ℓ_p-norm Multiple Kernel Learning with Low-Rank Kernels

A. Rakotomamonjy[1,*], S. Chanda[2,*]

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. Several algorithms and theoretical analyses have already been proposed in the literature, for low-rank Support Vector Machine or low-rank Kernel Ridge Regression but not for multiple kernel learning. This work bridges this gap by addressing 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 that optimization problem. We also provide partial theoretical results on the impact of the low-rank approximations over the kernel combination weights. Experimental evidences show that the proposed approach scales better than the SMO-MKL algorithm for tasks involving about several hundred thousands of examples. Experimental comparisons with interior point methods also prove the efficiency of the algorithm we propose.

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 good 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 as outputs a learned linear combination of some base kernels and a decision function [18, 4].

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 [26, 15, 12]. Improving the learning algorithm computational efficiency [23, 24, 28, 1, 25, 27] has also been on the focus of many interests. While these works have pushed the boundaries of multiple kernel learning method scalability, we believe that some efforts are still needed for making MKL capable of better handling large-scale 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 advantage of this approximation for accelerating the learning process [30, 9, 17]. As

*Corresponding author
Email address: alain.rakotomamonjy@insa-rouen.fr (A. Rakotomamonjy)
[1]Complete postal address : LITIS EA4108, University of Rouen, Avenue de l'université, 76800 Saint Etienne du Rouvray, France.
[2]Complete postal address : Department of Computer Science and Media Technology, Gjovik University College, Gjovik-2815, Norway

an example, for Kernel ridge regression, the Woodbury matrix inversion formula [11] makes it easy to derive that efficient algorithm. For SVM, Fine et al. [10] have proposed an interior point optimization method that can leverage on the low-rank kernel approximation. A recent work has also considered kernel approximation for multiple fisher discriminant analysis [29]. However, in this latter work, the kernel weights are not learned according to the data and the problem at hand, but defined according to the quality of the kernel approximation. While kernel matrix low-rank approximations are often computed without any supervision on the labels, some works also proposed to improve the kernel approximation by taking into account distance or similarity constraints over the training examples [16] or even by considering their labels [3]. In this work, we propose to make a another leap towards the goal of very large-scale multiple kernel learning. For this purpose, we develop an algorithm which is able to benefit from low-rank approximation of the *base* kernels used in a multiple kernel learning framework instead of directly learning a single low-rank kernel as in [16, 3]. In particular, we focus on ℓ_p-norm MKL because of its theoretical soundness [14] and because it has been proven to be useful in some practical applications [15]. Within this context of ℓ_p-norm MKL, we make the following contributions :

- we introduce a novel framework for MKL with low-rank kernels and we adapt, in a non-trivial way, a recently proposed technique for convex optimization to this problem of multiple kernel learning with low-rank kernels. 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 as it is essentially based on matrix-vector multiplications.

- we provide a theoretical result that bounds the difference between the minimizer of the MKL problem with the exact kernel and and the one with low-rank kernel approximation, of each component of the MKL decision function. This bound gives us intuitions on how the kernel approximations impact the weights in the kernel linear combination.

- We empirically demonstrate that our algorithm for solving the low-rank optimization problem is more efficient than interior point methods. In addition, we show 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 [28] while providing similar performance accuracies.

2. Framework

We introduce in this section the multiple kernel learning framework we are interested in. Let us 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 $\mathbf{Y}$ as 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

Algorithm 1 Alternate optimization algorithm 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

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 in [31] and [23] and extended by Kloft et al. [15] :

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

where each d_k regularizes the squared norm of f_k in the objective function. Hence, a smaller d_k indicates a smoother f_k. 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 respect to $\{f_k\}_k$ with fixed weights $\{d_k\}$ and then in optimizing the weights while the functions f_k are fixed [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

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

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

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

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 [15], it still suffers because of the size of kernel matrix, when large-scale learning problems are in play. The SMO-MKL algorithm [28] departs from this alternating optimization strategy and instead solves the dual of 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 [30, 17] or Incomplete Choleski decomposition [11]. 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 $\mathbf{V} = \mathbf{C}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$ 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 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 [8, 2]. In this paper, we have addressed algorithmic issues and have studied the impact of the kernel approximation on the minimizer of problem (2) on which the weight d_k depends. The theoretical study of the low-rank approximation impact on the performances is left for future works. 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 stands on developing an algorithm for large-scale multiple kernel learning by taking advantage of the low-rank decomposition of the base kernels. The algorithm we propose in the sequel follows 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 the 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 to be computed at each MKL iteration. This section presents the most efficient one when used within a MKL framework, which is based on approximating each kernel once and then in considering the sum of these low-rank kernels.

The approach 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

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

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 Fine et al. [10] for solving a low-rank kernel SVM. Indeed, because of the linear combination of kernels, the low-rank property may be lost as the rank of $\sum_k d_k \mathbf{V}_k \mathbf{V}_k^{\top}$ is upper bounded by $\sum_k \text{rank}(\mathbf{V}_k \mathbf{V}_k^{\top})$. Hence, the gain achieved through the rank$-R$ approximation of each $\mathbf{V}_k$ may be spoiled (for instance, one can easily construct a rank-1 kernel approximation of the each $\mathbf{V}_k\mathbf{V}_k^{\top}$ which sum $\sum_k \mathbf{V}_k\mathbf{V}_k^{\top}$ is of rank m.) Note that while SMO-MKL can be fed with the set of $\{\mathbf{V}_k\mathbf{V}_k^{\top}\}$ kernels, it is not able to take computational advantage of these low-rank approximations since each approximate kernel will still be considered as a full Gram matrix. This can be easily experimentally verified. 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 :

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

This optimization problem is a quadratic programming problem with $R\times m+\ell$ variables, $R\times m+1$ equality constraints and 2ℓ inequality constraints, 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 [21]. However, in this work, we depart from this classical route by using a proximal gradient algorithm for solving problem (6). In the next paragraphs, we first describe the algorithm before giving its motivations. Indeed, as they are related to computational complexities, they will be clearer after the algorithm exposition.

The proximal method that we are considering here is the generalized forward-backward splitting algorithm [22] which is tailored for minimizing the sum of an arbitrarily large number of convex functions as in

$$\min_{\mathbf{z}} F_0(\mathbf{z}) + \sum_{i=1}^{M} F_i(\mathbf{z})$$

where one of these functions (here F_0) is smooth with gradient Lipschitz of constant L while the others here the $\{F_i\}_{i=1}^M$ are possibly non-smooth but whose proximal operators can be simply computed [7].

The Generalized Forward-Backward (GFB) splitting algorithm [22] follows the same principle as the Forward-Backward splitting [6] 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 $\{Z_i\}$. 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. *[22, 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 $F_0 + \sum_{i=1}^M F_i$.

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 computation under weak conditions on the noise.

By defining the vector $\mathbf{z} = [\gamma_1; \gamma_2; \cdots; \gamma_m; \alpha]$, with $\mathbf{z} \in \mathbb{R}^{R \times m + \ell}$ we can see that this GFB algorithm can be straightforwardly applied to our problem by choosing $F_0(\mathbf{z})$ as the objective function $\frac{1}{2}\sum_k \gamma_k \top \gamma_k - \mathbf{1}^\top \alpha$ of problem (6). The constraints in that problem can form our functions F_1 and F_2 by choosing $F_1(\mathbf{z}) = \mathbb{I}_{\mathbf{b}_L \leq \mathbf{z} \leq \mathbf{b}_U}$ the indicator function on the box constraints delimited at each coordinate by the coordinates of $\mathbf{b}_L$ and $\mathbf{b}_U$ (in our case, we have $b_L = 0$ and $b_U = C$) and $F_2(\mathbf{z}) = \mathbb{I}_{\mathbf{A}\mathbf{z}=0}$, the indicator function on an affine set defined by a 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 the matrix inversion, 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 simple but essential 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 $m+2$ 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 [22]. However, we now have $m+1$ affine sets and a box constraint on which we have to project the vector $\mathbf{z}$. The m affine sets related to the linear constraints $\{\gamma_k = \sqrt{d_k}\mathbf{V}_k^\top \mathbf{Y}\alpha\}_{k=1}^m$, are formally defined as

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

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

$$\mathbf{A}_k = \left[\begin{array}{c|c|c|c|c} 0 & \cdots & \mathbf{I}_R & \cdots & 0 \end{array} \right] \quad -\sqrt{d_k}\mathbf{V}_k^\top \mathbf{Y} \left.\right] \tag{7}$$

with $\mathbf{I}_R$ being an identity matrix of size $R \times R$. The constraint related to $\alpha^\top \mathbf{y} = 0$ is defined as

$$\mathbf{A}_{m+1} = \left[\begin{array}{c|c|c|c|c} 0 & \cdots & 0 & \cdots & 0 \end{array} \right] \quad \mathbf{y}^\top \left.\right]$$

since the optimization variables are $[\gamma_1; \gamma_2; \cdots; \gamma_m; \alpha]$.

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 now of the compelling 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 our non-trivial application of the GFB algorithm to our particular problem is given in Algorithm 2. In this algorithm, $P_{b_L,b_U}(\cdot)$ defines the projection on the box constraints defined by the lowest coordinate b_L and highest coordinate b_U. Note that in practice we have set the step size ζ_n to 1 for all iterations since the Lipschitz constant of F_0 is 1.

Remark 1. *According to Raguet et al. [22, 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*

Algorithm 2 Efficient adaptation of the 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 $\{Z_i\}_{i=1}^{m+2}$ to random values where each Z_i is a vector of size $R \times m + \ell$
set $\mathbf{z}_0 = \frac{1}{M+2} \sum_{i=1}^{M+2} Z_i$.
set $[\gamma_0; \alpha_0] = \mathbf{z}_0$
precompute $\mathbf{A}_k$ and $\mathbf{A}_k^\top (\mathbf{A}_k \mathbf{A}_k^\top)^{-1}$ for $k = 1, \cdots, m+1$
$n \leftarrow 0$
repeat
  $\nabla F_0 = [\gamma_n; -\mathbf{1}]$
  % Compute the forward-backward steps related to the constraints $\{\gamma_k = \sqrt{d_k} \mathbf{V}_k^\top \mathbf{Y} \alpha\}_{k=1}^m$
  for $i = 1 \rightarrow m$ **do**
    $\mathbf{z}'_n = 2\mathbf{z}_n - Z_i - \zeta_n \nabla F_0$
    $Z_i \leftarrow Z_i + \lambda(\text{prox}_{\mathbb{I}_{\mathbf{A}_i \cdot = 0}}(\mathbf{z}'_n) - \mathbf{z}_n)$
  end for
  % Compute the forward-backward step related to the constraint $\alpha^\top \mathbf{y} = 0$
  $\mathbf{z}'_n = 2\mathbf{z}_n - Z_{m+1} - \zeta_n \nabla F_0$
  $Z_{m+1} \leftarrow Z_{m+1} + \lambda(\text{prox}_{\mathbb{I}_{\mathbf{A}_{m+1} \cdot = 0}}(\mathbf{z}') - \mathbf{z})$
  % Compute the forward-backward step related to the box constraint
  $\mathbf{z}'_n = 2\mathbf{z}_n - Z_{m+2} - \zeta_n \nabla F_0$
  $Z_{m+2} \leftarrow P_{\mathbf{b}_L, \mathbf{b}_U}(\mathbf{z}'_n)$
  % Define the next iterate of $\mathbf{z}$ as the average of the auxiliary variables
  $\mathbf{z}_{n+1} = \frac{1}{m+2} \sum_{i=1}^{m+2} Z_i$
  $[\gamma_{n+1}; \alpha_{n+1}] = \mathbf{z}_{n+1}$
  $n \leftarrow n + 1$
until convergence is met

$\{Z_i\}$, which would be in $(R \cdot m + \ell) \times R \cdot m$. In addition, while the computational complexity of each proximal operator is small, we have no guarantee 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.

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 over a standard interior-point approach. Indeed, for instance, in the interior-point approach proposed by Mehrotra [19], at each iteration, one needs to solve several linear systems of the size of the number of 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 they tends to provide (slightly) lower objective values. However, following the lines of Bottou and Bousquet [5], in large-scale machine learning settings, finding an approximate solution of the optimization problem is sufficient, and in this situation, we advocate the use of proximal gradient methods as they are far more efficient.

4. Analysis

Our objective in this section is to derive some theoretical understandings of the kernel approximation impact on the SVM problem (2) and thus on the multiple kernel learning. Similar analyses have already been carried out by Fine et al. [10] and Cortes et al. [8]. 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 norm difference between the minimizer of the exact SVM problem (2) and the minimizer of the approximated one (4). We justify our interest for the vector α as it plays a major role in the computation of the weights d_k (see Equation (3)). While we agree that bounding $\|\alpha - \alpha'\|$ 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 approximation 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 $\mathbf{K}'_k$ as a low-rank approximation of $\mathbf{K}_k$. Denote $\alpha^\star$ as the solution of problem (2) and α' the solution of the same problem but with kernel $\mathbf{K}'_d = \sum_k d_k \mathbf{K}'_k$. Then, we have,*

$$\|\alpha^\star - \alpha'\|_2 \leq \sqrt{\frac{2s}{\lambda_{min}}} \tag{8}$$

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

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

The complete proof is given in the supplementary material. It relies on bounds on variation of minimizers of perturbed strictly positive definite quadratic programming problems [20]. 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.

5. Numerical experiments

The objectives of these experiments are three-fold. At first, we want to provide empirical evidences that in large-scale situations the low-rank MKL method we propose is faster than the state of the art algorithm *SMO-MKL* while yielding to similar performances as long as the kernel approximations are accurate enough. Secondly, we provide support that proximal gradient algorithms are more efficient than interior points methods for solving problem (6). Finally, experiments depicting the gain in efficiency brought by our non-trivial application of the GFB algorithm compared to its naive implementation have also been reported.

5.1. General settings

Computations have been carried out on a 16-core Intel Xeon E5530@2.4 GHz machine with 24 GB of memory. All the codes have been written in Matlab and have been run on a single core of the above machine. For the *SMO-MKL*, we have used the C implementation of the authors.

Data sets	# examples	Dimension	comments
svmguide 1	7089	4	
mushrooms	8124	112	
magic	19020	10	
pokerhand	25010	10	class 0 against all
codrna	59535	8	
seismic	78823	50	class 3 against all
adult	48842	14	
ijcnn1	141691	22	

Table 1: Dataset statistics

We have considered datasets from the UCI and Libsvm repository and their statistics are summarized in Table 1. For each of these datasets, we have randomly split $70\% - 30\%$ the examples in training and test sets. 50 Gaussian kernels, with bandwidth σ ranging logarithmically from 10^0 to 10^2, have been considered. Since we are interested in relative performances, we have arbitrarily set $C = 1000$, which is a value that yields good accuracies. For *SMO-MKL*, we used the default stopping criterion and all its default parameter. Stopping criterion of our MKL algorithm is based on the maximal variation of the weights between two iterations, which should be lower than 10^{-4}. For our approach, 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.

5.2. Comparing with SMO-MKL

For supporting our claims, we have evaluated the computational speed (including the time for computing low-rank kernels) of our algorithm as well as the accuracy of the resulting decision function with respect to the rank of the kernel approximations. We show that for large-scale datasets our low-rank multiple kernel algorithm is more efficient than *SMO-MKL* while achieving similar or slightly worse performances. Note that the setting we are interested in is the large-scale number of examples/few kernels ones and in such a situation *SMO-MKL* is more efficient than the spectral projected gradient MKL [13].

Averaged performances (accuracy and running time) over 5 runs and increasing quality of approximation are reported in Figure 1 and Figure 2 for $p = 2$ and in Figure 3 and Figure 4 for sparse multiple kernel learning with $p = 1.1$. Note that we have chosen this latter value since *SMO-MKL* is not able to handle the exact sparse case $p = 1$. For the smallest datasets, running time comparison is in favour of the *SMO-MKL*, and we explain this fact through the advantage brought by the C implementation and the memory cache handling. As soon as, the number of examples is larger than 10^5 , our algorithm can be significantly faster (up to an order of magnitude in some cases) while achieving similar performances. Similar behaviors are observed for $p = 1.1$, and this can be expected as p does not influence the problem (6) we are solving.

Scalability with respects to the number of kernels is reported in Figure 5. Again, running time is in favour of our low-rank approach.

5.3. Proximal gradient vs interior-point

For all experiments described above, we were not looking for a precise solution of the optimization problem but as advocated by Bottou et al. [5], we were searching for approximate solutions of the problem (6).

The next experiments provide some empirical evidences on our claim regarding efficiency of the gradient proximal algorithm with respects to interior point (IP) methods for solving problem (6), at least for the precision we looked for. The interior point method we have considered is similar to the one proposed by Fine et al. [10]. We have implemented a Mehrotra predictor-corrector algorithm [19] but capable of handling a larger number of equality constraints. A careful attention has also been brought to the fact that the Hessian matrix of problem (6) is diagonal. Despite this, we emphasize that two linear systems involving a dense $R \times R$ positive definite matrix has to be solved at each iteration making the method computationally very expensive.

Remind that the stopping criterion of our low-rank SVM problem is based on relative variation of the objective value that should be lower than 10^{-4}. For the sake of fair comparison, we have early stopped the IP methods when the difference between their objective value and the one of the proximal method is less than 0.1%. Note that at this precision the infinite norm of the difference of the two minimizers is in the order of 10^{-6} . The experimental set-up is the same as for the large-scale experiments except that the rank R has been set to 500 so as to emphasize the difference in running times and that we have considered only 10 kernels and we have arbitrarily set $d_k = \frac{1}{m}$. We have compared these two methods on their efficiency for solving problem (6). Figure 6 depicts the running time of both algorithms as the number of training examples increases. We note that GFB is always more efficient with a gain factor varying from 1.1 to 8.

5.4. Splitting equality constraints

One of our contribution is to propose a non-trivial use of the GFB algorithm for solving our learning problem (6). This contribution is based on the idea that splitting the constraints in several blocks lead to reduce computational complexity. In what follows, we provide empirical supports of this fact.

The running time of two GFB algorithms, one which considers all linear constraints of problem (6) at once and one which splits the constraints (as described in Algorithm 2) have been reported in Figure 7. Again, we note the drastic gain in efficiency brought by our specific *GFB* approach compared to a naive *GFB*. We gain about 1 to 3 order of magnitude in efficiency.

6. Conclusion

The paper addresses the problem of large-scale MKL learning. In a large-scale setting, kernel methods usually resort to low-rank approximation of the original Gram matrices. In this work, we make the hypothesis that our base kernels for MKL are low-rank approximations of the original ones. We propose a novel MKL optimization problem which takes advantage of such approximations as well as a gradient proximal algorithm for solving the the related problem. Partial theoretical results dealing with 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.

As for future work, we plan to apply our work to real-world large-scale problems related to computer visions and audio signal classifications.

[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. In *Proceedings of the International Conference on Learning Theory*, 2013.

[3] F. Bach and M. Jordan. Predictive low-rank decomposition for kernel methods. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[4] 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.

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

[6] P. Combettes and V. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4:1168–1200, 2005.

[7] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In H. H. Bauschke, R. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. Springer-Verlag, 2010.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] A. Jain, SVN Vishwanathan, and M. Varma. Spg-gmkl: generalized multiple kernel learning with a million kernels. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 750–758. ACM, 2012.

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

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

[16] B. Kulis, M. Sustik, and I. Dhillon. Learning low-rank kernel matrices. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

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

[18] 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.

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

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

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

[22] H. Raguet, J. Fadili, and G. Peyré. Generalized forward-backward splitting. *SIAM Journal of Imaging Sciences*, to appear, 2013.

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

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

[25] T. Suzuki and R. Tomioka. SpicyMKL : A fast algorithm for multiple kernel learning with thousands of kernels. *Machine Learning*, 85:77–108, 2011.

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

[27] X. Tian, G. Gasso, and S. Canu. A multiple kernel framework for inductive semi-supervised svm learning. *Neurocomputing*, 90:46–58, 2012.

[28] 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.

[29] Z. Wang, W. Jie, and D. Gao. A novel multiple nystrm-approximating kernel discriminant analysis. *Neurocomputing*, 119(0):385 – 398, 2013.

[30] 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.

[31] 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.

7. Appendix

7.1. Proof of proposition 2

Let us first introduce the theorem which bounds the norm difference between the minimizer of a strictly convex quadratic programming problem and its perturbed version.

Theorem 1. *[20, Corollary 4.2] Let $\alpha^\star$ be the solution of strictly convex quadratic programming problem (P)*

$$\min_{\alpha \in C} \frac{1}{2}\alpha^\top Q\alpha + c^\top \alpha$$

with Q a positive definite matrix $\mathbb{R}^{\ell\times\ell}$, c a vector of $\mathbb{R}^\ell$ and C a non-empty convex set. Let α' be any global infimizer the perturbed version of this problem, where the perturbation stands in adding a function $p(\alpha)$ to the objective function, then the following inequality holds

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

with λ_{min} being the smallest eigenvalue of the matrix Q and s is so that $\sup_{\alpha\in C}|p(\alpha)| \leq s < \infty$.

Consider now $\alpha^\star$ as the solution of Problem (2) and α' the solution of problem

$$\min_{\alpha: y^\top\alpha = 0, 0\leq\alpha_i\leq C} \frac{1}{2}\alpha^\top \mathbf{Y}\mathbf{K}'_d\mathbf{Y}\alpha - \mathbf{1}^\top\alpha \tag{9}$$

where $\mathbf{K}'_d = \sum_k d_k\mathbf{K}'_k$, with $\mathbf{K}'_k$ a low-rank approximation of $\mathbf{K}_k$. Note that the above problem (9) is a perturbed version of Problem (2) with the following perturbation

$$p(\alpha) = \frac{1}{2}\alpha^\top\mathbf{Y}(\mathbf{K}'_d - \mathbf{K}_d)\mathbf{Y}\alpha$$

From the definition of s, in our specific case, we have

$$\infty > s \geq \sup_{\alpha: y^\top\alpha=0, C\geq\alpha_i\geq 0}\frac{1}{2}\alpha^\top\mathbf{Y}(\mathbf{K}'_d - \mathbf{K}_d)\mathbf{Y}\alpha$$

Let us now show that such s exists and that it can be upper-bounded by a term depending on $\|\mathbf{K}'_d - \mathbf{K}_d\|_F$.

First note that the perturbation function $p(\cdot)$ is continuous and that the constraints is subset of a closed subset of $\mathbb{R}^\ell$. Hence there exists s' so that

$$\sup_{\alpha: y^\top\alpha=0, 0\leq\alpha_i\leq C}|p(\alpha)| \leq \max_{0\leq\alpha_i\leq C}|p(\alpha)| = s' < \infty$$

Now, the following inequalities provide us with an upper-bound on $p(\alpha)$.

$$\begin{aligned} 2|p(\alpha)| &= \left|\alpha^\top \mathbf{Y}(\mathbf{K}'_d - \mathbf{K}_d)\mathbf{Y}\alpha\right| \\ &= \left|\text{tr}(\mathbf{Y}\alpha\alpha^\top \mathbf{Y}(\mathbf{K}'_d - \mathbf{K}_d))\right| \\ &\leq \|\mathbf{Y}\alpha\alpha^\top \mathbf{Y}\|_F \|\mathbf{K}'_d - \mathbf{K}_d\|_F \\ &= \|\alpha\|_2^2 \|\mathbf{K}'_d - \mathbf{K}_d\|_F \end{aligned}$$

Hence, we can choose s as

$$s = \max_{\alpha: y^\top \alpha = 0, C \geq \alpha_i \geq 0} \frac{1}{2}\|\alpha\|_2^2 (\|\mathbf{K}'_d - \mathbf{K}_d\|_F)$$

Then from Theorem 1 and the definition of s, we get the desired inequality given in Equation (8).

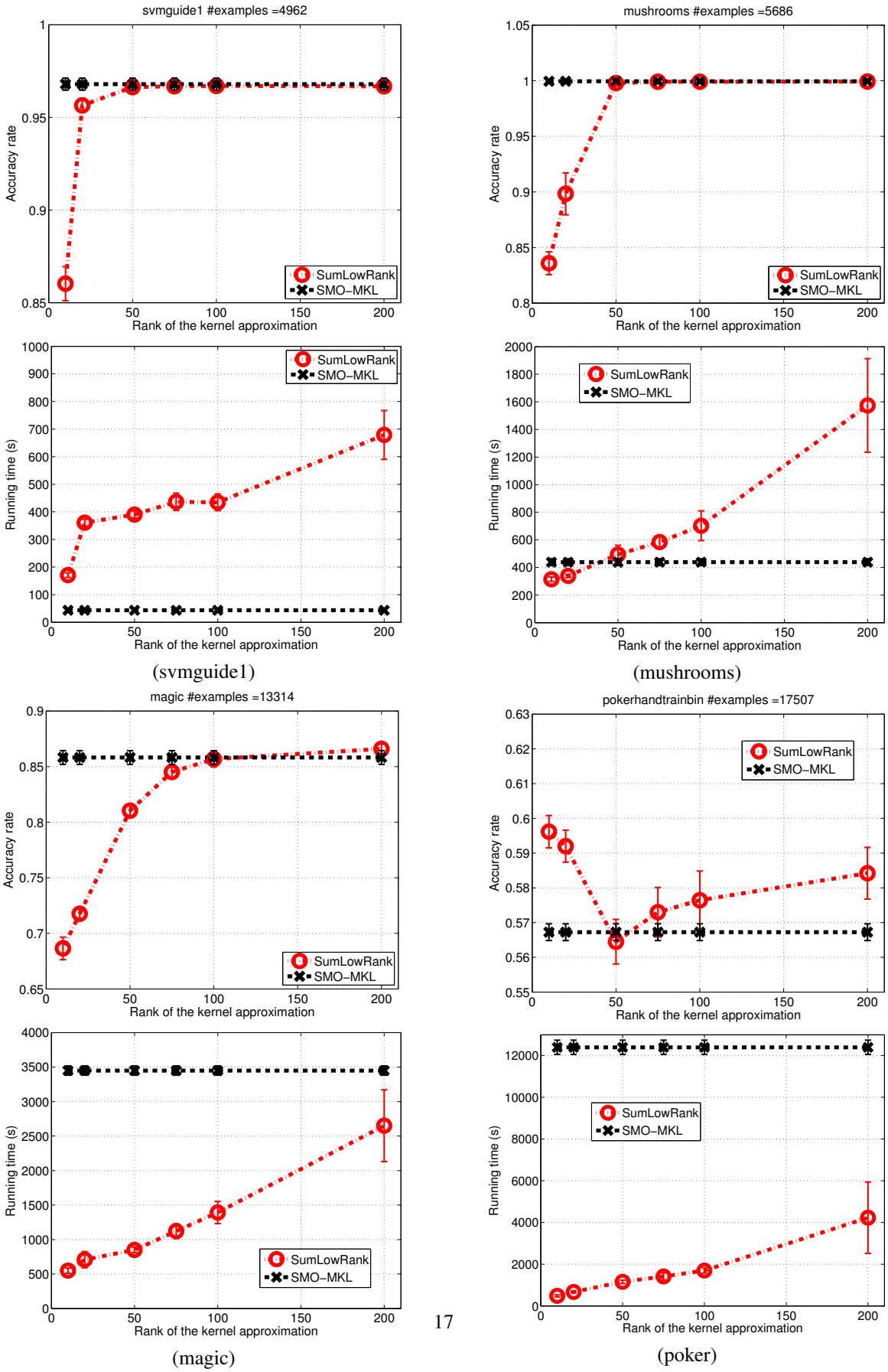
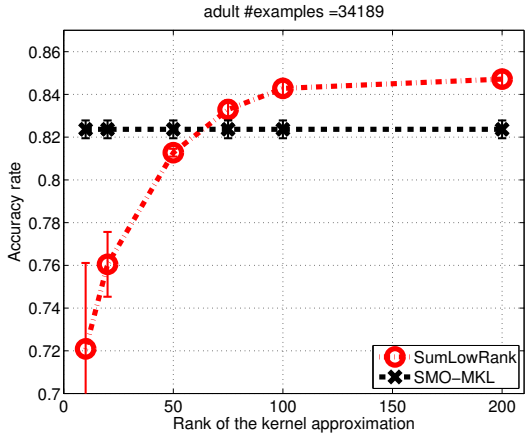
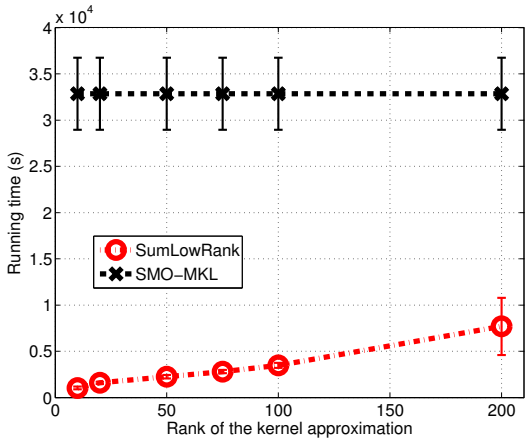
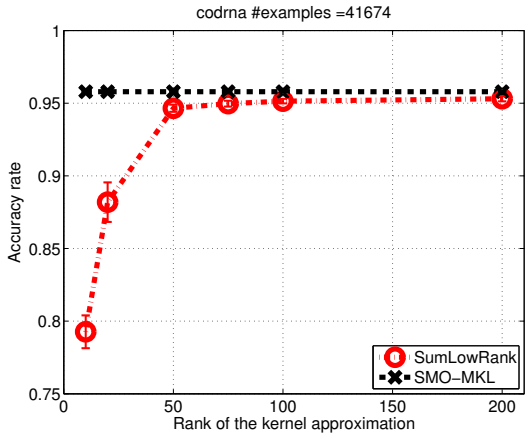


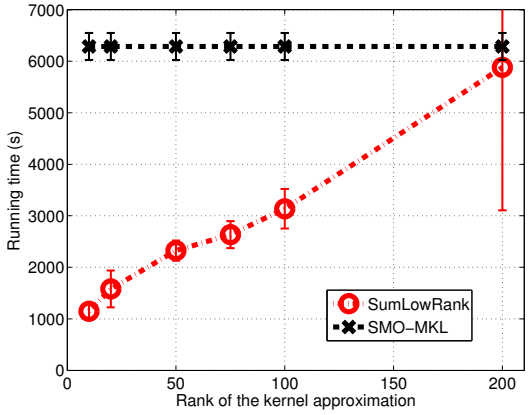
Figure 1: Comparing accuracy and running time of a SMO-MKL and our low-rank MKL (SumLowRank) algorithm for different datasets from increasing size. In this experiments, we have set $p = 2$. For each data set, the top panel reports the accuracy rate with respect to the rank of kernel approximations while the bottom one shows the running time with respect to rank of the approximation.



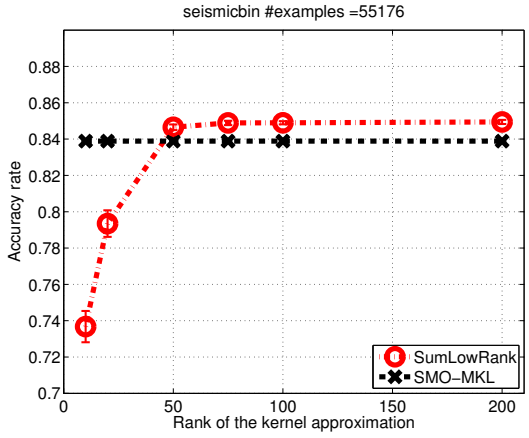


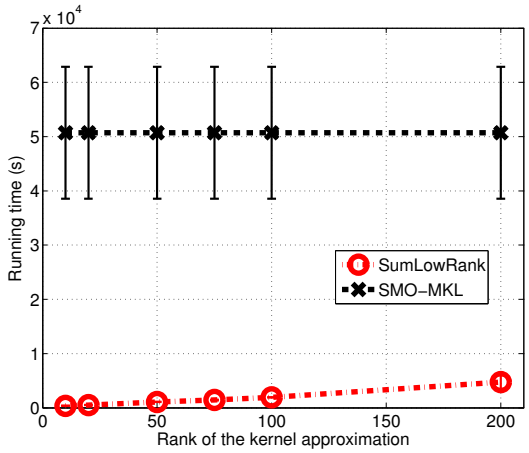
(adult)



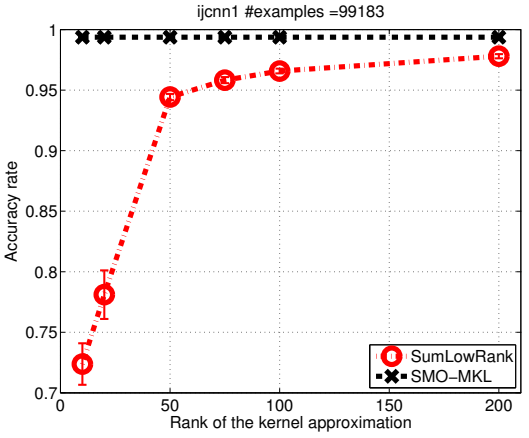


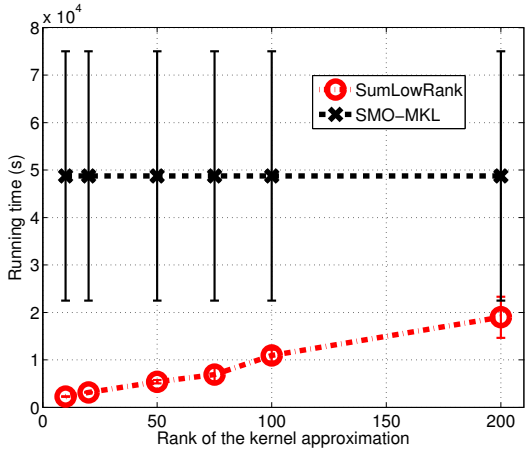
(codrna)





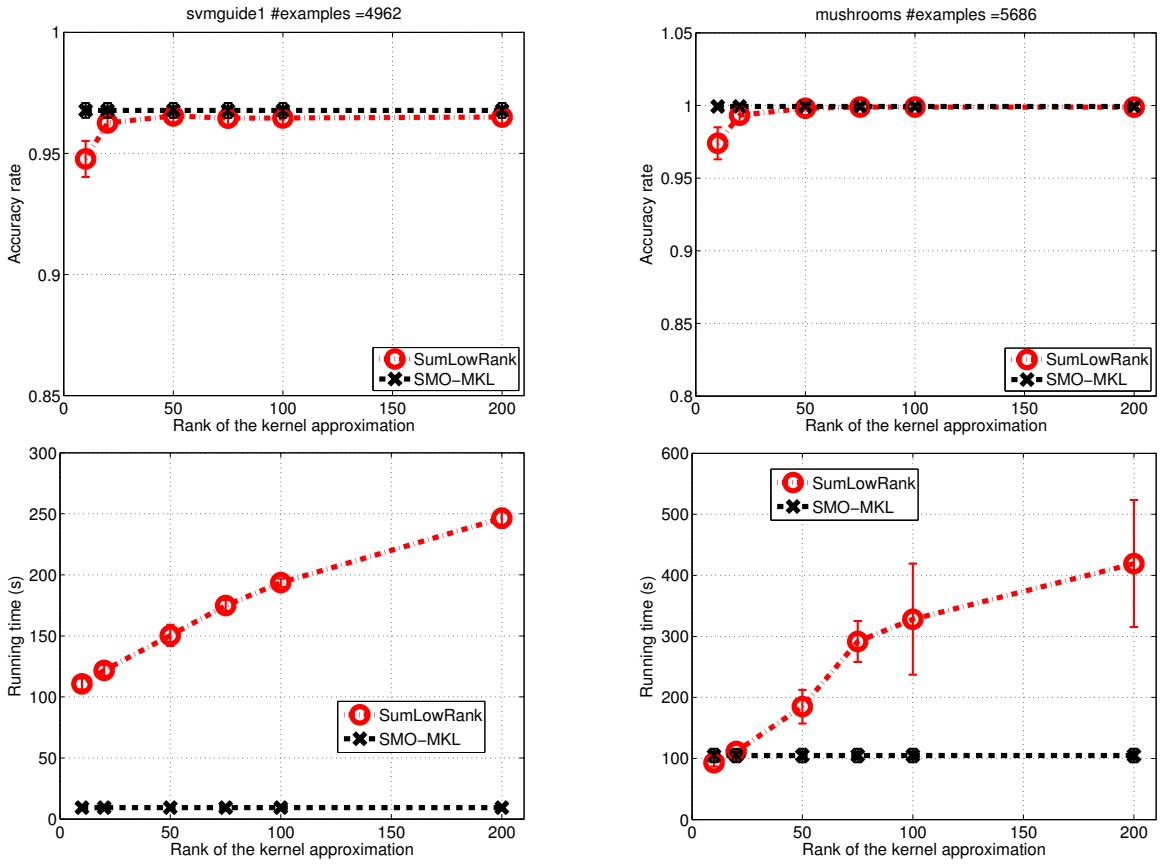
(seismic†)





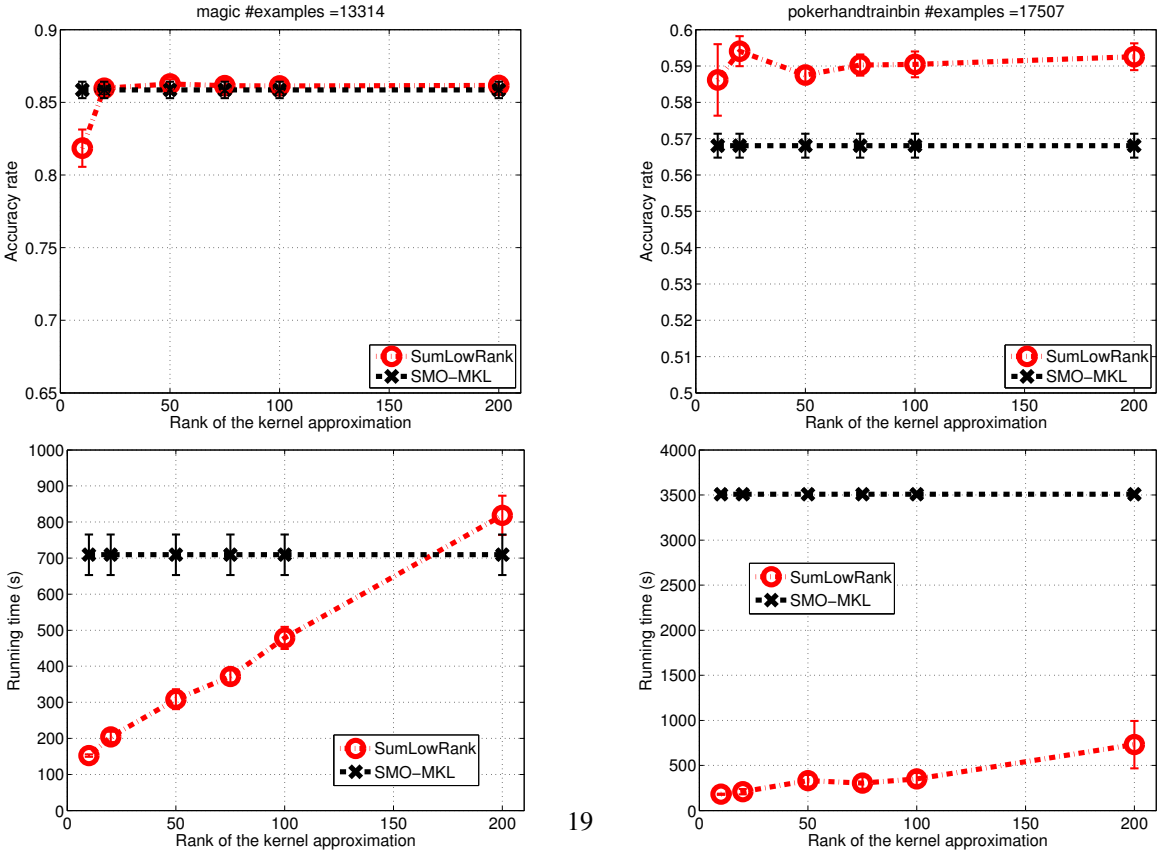
(ijcnn1)

Figure 2: Comparing accuracy and running time of a SMO-MKL and our low-rank MKL (SumLowRank) algorithm for different datasets from increasing size. In this experiments, we have set $p = 2$. For each data set, the top panel reports the accuracy rate with respect to the rank of kernel



(svmguide1)

(mushrooms)



(magic)

(poker)

Figure 3: Comparing accuracy and running time of a SMO-MKL and our low-rank MKL (SumLowRank) algorithm for different datasets from increasing size. In this experiments, we have set $p = 1.1$ and the number of kernels to 20. For each data set, the top panel reports the accuracy rate with respect to the rank of kernel approximations while the bottom one shows the running time with respect to rank of the approximation.

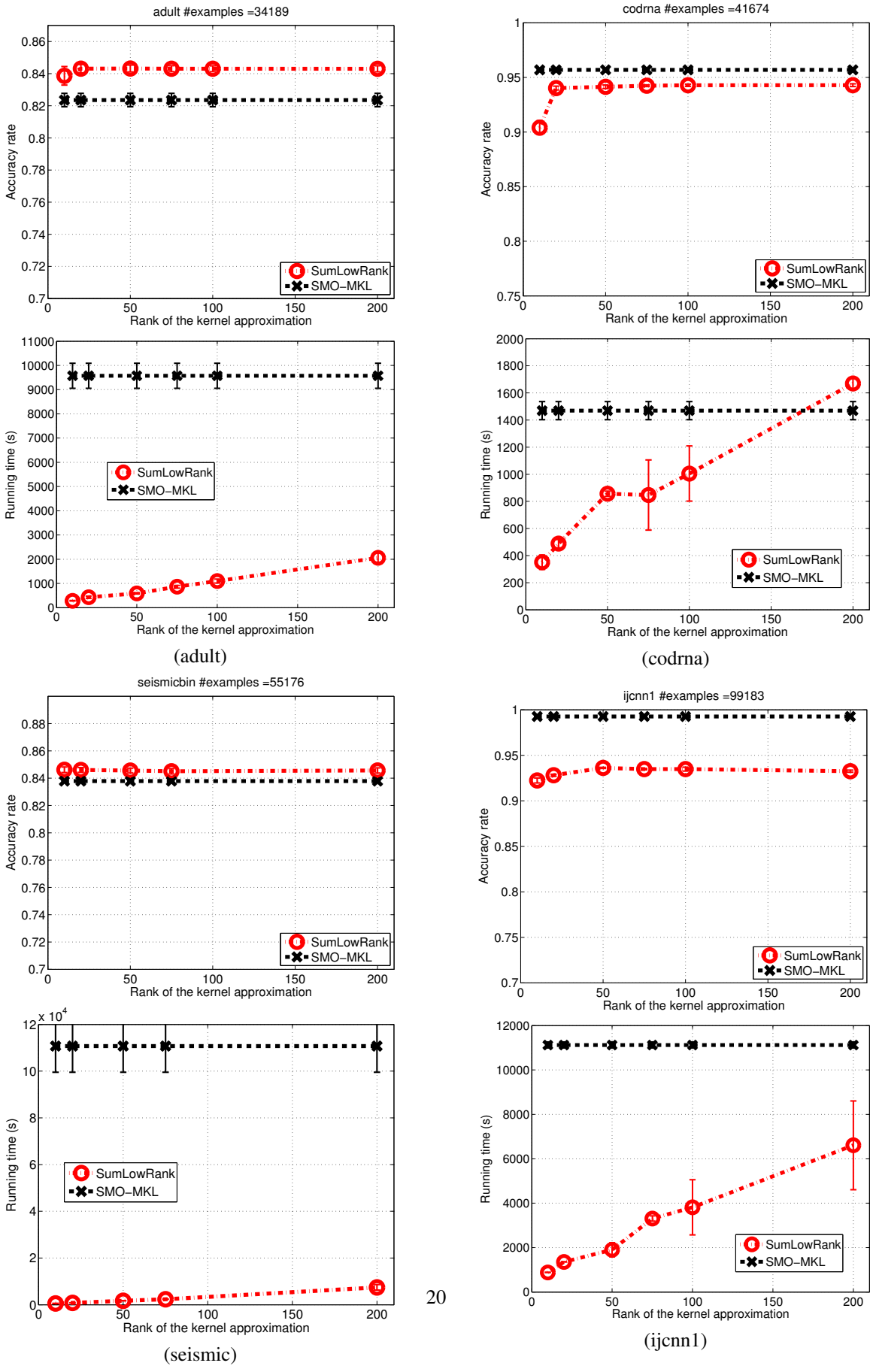
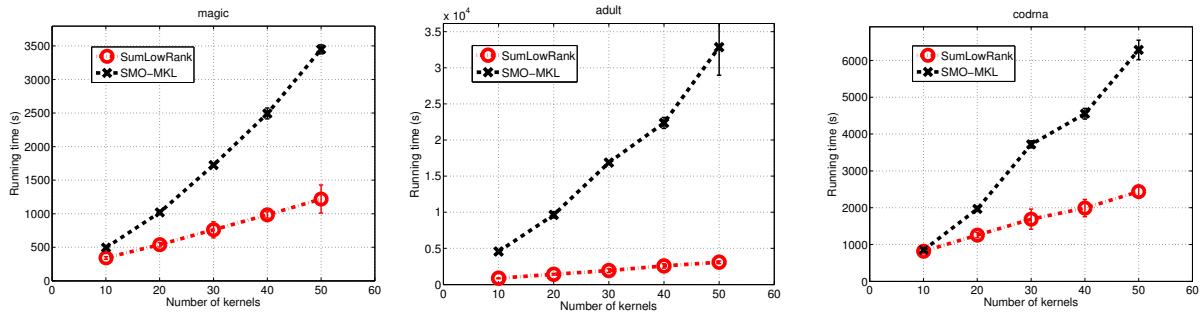
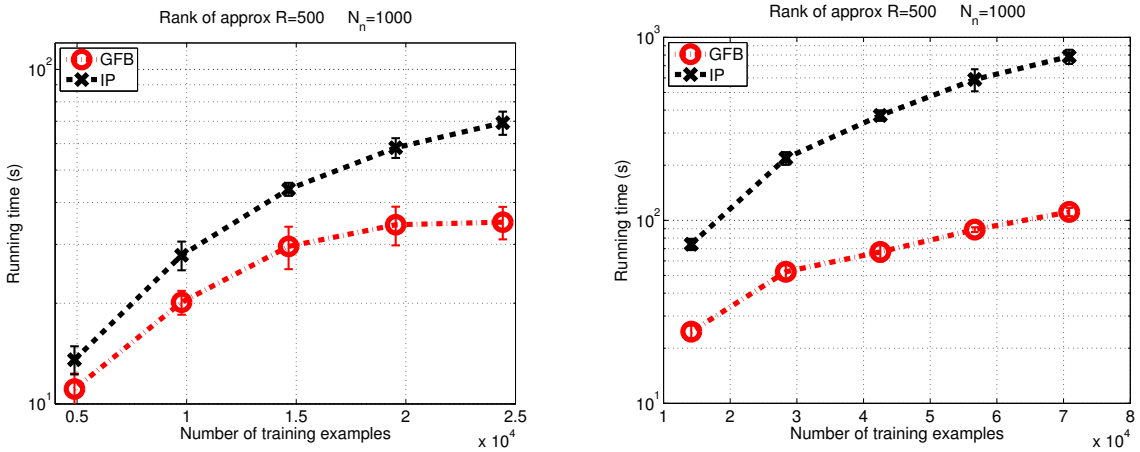


Figure 4: Comparing accuracy and running time of a SMO-MKL and our low-rank MKL (SumLowRank) algorithm for different datasets from increasing size. In this experiments, we have set $p = 1.1$ and the number of kernels to 20. For each data set, the top panel reports the accuracy rate with respect to the rank of kernel approximations while the bottom one shows the running time with respect to rank of the approximation.



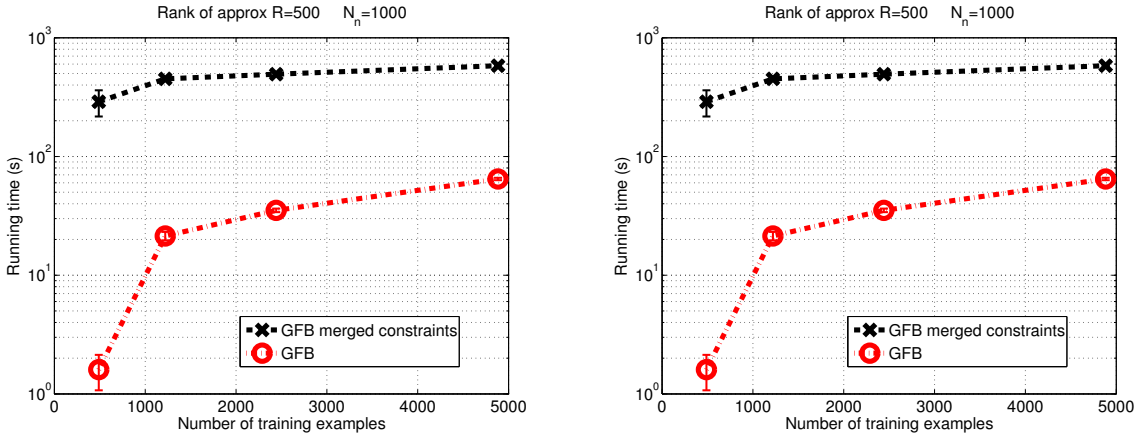
(magic) (adult) (codrna)

Figure 5: Running time with respect to the number of kernels of our low-rank MKL and the SMO-MKL. Rank of approximations have been fixed to 100 so that both algorithms provide similar performance accuracies.



(adult) (ijcnn1)

Figure 6: Comparing running time of an Mehrotra predictor-corrector IP method and a generalized forward backward algorithm for respectively solving problem ((6)) and its equivalent formulation. The number of kernels is $m = 10$ and each kernel has been approximated using rank-500 matrices



(adult) (ijcnn1)

Figure 7: Comparing running time of two usages of the generalized forward backward algorithm solving for problem (6). *GFB merged constraints* and *GFB* respectively refer to a naive implementation generalized forward-backward algorithm which handles all constraints at once and a GFB as described in Algorithm 2. The number of kernels is $m = 20$ and each kernel has been approximated using rank-500 matrices. All of these results in 5001 linear constraints to deal with.