ℓ_p-norm Multiple Kernel Learning with Low-Rank Kernels

Anonymous Author(s)
Affiliation
Address
email

Abstract

Kernel-based learning algorithms are well-known to poorly scale to large-scale applications. For such large tasks, a common solution is to use low-rank kernel approximation. While several algorithms and theoretical analyses have already been proposed in the literature, for low-rank Support Vector Machine or low-rank Kernel Ridge Regression, this work addresses the problem of scaling ℓ_p-norm multiple kernel for large learning tasks using low-rank kernel approximations. Our contributions stand on proposing a novel optimization problem, which takes advantage of the low-rank kernel approximations and on introducing a proximal gradient algorithm for solving two different forms of that problem. We also provide theoretical results on the impact of the low-rank approximations over the kernel combination weights. Experimental evidences show that the proposed approaches scale better than the SMO-MKL algorithm for tasks involving about several hundred thousands of examples.

1 Introduction

Kernel methods such as Support Vector Machines or Kernel Rigde Regression have now become classical tools for classification or regression problems. In these methods, the choice of the kernel is of primary importance for achieving performances. Multiple kernel learning (MKL) is a learning framework that transfers the choice of the kernel from the practitioner to the algorithm since the latter provides a learned linear combination of some bases kernels and a decision function[13, 3].

Since the last few years, the literature on MKL have been flourishing and many efforts have been spent on proposing and analyzing novel regularizers for kernel combinations [22, 11, 9]. Improving the learning algorithm computational efficiencies [19, 20, 23, 1, 21] have also been on the focus of many interests. While these works have pushed the boundaries of the scalability of multiple kernel learning methods, we believe that some efforts are still needed for making MKL capable of handling large learning problems.

One common way for handling large-scale problems in kernel methods is to consider low-rank approximation of the kernel matrix and to take advantages of this approximation for accelerating the learning process [24, 6, 12]. As an example, for Kernel ridge regression, the Woodbury matrix inversion formula [8] makes it easy to derive that efficient algorithm. For SVM, **(author?)** [7] have proposed an interior point optimization method that can profit from the low-rank kernel approximation. In this work, we propose to make a leap towards the goal of very large-scale multiple kernel learning by developing algorithms which are able to benefit from low-rank approximation of the *base* kernels used in a multiple kernel learning framework. In particular, we focus on ℓ_p-norm MKL because of its theoretical soundness [10] and because it has been proven to be useful in some practical applications [11]. Within this context of ℓ_p norm MKL, we make the following contributions :

- we introduce a framework for MKL with low-rank kernels and propose a proximal gradient algorithm for solving the learning problem. By exhibiting a trick on projection on affine sets, we are able to drastically reduce the computational complexity of our algorithm. In addition, our approach is parallelizable, a property that comes from the nature of the algorithm itself and because the approach is essentially based on matrix-vector multiplications.
- we provide a theoretical result that bounds the difference between norms, for the exact kernel and its low-rank kernel approximation, of each component of the MKL decision function. This bound gives us intuitions on how the kernel approximation impact the weights in the kernel linear combination.
- We empirically demonstrate that our MKL with low-rank kernel approach can be significantly faster than state-of-the-art ℓ_p MKL solvers such as the *SMO-MKL* algorithm. For instance, we are able to solve a MKL problem with 10 kernels and about 100 thousands examples or 5 kernels and 500 thousands examples in about than half an hour.

2 Framework

We introduce in this section the multiple kernel learning framework we are interested in.

2.1 MKL framework

We consider a classification learning problem from data $\{\mathbf{x}_i, y_i\}_{i=1}^{\ell}$ where $\mathbf{x}_i$ belongs to some input space $\mathcal{X}$ and $y_i = \{+1, -1\}$ denoting the class label of examples $\{\mathbf{x}_i\}$. We denote as $\mathbf{Y}$ the diagonal matrix composed from the label vector.

In a multiple kernel learning framework, which involves a linear combination of m kernels, one looks for a decision function of the form $f(\mathbf{x}) = \sum_{k=1}^{m} f_k(\mathbf{x}) + b$ with each function $f_k \in \mathcal{H}_{\mathbf{K}_k}$, $\mathcal{H}_{\mathbf{K}_k}$ being the RKHS associated with positive definite kernel $\mathbf{K}_k$. One way to learn these functions $f_k(\cdot)$ is to solve the following MKL primal problem as first proposed by **(author?)** [26] and **(author?)** [19] and extended by **(author?)** [11] :

$$\begin{array}{ll} \min\limits_{d_k \geq 0, f_k, b, \xi_i \geq 0} & \sum_k \frac{\|f_k\|^2_{\mathcal{H}_{\mathbf{K}_k}}}{d_k} + C \sum_i \xi_i \\ \text{with} & y_i \sum_k f_k(x_i) + y_i b \geq 1 - \xi_i \quad \forall i \\ & \|\mathbf{d}\|_p^2 \leq 1 \end{array} \tag{1}$$

where each d_k regularizes the squared norm of f_k in the objective function. Hence, the smaller d_k is, the smoother f_k should be. The p-norm constraint on the weight vector $\mathbf{d}$ controls the amplitudes of $\{d_k\}$ while the choice of p eventually enforces the kernel linear combination to be sparse ($p = 1$), non-sparse ($p > 1$) or have equal weights ($p = \infty$). Problem (1) is usually solved by considering an alternating optimization strategy which consists in optimizing a slave problem with respects to $\{f_k\}_k$ with fixed weights $\{d_k\}$ and then in optimizing the weights while the functions f_k are fixed [11, 25, 15]. This alternating strategy, depicted in Algorithm 1, has been proved to be efficient owing to several features. The first one is that the slave problem boils down to be an SVM problem where the kernel matrix is a fixed combination of kernels

$$\min_{\alpha : y^\top \alpha = 0, C \geq \alpha_i \geq 0} \frac{1}{2} \alpha^\top \mathbf{Y} \Big(\sum_k d_k \mathbf{K}_k \Big) \mathbf{Y} \alpha - \mathbf{1}^\top \alpha \tag{2}$$

and thus off-the-shelf efficient SVM solvers can be used. The other point is that the optimization with respects to a weight d_k admits a closed-form solution :

$$d_k = \frac{\|f_k\|_{\mathcal{H}_{\mathbf{K}_k}}^{\frac{2}{p+1}}}{(\sum_{k=1}^{m} \|f_k\|_{\mathcal{H}_{\mathbf{K}_k}}^{\frac{2p}{p+1}})^{\frac{1}{p}}} \tag{3}$$

where $f_k(\cdot) = d_k \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{K}_k(\mathbf{x}_i, \cdot)$. Although this strategy can be made further efficient, by for instance interleaving the SVM and the weight optimization solvers [11], it still suffers because of the kernel matrix size, when large-scale learning problems are in play. The SMO-MKL algorithm of **(author?)** [23] departs from this alternating optimization strategy and instead solves the dual of

Algorithm 1 Alternate optimization for ℓ_p MKL

1: set k=1, initialize the kernel weights $\{d_k\}_k$
2: **repeat**
3: optimize an SVM with kernel $\sum_k d_k \mathbf{K}_k$
4: update $\{d_k\}$ according to Equation 3
5: **until** convergence is met

a modified version of Problem (1). A very nice feature of this latter algorithm is that it can handle large-scale problems since kernel matrix entries can be computed on the fly.

One way to deal with such large-scale training datasets in kernel-based learning algorithms is to resort to low-rank kernel approximation of the kernel matrix. Several methods are available for performing low-rank approximation of a semi-definite positive matrix including Nystrom methods [24, 12] or Incomplete Choleski decomposition [8]. In this study, we will consider Nystrom method that we briefly remind in what follows. Suppose we want to approximate a positive definite Gram matrix $\mathbf{K}$. A rank$-R$ Nystrom approximation $\tilde{\mathbf{K}}$ of $\mathbf{K}$ is obtained by randomly sampling N_n training examples among all available ones and by defining

$$\tilde{\mathbf{K}} = \mathbf{C}\mathbf{W}_R^{\dagger}\mathbf{C}^{\top} = \mathbf{V}\mathbf{V}^{\top}$$

where $\mathbf{C}$ is the matrix formed by the columns of the matrix $\mathbf{K}$ related to the N_n samples, $\mathbf{W}$ is the Gram matrix of those examples, $\mathbf{W}_R^{\dagger}$ being the pseudo-inverse of the best rank-R approximation of $\mathbf{W}$ with $R \leq N_n$ and $\mathbf{V}$ a matrix of size $\ell \times R$ so that $\mathbf{V} = \mathbf{C}\mathbf{L}\mathbf{D}$ with $\mathbf{D}$ the diagonal matrix of size $R \times R$ composed of the largest R eigenvalues of $\mathbf{W}$ and $\mathbf{L}$ the matrix of the corresponding eigenvectors. Computational complexity of the Nystrom method is dominated by the pseudo-inverse computation which is $\mathcal{O}(N_n^3)$.

The impact of the kernel approximation on the kernel method's performance is of course of interest, since using low-rank approximation boils down in trading some computational efforts against eventually some losses of generalization performances. For some families of kernel methods, these impacts have been formally analyzed by **(author?)** [5] and **(author?)** [2]. In this paper, we have addressed algorithmic issues and have studied the impact of the approximation on the norm of $\|f_k\|$, on which the weight d_k depends. We have left for future works the theoretical study of the low-rank approximation on the performances. However, our experimental results show that when the rank of the approximation becomes sufficiently large, the loss in generalization performances can become small compared to the use of the full kernel matrix.

Our contribution is thus to develop algorithms for large-scale multiple kernel learning by taking advantage of the low-rank decomposition of the base kernels. The algorithms we propose in the sequel follow the classical trend in MKL which consists in optimizing alternatively the inner SVM problem with fixed kernel weights and then in optimizing these weights while keeping the SVM optimization variables fixed. Hence, our main objective is to solve efficiently problem (2) given that we are provided with some Nystrom low-rank approximations $\{\mathbf{V}_k\}_{k=1}^m$ or all the $\{\mathbf{C}_k\}$ and $\{\mathbf{W}_k\}$ of all base kernels $\{\mathbf{K}_k\}_{k=1}^m$. For a sake of clarity and simplicity, we suppose that all $\mathbf{V}_k$ are of the same dimensions.

3 Algorithms

There exists two ways for solving the SVM problem (2) given low-rank kernel approximations. Indeed, the sum of kernel can either be approximated as a sum of low-rank kernels or as a low-rank approximation of $\sum_k d_k \mathbf{K}_k$ that needs be computed at each iteration. This section introduces these two algorithms we have proposed for solving the MKL problem with low-rank kernels. The merits and disavantages of both approaches will also be discussed.

3.1 Sum of low-rank approximation

The first approach we devise is based on the idea that low-rank approximation of each single kernel can be computed beforehand and the sum of kernel $\mathbf{K}_d$ is then approximated as the weighted sum of

low-rank approximation of each single kernel. Formally, we consider the following approximation :

$$\mathbf{K}_d = \sum_{k=1}^{m} d_k \mathbf{K}_k \approx \sum_{k=1}^{m} d_k \mathbf{C}_k \mathbf{W}_k^{\dagger} \mathbf{C}_k^{\top} = \sum_{k=1}^{m} d_k \mathbf{V}_k \mathbf{V}_k^{\top}$$

From this kernel approximation, Problem (2) boils down to be

$$\begin{array}{ll} \min_{\alpha} & \frac{1}{2}\alpha^{\top}\mathbf{Y}\Big(\sum_k d_k \mathbf{V}_k \mathbf{V}_k^{\top}\Big)\mathbf{Y}\alpha - \mathbf{1}^{\top}\alpha \\ \text{st} & y^{\top}\alpha = 0 \\ & 0 \leq \alpha_i \leq C \end{array} \tag{4}$$

and each $\|f_k\|^2_{\mathcal{H}_{\mathbf{K}_k}}$ can be obtained as :

$$\|f_k\|^2_{\mathcal{H}_{\mathbf{K}_k}} = d_k \alpha^{\top}\mathbf{Y}\mathbf{V}_k\mathbf{V}_k^{\top}\mathbf{Y}\alpha \tag{5}$$

Let us highlight that Problem (4), because of the sum of kernels, does not fit into the framework introduced by **(author?)** [7] for solving a low-rank kernel SVM. In addition, because of the linear combination of kernels, the low-rank property may have been spoiled. Indeed, the rank of $\sum_k d_k \mathbf{V}_k \mathbf{V}_k^{\top}$ is upper bounded by $\sum_k \text{rank}(\mathbf{V}_k \mathbf{V}_k^{\top})$. Our objective is thus to derive an algorithm that is able to leverage from the low-rank approximations $\{\mathbf{V}_k\}$ of all base kernel matrices.

For this purpose, we propose an equivalent formulation of Problem (4) by introducing some auxiliary variables and additional equality constraints :

$$\begin{array}{ll} \min_{\alpha,\{\gamma_k\}} & \frac{1}{2}\sum_{k=1}^{m}\gamma_k^{\top}\gamma_k - \mathbf{1}^{\top}\alpha \\ & \gamma_k = \sqrt{d_k}\mathbf{V}_k^{\top}\mathbf{Y}\alpha \qquad \forall k \in 1,\cdots,m \\ & y^{\top}\alpha = 0 \\ & 0 \leq \alpha_i \leq C \qquad \forall i \in 1,\cdots,\ell \end{array} \tag{6}$$

This optimization problem is a quadratic programming problem with $R \times m + \ell$ variables, $R \times m + 1$ equality constraints and 2ℓ inequality contraints, thus when the number ℓ of training examples is large, we have to face with a pretty large-scale problem. We can resort to classical QP solvers that handle such large-scale problems, for instance interior point methods [17]. However, in this work, we depart from this classical route by using instead a proximal gradient algorithm for solving problem (6). After having described the algorithm, we will make clearer the reasons for this choice.

The proximal method we are considering is the generalized forward-backward splitting algorithm [18] which is tailored for minimizing the sum of arbitrarily large number of convex functions $F_0(\mathbf{z}) + \sum_{i=1}^{M} F_i(\mathbf{z})$ where one of these functions (here F_0) is smooth and gradient Lipschitz of constant L while the others are (possibly) non-smooth but whose proximal operators can be simply computed. The Generalized Forward-Backward (GFB) splitting algorithm [18] follows the same principle as the Forward-Backward splitting in the sense that at a given iteration with an iterate $\mathbf{z}_n$, it first takes an explicit gradient step and then makes a implicit step where proximal operators are computed. The main feature of the Generalized Forward-Backward (GFB) splitting algorithm is that since we have several proximal operators related to the constraints or regularizers to compute, these computations are applied in parallel on some auxiliary variables. These variables are finally averaged so as to yield the next iterate $\mathbf{z}_{n+1}$. The following (simplified) proposition makes the GFB algorithm explicit :

Proposition 1 *[18, Th 2.1] Let $\{F_i\}_{i=0}^{M}$ be $M+1$ convex lower semi-continuous functions of $\mathbb{R}^d$ such that a standard domain qualification holds and such that the set of minimizers of $\sum_{i=0}^{M} F_i(\mathbf{z})$ is not empty. Set $\{Z_i\}_{i=1}^{M} \in \mathbb{R}^d$, $\mathbf{z}_0 = \frac{1}{M}\sum_{i=1}^{M} Z_i$ and build at each iteration $n \geq 0$*

$$Z_i \leftarrow Z_i + \lambda_n \left(prox_{M\zeta_n F_i}(2\mathbf{z}_n - Z_i - \zeta_n \nabla F_0(\mathbf{z}_n)) - \mathbf{z}_n\right)$$

for all $i \in 1,\cdots,M$, and

$$\mathbf{z}_{n+1} \leftarrow \frac{1}{M}\sum_{i=1}^{M} Z_i$$

with $\zeta_n \in]0, \frac{2}{L}[$ and $\lambda_n \in]0,1]$, then every sequence $\{\mathbf{z}_n\}$ generated by this algorithm weakly converges towards a minimizer of $\sum_{i=0}^{M} F_i$.

Note that this algorithm is exactly the forward-backward splitting algorithm when $M = 1$ and in the same way, it is robust to noise on the proximal operators and on the gradient computations as long as the noise norm series converge.

By defining the vector $\mathbf{z} = [\gamma_1; \gamma_2; \cdots; \gamma_m; \alpha]$ we can see that this GFB algorithm can be straightforwardly applied to our problem by choosing $F_0(\mathbf{z}) = \frac{1}{2}\sum_k \gamma_k{}^\top\gamma_j - \mathbf{1}^\top\alpha$, $F_1(\mathbf{z}) = \mathbb{I}_{\mathbf{b}_L \leq \mathbf{z} \leq \mathbf{b}_U}$ the indicator function on the box constraints delimited at each coordinate by the coordinates of $\mathbf{b}_L$ and $\mathbf{b}_U$ and $F_2(\mathbf{z}) = \mathbb{I}_{\mathbf{A}\mathbf{z}=0}$, the indicator function on an affine set defined by the matrix $\mathbf{A}$. In our case, $\mathbf{A}$ is the matrix of size $(R \times m + 1) \times (R \times m + \ell)$ defining the linear equality constraints as given in Problem (6). The proximal operators of $F_1(\cdot)$ and $F_2(\cdot)$ are simple to derive and in particular, we have :

$$\text{prox}_{\mathbb{I}_{\mathbf{A}\cdot=0}}(\mathbf{v}) = \mathbf{v} - \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}\mathbf{A}\mathbf{v}$$

Computing the term $\mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}\mathbf{A}$ involved in the proximal operator, at a cost of the order of $\mathcal{O}(m^3R^3 + m^2R^2\ell)$, is the main burden of the algorithm, although it is computed only once. Applying the proximal operator to a vector $\mathbf{v}$ involves about $\mathcal{O}(m^2R^2)$ multiplications. Hence, because of this matrix inversion needed for the projection on the affine set, the number of equality constraints is critical as it introduces a cubic dependency with respect to the number of kernels. In order to break down this complexity, we take advantage of one property of the GFB algorithm. Indeed, since this algorithm can handle any number of constraints and by noticing that the following equality holds

$$\mathbb{I}_{\mathbf{A}\mathbf{z}=0} = \sum_{j=1} \mathbb{I}_{\mathbf{A}_{C_j,:}\mathbf{z}=0}$$

where the $\{C_j\}$ forms a partition of the row index of $\mathbf{A}$, one can consider the constraints in problem (6) as the intersection of the sets defined by : $\{\gamma_k = \sqrt{d_k}\mathbf{V}_k^\top\mathbf{Y}\alpha\}_{k=1}^m$, $\alpha^\top\mathbf{y} = 0$ and $0 \preceq \alpha \preceq C\mathbf{1}$. Basically, we have splitted the matrix $\mathbf{A}$ in submatrices $\mathbf{A}_k$ which applies only on some components of $\mathbf{z}$. Under this novel perspective, Problem (6) still fits into the framework of the generalized forward-backward splitting [18]. However, we now have $m + 1$ affine sets on which we have to project the vector $\mathbf{z}$ and a box constraint. The affine sets we are interesting in, are formally defined as

$$\mathbb{I}_{\mathbf{A}_k\mathbf{z}=0}$$

with, for $k \in 1, \cdots, m$

$$\mathbf{A}_k = \left[\ 0 \ \middle|\ \cdots \ \middle|\ \mathbf{I}_R \ \middle|\ \cdots \ \middle|\ 0 \ \middle|\ -\sqrt{d_k}\mathbf{V}_k^\top\mathbf{Y}\ \right] \tag{7}$$

with $\mathbf{I}_R$ being an identity matrix of size $R \times R$ and

$$\mathbf{A}_{m+1} = \left[\ 0 \ \middle|\ \cdots \ \middle|\ 0 \ \middle|\ \cdots \ \middle|\ 0 \ \middle|\ \mathbf{y}^\top\ \right]$$

Because of its peculiar structure $\text{prox}_{\mathbb{I}_{\mathbf{A}_k\cdot=0}}(\mathbf{z})$ leaves unchanged all components but γ_k and α. The computational complexity for computing $\mathbf{A}_k^\top(\mathbf{A}_k\mathbf{A}_k^\top)^{-1}\mathbf{A}_k$ is of the order of $\mathcal{O}(R^3 + R^2\ell)$. And, since we have to compute this proximal operator m times, the complexity is now only linear in m. A detailed instantiation of the GFB algorithm to our particular problem is given in Algorithm 2.

Remark 1 *According to* **(author?)** *[18, Th 2.1], the sequence $\{\mathbf{z}_n\}$ can be made strongly convergent to the minimizer of Problem (6) if its objective function is uniformly convex. This can be easily achieved through the addition of a term $\frac{\eta}{2}\alpha^\top\alpha$ with $\eta > 0$ at the expense of slightly perturbing the original problem.*

Remark 2 *In the above algorithm, we have decided to split the original linear equality constraints in m affine sets as they are naturally related to a given low-rank kernel approximation. However, we could have chosen different number of sets. For instance, the extreme case is the case where a proximal operator is associated to a single linear equality constraint making the computation of the related $(\mathbf{A}_k\mathbf{A}_k)^{-1}$ in $\mathcal{O}(1 + \ell)$. The number of proximal operators involved in the* for-loop *of the algorithm is now $R \cdot m$ but each application of the proximal operator only costs $\mathcal{O}(1 + \ell)$. The drawback of this approach is that the memory complexity drastically increases since we have to store all the copies $\{Z_i\}$, which would be in $(R \cdot m + \ell) \times R \cdot m$. In addition, while the computational complexity of each proximal operator is small, nothing is guaranteed regarding the number of iterations needed by the global algorithm to converge towards a solution. Hence a trade-off has be made. We have not thoroughly analyzed this issue in this work and have left it for future works.*

Algorithm 2 Generalized forward-backward for SVM with sum of low-rank kernels

Input : $\{\mathbf{V}_k\}_k$ matrices so that $\mathbf{K}_k \approx \mathbf{V}_k\mathbf{V}_k^\top$.
set $\zeta_n < 1$, $\lambda = 1$
initialize auxiliary variables $\{Z_i\}_{i=1}^{m+2}$.
initialize $\mathbf{z}_0 = \frac{1}{M+2}\sum_i Z_i$.
precompute $\mathbf{A}_k$ and $\mathbf{A}_k^\top(\mathbf{A}_k\mathbf{A}_k^\top)^{-1}$, $k = 1, \cdot, m+2$
$n \leftarrow 0$
repeat
  $\nabla f = [\gamma_n; -\mathbf{1}]$
  for $i = 1 \to m$ **do**
    $\mathbf{z}'_n = 2\mathbf{z}_n - Z_i - \zeta_n \nabla f$
    $Z_i \leftarrow Z_i + \lambda(\text{prox}_{\mathbb{I}_{\mathbf{A}_i \cdot = 0}}(\mathbf{z}'_n) - \mathbf{z}_n)$
  end for
  $\mathbf{z}'_n = 2\mathbf{z}_n - Z_{m+1} - \zeta_n \nabla f$
  $Z_{m+1} \leftarrow Z_{m+1} + \lambda(\text{prox}_{\mathbb{I}_{\mathbf{A}_{m+1} \cdot = 0}}(\mathbf{z}') - \mathbf{z})$
  $\mathbf{z}'_n = 2\mathbf{z}_n - Z_{m+2} - \zeta_n \nabla f$
  $Z_{m+2} \leftarrow P_{\mathbf{b}_L, \mathbf{b}_U}(\mathbf{z}'_n)$
  $\mathbf{z}_{n+1} = \frac{1}{m+2}\sum_{i=1}^{m+2} Z_i$
  $[\gamma_{n+1}; \alpha_{n+1}] = \mathbf{z}_{n+1}$
  $n \leftarrow n + 1$
until convergence is met

This possibility of splitting the linear constraints and then in dealing with each constraint separately is one of the main reason that makes us prefer the Generalized Forward-Backward algorithm to a standard interior-point approach. Indeed, for instance, in the interior-point approach proposed by **(author?)** [14], at each iteration, one needs to solve several linear systems of size of the linear constraints (in our case $R \times m$). While a careful implementation, using for instance a Choleski factorization and a proper use of the diagonal Hessian, helps in reducing computational complexity, the cubic dependency in the number of kernels can hardly be avoided.

Let us emphasize that from our experimental study of interior point and proximal methods, when an accurate solution of problem (6) is needed, IP methods may do a better job as it tends to provide (slightly) lower objective values. However, following the lines of **(author?)** [4], in large-scale machine learning settings, finding an approximate solution of the problem may be sufficient, and in this situation, we advocate the use of proximal gradient methods.

3.2 Low-rank decomposition of kernel sum

As we have at our disposal all the matrices $\{\mathbf{C}_k\}$ and $\{\mathbf{W}_k\}$ of all the *base* kernels, we can also consider the other way around approach which consists in building a Nystrom approximation of the kernel $\mathbf{K}_d = \sum_{k=1}^m d_k\mathbf{K}_k$. Hence, we can obtain the following approximation $\tilde{\mathbf{K}}_d$ of $\mathbf{K}_d$

$$\mathbf{K}_d \approx \tilde{\mathbf{K}}_d = \mathbf{C}_d\mathbf{W}_{d,R}^\dagger\mathbf{C}_d = \mathbf{V}_d\mathbf{V}_d^\top$$

where $\mathbf{C}_d$ and $\mathbf{W}_{d,R}^\dagger$ are defined as above but apply to the sum of kernels instead of a single kernel. $\mathbf{V}_d$ is a $\ell \times R$ matrix.

From this kernel approximation, Problem (2) boils down to be

$$\min_{\alpha: y^\top\alpha = 0, C \geq \alpha_i \geq 0} \frac{1}{2}\alpha^\top\mathbf{Y}\mathbf{V}_d\mathbf{V}_d^\top\mathbf{Y}\alpha - \mathbf{1}^\top\alpha \tag{8}$$

and each $\|f_k\|^2_{\mathcal{H}_{\mathbf{K}_k}}$ can still be written as in Equation (5).

Note that the optimization problem (8) has exactly the same structure than problem (4). Applying the same problem transformation leads to an optimization problem similar to (6) where the number of linear constraints is limited to $R + 1$ instead of $m \times R + 1$. Because of this similarity, we have also applied a generalized forward backward algorithm to solve it.

Because of the lower number of constraints, we can expect that the resolution of each problem (8) costs less than problem (4). However, at each iteration of the MKL algorithm, we have to perform an eigendecomposition of the matrix $\sum_k d_k \mathbf{K}_k$ of size $N_n \times N_n$. Hence, we expect this approach to be effective especially when the number of sampled columns N_n is small.

Remark 3 *We could have also solved Problem (8) using a low-rank SVM approach as the one proposed by* **(author?)** *[7]. For the same reasons as above, the latter method is slower than our GFB approach and some experimental analysis, provided in the supplementary material, also corrobate this findings*

4 Analysis

Our objective in this section is to derive some theoretical understandings of the kernel approximation impact on the multiple kernel learning. Similar analyses have already been carried out by **(author?)** [7] and **(author?)** [5] for SVM. The former work proposed a bound on the variation of the SVM optimization problem objective value when the kernel is replaced by an approximated one. The latter one instead, analyzed how the decision function $f(\mathbf{x})$ varies with respect to the Frobenius norm of kernel difference. In our work, we focus our interest on the variation of $\|f_k\|_{\mathcal{H}_{\mathbf{K}_k}} - \|f'_k\|_{\mathcal{H}_{\mathbf{K}'_k}}$, where f_k and f'_k are respectively defined as $f_k(\mathbf{x}) = d_k \sum_i \alpha_i y_i \mathbf{K}_k(\mathbf{x}, \mathbf{x}_i)$, $f'_k(\mathbf{x}) = d_k \sum_i \alpha_i y_i \mathbf{K}'_k(\mathbf{x}, \mathbf{x}_i)$ with $\mathbf{K}'_k$ being a given approximation of $\mathbf{K}_k$ and $\mathcal{H}_{\mathbf{K}'_k}$ the corresponding RKHS. We justify our interest on the norm of $\|f_k\|_{\mathcal{H}_{\mathbf{K}_k}}$ since it plays a major role in the computation of the weights d_k. While we agree that bounding $\Big|\|f_k\|_{\mathcal{H}_{\mathbf{K}_k}} - \|f'_k\|_{\mathcal{H}_{\mathbf{K}'_k}}\Big|$ does not tell how $|d_k - d'_k|$ varies, it provides a partial result that gives us some intuitions about the key components in the kernel approximations that help in reducing the norm difference.

Proposition 2 *Suppose that $\mathbf{K}_d = \sum_k d_k \mathbf{K}_k$ is positive definite. Denote as $\tilde{\mathbf{K}}_k$ a low-rank approximation of $\mathbf{K}_k$ and $\mathbf{K}'_k = \tilde{\mathbf{K}}_k + \varepsilon I$ with ε being a small positive value. Denote as α the solution of problem (2) and α' the solution of the same problem but with kernel $\mathbf{K}'_d = \sum_k d_k \mathbf{K}'_k = \sum_k d_k \tilde{\mathbf{K}}_k + \varepsilon^\star I$, with $\varepsilon^\star = \varepsilon \sum_k d_k$. There exists two constants $M_{\mathbf{K}_k}$ and $M_{\mathbf{K}'_k}$ so that :*

$$\Big|\|f_k\|_{\mathcal{H}_{\mathbf{K}_k}} - \|f'_k\|_{\mathcal{H}_{\mathbf{K}'_k}}\Big| \leq (M_{\mathbf{K}_k} + M_{\mathbf{K}'_k})\sqrt{\frac{2s}{\lambda_{min}}} \tag{9}$$

where λ_{min} is the smallest eigenvalue of $\mathbf{K}_d$ and s so that

$$s = \max_{\alpha : y^\top \alpha = 0, 0 \leq \alpha_i \leq C} \frac{1}{2}\|\alpha\|_2^2 \Big(\Big\| \sum_k d_k (\tilde{\mathbf{K}}_k - \mathbf{K}_k) \Big\|_F + \varepsilon^\star \Big)$$

Proof : (sketch) The complete proof in given in the supplementary material. It relies on two main intermediate results : norm equivalence in finite dimensional spaces and bounds on variation of minimizers of perturbed strictly positive definite quadratic programming problems [16]. From the norm equivalence, we show that

$$\Big|\|f_k\|_{H_{\mathbf{K}_k}} - \|f'_k\|_{H_{\mathbf{K}'_k}}\Big| \leq (M_{\mathbf{K}_k} + M_{\mathbf{K}'_k})\|\alpha - \alpha'\|_2 \tag{10}$$

where $M_{\mathbf{K}}$ and $M_{\mathbf{K}'_k}$ are the upper bound constants of some norm equivalence, which means that for all $\mathbf{u} \in \mathbb{R}^\ell$, we have

$$\|\mathbf{u}\|_{d_k^2 \mathbf{Y}\mathbf{K}_k\mathbf{Y}} \leq M_{\mathbf{K}_k}\|\mathbf{u}\|_2 \text{ and } \|\mathbf{u}\|_{\mathbf{d}_k^2 Y \mathbf{K}'_k \mathbf{Y}} \leq M_{\mathbf{K}'_k}\|\mathbf{u}\|_2$$

with $\|\mathbf{u}\|_{d_k^2 \mathbf{Y}\mathbf{K}_k\mathbf{Y}}$ being the norm induced by the kernel $\mathbf{Y}\mathbf{K}_k\mathbf{Y}$. Then, a pertubation result from the work of **(author?)** [16] tells us that there exists a constant s, that depends on the perturbation, so that

$$\|\alpha - \alpha'\|_2 \leq \sqrt{\frac{2s}{\lambda_{min}}}$$

which, combined with Equation (10) leads to the inequality that concludes the proof. ■

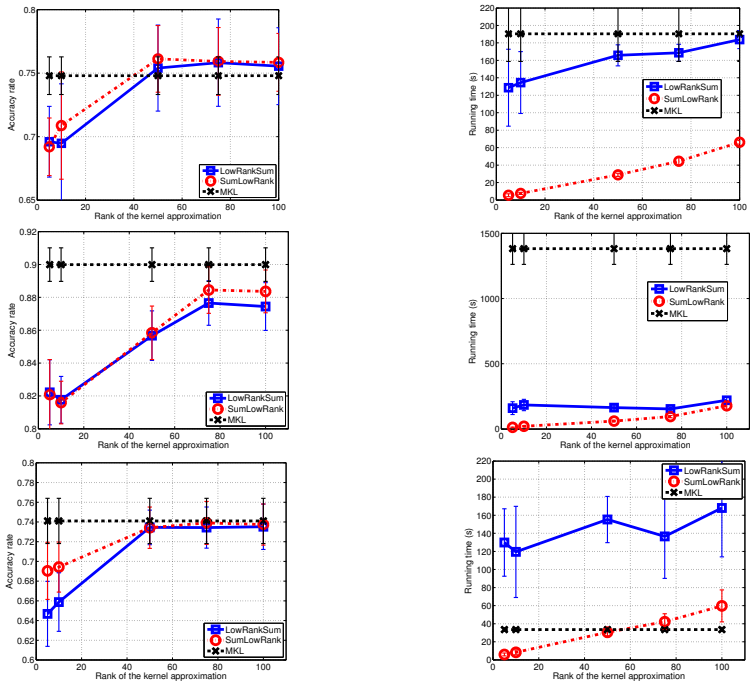


Figure 1: Comparing accuracy and running time of a plain MKL and our two low-rank MKL algorithms for small-scale learning problems. (left column) accuracy rate with respect to the rank of kernel approximations. (right column) running time with respect to rank of the approximation. (from top to bottom) the *German*, *Splice* and *Yeast* datasets involving respectively 800, 2540 and 1187 training examples and 20 kernels.

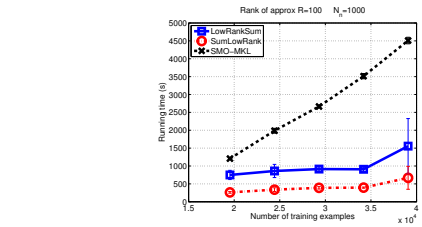
This proposition gives us the intuition that the impact of the low-rank kernel approximation is controlled by one term independent of the approximation (λ_{min}) and by the Frobenius norm of the kernel difference.

Note that this theoretical result applies to our problem only when ε goes towards 0. The introduction of ε is mandatory so as to render $\mathbf{K}'_k$ positive definite, which implies the uniqueness of α' and the existence of the norm $\|\cdot\|_{\mathbf{K}'_k}$.

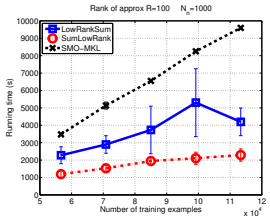
5 Numerical experiments

The main objective of our experiments is to provide empirical evidences that (i) in large-scale situations the low-rank MKL methods we propose are significantly faster than state of the art algorithms like the *SMO-MKL* and (ii) they can yield to performances similar to those of usual MKL methods, if the kernel approximation is accurate enough. An experiment providing a proof of concept about the multithreading capabilities of our algorithms has also been reported. Some other experiments, depicted in the supplementary material have also been carried out in order to prove that IP methods are less efficient than our proximal gradient algorithms, at least for the precision we are looking for.

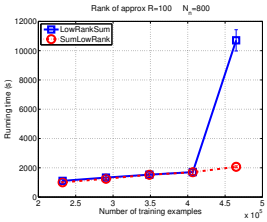
All computations have been carried out on a 16-core Intel Xeon machine with 24 GB of memory. All the codes have been written in Matlab and run on a single core of the above machine. For the *SMO-MKL*, we have used the C implementation of the authors. All reported results consider $p = 1$ for our MKL approaches and $p = 1.1$ for the SMO-MKL so as to keep the learning problem smooth. Similar results as those presented below have been obtained with $p = 2$ but not reported due to space constraints.



(adult)



(ijcnn1)



(covertype)

Figure 2: Running time with respect to the number of training examples of our two low-rank MKL algorithms and the SMO-MKL for large-scale learning problems.

	Algorithms		
Dataset	SumLR	LowRS	SMO
adult	83.92 ± 0.3	83.98 ± 0.3	84.07 ± 0.4
ijcnn1	92.36 ± 0.0	92.31 ± 0.1	99.23 ± 0.1
cover	74.91 ± 0.2	74.84 ± 0.2	-

Table 1: Comparing accuracy for large-scale problems. C has been set to 1000 and 80% of the examples have been used as training examples and the rest for testing.

5.1 Accuracy with respect to low-rank kernel approximation

We have investigated the impact of low-rank kernel approximation on the decision function accuracy. For this purpose, we have compared the accuracy of our algorithms, denoted as *SumLR* (for the one that considers sum of low-rank kernel) and *LowRS* (for the one which approximates the sum of kernel at each iteration), to the one of a MKL which uses an analytical updates of the weigths $\{d_k\}$ [25, 11]. We have considered three small-scale datasets from the UCI repository : *German*, *Splice* and *Yeast*. For each of these datasets, we have randomly split $80\% - 20\%$ the examples in training and test set. 20 Gaussian kernels, with bandwidth σ ranging logarithmically from 10^0 to $10^{1.5}$, have been pre-computed. Since we are interested in relative performances, we have arbitrarily set $C = 1000$, which is a value that yields good accuracies. Stopping criterion of all MKL algorithms are based on the maximal variation of the weights between two iterations, which should be lower than 10^{-4}. For our approaches, the inner low-rank SVM algorithm is stopped when the relative variation of its objective value is lower than 10^{-4} or when 3000 iterations have been reached.

Averaged performances (accuracy and running time) over 20 runs and increasing quality of approximation are reported in Figure 1. As expected, the accuracy of our algorithms increases along with the rank of the kernel approximations. We can see that for most problems, approximating each base kernel with a rank-100 kernel is enough for achieving accuracy similar to those provided by the full kernels. The worst situation is obtained for the *Splice* dataset as a slight loss of accuracy occurs (about 2%. Regarding running time, we note that the sum of low-rank kernel approach is more efficient than the two other algorithms except in few situations for the *Yeast* dataset.

5.2 Running time in large-scale settings

We now turn our attention on large-scale learning problems where the base kernel matrices can not be pre-computed as they do no fit in memory. In such a situation, we resort to algorithms such as

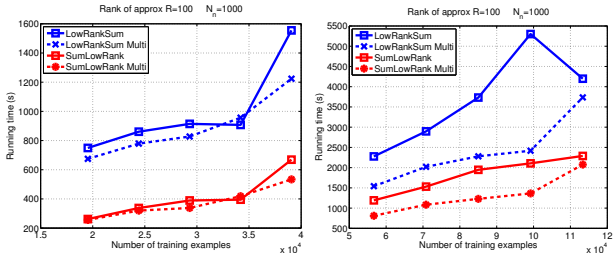


Figure 3: Comparing gain in running time when allowing multithreading and parallel computing in the large-scale setting experiments (left) Adult. (right) ijcnn1

the *SMO-MKL* which computes the kernel entries on the fly. We now compare the efficiency of this algorithm to our approach.

For these experiments, we have considered the following UCI datasets : *Adult*, *Ijcnn1*, and *Covertype*. Note that we have turned the last dataset into a binary classification problem by considering class 1 against all others. We followed exactly the same experimental set-up as above except that we have considered only 10 different bandwidths (in the same range) of the Gaussian kernel. For the *Covertype* problem, due to memory constraints, we have reduced the number of sampled columns N_n to 800.

For each dataset and algorithm, running time averaged over 5 iterations of have been plotted with respect to the number of training examples in Figure 2. The first two plots depict the results achieved on *adult* and *ijcnn1*. Both of our approaches are significantly faster than *SMO-MKL* and the gain in efficiency increases along with the number of training examples. For instance, for *adult*, with $38K$ training examples, our *SumLR* algorithm is nearly an order of magnitude faster while for *ijcnn1*, with $110K$ examples, it is about 5 times more efficient.

For the *Covertype* dataset, results of the *SMO-MKL* have not been reported because the algorithm did not converge after 48 hours even for $200K$ training examples. With our algorithms, we were able to solve the learning problems in about half an hour for $230K$ examples and less than an hour for $400K$ examples using the *SumLR* approach. To the best of our knowledge, this is a state-of-the-art efficiency result for ℓ_p multiple kernel learning. The large increase in running time occuring at $460K$ examples for *LowRS* is due to some memory swapping.

One may argue that this gain in efficiency has to be pondered by some losses in accuracy of the classifier. Table 1 compares the accuracy rate of for these 3 algorithms using 80% of training examples and the rest for testing. We can remark that for *adult* this loss is nearly neglectable while for *ijcnn1*, it goes up to 7%. For *Covertype*, the comparison is again in our favor since the *SMO-MKL* has output any results. Hence, on a given (tight) budget of time, it seems preferable to consider our low-rank MKL approaches.

5.3 On the influence of parallelization

We also provide some results showing that our algorithms can benefit from multithreading or parallel computations. For this experiment, we have replicated the large-scale experiments but allowed Matlab to use its multithreading capabilities as well as its parallel computing framework through the *parfor* instruction while computing lines 9 to 12 of Algorithm 2.

Figure 3 shows the gain in performance yielded by multi-threading and parallel computing for our two approaches. We remark that this gain is rather small but effective especially for large number of training examples. Careful analysis of the running time brought to our attention that the parallel computing framework of Matlab does not provide us any gain in efficiency. We thus conjecture that a better handling (with appropriate tools) of the parallel computing will help us to further improve the global computational efficiency of our algorithm.

6 Conclusion

The paper addresses the problem of large-scale MKL learning when the base kernels are low-rank approximation of the original ones. We propose a novel optimization problem which takes advantage of such approximations as well as two gradient proximal algorithms for solving two versions of the problem.

Theoretical results on the impact of the kernel approximation on the kernel combination weights are also provided as well as empirical evidences on the better scalability of our approaches compared to state-of-the-art MKL solvers.

References

[1] J. Aflalo, A. Ben-Tal, C. Bhattacharyya, J. Saketha Nath, and S. Raman. Variable sparsity kernel learning. *Journal of Machine Learning Research*, 12:565–592, 2011.

[2] F. Bach. Sharp analysis of low-rank kernel matrix approximations. Technical report, INRIA HAL-00723365, 2012.

[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*, pages 41–48, 2004.

[4] L. Bottou and O/ Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, volume 20. MIT Press, Cambridge, MA, 2008.

[5] C. Cortes, M. Mohri, and A. Talwalkar. On the impact of kernel approximation on learning accuracy. In *Conference on Artificial Intelligence and Statistics*, pages 113–120, 2010.

[6] P. Drineas and M.W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153–2175, 2005.

[7] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *The Journal of Machine Learning Research*, 2:243–264, 2002.

[8] G.H. Golub and C.F. Van Loan. *Matrix computations*, volume 3. Johns Hopkins University Press, 1996.

[9] C. Hinrichs, V. Singh, J. Peng, and S.C. Johnson. Q-mkl: Matrix-induced regularization in multi-kernel learning with applications to neuroimaging. In *Advances in Neural Information Processing Systems*, 2012.

[10] M. Kloft and G. Blanchard. On the convergence rate of p-norm multiple kernel learning. *Journal of Machine Learning Research*, 13:2465–2501, 2012.

[11] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien. lp-norm multiple kernel learning. *Journal of Machine Learning Research*, 12:953–997, 2011.

[12] S. Kumar, M. Mohri, and A. Talwalkar. Sampling methods for the nyström method. *The Journal of Machine Learning Research*, 98888:981–1006, 2012.

[13] 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.

[14] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2(4):575–601, 1992.

[15] J.S. Nath, G. Dinesh, S. Raman, C. Bhattacharyya, A. Ben-Tal, and K.R. Ramakrishnan. On the algorithmics and applications of a mixed-norm based kernel learning formulation. *Advances in neural information processing systems*, 22:844–852, 2009.

[16] HX Phu and VM Pho. Some properties of boundedly perturbed strictly convex quadratic functions. *Optimization*, 61(1):67–88, 2012.

[17] F.A. Potra and S.J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1):281–302, 2000.

[18] H. Raguet, J. Fadili, and G. Peyré. Generalized forward-backward splitting. *arXiv preprint arXiv:1108.4404*, 2011.

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

[20] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7(1):1531–1565, 2006.

[21] T. Suzuki and R. Tomioka. SpicyMKL : A fast algorithm for multiple kernel learning with thousands of kernels. *Machine Learning*, to appear, 2011.

[22] M. Szafranski, Y. Grandvalet, and A. Rakotomamonjy. Composite kernel learning. *Machine Learning Journal*, 79(1-2):73–103, 2010.

[23] S.V.N Vishwanathan, Z. Sun, N. Theera-Ampornpunt, and M. Varma. Multiple kernel learning and the smo algorithm. In *Advances in Neural Information Processing Systems 23*, 2010.

[24] C.K.I. Williams and M. Seeger. Using the nystrom method to speed up kernel machines. *Advances in neural information processing systems*, pages 682–688, 2001.

[25] Z. Xu, R. Jin, H. Yang, I. King, and M. Lyu. Simple and efficient multiple kernel learning by group lasso. In *Proc. of 27th International Conference on Machine Learning*, pages 1–8, 2010.

[26] A. Zien and C.S. Ong. Multiclass Multiple Kernel Learning. In *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pages 1191–1198, 2007.