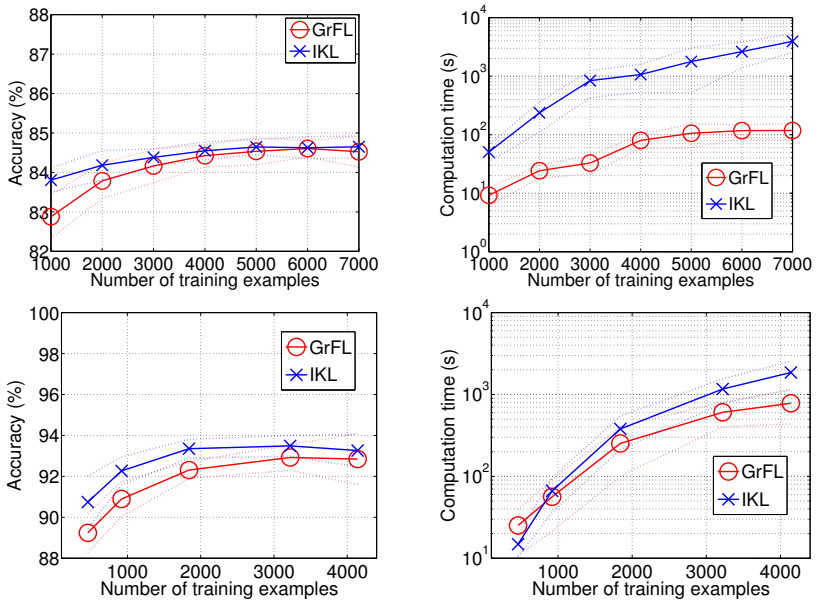


**Fig. 3** Performance accuracy and computational time vs number of training examples for (left) *Adult*. (right) *spamdata*.

between the two approaches decreases but is still manifest. Compared to the embedded feature selection approach *selected feat*, our GrFL method does not provide any gain in performance but its running time is reduced by an order of magnitude (although we already limited the number of features in the *selected feat* approach).

### 6.3 Medium-scale and large-scale datasets

We have compared the IKL algorithm of Gehler et al. [21] and our approximate IKL on two medium-scale binary classification datasets *Adult*( reduced to randomly selected 10000 examples) and *spamdata*. We do not expect our approximated kernel to perform better than IKL but rather to be more **efficient** and to be **cheaper** to evaluate. For our approximate IKL, we have randomly sampled 20 values of Gaussian bandwidth in the interval $[5, 100]$ and for each of these values, we have randomly sampled 59 directions $\mathbf{v}$. Among all these features, the one violating the most its constraints is included in the active feature set. Note that since $\mathbf{z_v}(\mathbf{x})$ is composed of two features, these latter are jointly regularized in our model by means of a $\ell_1 - \ell_2$ mixed-norm for $\Omega_p(\mathbf{w})$.

Figure 3 depicts the performance comparisons of the two approaches. These plots show that while our *GrFL* algorithm induces some slight loss of performances in small-scale situations, in other situations, substantial gain in computational time may be obtained. Furthermore, as shown in Figure 4, the number of features selected by our algorithm is by an order or magnitude lower than the number of support vectors selected by IKL.

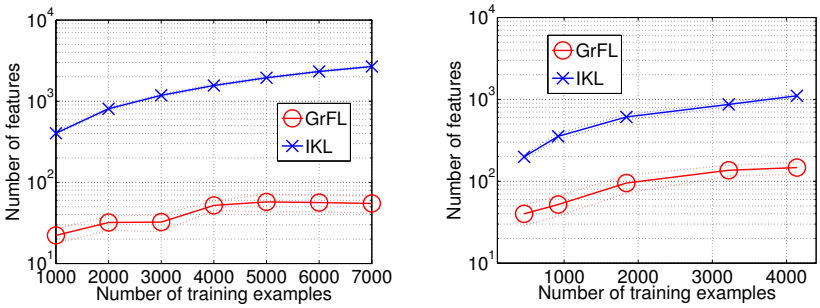


**Fig. 4** Comparison of the number of features kept for the final decision functions of our algorithm and IKL for dataset (left) *Adult*. (right) *spamdata*.

| | Adult | | | IJCNN1 | | |
|---|---|---|---|---|---|---|
| # feat | GrFL | GrFL-M | CSI | GrFL | GrFL-M | CSI |
| 10 | **83.82** | 83.77 | 83.38 | **92.06** | 91.96 | 91.03 |
| 50 | 84.76 | **84.86** | 84.58 | **97.05** | 96.97 | 92.19 |
| 100 | 84.98 | **85.00** | 84.84 | 97.97 | **98.02** | 93.29 |
| 500 | 85.24 | **85.30** | 85.04 | – | – | – |
| | Adult | | | IJCNN1 | | |
| ratio | GrFL | GrFL-M | CSI | GrFL | GrFL-M | CSI |
| 0.1 | 84.23 | 84.34 | **84.54** | 96.27 | **96.67** | 93.38 |
| 0.3 | 84.78 | **84.87** | 84.72 | 97.40 | **97.77** | 93.23 |
| 0.5 | 84.91 | **84.95** | 84.74 | 97.75 | **97.96** | 93.32 |
| 0.7 | 84.98 | **85.00** | 84.84 | 97.97 | **98.02** | 93.29 |

**Table 4** Accuracy in % on two large scale problems. Comparing single (GrFL) and multiple kernel (GrFL-M) feature approximation to low-rank decomposition. The left table gives algorithms performance for a given number of features while the right table shows performances for increasing number of training examples and 100 features. Performances in **bold** are significantly better than those of CSI according to a Wilcoxon signed rank test with $p = 0.05$.

Our last experiment shows that our algorithm can handle large scale kernelized problems and that it is competitive compared to other large-scale kernel machines like the low-rank decomposition method of Bach [6]. For this purpose, we have considered the *Adult* and *IJCNN1* datasets which respectively have 48842 and 141691 examples and 14 and 22 features. For these two datasets, we have compared the performance of our algorithm using both single Gaussian kernel approximation (*GrFL*) and multiple Gaussian kernel approximation (*GrFL-M*) sampling strategies and the low-rank decomposition method of Bach et al. denoted as (CSI). For all methods, the considered Gaussian kernel bandwidth is chosen among $\sigma^2 = \{2.5, 5, 10, 15\}$, and the regularization parameters for *GrFL* and *GrFL-M* have been selected among the values $[10, 50, 100, 500]$ while for CSI, we chose from $[10^{-7}, 10^{-6}, \cdots, 10^{-3}]$. Note that for our methods, we have sampled at each constraint violation checking, 50 features $\mathbf{z}_v$ for each possible bandwidth value which means that for *GrFL* and *GrFL-M*, we evaluated respectively 50 and 200 features.

Table 4 reports the average performance over 10 different splits of the data. At first, we are interested in accuracy performance for a given budget of features (from 10 to 500) using 70% of the examples for training and the rest for testing.

Then, we have fixed the maximal number of features at 100 and have made varied the ratio number of training examples vs number of examples. We can see that for most situations, our approach using the multiple kernel sampling strategy performs significantly better than other methods. For the *IJCNN1* dataset, for both sampling strategies, we get better performances of about 4% better than CSI. Regarding running time, building the models *GrFL* and *GrFL-M* with at most 50 features using 70% of the examples for training take about 330 and 410 seconds for the Adult dataset and 1300 and 1400 seconds for the *IJCNN1* dataset using a (non-optimized) Matlab code on a single core Xeon machine with 24 Gbytes of memory. With just a slight computational overhead, we are thus able to mix features derived from different kernels and achieve better performances.

## 7 Conclusions

We proposed in this paper, a novel framework and a novel algorithm for learning with infinitely many features, which has been made possible by using sparsity-inducing regularizers. We have analyzed the optimality conditions of such regularized problems with infinite number of features. From these analyzes, we derived a simple algorithm which iteratively selects feature parameters. The algorithm, built around an alternating direction methods of multipliers approach, is generic enough to handle different loss functions and different types of regularizers, as well as single or multiple task learning paradigm.

We also discussed two useful situations where features are indeed infinitely many : continuously parametrized Gabor features and randomized kernel approximations. We reported experimental results showing the benefits that can be achieved by learning the feature parameters using our framework instead of pre-fixing them in advance.

For future works, we plan to provide a theoretical analysis of this learning framework. In particular, taking inspirations from related works [16, 40], we are interested in studying its generalization performance. In addition, we plan to better analyze the learning problem by providing conditions on the families of features for having a solution with finite number of features and to deploy our algorithm on real-world applications dealing in particular to computer vision problems.

## References

1. A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.
2. A. Argyriou, R. Hauser, C. Micchelli, and M. Pontil. A dc-programming algorithm for kernel selection. In *In Proc. International Conference in Machine Learning*, 2006.
3. A. Argyriou, A. Maurer, and M. Pontil. An algorithm for transfer learning in a heterogeneous environment. In *Proceedings of the European Conference on Machine Learning*, 2008.
4. A. Argyriou, C. Micchelli, and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. In *Proceedings of COLT'2005*, pages 338–352, 2005.

5. A. Argyriou, C. Micchelli, and M. Pontil. When is there a representer theorem? vector versus matrix regularizers. *Journal of Machine Learning Research*, 10:2507–2529, 2009.
6. F. Bach and M. Jordan. Predictive low-rank decomposition for kernel methods. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.
7. A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2:183–202, 2009.
8. Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.
9. F. Bianconi and A. Fernandez. Evaluation of the effects of gabor filter parameters on texture classification. *Pattern Recognition*, 40(12):3325–3335, 2007.
10. A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.
11. S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and trends in machine learning*, 3:1–122, 2011.
12. O. Chapelle, P. S. and. S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. Boosted multi-task learning. *Machine Learning*, 85(1-2):149–173, 2011.
13. 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.
14. W.-P. Choi, S.-H. Tse, K.-W. Wong, and K.-M. Lam. Simplified gabor wavelets for human face recognition. *Pattern Recognition*, 41(3):1186–1199, 2008.
15. D. Clausi and M. Jernigan. Designing gabor filters for optimal texture separability. *Pattern Recognition*, 33(11):1835–1849, 2000.
16. C. Cortes, M. Mohri, and A. Rostamizadeh. Generalization bounds for learning kernels. In *Proceedings of the 27th Annual International Conference on Machine Learning*, 2010.
17. A. Demiriz, K. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:225–254, 2002.
18. J. Eckstein and D. Bertsekas. On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 5:293–318, 1992.
19. T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Proceedings of the tenth Conference on Knowledge Discovery and Data mining*, 2004.
20. P. Gehler and S. Nowozin. Infinite kernel learning. In *NIPS workshop on Automatic Selection of Kernel Parameters*, 2008.
21. P. Gehler and S. Nowozin. Let the kernel figure it out: Principled learning of pre-processing for kernel classifiers. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.
22. A. Jalali, P. Ravikumar, S. Sanghavi, and C. Ruan. A dirty model for multitask learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.
23. G. Lanckriet, N. Cristianini, L. El Ghaoui, P. Bartlett, and M. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.
24. H. Lee, Y. Largman, P. Pham, and A. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information and Processing Systems*, 2009.

25. L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent in function space. In *in Neural information processing systems*, 1999.
26. J. Mota, J. Xavier, P. Aguiar, and M. P uschel. A proof of convergence for the alternating direction method of multipliers applied to polyhedral-constrained functions. Technical report, ArXiv:1112.2295, 2011.
27. J. Neumann, C. Schnorr, and G. Steidl. Efficient wavelet adaptation for hybrid wavelet-large margin classifiers. *Pattern Recognition*, 38(11):1815–1830, 2005.
28. J. Nocedal and S. Wright. *Numerical optimization*. Springer, 2000.
29. G. Obozinski, B. Taskar, and M. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20:231–252, 2010.
30. S. Ozogur-Akyuz and G. Weber. Learning with infinitely many kernels via semi-infinite programming. In *In Proceedings of Euro mini conference on "Continuous Optimization and Knowledge Based Technologies"*, pages 342–348, 2008.
31. A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1177–1184, 2007.
32. A. Rakotomamonjy, R. Flamary, G. Gasso, and S. Canu. $\ell_p - \ell_q$ penalty for sparse linear and sparse multiple kernel multi-task learning,. *IEEE Trans. on Neural Networks*, to appear, 2011.
33. S. Rosset, G. Swirszcz, N. Srebro, and J. Zhu. $\ell_1$ regularization in infinite dimensional feature spaces. In *Proceedings of Computational Learning Theory*, pages 544–558, 2007.
34. T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):411–426, 2007.
35. C. Shen and H. Li. On the dual formulation of boosting algorithms. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 32(12):2216–2231, 2010.
36. A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.
37. J. Wang, H. Lu, K. Plataniotis, and J. Lu. Gaussian kernel optimization for pattern recognition. *Pattern Recognition*, 42(7):1237–1247, 2009.
38. D.-Y. Yeung, H. Chang, and G. Dai. Learning the kernel matrix by maximizing a kfd-based class separability criterion. *Pattern Recognition*, 40(7):2021–2028, 2007.
39. F. Yger and A. Rakotomamonjy. Wavelet kernel learning. *Pattern Recognition*, 5, 2011.
40. Y. Ying and C. Campbell. Generalization bounds for learning the kernel. In *Proceed- ings of 22nd Annual Conference on Learning Theory (COLT)*, 2009.