

# SimpleMKL

Alain Rakotomamonjy, Francis Bach, Stephane Canu, Yves Grandvalet

▶ **To cite this version:**

Alain Rakotomamonjy, Francis Bach, Stephane Canu, Yves Grandvalet. SimpleMKL. Journal of Machine Learning Research, 2008, 9, pp.2491-2521. hal-00218338v2

**HAL Id: hal-00218338**
**https://hal.science/hal-00218338v2**

Submitted on 8 Sep 2008

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Journal of Machine Learning Research X (2008) 1-34 Submitted 01/08; Revised 08/08; Published XX/XX

# SimpleMKL

**Alain Rakotomamonjy** ALAIN.RAKOTOMAMONJY@INSA-ROUEN.FR
*LITIS EA 4108*
*Université de Rouen*
*76800 Saint Etienne du Rouvray, France*

**Francis R. Bach** FRANCIS.BACH@MINES.ORG
*INRIA - WILLOW Project - Team*
*Laboratoire d'Informatique de l'Ecole Normale Supérieure(CNRS/ENS/INRIA UMR 8548)*
*45, Rue d'Ulm, 75230 Paris, France*

**Stéphane Canu** STEPHANE.CANU@INSA-ROUEN.FR
*LITIS EA 4108*
*INSA de Rouen*
*76801 Saint Etienne du Rouvray, France*

**Yves Grandvalet** YVES.GRANDVALET@UTC.FR
*Idiap Research Institute, Centre du Parc, 1920 Martigny, Switzerland*
*Heudiasyc, CNRS/Université de Technologie de Compiègne (UMR 6599), 60205 Compiègne, France*

**Editor:** Nello Cristianini

## Abstract

Multiple kernel learning (MKL) aims at simultaneously learning a kernel and the associated predictor in supervised learning settings. For the support vector machine, an efficient and general multiple kernel learning algorithm, based on semi-infinite linear progamming, has been recently proposed. This approach has opened new perspectives since it makes MKL tractable for large-scale problems, by iteratively using existing support vector machine code. However, it turns out that this iterative algorithm needs numerous iterations for converging towards a reasonable solution. In this paper, we address the MKL problem through a weighted 2-norm regularization formulation with an additional constraint on the weights that encourages sparse kernel combinations. Apart from learning the combination, we solve a standard SVM optimization problem, where the kernel is defined as a linear combination of multiple kernels. We propose an algorithm, named SimpleMKL, for solving this MKL problem and provide a new insight on MKL algorithms based on mixed-norm regularization by showing that the two approaches are equivalent. We show how SimpleMKL can be applied beyond binary classification, for problems like regression, clustering (one-class classification) or multiclass classification. Experimental results show that the proposed algorithm converges rapidly and that its efficiency compares favorably to other MKL algorithms. Finally, we illustrate the usefulness of MKL for some regressors based on wavelet kernels and on some model selection problems related to multiclass classification problems.

©2008 Rakotomamonjy et al..

## 1. Introduction

During the last few years, kernel methods, such as support vector machines (SVM) have proved to be efficient tools for solving learning problems like classification or regression (Schölkopf and Smola, 2001). For such tasks, the performance of the learning algorithm strongly depends on the data representation. In kernel methods, the data representation is implicitly chosen through the so-called kernel $K(x, x')$. This kernel actually plays two roles: it defines the similarity between two examples $x$ and $x'$, while defining an appropriate regularization term for the learning problem.

Let $\{x_i, y_i\}_{i=1}^{\ell}$ be the learning set, where $x_i$ belongs to some input space $\mathcal{X}$ and $y_i$ is the target value for pattern $x_i$. For kernel algorithms, the solution of the learning problem is of the form

$$f(x) = \sum_{i=1}^{\ell} \alpha_i^\star K(x, x_i) + b^\star, \tag{1}$$

where $\alpha_i^\star$ and $b^\star$ are some coefficients to be learned from examples, while $K(\cdot, \cdot)$ is a given positive definite kernel associated with a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$.

In some situations, a machine learning practitioner may be interested in more flexible models. Recent applications have shown that using multiple kernels instead of a single one can enhance the interpretability of the decision function and improve performances (Lanckriet et al., 2004a). In such cases, a convenient approach is to consider that the kernel $K(x, x')$ is actually a convex combination of *basis* kernels:

$$K(x, x') = \sum_{m=1}^{M} d_m K_m(x, x') \ , \quad \text{with } d_m \geq 0 \ , \ \sum_{m=1}^{M} d_m = 1 \ ,$$

where $M$ is the total number of kernels. Each basis kernel $K_m$ may either use the full set of variables describing $x$ or subsets of variables stemming from different data sources (Lanckriet et al., 2004a). Alternatively, the kernels $K_m$ can simply be classical kernels (such as Gaussian kernels) with different parameters. Within this framework, the problem of data representation through the kernel is then transferred to the choice of weights $d_m$.

Learning both the coefficients $\alpha_i$ and the weights $d_m$ in a single optimization problem is known as the multiple kernel learning (MKL) problem. For binary classification, the MKL problem has been introduced by Lanckriet et al. (2004b), resulting in a quadratically constrained quadratic programming problem that becomes rapidly intractable as the number of learning examples or kernels become large.

What makes this problem difficult is that it is actually a convex but non-smooth minimization problem. Indeed, Bach et al. (2004a) have shown that the MKL formulation of Lanckriet et al. (2004b) is actually the dual of a SVM problem in which the weight vector has been regularized according to a mixed $(\ell_2, \ell_1)$-norm instead of the classical squared $\ell_2$-norm. Bach et al. (2004a) have considered a smoothed version of the problem for which they proposed a SMO-like algorithm that enables to tackle medium-scale problems.

Sonnenburg et al. (2006) reformulate the MKL problem of Lanckriet et al. (2004b) as a semi-infinite linear program (SILP). The advantage of the latter formulation is that the algorithm addresses the problem by iteratively solving a classical SVM problem with a single kernel, for which many efficient toolboxes exist (Vishwanathan et al., 2003, Loosli et al.,

2005, Chang and Lin, 2001), and a linear program whose number of constraints increases along with iterations. A very nice feature of this algorithm is that is can be extended to a large class of convex loss functions. For instance, Zien and Ong (2007) have proposed a multiclass MKL algorithm based on similar ideas.

In this paper, we present another formulation of the multiple learning problem. We first depart from the primal formulation proposed by Bach et al. (2004a) and further used by Bach et al. (2004b) and Sonnenburg et al. (2006). Indeed, we replace the mixed-norm regularization by a weighted $\ell_2$-norm regularization, where the sparsity of the linear combination of kernels is controlled by a $\ell_1$-norm constraint on the kernel weights. This new formulation of MKL leads to a smooth and convex optimization problem. By using a variational formulation of the mixed-norm regularization, we show that our formulation is equivalent to the ones of Lanckriet et al. (2004b), Bach et al. (2004a) and Sonnenburg et al. (2006).

The main contribution of this paper is to propose an efficient algorithm, named SimpleMKL, for solving the MKL problem, through a primal formulation involving a weighted $\ell_2$-norm regularization. Indeed, our algorithm is simple, essentially based on a gradient descent on the SVM objective value. We iteratively determine the combination of kernels by a gradient descent wrapping a standard SVM solver, which is SimpleSVM in our case. Our scheme is similar to the one of Sonnenburg et al. (2006), and both algorithms minimize the same objective function. However, they differ in that we use reduced gradient descent in the primal, whereas Sonnenburg et al.'s SILP relies on cutting planes. We will empirically show that our optimization strategy is more efficient, with new evidences confirming the preliminary results reported in Rakotomamonjy et al. (2007).

Then, extensions of SimpleMKL to other supervised learning problems such as regression SVM, one-class SVM or multiclass SVM problems based on pairwise coupling are proposed. Although it is not the main purpose of the paper, we will also discuss the applicability of our approach to general convex loss functions.

This paper also presents several illustrations of the usefulness of our algorithm. For instance, in addition to the empirical efficiency comparison, we also show, in a SVM regression problem involving wavelet kernels, that automatic learning of the kernels leads to far better performances. Then we depict how our MKL algorithm behaves on some multiclass problems.

The paper is organized as follows. Section 2 presents the functional settings of our MKL problem and its formulation. Details on the algorithm and discussion of convergence and computational complexity are given in Section 3. Extensions of our algorithm to other SVM problems are discussed in Section 4 while experimental results dealing with computational complexity or with comparison with other model selection methods are presented in Section 5.

A SimpleMKL toolbox based on Matlab code is available at `http://www.mloss.org`. This toolbox is an extension of our SVM-KM toolbox (Canu et al., 2003).

## 2. Multiple Kernel Learning Framework

In this section, we present our MKL formulation and derive its dual. In the sequel, $i$ and $j$ are indices on examples, wheras $m$ is the kernel index. In order to lighten notations,

we omit to specify that summations on $i$ and $j$ go from 1 to $\ell$, and that summations on $m$ go from 1 to $M$.

### 2.1 Functional framework

Before entering into the details of the MKL optimization problem, we first present the functional framework adopted for multiple kernel learning. Assume $K_m, m = 1, ..., M$ are $M$ positive definite kernels on the same input space $\mathcal{X}$, each of them being associated with an RKHS $\mathcal{H}_m$ endowed with an inner product $\langle \cdot, \cdot \rangle_m$. For any $m$, let $d_m$ be a non-negative coefficient and $\mathcal{H}'_m$ be the Hilbert space derived from $\mathcal{H}_m$ as follows:

$$\mathcal{H}'_m = \{f \mid f \in \mathcal{H}_m : \frac{\|f\|_{\mathcal{H}_m}}{d_m} < \infty\} \ ,$$

endowed with the inner product

$$\langle f, g \rangle_{\mathcal{H}'_m} = \frac{1}{d_m} \langle f, g \rangle_m \ .$$

In this paper, we use the convention that $\frac{x}{0} = 0$ if $x = 0$ and $\infty$ otherwise. This means that, if $d_m = 0$ then a function $f$ belongs to the Hilbert space $\mathcal{H}'_m$ only if $f = 0 \in \mathcal{H}_m$. In such a case, $\mathcal{H}'_m$ is restricted to the null element of $\mathcal{H}_m$.

Within this framework, $\mathcal{H}'_m$ is a RKHS with kernel $K(x, x') = d_m \ K_m(x, x')$ since

$$\begin{aligned} \forall f \in \mathcal{H}'_m \subseteq \mathcal{H}_m \ , \quad f(x) &= \langle f(\cdot), K_m(x, \cdot) \rangle_m \\ &= \frac{1}{d_m} \langle f(\cdot), d_m K_m(x, \cdot) \rangle_m \\ &= \langle f(\cdot), d_m K_m(x, \cdot) \rangle_{\mathcal{H}'_m} \ . \end{aligned}$$

Now, if we define $\mathcal{H}$ as the direct sum of the spaces $\mathcal{H}'_m$, i.e.,

$$\mathcal{H} = \bigoplus_{m=1}^{M} \mathcal{H}'_m \ ,$$

then, a classical result on RKHS (Aronszajn, 1950) says that $\mathcal{H}$ is a RKHS of kernel

$$K(x, x') = \sum_{m=1}^{M} d_m K_m(x, x') \ .$$

Owing to this simple construction, we have built a RKHS $\mathcal{H}$ for which any function is a sum of functions belonging to $\mathcal{H}_m$. In our framework, MKL aims at determining the set of coefficients $\{d_m\}$ within the learning process of the decision function. The multiple kernel learning problem can thus be envisioned as learning a predictor belonging to an adaptive hypothesis space endowed with an adaptive inner product. The forthcoming sections explain how we solve this problem.

### 2.2 Multiple kernel learning primal problem

In the SVM methodology, the decision function is of the form given in equation (1), where the optimal parameters $\alpha_i^\star$ and $b^\star$ are obtained by solving the dual of the following optimization problem:

$$\begin{array}{ll} \min\limits_{f,b,\xi} & \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum\limits_i \xi_i \\ \text{s.t.} & y_i(f(x_i)+b) \geq 1-\xi_i \quad \forall i \\ & \xi_i \geq 0 \quad \forall i \ . \end{array}$$

In the MKL framework, one looks for a decision function of the form $f(x)+b = \sum_m f_m(x) + b$, where each function $f_m$ belongs to a different RKHS $\mathcal{H}_m$ associated with a kernel $K_m$. According to the above functional framework and inspired by the multiple smoothing splines framework of Wahba (1990, chap. 10), we propose to address the MKL SVM problem by solving the following convex problem (see proof in appendix), which we will be referred to as the primal MKL problem:

$$\begin{array}{ll} \min\limits_{\{f_m\},b,\xi,d} & \frac{1}{2}\sum\limits_m \frac{1}{d_m}\|f_m\|_{\mathcal{H}_m}^2 + C\sum\limits_i \xi_i \\ \text{s.t.} & y_i\sum\limits_m f_m(x_i) + y_i b \geq 1-\xi_i \quad \forall i \\ & \xi_i \geq 0 \quad \forall i \\ & \sum\limits_m d_m = 1 \ , \quad d_m \geq 0 \quad \forall m \ , \end{array} \tag{2}$$

where each $d_m$ controls the squared norm of $f_m$ in the objective function.

The smaller $d_m$ is, the smoother $f_m$ (as measured by $\|f_m\|_{\mathcal{H}_m}$) should be. When $d_m = 0$, $\|f_m\|_{\mathcal{H}_m}$ has also to be equal to zero to yield a finite objective value. The $\ell_1$-norm constraint on the vector $d$ is a sparsity constraint that will force some $d_m$ to be zero, thus encouraging sparse basis kernel expansions.

### 2.3 Connections with mixed-norm regularization formulation of MKL

The MKL formulation introduced by Bach et al. (2004a) and further developed by Sonnenburg et al. (2006) consists in solving an optimization problem expressed in a functional form as

$$\begin{array}{ll} \min\limits_{\{f\},b,\xi} & \frac{1}{2}\left(\sum\limits_m \|f_m\|_{\mathcal{H}_m}\right)^2 + C\sum\limits_i \xi_i \\ \text{s.t.} & y_i\sum\limits_m f_m(x_i) + y_i b \geq 1-\xi_i \ \forall i \\ & \xi_i \geq 0 \ \forall i. \end{array} \tag{3}$$

Note that the objective function of this problem is not smooth since $\|f_m\|_{\mathcal{H}_m}$ is not differentiable at $f_m = 0$. However, what makes this formulation interesting is that the mixed-norm penalization of $f = \sum_m f_m$ is a soft-thresholding penalizer that leads to a sparse solution, for which the algorithm performs kernel selection (Bach, 2008). We have stated in the previous section that our problem should also lead to sparse solutions. In the following, we show that the formulations (2) and (3) are equivalent.

For this purpose, we simply show that the variational formulation of the mixed-norm regularization is equal to the weighted 2-norm regularization, (which is a particular case of a more general equivalence proposed by Micchelli and Pontil (2005)) i.e., by Cauchy-Schwartz inequality, for any vector $d$ on the simplex:

$$\begin{aligned}\left(\sum_m \|f_m\|_{\mathcal{H}_m}\right)^2 &= \left(\sum_m \frac{\|f_m\|_{\mathcal{H}_m}}{d_m^{1/2}} d_m^{1/2}\right)^2 \\ &\leqslant \left(\sum_m \frac{\|f_m\|^2_{\mathcal{H}_m}}{d_m}\right)\left(\sum_m d_m\right) \\ &\leqslant \sum_m \frac{\|f_m\|^2_{\mathcal{H}_m}}{d_m} \ ,\end{aligned}$$

where equality is met when $d_m^{1/2}$ is proportional to $\|f_m\|_{\mathcal{H}_m}/d_m^{1/2}$, that is:

$$d_m = \frac{\|f_m\|_{\mathcal{H}_m}}{\sum_q \|f_q\|_{\mathcal{H}_q}} \ , \tag{4}$$

which leads to

$$\min_{d_m \geq 0, \sum_m d_m = 1} \sum_m \frac{\|f_m\|^2_{\mathcal{H}_m}}{d_m} = \left(\sum_m \|f_m\|_{\mathcal{H}_m}\right)^2 \ . \tag{5}$$

Hence, owing to this variational formulation, the non-smooth mixed-norm objective function of problem (3) has been turned into a smooth objective function in problem (2). Although the number of variables has increased, we will see that this problem can be solved more efficiently.

### 2.4 The MKL dual problem

The dual problem is a key point for deriving MKL algorithms and for studying their convergence properties. Since our primal problem (2) is equivalent to the one of Bach et al. (2004a), they lead to the same dual. However, our primal formulation being convex and differentiable, it provides a simple derivation of the dual, that does not use conic duality.

The Lagrangian of problem (2) is

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\sum_m \frac{1}{d_m}\|f_m\|^2_{\mathcal{H}_m} + C\sum_i \xi_i + \sum_i \alpha_i \left(1 - \xi_i - y_i \sum_m f_m(x_i) - y_i b\right) - \sum_i \nu_i \xi_i \\ &\quad + \lambda\left(\sum_m d_m - 1\right) - \sum_m \eta_m d_m \ ,\end{aligned} \tag{6}$$

where $\alpha_i$ and $\nu_i$ are the Lagrange multipliers of the constraints related to the usual SVM problem, whereas $\lambda$ and $\eta_m$ are associated to the constraints on $d_m$. When setting to zero

the gradient of the Lagrangian with respect to the primal variables, we get the following

$$\begin{array}{ll} \text{(a)} & \frac{1}{d_m} f_m(\cdot) = \sum_i \alpha_i y_i K_m(\cdot, x_i) \ , \quad \forall m \\ \text{(b)} & \sum_i \alpha_i y_i = 0 \\ \text{(c)} & C - \alpha_i - \nu_i = 0 \ , \quad \forall i \\ \text{(d)} & -\frac{1}{2} \frac{\|f_m\|^2_{\mathcal{H}_m}}{d_m^2} + \lambda - \eta_m = 0 \ , \quad \forall m \ . \end{array} \tag{7}$$

We note again here that $f_m(\cdot)$ has to go to 0 if the coefficient $d_m$ vanishes. Plugging these optimality conditions in the Lagrangian gives the dual problem

$$\begin{array}{ll} \max\limits_{\alpha_i, \lambda} & \sum_i \alpha_i - \lambda \\ \text{s.t.} & \sum_i \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \quad \forall i \\ & \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K_m(x_i, x_j) \leq \lambda \ , \quad \forall m \ . \end{array} \tag{8}$$

This dual problem[1] is difficult to optimize due to the last constraint. This constraint may be moved to the objective function, but then, the latter becomes non-differentiable causing new difficulties (Bach et al., 2004a). Hence, in the forthcoming section, we propose an approach based on the minimization of the primal. In this framework, we benefit from differentiability which allows for an efficient derivation of an approximate primal solution, whose accuracy will be monitored by the duality gap.

## 3. Algorithm for solving the MKL primal problem

One possible approach for solving problem (2) is to use the alternate optimization algorithm applied by Grandvalet and Canu (1999, 2003) in another context. In the first step, problem (2) is optimized with respect to $f_m$, $b$ and $\xi$, with $d$ fixed. Then, in the second step, the weight vector $d$ is updated to decrease the objective function of problem (2), with $f_m$, $b$ and $\xi$ being fixed. In Section 2.3, we showed that the second step can be carried out in closed form. However, this approach lacks convergence guarantees and may lead to numerical problems, in particular when some elements of $d$ approach zero (Grandvalet, 1998). Note that these numerical problems can be handled by introducing a perturbed version of the alternate algorithm as shown by Argyriou et al. (2008).

Instead of using an alternate optimization algorithm, we prefer to consider here the following constrained optimization problem:

$$\min_d J(d) \quad \text{such that} \quad \sum_{m=1}^M d_m = 1, \ d_m \geq 0 \ , \tag{9}$$

1. Note that Bach et al. (2004a) formulation differs slightly, in that the kernels are weighted by some pre-defined coefficients that were not considered here.

where

$$J(d) = \left\{ \begin{array}{ll} \min\limits_{\{f\},b,\xi} & \frac{1}{2}\sum\limits_m \frac{1}{d_m}\|f_m\|^2_{\mathcal{H}_m} + C\sum\limits_i \xi_i \quad \forall i \\ \text{s.t.} & y_i \sum\limits_m f_m(x_i) + y_i b \geq 1 - \xi_i \\ & \xi_i \geq 0 \quad \forall i \ . \end{array} \right. \tag{10}$$

We show below how to solve problem (9) on the simplex by a simple gradient method. We will first note that the objective function $J(d)$ is actually an optimal SVM objective value. We will then discuss the existence and computation of the gradient of $J(\cdot)$, which is at the core of the proposed approach.

### 3.1 Computing the optimal SVM value and its derivatives

The Lagrangian of problem (10) is identical to the first line of equation (6). By setting to zero the derivatives of this Lagrangian according to the primal variables, we get conditions (7) (a) to (c), from which we derive the associated dual problem

$$\begin{array}{ll} \max\limits_{\alpha} & -\frac{1}{2}\sum\limits_{i,j}\alpha_i\alpha_j y_i y_j \sum\limits_m d_m K_m(x_i,x_j) + \sum\limits_i \alpha_i \\ \text{with} & \sum\limits_i \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \quad \forall i \ , \end{array} \tag{11}$$

which is identified as the standard SVM dual formulation using the combined kernel $K(x_i,x_j) = \sum_m d_m K_m(x_i,x_j)$. Function $J(d)$ is defined as the optimal objective value of problem (10). Because of strong duality, $J(d)$ is also the objective value of the dual problem:

$$J(d) = -\frac{1}{2}\sum_{i,j}\alpha_i^\star\alpha_j^\star y_i y_j \sum_m d_m K_m(x_i,x_j) + \sum_i \alpha_i^\star \ , \tag{12}$$

where $\alpha^\star$ maximizes (11). Note that the objective value $J(d)$ can be obtained by any SVM algorithm. Our method can thus take advantage of any progress in single kernel algorithms. In particular, if the SVM algorithm we use is able to handle large-scale problems, so will our MKL algorithm. Thus, the overall complexity of SimpleMKL is tied to the one of the single kernel SVM algorithm.

From now on, we assume that each Gram matrix $(K_m(x_i,x_j))_{i,j}$ is positive definite, with all eigenvalues greater than some $\eta > 0$ (to enforce this property, a small ridge may be added to the diagonal of the Gram matrices). This implies that, for any admissible value of $d$, the dual problem is strictly concave with convexity parameter $\eta$ (Lemaréchal and Sagastizabal, 1997). In turn, this strict concavity property ensures that $\alpha^\star$ is unique, a characteristic that eases the analysis of the differentiability of $J(\cdot)$.

Existence and computation of derivatives of optimal value functions such as $J(\cdot)$ have been largely discussed in the literature. For our purpose, the appropriate reference is Theorem 4.1 in Bonnans and Shapiro (1998), which has already been applied by Chapelle et al. (2002) for tuning squared-hinge loss SVM. This theorem is reproduced in the appendix for self-containedness. In a nutshell, it says that differentiability of $J(d)$ is ensured by

the unicity of $\alpha^\star$, and by the differentiability of the objective function that gives $J(d)$. Furthermore, the derivatives of $J(d)$ can be computed as if $\alpha^\star$ were not to depend on $d$. Thus, by simple differentiation of the dual function (11) with respect to $d_m$, we have:

$$\frac{\partial J}{\partial d_m} = -\frac{1}{2}\sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j K_m(x_i, x_j) \qquad \forall m \ . \tag{13}$$

We will see in the sequel that the applicability of this theorem can be extended to other SVM problems. Note that complexity of the gradient computation is of the order of $m \cdot n_{SV}^2$, with $n_{SV}$ being the number of support vectors for the current $d$.

### 3.2 Reduced gradient algorithm

The optimization problem we have to deal with in (9) is a non-linear objective function with constraints over the simplex. With our positivity assumption on the kernel matrices, $J(\cdot)$ is convex and differentiable with Lipschitz gradient (Lemaréchal and Sagastizabal, 1997). The approach we use for solving this problem is a reduced gradient method, which converges for such functions (Luenberger, 1984).

Once the gradient of $J(d)$ is computed, $d$ is updated by using a descent direction ensuring that the equality constraint and the non-negativity constraints on $d$ are satisfied. We handle the equality constraint by computing the reduced gradient (Luenberger, 1984, Chap. 11). Let $d_\mu$ be a non-zero entry of $d$, the reduced gradient of $J(d)$, denoted $\nabla_{red} J$, has components:

$$[\nabla_{red} J]_m = \frac{\partial J}{\partial d_m} - \frac{\partial J}{\partial d_\mu} \ \forall m \neq \mu \ , \text{ and } [\nabla_{red} J]_\mu = \sum_{m \neq \mu} \left( \frac{\partial J}{\partial d_\mu} - \frac{\partial J}{\partial d_m} \right) \ .$$

We chose $\mu$ to be the index of the largest component of vector $d$, for better numerical stability (Bonnans, 2006).

The positivity constraints have also to be taken into account in the descent direction. Since we want to minimize $J(\cdot)$, $-\nabla_{red} J$ is a descent direction. However, if there is an index $m$ such that $d_m = 0$ and $[\nabla_{red} J]_m > 0$, using this direction would violate the positivity constraint for $d_m$. Hence, the descent direction for that component is set to 0. This gives the descent direction for updating $d$ as

$$D_m = \begin{cases} 0 & \text{if } d_m = 0 \text{ and } \frac{\partial J}{\partial d_m} - \frac{\partial J}{\partial d_\mu} > 0 \\ -\frac{\partial J}{\partial d_m} + \frac{\partial J}{\partial d_\mu} & \text{if } d_m > 0 \text{ and } m \neq \mu \\ \sum_{g \neq \mu, d_\nu > 0} \left( \frac{\partial J}{\partial d_\nu} - \frac{\partial J}{\partial d_\mu} \right) & \text{for } m = \mu \ . \end{cases} \tag{14}$$

The usual updating scheme is $d \leftarrow d + \gamma D$ , where $\gamma$ is the step size. Here, as detailed in Algorithm 1, we go one step beyond: once a descent direction $D$ has been computed, we first look for the maximal admissible step size in that direction and check whether the objective value decreases or not. The maximal admissible step size corresponds to a component, say $d_\nu$, set to zero. If the objective value decreases, $d$ is updated, we set $D_\nu = 0$ and normalize $D$ to comply with the equality constraint. This procedure is repeated until the objective value

**Algorithm 1** SimpleMKL algorithm

set $d_m = \frac{1}{M}$ for $m = 1, \ldots, M$
**while** stopping criterion not met **do**
  compute $J(d)$ by using an SVM solver with $K = \sum_m d_m K_m$
  compute $\frac{\partial J}{\partial d_m}$ for $m = 1, \ldots, M$ and descent direction $D$ (14).
  set $\mu = \underset{m}{\operatorname{argmax}}\, d_m$, $J^\dagger = 0$, $d^\dagger = d$, $D^\dagger = D$
  **while** $J^\dagger < J(d)$ **do** {descent direction update}
    $d = d^\dagger$, $D = D^\dagger$
    $\nu = \underset{\{m|D_m<0\}}{\operatorname{argmin}} -d_m/D_m$, $\gamma_{\max} = -d_\nu/D_\nu$
    $d^\dagger = d + \gamma_{\max} D$, $D^\dagger_\mu = D_\mu - D_\nu$, $D^\dagger_\nu = 0$
    compute $J^\dagger$ by using an SVM solver with $K = \sum_m d^\dagger_m K_m$
  **end while**
  line search along $D$ for $\gamma \in [0, \gamma_{\max}]$ {calls an SVM solver for each $\gamma$ trial value}
  $d \leftarrow d + \gamma D$
**end while**

stops decreasing. At this point, we look for the optimal step size $\gamma$, which is determined by using a one-dimensional line search, with proper stopping criterion, such as Armijo's rule, to ensure global convergence.

In this algorithm, computing the descent direction and the line search are based on the evaluation of the objective function $J(\cdot)$, which requires solving an SVM problem. This may seem very costly but, for small variations of $d$, learning is very fast when the SVM solver is initialized with the previous values of $\alpha^\star$ (DeCoste and Wagstaff., 2000). Note that the gradient of the cost function is not computed after each update of the weight vector $d$. Instead, we take advantage of an easily updated descent direction as long as the objective value decreases. We will see in the numerical experiments that this approach saves a substantial amount of computation time compared to the usual update scheme where the descent direction is recomputed after each update of $d$. Note that we have also investigated gradient projection algorithms (Bertsekas, 1999, Chap 2.3), but this turned out to be slightly less efficient than the proposed approach, and we will not report these results.

The algorithm is terminated when a stopping criterion is met. This stopping criterion can be either based on the duality gap, the KKT conditions, the variation of $d$ between two consecutive steps or, even more simply, on a maximal number of iterations. Our implementation, based on the duality gap, is detailed in the forthcoming section.

### 3.3 Optimality conditions

In a convex constrained optimization algorithm such as the one we are considering, we have the opportunity to check for proper optimality conditions such as the KKT conditions or the duality gap (the difference between primal and dual objective values), which should be zero at the optimum. From the primal and dual objectives provided respectively in (2) and (8), the MKL duality gap is

$$\text{DualGap} = J(d^\star) - \sum_i \alpha_i^\star + \frac{1}{2}\max_m \sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j K_m(x_i, x_j) \ , \ ,$$

where $d^\star$ and $\{\alpha_i^\star\}$ are optimal primal and dual variables, and $J(d^\star)$ depends implicitly on optimal primal variables $\{f_m^\star\}$, $b^\star$ and $\{\xi_i^\star\}$. If $J(d^\star)$ has been obtained through the dual problem (11), then this MKL duality gap can also be computed from the single kernel SVM algorithm duality gap $DG_{\text{SVM}}$. Indeed, equation (12) holds only when the single kernel SVM algorithm returns an exact solution with $DG_{\text{SVM}} = 0$. Otherwise, we have

$$DG_{\text{SVM}} = J(d^\star) + \frac{1}{2}\sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j \sum_m d_m^\star K_m(x_i, x_j) - \sum_i \alpha_i^\star$$

then the MKL duality gap becomes

$$\text{DualGap} = DG_{\text{SVM}} - \frac{1}{2}\sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j \sum_m d_m^\star K_m(x_i, x_j) + \frac{1}{2}\max_m \sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j K_m(x_i, x_j) \ .$$

Hence, it can be obtained with a small additional computational cost compared to the SVM duality gap.

In iterative procedures, it is common to stop the algorithm when the optimality conditions are respected up to a tolerance threshold $\varepsilon$. Obviously, SimpleMKL has no impact on $DG_{\text{SVM}}$, hence, one may assume, as we did here, that $DG_{\text{SVM}}$ needs not to be monitored. Consequently, we terminate the algorithm when

$$\max_m \sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j K_m(x_i, x_j) - \sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j \sum_m d_m^\star K_m(x_i, x_j) \leq \varepsilon \ . \tag{15}$$

For some of the other MKL algorithms that will be presented in Section 4, the dual function may be more difficult to derive. Hence, it may be easier to rely on approximate KKT conditions as a stopping criterion. For the general MKL problem (9), the first order optimality conditions are obtained through the KKT conditions:

$$\begin{aligned} \frac{\partial J}{\partial d_m} + \lambda - \eta_m &= 0 \quad \forall m \\ \eta_m \cdot d_m &= 0 \quad \forall m \ , \end{aligned}$$

where $\lambda$ and $\{\eta_m\}$ are respectively the Lagrange multipliers for the equality and inequality constraints of (9). These KKT conditions imply

$$\begin{aligned} \frac{\partial J}{\partial d_m} &= -\lambda \quad \text{if } d_m > 0 \\ \frac{\partial J}{\partial d_m} &\geq -\lambda \quad \text{if } d_m = 0 \ . \end{aligned}$$

However, as Algorithm 1 is not based on the Lagrangian formulation of problem (9), $\lambda$ is not computed. Hence, we derive approximate necessary optimality conditions to be used for termination criterion. Let's define $dJ_{\text{min}}$ and $dJ_{\text{max}}$ as

$$dJ_{\text{min}} = \min_{\{d_m | d_m > 0\}} \frac{\partial J}{\partial d_m} \quad \text{and} \quad dJ_{\text{max}} = \max_{\{d_m | d_m > 0\}} \frac{\partial J}{\partial d_m} \ ,$$

then, the necessary optimality conditions are approximated by the following termination conditions:

$$|dJ_{\min} - dJ_{\max}| \leq \varepsilon \quad \text{and} \quad \frac{\partial J}{\partial d_m} \geq dJ_{\max} \quad \text{if } d_m = 0$$

In other words, we are considered at the optimum when the gradient components for all positive $d_m$ lie in a $\varepsilon$-tube and when all gradient components for vanishing $d_m$ are outside this tube. Note that these approximate necessary optimality conditions are available right away for any differentiable objective function $J(d)$.

### 3.4 Cutting Planes, Steepest Descent and Computational Complexity

As we stated in the introduction, several algorithms have been proposed for solving the original MKL problem defined by Lanckriet et al. (2004b). All these algorithms are based on equivalent formulations of the same dual problem; they all aim at providing a pair of optimal vectors $(d, \alpha)$.

In this subsection, we contrast SimpleMKL with its closest relative, the SILP algorithm of Sonnenburg et al. (2005, 2006). Indeed, from an implementation point of view, the two algorithms are alike, since they are wrapping a standard single kernel SVM algorithm. This feature makes both algorithms very easy to implement. They, however, differ in computational efficiency, because the kernel weights $d_m$ are optimized in quite different ways, as detailed below.

Let us first recall that our differentiable function $J(d)$ is defined as:

$$J(d) = \left\{ \begin{array}{ll} \max\limits_{\alpha} & -\frac{1}{2} \sum\limits_{i,j} \alpha_i \alpha_j y_i y_j \sum\limits_m d_m K_m(x_i, x_j) + \sum\limits_i \alpha_i \\ \text{with} & \sum\limits_i \alpha_i y_i = 0, \qquad C \geq \alpha_i \geq 0 \quad \forall i \ , \end{array} \right.$$

and both algorithms aim at minimizing this differentiable function. However, using a SILP approach in this case, does not take advantage of the smoothness of the objective function.

The SILP algorithm of Sonnenburg et al. (2006) is a *cutting plane* method to minimize $J$ with respect to $d$. For each value of $d$, the best $\alpha$ is found and leads to an affine lower bound on $J(d)$. The number of lower bounding affine functions increases as more $(d, \alpha)$ pairs are computed, and the next candidate vector $d$ is the minimizer of the current lower bound on $J(d)$, that is, the maximum over all the affine functions. Cutting planes method do converge but they are known for their instability, notably when the number of lower-bounding affine functions is small: the approximation of the objective function is then loose and the iterates may oscillate (Bonnans et al., 2003). Our steepest descent approach, with the proposed line search, does not suffer from instability since we have a differentiable function to minimize. Figure 1 illustrates the behaviour of both algorithms in a simple case, with oscillations for cutting planes and direct convergence for gradient descent.

Section 5 evaluates how these oscillations impact on the computational time of the SILP algorithm on several examples. These experiments show that our algorithm needs less costly gradient computations. Conversely, the line search in the gradient base approach requires more SVM retrainings in the process of querying the objective function. However,

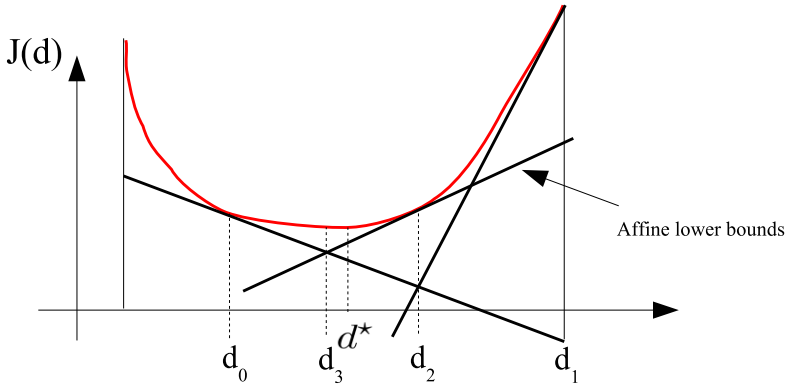



Figure 1: Illustrating three iterations of the SILP algorithm and a gradient descent algorithm for a one-dimensional problem. This dimensionality is not representative of the MKL framework, but our aim is to illustrate the typical oscillations of cutting planes around the optimal solution (with iterates $d_0$ to $d_3$). Note that computing an affine lower bound at a given $d$ requires a gradient computation. Provided the step size is chosen correctly, gradient descent converges directly towards the optimal solution without overshooting (from $d_0$ to $d^\star$).

the computation time per SVM training is considerably reduced, since the gradient based approach produces estimates of $d$ on a smooth trajectory, so that the previous SVM solution provides a good guess for the current SVM training. In SILP, with the oscillatory subsequent approximations of $d^\star$, the benefit of warm-start training severely decreases.

### 3.5 Convergence Analysis

In this paragraph, we briefly discuss the convergence of the algorithm we propose. We first suppose that problem (10) is always exactly solved, which means that the duality gap of such problem is 0. With such conditions, the gradient computation in (13) is exact and thus our algorithm performs reduced gradient descent on a continuously differentiable function $J(\cdot)$ (remember that we have assumed that the kernel matrices are positive definite) defined on the simplex $\{d | \sum_m d_m = 1, d_m \geq 0\}$, which does converge to the global minimum of $J$ (Luenberger, 1984).

However, in practice, problem (10) is not solved exactly since most SVM algorithms will stop when the duality gap is smaller than a given $\varepsilon$. In this case, the convergence of our projected gradient method is no more guaranteed by standard arguments. Indeed, the output of the approximately solved SVM leads only to an $\varepsilon$-subgradient (Bonnans et al., 2003, Bach et al., 2004a). This situation is more difficult to analyze and we plan to address it thoroughly in future work (see for instance D'Aspremont (2008) for an example of such analysis in a similar context).

## 4. Extensions

In this section, we discuss how the proposed algorithm can be simply extended to other SVM algorithms such as SVM regression, one-class SVM or pairwise multiclass SVM algorithms. More generally, we will discuss other loss functions that can be used within our MKL algorithms.

### 4.1 Extensions to other SVM Algorithms

The algorithm we described in the previous section focuses on binary classification SVMs, but it is worth noting that our MKL algorithm can be extended to other SVM algorithms with only little changes. For SVM regression with the $\varepsilon$-insensitive loss, or clustering with the one-class soft margin loss, the problem only changes in the definition of the objective function $J(d)$ in (10).

For SVM regression (Vapnik et al., 1997, Schölkopf and Smola, 2001), we have

$$J(d) = \left\{ \begin{array}{ll} \min\limits_{f_m, b, \xi_i} & \frac{1}{2}\sum\limits_m \frac{1}{d_m}\|f_m\|^2_{\mathcal{H}_m} + C\sum\limits_i (\xi_i + \xi_i^*) \\ \text{s.t.} & y_i - \sum\limits_m f_m(x_i) - b \leq \varepsilon + \xi_i \quad \forall i \\ & \sum\limits_m f_m(x_i) + b - y_i \leq \varepsilon + \xi_i^* \quad \forall i \\ & \xi_i \geq 0, \xi_i^* \leq 0 \quad \forall i \quad , \end{array} \right. \tag{16}$$

and for one-class SVMs (Schölkopf and Smola, 2001), we have:

$$J(d) = \left\{ \begin{array}{ll} \min\limits_{f_m, b, \xi_i} & \frac{1}{2}\sum\limits_m \frac{1}{d_m}\|f_m\|^2_{\mathcal{H}_m} + \frac{1}{\nu\ell}\sum\limits_i \xi_i - b \\ \text{s.t.} & \sum\limits_m f_m(x_i) \geq b - \xi_i \\ & \xi_i \geq 0 \ . \end{array} \right. \tag{17}$$

Again, $J(d)$ can be defined according to the dual functions of these two optimization problems, which are respectively

$$J(d) = \left\{ \begin{array}{ll} \max\limits_{\alpha, \beta} & \sum\limits_i (\beta_i - \alpha_i) y_i - \varepsilon \sum\limits_i (\beta_i + \alpha_i) - \frac{1}{2}\sum\limits_{i,j} (\beta_i - \alpha_i)(\beta_j - \alpha_j) \sum\limits_m d_m K_m(x_i, x_j) \\ \text{with} & \sum\limits_i (\beta_i - \alpha_i) = 0 \\ & 0 \leq \alpha_i \ , \ \beta_i \leq C, \quad \forall i \ , \end{array} \right. \tag{18}$$

and

$$J(d) = \left\{ \begin{array}{ll} \max\limits_{\alpha} & -\frac{1}{2}\sum\limits_{i,j} \alpha_i \alpha_j \sum\limits_m d_m K_m(x_i, x_j) \\ \text{with} & 0 \leq \alpha_i \leq \frac{1}{\nu\ell} \quad \forall i \\ & \sum\limits_i \alpha_i = 1 \ , \end{array} \right. \tag{19}$$

where $\{\alpha_i\}$ and $\{\beta_i\}$ are Lagrange multipliers.
Then, as long as $J(d)$ is differentiable, a property strictly related to the strict concavity of its dual function, our descent algorithm can still be applied. The main effort for the extension of our algorithm is the evaluation of $J(d)$ and the computation of its derivatives. Like for binary classification SVM, $J(d)$ can be computed by means of efficient off-the-shelf SVM solvers and the gradient of $J(d)$ is easily obtained through the dual problems. For SVM regression, we have:

$$\frac{\partial J}{\partial d_m} = -\frac{1}{2}\sum_{i,j}(\beta_i^\star - \alpha_i^\star)(\beta_j^\star - \alpha_j^\star)K_m(x_i, x_j) \qquad \forall m \ , \tag{20}$$

and for one-class SVM, we have:

$$\frac{\partial J}{\partial d_m} = -\frac{1}{2}\sum_{i,j}\alpha_i^\star\alpha_j^\star K_m(x_i, x_j) \qquad \forall m \ , \tag{21}$$

where $\alpha_i^\star$ and $\beta_i^\star$ are the optimal values of the Lagrange multipliers. These examples illustrate that extending SimpleMKL to other SVM problems is rather straighforward. This observation is valid for other SVM algorithms (based for instance on the $\nu$ parameter, a squared hinge loss or squared-$\varepsilon$ tube) that we do not detail here. Again, our algorithm can be used provided $J(d)$ is differentiable, by plugging in the algorithm the function that evaluates the objective value $J(d)$ and its gradient. Of course, the duality gap may be considered as a stopping criterion if it can be computed.

### 4.2 Multiclass Multiple Kernel Learning

With SVMs, multiclass problems are customarily solved by combining several binary classifiers. The well-known *one-against-all* and *one-against-one* approaches are the two most common ways for building a multiclass decision function based on pairwise decision functions. Multiclass SVM may also be defined right away as the solution of a global optimization problem (Weston and Watkins, 1999, Crammer and Singer, 2001), that may also be addressed with structured-output SVM (Tsochantaridis et al., 2005). Very recently, an MKL algorithm based on structured-output SVM has been proposed by Zien and Ong (2007). This work extends the work of Sonnenburg et al. (2006) to multiclass problems, with an MKL implementation still based on a QCQP or SILP approach.

Several works have compared the performance of multiclass SVM algorithms (Duan and Keerthi, 2005, Hsu and Lin, 2002, Rifkin and Klautau, 2004). In this subsection, we do not deal with this aspect; we explain how SimpleMKL can be extended to pairwise SVM multiclass implementations. The problem of applying our algorithm to structured-output SVM will be briefly discussed later.

Suppose we have a multiclass problem with $P$ classes. For a *one-against-all* multiclass SVM, we need to train $P$ binary SVM classifiers, where the $p$-th classifier is trained by considering all examples of class $p$ as positive examples while all other examples are considered negative. For a *one-against-one* multiclass problem, $P(P-1)/2$ binary SVM classifiers are built from all pairs of distinct classes. Our multiclass MKL extension of SimpleMKL differs from the binary version only in the definition of a new cost function $J(d)$. As we now look

for the combination of kernels that *jointly* optimizes all the pairwise decision functions, the objective function we want to optimize according to the kernel weights $\{d_m\}$ is:

$$J(d) = \sum_{p \in \mathcal{P}} J_p(d) \ ,$$

where $\mathcal{P}$ is the set of all pairs to be considered, and $J_p(d)$ is the binary SVM objective value for the classification problem pertaining to pair $p$.

Once the new objective function is defined, the lines of Algorithm 1 still apply. The gradient of $J(d)$ is still very simple to obtain, since owing to linearity, we have:

$$\frac{\partial J}{\partial d_m} = -\frac{1}{2} \sum_{p \in \mathcal{P}} \sum_{i,j} \alpha^\star_{i,p} \alpha^\star_{j,p} y_i y_j K_m(x_i, x_j) \qquad \forall m \ , \tag{22}$$

where $\alpha_{j,p}$ is the Lagrange multiplier of the $j$-th example involved in the $p$-th decision function. Note that those Lagrange multipliers can be obtained *independently* for each pair.

The approach described above aims at finding the combination of kernels that *jointly* optimizes all binary classification problems: this one set of features should maximize the sum of margins. Another possible and straightforward approach consists in running independently SimpleMKL for each classification task. However, this choice is likely to result in as many combinations of kernels as there are binary classifiers.

### 4.3 Other loss functions

Multiple kernel learning has been of great interest and since the seminal work of Lanckriet et al. (2004b), several works on this topic have flourished. For instance, multiple kernel learning has been transposed to least-square fitting and logistic regression (Bach et al., 2004b). Independently, several authors have applied mixed-norm regularization, such as the additive spline regression model of Grandvalet and Canu (1999). This type of regularization, which is now known as the *group lasso*, may be seen as a linear version of multiple kernel learning (Bach, 2008). Several algorithms have been proposed for solving the group lasso problem. Some of them are based on projected gradient or on coordinate descent algorithm. However, they all consider the non-smooth version of the problem.

We previously mentioned that Zien and Ong (2007) have proposed an MKL algorithm based on structured-output SVMs. For such problem, the loss function, which differs from the usual SVM hinge loss, leads to an algorithm based on cutting planes instead of the usual QP approach.

Provided the gradient of the objective value can be obtained, our algorithm can be applied to group lasso and structured-output SVMs. The key point is whether the theorem of Bonnans et al. (2003) can be applied or not. Although we have not deeply investigated this point, we think that many problems comply with this requirement, but we leave these developments for future work.

### 4.4 Approximate regularization path

SimpleMKL requires the setting of the usual SVM hyperparameter $C$, which usually needs to be tuned for the problem at hand. For doing so, a practical and useful technique

is to compute the so-called regularization path, which describes the set of solutions as $C$ varies from 0 to $\infty$.

Exact path following techniques have been derived for some specific problems like SVMs or the lasso (Hastie et al., 2004, Efron et al., 2004). Besides, regularization paths can be sampled by predictor-corrector methods (Rosset, 2004, Bach et al., 2004b).

For model selection purposes, an approximation of the regularization path may be sufficient. This approach has been applied for instance by Koh et al. (2007) in regularized logistic regression.

Here, we compute an approximate regularization path based on a warm-start technique. Suppose, that for a given value of $C$, we have computed the optimal $(d^\star, \alpha^\star)$ pair; the idea of a warm-start is to use this solution for initializing another MKL problem with a different value of $C$. In our case, we iteratively compute the solutions for decreasing values of $C$ (note that $\alpha^\star$ has to be modified to be a feasible initialization of the more constrained SVM problem).

## 5. Numerical experiments

In this experimental section, we essentially aim at illustrating three points. The first point is to show that our gradient descent algorithm is efficient. This is achieved by binary classification experiments, where SimpleMKL is compared to the SILP approach of Sonnenburg et al. (2006). Then, we illustrate the usefulness of a multiple kernel learning approach in the context of regression. The examples we use are based on wavelet-based regression in which the multiple kernel learning framework naturally fits. The final experiment aims at evaluating the multiple kernel approach in a model selection problem for some multiclass problems.

### 5.1 Computation time

The aim of this first set of experiments is to assess the running times of SimpleMKL. [2] First, we compare with SILP regarding the time required for computing a single solution of MKL with a given $C$ hyperparameter. Then, we compute an approximate regularization path by varying $C$ values. We finally provide hints on the expected complexity of SimpleMKL, by measuring the growth of running time as the number of examples or kernels increases.

#### 5.1.1 Time needed for reaching a single solution

In this first benchmark, we put SimpleMKL and SILP side by side, for a fixed value of the hyperparameter $C$ ($C = 100$). This procedure, which does not take into account a proper model selection procedure, is not representative of the typical use of SVMs. It is however relevant for the purpose of comparing algorithmic issues.

The evaluation is made on five datasets from the UCI repository: *Liver*, *Wpbc*, *Ionosphere*, *Pima*, *Sonar* (Blake and Merz, 1998). The candidate kernels are:

2. All the experiments have been run on a Pentium D-3 GHz with 3 GB of RAM.

- Gaussian kernels with 10 different bandwidths $\sigma$, on all variables and on each single variable;
- polynomial kernels of degree 1 to 3, again on all and each single variable.

All kernel matrices have been normalized to unit trace, and are precomputed prior to running the algorithms.

Both SimpleMKL and SILP wrap an SVM dual solver based on SimpleSVM, an active constraints method written in Matlab (Canu et al., 2003). The descent procedure of SimpleMKL is also implemented in Matlab, whereas the linear programming involved in SILP is implemented in the publicly available toolbox LPSOLVE (Berkelaar et al., 2004).

For a fair comparison, we use the same stopping criterion for both algorithms. They halt when, either the duality gap is lower than 0.01, or the number of iterations exceeds 2000. Quantitatively, the displayed results differ from the preliminary version of this work, where the stopping criterion was based on the stabilization of the weights, but they are qualitatively similar (Rakotomamonjy et al., 2007).

For each dataset, the algorithms were run 20 times with different train and test sets (70% of the examples for training and 30% for testing). Training examples were normalized to zero mean and unit variance.

In Table 1, we report different performance measures: accuracy, number of selected kernels and running time. As the latter is mainly spent in querying the SVM solver and in computing the gradient of $J$ with respect to $d$, the number of calls to these two routines is also reported.

Both algorithms are nearly identical in performance accuracy. Their number of selected kernels are of same magnitude, although SimpleMKL tends to select 10 to 20% more kernels. As both algorithms address the same convex optimization problem, with convergent methods starting from the same initialization, the observed differences are only due to the inaccuracy of the solution when the stopping criterion is met. Hence, the trajectories chosen by each algorithm for reaching the solution, detailed in Section 3.4, explain the differences in the number of selected kernels. The updates of $d_m$ based on the descent algorithm of SimpleMKL are rather conservative (small steps departing from $1/M$ for all $d_m$), whereas the oscillations of cutting planes are likely to favor extreme solutions, hitting the edges of the simplex.

This explanation is corroborated by Figure 2, which compares the behavior of the $d_m$ coefficients through time. The instability of SILP is clearly visible, with very high oscillations in the first iterations and a noticeable residual noise in the long run. In comparison, the trajectories for SimpleSVM are much smoother.

If we now look at the overall difference in computation time reported in Table 1, clearly, on all data sets, SimpleSVM is faster than SILP, with an average gain factor of about 5. Furthermore, the larger the number of kernels is, the larger the speed gain we achieve. Looking at the last column of Table 1, we see that the main reason for improvement is that SimpleMKL converges in fewer iterations (that is, gradient computations). It may seem surprising that this gain is not conterbalanced by the fact that SimpleMKL requires many more calls to the SVM solver (on average, about 4 times). As we stated in Section 3.4, when the number of kernels is large, computing the gradient may be expensive compared to SVM retraining with warm-start techniques.

Table 1: Average performance measures for the two MKL algorithms and a plain gradient descent algorithm.

Liver $\ell = 241$ $M = 91$

| Algorithm | # Kernel | Accuracy | Time (s) | # SVM eval | # Gradient eval |
|---|---|---|---|---|---|
| SILP | 10.6 ± 1.3 | 65.9 ± 2.6 | 47.6 ± 9.8 | 99.8 ± 20 | 99.8 ± 20 |
| SimpleMKL | 11.2 ± 1.2 | 65.9 ± 2.3 | 18.9 ± 12.6 | 522 ± 382 | 37.0 ± 26 |
| Grad. Desc. | 11.6 ± 1.3 | 66.1 ± 2.7 | 31.3 ± 14.2 | 972 ± 630 | 103 ± 27 |

Pima $\ell = 538$ $M = 117$

| Algorithm | # Kernel | Accuracy | Time (s) | # SVM eval | # Gradient eval |
|---|---|---|---|---|---|
| SILP | 11.6 ± 1.0 | 76.5 ± 2.3 | 224 ± 37 | 95.6 ± 13 | 95.6 ± 13 |
| SimpleMKL | 14.7 ± 1.4 | 76.5 ± 2.6 | 79.0 ± 13 | 314 ± 44 | 24.3 ± 4.8 |
| Grad. Desc. | 14.8 ± 1.4 | 75.5 ± 2.5 | 219 ± 24 | 873 ± 147 | 118 ± 8.7 |

Ionosphere $\ell = 246$ $M = 442$

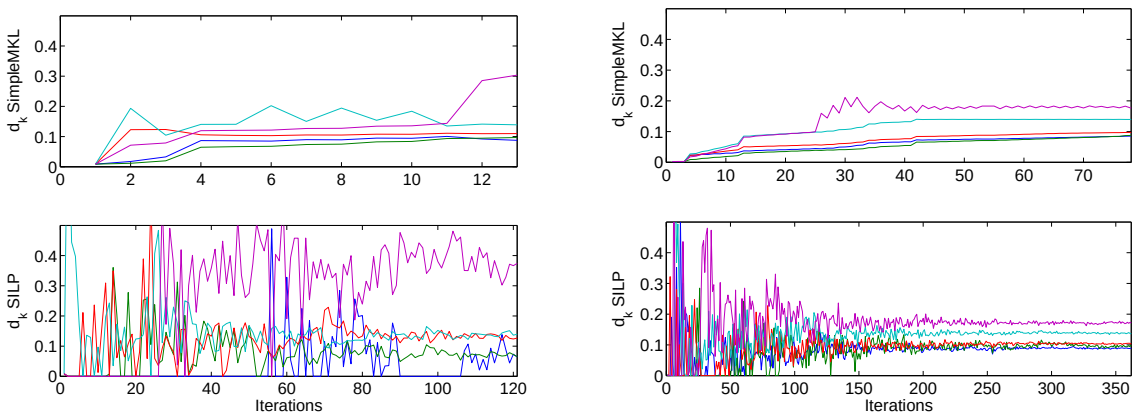
| Algorithm | # Kernel | Accuracy | Time (s) | # SVM eval | # Gradient eval |
|---|---|---|---|---|---|
| SILP | 21.6 ± 2.2 | 91.7 ± 2.5 | 535 ± 105 | 403 ± 53 | 403 ± 53 |
| SimpleMKL | 23.6 ± 2.6 | 91.5 ± 2.5 | 123 ± 46 | 1170 ± 369 | 64 ± 25 |
| Grad. Desc. | 22.9 ± 3.2 | 92.1 ± 2.5 | 421 ± 61.9 | 4000 ± 874 | 478 ± 38 |

Wpbc $\ell = 136$ $M = 442$

| Algorithm | # Kernel | Accuracy | Time (s) | # SVM eval | # Gradient eval |
|---|---|---|---|---|---|
| SILP | 13.7 ± 2.5 | 76.8 ± 1.2 | 88.6 ± 32 | 157 ± 44 | 157 ± 44 |
| SimpleMKL | 15.8 ± 2.4 | 76.7 ± 1.2 | 20.6 ± 6.2 | 618 ± 148 | 24 ± 10 |
| Grad. Desc. | 16.8 ± 2.8 | 76.9 ± 1.5 | 106 ± 6.1 | 2620 ± 232 | 361 ± 16 |

Sonar $\ell = 146$ $M = 793$

| Algorithm | # Kernel | Accuracy | Time (s) | # SVM eval | # Gradient eval |
|---|---|---|---|---|---|
| SILP | 33.5 ± 3.8 | 80.5 ± 5.1 | 2290± 864 | 903 ± 187 | 903 ± 187 |
| SimpleMKL | 36.7 ± 5.1 | 80.6 ± 5.1 | 163 ± 93 | 2770 ± 1560 | 115 ± 66 |
| Grad. Desc. | 35.7 ± 3.9 | 80.2 ± 4.7 | 469 ± 90 | 7630 ± 2600 | 836 ± 99 |




*Pima* *Ionosphere*

Figure 2: Evolution of the five largest weights $d_m$ for SimpleMKL and SILP; left row: *Pima*; right row: *Ionosphere*.

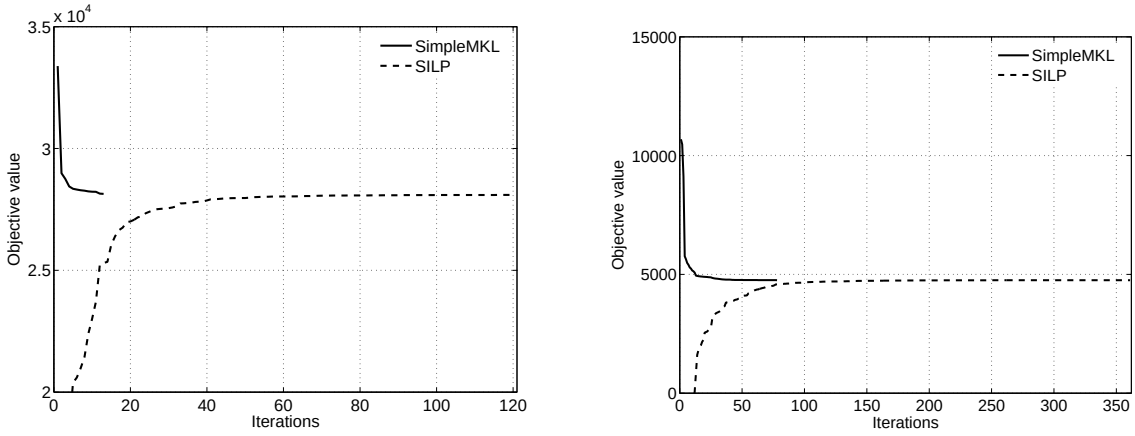
To understand why, with this large number of calls to the SVM solver, SimpleMKL is still much faster than SILP, we have to look back at Figure 2. On the one hand, the large variations in subsequents $d_m$ values for SILP, entail that subsequent SVM problems are not likely to have similar solutions: a warm-start call to the SVM solver does not help much. On the other hand, with the smooth trajectories of $d_m$ in SimpleMKL, the previous SVM solution is often a good guess for the current problem: a warm-start call to the SVM solver results in much less computation than a call from scratch.

Table 1 also shows the results obtained when replacing the update scheme described in Algorithm 1 by a usual reduced gradient update, which, at each iteration, modifies $d$ by computing the optimal step size on the descent direction $D$ (14). The training of this variant is considerably slower than SimpleMKL and is only slightly better than SILP. We see that the gradient descent updates require many more calls to the SVM solver and a number of gradient computations comparable with SILP. Note that, compared to SILP, the numerous additional calls to the SVM solver have not a drastic effect on running time. The gradient updates are stable, so that they can benefit from warm-start contrary to SILP.

To end this first series of experiments, Figure 3 depicts the evolution of the objective function for the data sets that were used in Figure 2. Besides the fact that SILP needs more iterations for achieving a good approximation of the final solution, it is worth noting that the objective values rapidly reach their steady state while still being far from convergence, when $d_m$ values are far from being settled. Thus, monitoring objective values is not suitable to assess convergence.

#### 5.1.2 Time needed for getting an approximate regularization path

In practice, the optimal value of $C$ is unknown, and one has to solve several SVM problems, spanning a wide range of $C$ values, before choosing a solution according to some model selection criterion like the cross-validation error. Here, we further pursue the comparison




*Pima* *Ionosphere*

Figure 3: Evolution of the objective values for SimpleSVM and SILP; left row: *Pima*; right row: *Ionosphere*.

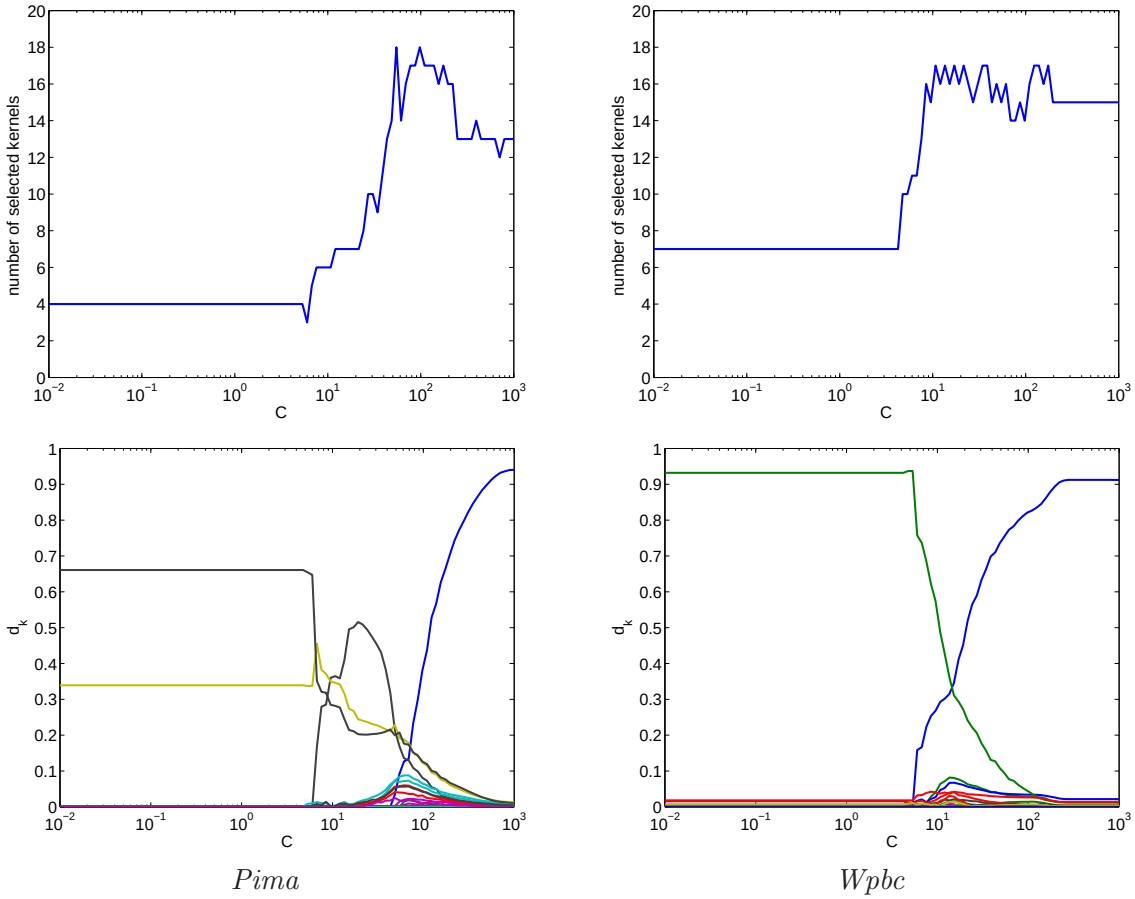
of the running times of SimpleMKL and SILP, in a series of experiments that include the search for a sensible value of $C$.

In this new benchmark, we use the same data sets as in the previous experiments, with the same kernel settings. The task is only changed in the respect that we now evaluate the running times needed by both algorithms to compute an approximate regularization path.

For both algorithms, we use a simple warm-start technique, which consists in using the optimal solutions $\{d_m^\star\}$ and $\{\alpha_i^\star\}$ obtained for a given $C$ to initialize a new MKL problem with $C + \Delta C$ (DeCoste and Wagstaff., 2000). As described in Section 4.4, we start from the largest $C$ and then approximate the regularization path by decreasing its value. The set of $C$ values is obtained by evenly sampling the interval $[0.01, 1000]$ on a logarithmic scale.

Figure 4 shows the variations of the number of selected kernels and the values of $d$ along the regularization path for the *Pima* and *Wpbc* datasets. The number of kernels is not a monotone function of $C$: for small values of $C$, the number of kernels is somewhat constant, then, it rises rapidly. There is a small overshooth before reaching a plateau corresponding to very high values of $C$. This trend is similar for the number of leading terms in the kernel weight vector $d$. Both phenomenon were observed consistently over the datasets we used.

Table 2 displays the average computation time (over 10 runs) required for building the approximate regularization path. As previously, SimpleMKL is more efficient than SILP, with a gain factor increasing with the number of kernels in the combination. The range of gain factors, from 5.9 to 23, is even more impressive than in the previous benchmark. SimpleMKL benefits from the continuity of solutions along the regularization path, whereas SILP does not take advantage of warm starts. Even provided with a good initialization, it needs many cutting planes to stabilize.




*Pima* *Wpbc*

Figure 4: Regularization paths for $d_m$ and the number of selected kernels versus $C$; left row: *Pima*; right row: *Wpbc*.

Table 2: Average computation time (in seconds) for getting an approximate regularization path. For the *Sonar* data set, SILP was extremely slow, so that regularization path was computed only once.

| Dataset | SimpleMKL | SILP | Ratio |
|---|---|---|---|
| Liver | 148 ± 37 | 875 ± 125 | 5.9 |
| Pima | 1030 ± 195 | 6070 ± 1430 | 5.9 |
| Ionosphere | 1290 ± 927 | 8840 ± 1850 | 6.8 |
| Wpbc | 88 ± 16 | 2040 ± 544 | 23 |
| Sonar | 625 ± 174 | $1.52 \cdot 10^5$ (*) | 243 |

#### 5.1.3 More on SimpleMKL running times

Here, we provide an empirical assessment of the expected complexity of SimpleMKL on different data sets from the UCI repository. We first look at the situation where kernel matrices can be pre-computed and stored in memory, before reporting experiments where the memory are too high, leading to repeated kernel evaluations.

In a first set of experiments, we use Gaussian kernels, computed on random subsets of variables and with random width. These kernels are precomputed and stored in memory, and we report the average CPU running times obtained from 20 runs differing in the random draw of training examples. The stopping criterion is the same as in the previous section: a relative duality gap less than $\varepsilon = 0.01$.

The first two rows of Figure 5 depicts the growth of computation time as the number of kernel increases. We observe a nearly linear trend for the four learning problems. This growth rate could be expected considering the linear convergence property of gradient techniques, but the absence of overhead is valuable.

The last row of Figure 5 depicts the growth of computation time as the number of examples increases. Here, the number of kernels is set to 10. In these plots, the observed trend is clearly superlinear. Again, this trend could be expected, considering that SVM expected training times are superlinear in the number of training examples. As we already mentioned, the complexity of SimpleMKL is tightly linked to the one of SVM training (for some examples of single kernel SVM running time, one can refer to the work of Loosli and Canu (2007)).

When all the kernels used for MKL cannot be stored in memory, one can resort to a decomposition method. Table 3 reports the average computation times, over 10 runs, in this more difficult situation. The large-scale SVM scheme of Joachims (1999) has been implemented, with basis kernels recomputed whenever needed. This approach is computationally expensive but goes with no memory limit. For these experiments, the stopping criterion is based on the variation of the weights $d_m$. As shown in Figure 2, the kernel weights rapidly reach a steady state and many iterations are spent for fine tuning the weight and reach the duality gap tolerance. Here, we trade the optimality guarantees provided by the duality gap for substantial computational time savings. The algorithm terminates when the kernel weights variation is lower than 0.001.

Results reported in Table 3 just aim at showing that medium and large-scale situations can be handled by SimpleMKL. Note that Sonnenburg et al. (2006) have run a modified version of their SILP algorithm on a larger scale datasets. However, for such experiments, they have taken advantage of some specific feature map properties. And, as they stated, for general cases where kernel matrices are dense, they have to rely on the SILP algorithm we used in this section for efficiency comparison .

### 5.2 Multiple kernel regression examples

Several research papers have already claimed that using multiple kernel learning can lead to better generalization performances in some classification problems (Lanckriet et al., 2004a, Zien and Ong, 2007, Harchaoui and Bach, 2007). This next experiment aims at illustrating this point but in the context of regression. The problem we deal with is a classical univariate regression problem where the design points are irregular (D'Amato et al.,

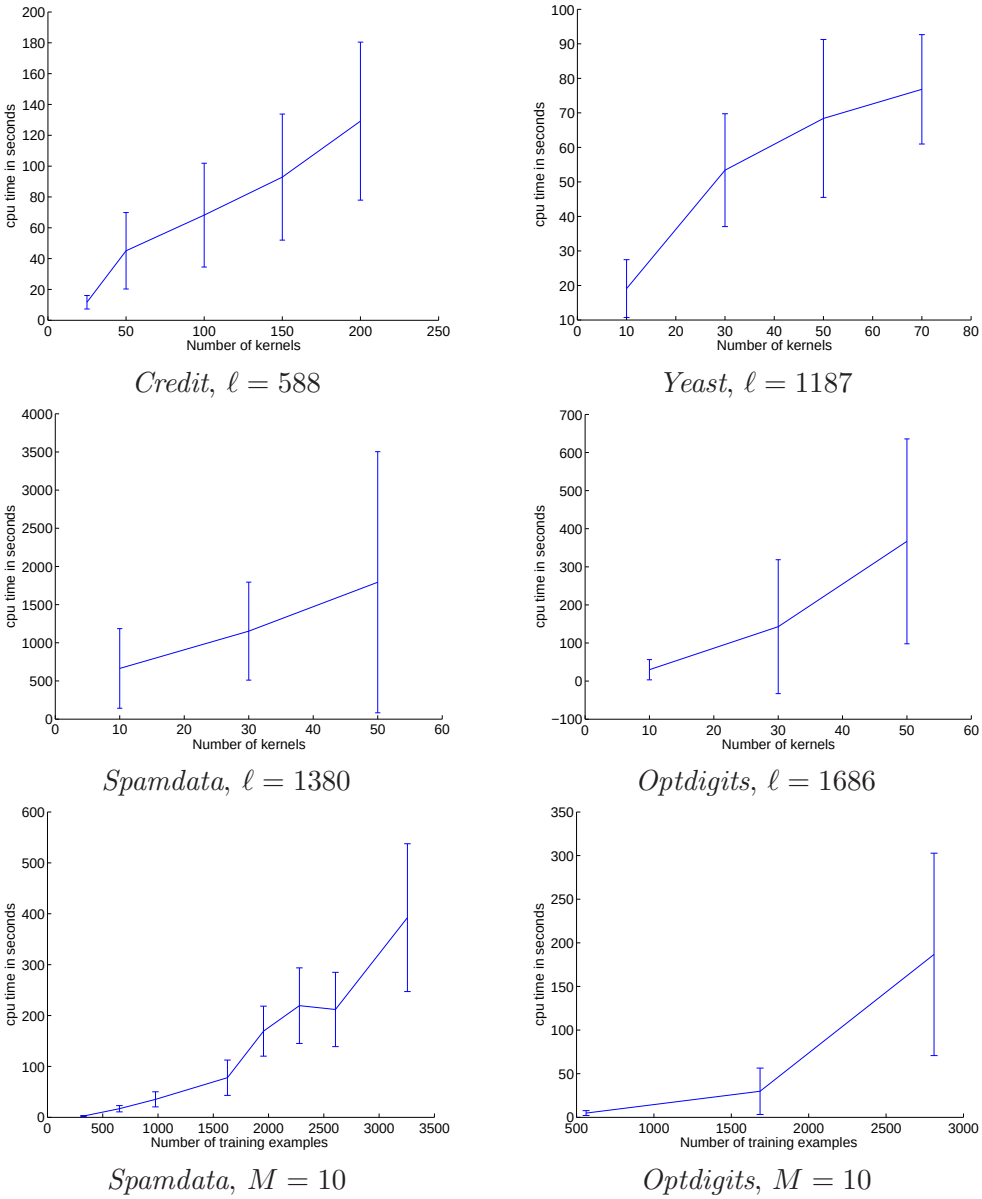



Figure 5: SimpleMKL average computation times for different datasets; top two rows: number of training examples fixed, number of kernels varying; bottom row: number of training examples varying, number of kernels fixed.

Table 3: Average computation time needed by SimpleSVM using decomposition methods.

| Dataset | Nb Examples | # Kernel | Accuracy (%) | Time (s) |
|---|---|---|---|---|
| Yeast | 1335 | 22 | 77.25 | 1130 |
| Spamdata | 4140 | 71 | 93.49 | 34200 |

2006). Furthermore, according to equation (16), we look for the regression function $f(x)$ as a linear combination of functions each belonging to a wavelet based reproducing kernel Hilbert space.

The algorithm we use is a classical SVM regression algorithm with multiple kernels where each kernel is built from a set of wavelets. These kernels have been obtained according to the expression:

$$K(x,x') = \sum_j \sum_s \frac{1}{2^j} \psi_{j,s}(x)\psi_{j,s}(x')$$

where $\psi(\cdot)$ is a mother wavelet and $j$,$s$ are respectively the dilation and translation parameters of the wavelet $\psi_{j,s}(\cdot)$. The theoretical details on how such kernels can been built are available in D'Amato et al. (2006), Rakotomamonjy and Canu (2005), Rakotomamonjy et al. (2005).

Our hope when using multiple kernel learning in this context is to capture the multiscale structure of the target function. Hence, each kernel involved in the combination should be weighted accordingly to its correlation to the target function. Furthermore, such a kernel has to be built according to the multiscale structure we wish to capture. In this experiment, we have used three different choices of multiple kernels setting. Suppose we have a set of wavelets with $j \in [j_{min}, j_{max}]$ and $s \in\in [s_{min}, s_{max}]$.

First of all, we have build a single kernel from all the wavelets according to the above equation. Then we have created kernels from all wavelets of a given scale (dilation)

$$K_{Dil,J}(x,x') = \sum_{s=s_{min}}^{s_{max}} \frac{1}{2^j} \psi_{J,s}(x)\psi_{J,s}(x') \qquad \forall J \in [j_{min}, j_{max}]$$

and lastly, we have a set of kernels, where each kernel is built from wavelets located at a given scale and given time-location:

$$K_{Dil-Trans,J,\mathcal{S}}(x,x') = \sum_{s=\mathcal{S}} \frac{1}{2^j} \psi_{J,s}(x)\psi_{J,s}(x') \qquad \forall J \in [j_{min}, j_{max}]$$

where $\mathcal{S}$ is a given set of translation parameter. These sets are built by splitting the full translation parameters index in contiguous and non-overlapping index. The mother wavelet we used is a *Symmlet* Daubechies wavelet with 6 vanishing moments. The resolution levels of the wavelet goes from $j_{min} = -3$ to $j_{min} = 6$. According to these settings, we have 10 dilation kernels and 48 dilation-translation kernels.

We applied this MKL SVM regression algorithm to simulated datasets which are well-known functions in the wavelet literature (Antoniadis and Fan, 2001). Each signal length is 512 and a Gaussian independent random has been added to each signal so that the signal to noise ratio is equal to 5. Examples of the true signals and their noisy versions are displayed in Figure 6. Note that the *LinChirp* and *Wave* signals present some multiscale features that should suit well to an MKL approach.

Performance of the different multiple kernel settings have been compared according to the following experimental setting. For each training signal, we have estimated the regularization parameter $C$ of the MKL SVM regression by means of a validation procedure. The 512 samples have been randomly separated in a learning and a validation sets. Then, by

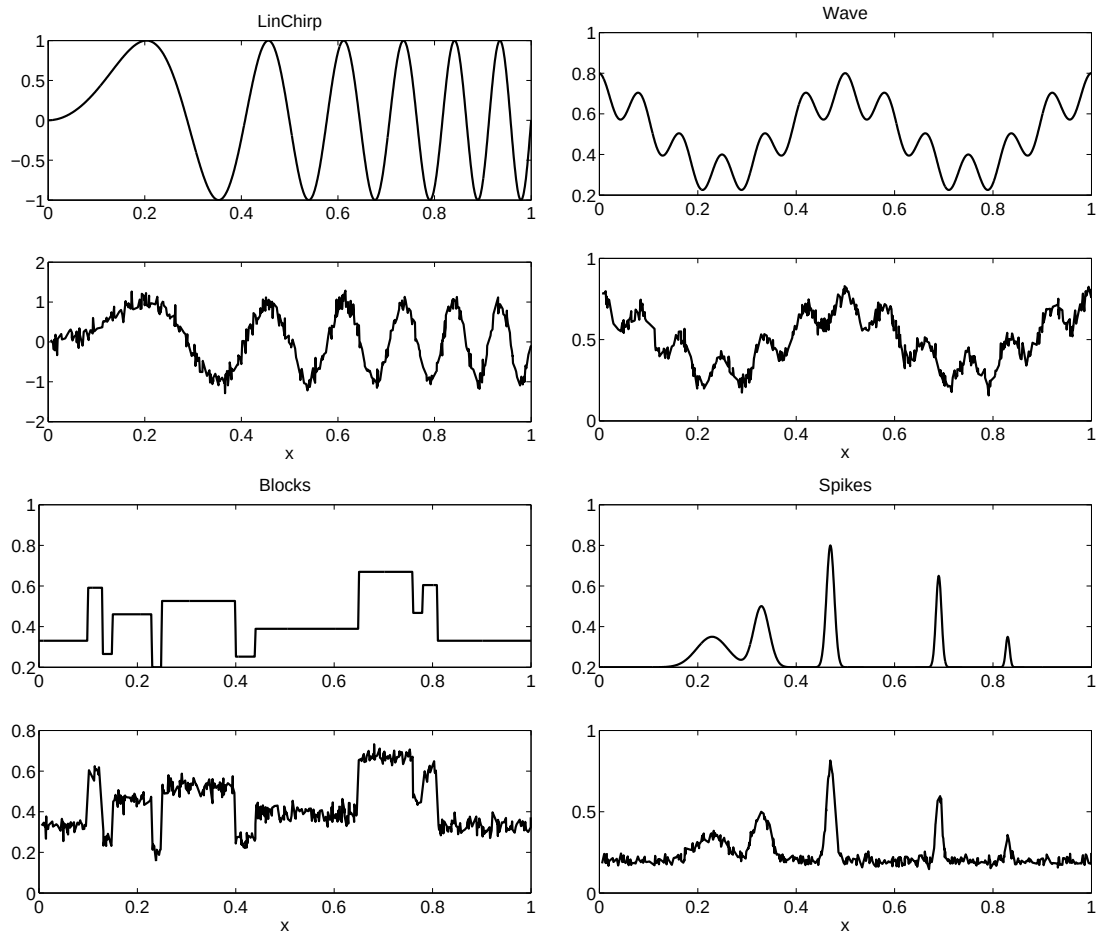



Figure 6: Examples of signals to approximate in the regression problem. (top-left) LinChirp. (top-right) Wave. (bottom-left) Blocks. (bottom-right) Spikes. For each figure, the top plot depicts the true signal while the bottom one presents an example of their randomly sampled noisy versions.

means of an approximate regularization path as described in Section 4.4, we learn different regression functions for 20 samples of $C$ logarithmically sampled on the interval $[0.01, 1000]$. This is performed for 5 random draws of the learning and validation sets. The $C$ value that gives the lowest average normalized mean-square error is considered as the optimal one. Finally, we use all the samples of the training signal and the optimal $C$ value to train an MKL SVM regression. The quality of the resulting regression function is then evaluated with respect to 1000 samples of the true signal. For all the simulations the $\varepsilon$ has been fixed to 0.1.

Table 4 summarizes the generalization performances achieved by the three different kernel settings. As expected, using a multiple kernel learning setting outperforms the single kernel setting especially when the target function presents multiscale structure. This is

Table 4: Normalized Mean Square error for the data described in Figure 6. The results are averaged over 20 runs. The first column give the performance of a SVM regression using a single kernel which is the average sum of all the kernels used for the two other results. Results corresponding to the columns *Kernel Dil* and *Kernel Dil-Trans* are related to MKL SVM regression with multiple kernels. # Kernel denotes the number of kernels selected by SimpleMKL.

| | Single Kernel | Kernel *Dil* | | Kernel *Dil-Trans* | |
|---|---|---|---|---|---|
| Dataset | Norm. MSE (%) | #Kernel | Norm. MSE | #Kernel | Norm. MSE |
| LinChirp | 1.46 ± 0.28 | 7.0 | 1.00 ± 0.15 | 21.5 | 0.92 ± 0.20 |
| Wave | 0.98 ± 0.06 | 5.5 | 0.73 ± 0.10 | 20.6 | 0.79 ± 0.07 |
| Blocks | 1.96 ± 0.14 | 6.0 | 2.11 ± 0.12 | 19.4 | 1.94 ± 0.13 |
| Spike | 6.85 ± 0.68 | 6.1 | 6.97 ± 0.84 | 12.8 | 5.58 ± 0.84 |

noticeable for the *LinChirp* and *Wave* dataset. Interestingly, for these two signals, performances of the multiple kernel settings also depend on the signal structure. Indeed, *Wave* presents a frequency located structure while *LinChirp* has a time and frequency located structure. Therefore, it is natural that the Dilation set of kernels performs better than the Dilation-Translation ones for *Wave*. Figure 7 depicts an example of multiscale regression function obtained when using the Dilation set of kernels. These plots show how the kernel weights adapt themselves to the function to estimate. For the same reason of adaptivity to the signal, the Dilation-Translation set of kernels achieves better performances for *Wave* and *Spikes*. We also notice that for the *Blocks* signal using multiple kernels only slightly improves performance compared to a single kernel.

### 5.3 Multiclass problem

For selecting the kernel and regularization parameter of a SVM, one usually tries all pairs of parameters and picks the couple that achieves the best cross-validation performance. Using an MKL approach, one can instead let the algorithm combine all available kernels (obtained by sampling the parameter) and just selects the regularization parameter by cross-validation. This last experiment aims at comparing on several multi-class datasets problem, these two model selection approaches (using MKL and CV) for choosing the kernel. Thus, we evaluate the two methods on some multiclass datasets taken from the UCI collection: *dna*, *waveform*, *image segmentation* and *abe* a subset problem of the Letter dataset corresponding to the classes A, B and E. Some information about the dataset are given in Table 5. For each dataset, we divide the whole data into a training set and a test set. This random splitting has been performed 20 times. For ease of comparison with previous works, we have used the splitting proposed by Duan and Keerthi (2005) and available at `http://www.keerthis.com/multiclass.html`. Then we have just computed the performance of SimpleMKL and report their results for the CV approach.

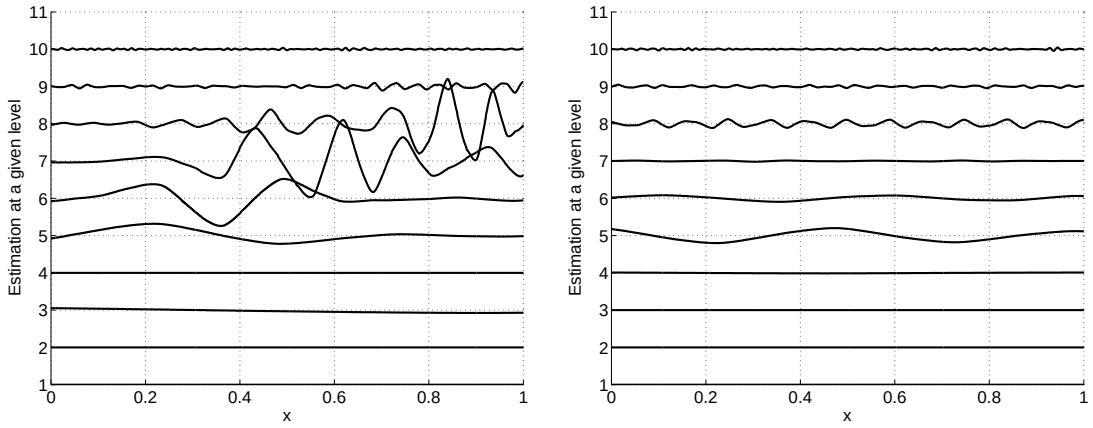



Figure 7: Examples of multiscale analysis of the *LinChirp* signal (left) and the *Wave* signal (right) when using Dilation based multiple kernels. The plots show how each function $f_m(\cdot)$ of the estimation focuses on a particular scale of the target function. The y-axis denotes the scale $j$ of the wavelet used for building the kernel. We can see that some low resolution space are not useful for the target estimation.

Table 5: Summary of the multiclass datasets and the training set size used.

| | | | Training Set Size | |
|---|---|---|---|---|
| Dataset | #Classes | # examples | Medium | Large |
| ABE | 3 | 2323 | 560 | 1120 |
| DNA | 3 | 3186 | 500 | 1000 |
| SEG | 7 | 2310 | 500 | 1000 |
| WAV | 3 | 5000 | 300 | 600 |

In our MKL *one-against-all* approach, we have used a polynomial kernel of degree 1 to 3 and Gaussian kernel for which $\sigma$ belongs to $[0.5, 1, 2, 5, 7, 10, 12, 15, 17, 20]$. For the regularization parameter $C$, we have 10 samples over the interval $[0.01, 10000]$. Note that Duan and Keerthi (2005) have used a more sophisticated sampling strategy based on a coarse sampling of $\sigma$ and $C$ and followed by fine-tuned sampling procedure. They also select the same couple of $C$ and $\sigma$ over all pairwise decision functions. Similarly to Duan and Keerthi (2005), the best hyperparameter $C$ has been tuned according to a five-fold cross-validation. According to this best $C$, we have learned an MKL all the full training set and evaluated the resulting decision function on the test set.

The comparison results are summarized on Table 6. We can see that the generalization performances of an MKL approach is either similar or better than the performance obtained when selecting the kernel through cross-validation, even though we have roughly searched the kernel and regularization parameter space. Hence, we can deduce that MKL

Table 6: Comparison of the generalization performances of an MKL approach and a cross-validation approach for selecting models in some multiclass problems. We have reported the average (over 20 runs) the test set errors of our algorithm while the errors obtained for the SV approach have been extracted from Duan and Keerthi (2005). Results also depend on the training set sizes.

| | Training set size | | | |
|---|---|---|---|---|
| | Medium | | Large | |
| Dataset | MKL | CV | MKL | CV |
| ABE | 0.73 ± 0.28 (16) | 0.96 ± 0.36 | 0.44 ± 0.67 (11) | 0.46 ± 0.20 |
| DNA | 7.69 ± 0.76 (11) | 7.84 ± 0.79 | 5.59 ± 0.55 (10) | 5.59 ± 0.39 |
| SEG | 6.52 ± 0.76 (10) | 6.51 ± 0.99 | 4.71 ± 0.67 (13) | 4.89 ± 0.71 |
| WAV | 15.18 ± 0.90 (15) | 15.43 ± 0.97 | 14.26 ± 0.68 (8) | 14.09 ± 0.55 |

can favorably replace cross-validation on kernel parameters. This result based on empirical observations is in accordance with some other works (Lanckriet et al., 2004b, Fung et al., 2004, Kim et al., 2006). However, we think that MKL and thus SimpleMKL in particular, can be better exploited and thus performs better than cross-validation when the kernels have been obtained from heterogenous source as described for instance in Lanckriet et al. (2004a), Zien and Ong (2007), Harchaoui and Bach (2007).

## 6. Conclusion

In this paper, we introduced SimpleMKL, a novel algorithm for solving the Multiple Kernel Learning problem. Our formalization of the MKL problem results in a smooth and convex optimization problem, which is actually equivalent to other MKL formulations available in the literature. The main added value of the smoothness of our new objective function is that descent methods become practical and efficient means to solve the optimization problem that wraps a single kernel SVM solver. We provide optimality conditions, analyze convergence and computational complexity issues for binary classification. The SimpleMKL algorithm and the resulting analyses can be easily be transposed to SVM regression, one-class SVM and multiclass SVM to name a few.

We provide experimental evidence that SimpleMKL is significantly more efficient than the state-of-the art SILP approach (Sonnenburg et al., 2006). This efficiency permits to demonstrate the usefulness of our algorithm on wavelet kernel based regression. We also illustrate in multiclass problems that MKL is a viable alternative to cross-validation for selecting a model.

Possible extensions of this work include other learning problems, such as semi-supervised learning or kernel eigenvalue problem like kernel Fisher discriminant analysis. We also plan to explore two different ways to speed up the algorithm. As a first direction, we will

investigate ways to obtain a better the descent direction, for example with second-order methods. Note however that computing the Hessian needs the derivative of the dual variable with respects to the weights $d$. This operation requires solving a linear system (Chapelle et al., 2002) and thus may produce some computational overhead. The second direction is motivated by the observation that most of the computational load is to the computation of the kernel combination. Hence, coordinate-wise optimizers may provide promising routes for improvements.

## Acknowlegments

We would like to thank the anonymous reviewers for their useful comments. This work was supported in part by the IST Program of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. Alain Rakotomamonjy, Francis Bach and Stéphane Canu were also supported by French grants from the Agence Nationale de la Recherche (KernSig for AR and SC, MGA for FB).

## References

A. Antoniadis and J. Fan. Regularization by Wavelet Approximations. *J. American Statistical Association*, 96:939–967, 2001.

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning, to appear*, 2008.

N. Aronszajn. Theory of reproducing kernels. *Trans. Am. Math. Soc.*, (68):337–404, 1950.

F. Bach. Consistency of the group Lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

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, 2004a.

F. Bach, R. Thibaux, and M. Jordan. Computing regularization paths for learning multiple kernels. In *Advances in Neural Information Processing Systems*, volume 17, pages 41–48, 2004b.

M. Berkelaar, K. Eikland, and P. Notebaert. *Lpsolve, Version 5.1.0.0*, 2004. URL `http://lpsolve.sourceforge.net/5.5/`.

D. Bertsekas. *Nonlinear programming*. Athena scientific, 1999.

C. Blake and C. Merz. UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

F. Bonnans. *Optimisation continue*. Dunod, 2006.

J.F. Bonnans and A. Shapiro. Optimization problems with pertubation : A guided tour. *SIAM Review*, 40(2):202–227, 1998.

J.F. Bonnans, J.C Gilbert, C. Lemaréchal, and C.A Sagastizbal. *Numerical Optimization Theoretical and Practical Aspects*. Springer, 2003.

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

S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy. SVM and kernel methods Matlab toolbox. LITIS EA4108, INSA de Rouen, Rouen, France, 2003. URL `http://asi.insa-rouen.fr/enseignants/~arakotom/toolbox/index.html`.

C-C. Chang and C-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukerjhee. Choosing multiple parameters for SVM. *Machine Learning*, 46(1-3):131–159, 2002.

K. Crammer and Y. Singer. On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

A. D'Amato, A. Antoniadis, and M. Pensky. Wavelet kernel penalized estimation for non-equispaced design regression. *Statistics and Computing*, 16:37–56, 2006.

A. D'Aspremont. Smooth Optimization with Approximate Gradient. *SIAM Journal on Optimization, To appear*, 2008.

D. DeCoste and K. Wagstaff. Alpha seeding for support vector machines. In *International Conference on Knowledge Discovery and Data Mining*, 2000.

K. Duan and S. Keerthi. Which Is the Best Multiclass SVM Method? An Empirical Study. In *Multiple Classifier Systems*, pages 278–285, 2005. URL `http://www.keerthis.com/multiclass.html`.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression (with discussion). *Annals of statistics*, 32(2):407–499, 2004.

G. Fung, M. Dundar, J. Bi, and B. Rao. a fast iterative algorithm for Fisher discriminant using heterogeneous kernels. In *Proceeedins of the 21th International Conference on Machine Learning*, 2004.

Y. Grandvalet. Least absolute shrinkage is equivalent to quadratic penalization. In L. Niklasson, M. Bodén, and T. Ziemske, editors, *ICANN'98*, volume 1 of *Perspectives in Neural Computing*, pages 201–206. Springer, 1998.

Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in svms. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2003.

Y. Grandvalet and S. Canu. *Outcomes of the equivalence of adaptive ridge with least absolute shrinkage*. MIT Press, 1999.

Z. Harchaoui and F. Bach. Image classification with segmentation graph kernels. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.

T. Joachims. Making large-scale SVM learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advanced in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press, 1999.

S.-J. Kim, A. Magnani, and S. Boyd. Optimal kernel selection in kernel Fisher discriminant analysis. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006.

K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale $\ell_1$-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

G. Lanckriet, T. De Bie, N. Cristianini, M. Jordan, and W. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20:2626–2635, 2004a.

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, 2004b.

C. Lemaréchal and C. Sagastizabal. Practical aspects of moreau-yosida regularization : theoretical preliminaries. *SIAM Journal of Optimization*, 7:867–895, 1997.

G. Loosli and S. Canu. Comments on the "Core Vector Machines: Fast SVM Training on Very Large Data Sets". *Journal of Machine Learning Research*, 8:291–301, February 2007.

G. Loosli, S. Canu, S. Vishwanathan, A. Smola, and M. Chattopadhyay. Boîte à outils SVM simple et rapide. *Revue d'Intelligence Artificielle*, 19(4-5):741–767, 2005.

D. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 1984.

C. Micchelli and M. Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

A. Rakotomamonjy and S. Canu. Frames, reproducing kernels, regularization and learning. *Journal of Machine Learning Research*, 6:1485–1515, 2005.

A. Rakotomamonjy, X. Mary, and S. Canu. Non parametric regression with wavelet kernels. *Applied Stochastics Model for Business and Industry*, 21(2):153–163, 2005.

A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In Zoubin Ghahramani, editor, *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007)*, pages 775–782. Omnipress, 2007.

R. Rifkin and A. Klautau. In Defense of One-Vs-All Classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

S. Rosset. Tracking Curved Regularized Optimization Solution Paths. In *Advances in Neural Information Processing Systems*, 2004.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2001.

S. Sonnenburg, G. Rätsch, and C. Schäfer. A general and efficient algorithm for multiple kernel learning. In *Advances in Neural Information Processing Systems*, volume 17, pages 1–8, 2005.

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

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

V. Vapnik, S. Golowich, and A. Smola. *Support Vector Method for function estimation, Regression estimation and Signal processing*, volume Vol. 9. MIT Press, Cambridge, MA, neural information processing systems, edition, 1997.

S. V. N. Vishwanathan, A. J. Smola, and M. Murty. SimpleSVM. In *International Conference on Machine Learning*, 2003.

G. Wahba. *Spline Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, 1990.

J. Weston and C. Watkins. Multiclass support vector machines. In *Proceedings of ESANN99, Brussels*. D. Facto Press, 1999.

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.

## Appendix

### Proof of convexity of the weighted squared norm MKL formulation

The convexity of the MKL problem (2) introduced in section 2.2 will be established if we prove the convexity of

$$J(f,t) = \frac{1}{t}\langle f, f\rangle_{\mathcal{H}} \quad \text{where } f \in \mathcal{H} \text{ and } t \in \mathbb{R}^{*+}$$

Since $J(f,t)$ is differentiable with respects to its arguments, we only have to make sure that the first order conditions for convexity are verified. As the convexity of the domain of $J$ is trivial, we verify that, for any $(f,t)$ and $(g,s) \in \mathcal{H} \times \mathbb{R}^{*+}$, the following holds:

$$J(g,s) \geq J(f,t) + \langle \nabla_f J, g - f\rangle_{\mathcal{H}} + (s-t)\nabla_t J \ .$$

As $\nabla_f J = \frac{2}{t} f$ and $\nabla_t J = -\frac{1}{t^2}\langle f, f\rangle_{\mathcal{H}}$, this inequality can be written as

$$\begin{aligned} & \frac{1}{s}\langle g, g\rangle_{\mathcal{H}} \geq \frac{2}{t}\langle f, g\rangle_{\mathcal{H}} - \frac{s}{t^2}\langle f, f\rangle_{\mathcal{H}} \ , \\ \Leftrightarrow \quad & \langle t\, g - s\, f, t\, g - s\, f\rangle_{\mathcal{H}} \geq 0 \ , \end{aligned}$$

where we used that $s$ and $t$ are positive. The above inequality holds since the scalar product on the left-hand-side is a norm. Hence problem (2) minimizes the sum of convex functions on a convex set; it is thus convex. Note that when $\mathcal{H}$ is a finite dimension space, the function $J(f, t)$ is known as the perspective of $f$, whose convexity is proven in textbooks (Boyd and Vandenberghe, 2004).

**Differentiability of optimal value function**

The algorithm we propose for solving the MKL problem heavily relies on the differentiability of the optimal value of the primal SVM objective function. For the sake of self-containedness, we reproduce here a theorem due to Bonnans and Shapiro (1998) that allows us to compute the derivatives of $J(d)$ defined in (10).

**Theorem 1** *(Bonnans and Shapiro, 1998) Let $X$ be a metric space and $U$ be a normed space. Suppose that for all $x \in X$ the function $f(x, \cdot)$ is differentiable, that $f(x, u)$ and $D_u f(x, u)$ the derivative of $f(x, \cdot)$ are continuous on $X \times U$ and let $\Phi$ be a compact subset of $X$. Let define the optimal value function as $v(u) = \inf_{x\in\Phi} f(x, u)$. The optimal value function is directionally differentiable. Furthermore, if for $u^0 \in U$, $f(\cdot, u^0)$ has a unique minimizer $x^0$ over $\Phi$ then $v(u)$ is differentiable at $u^0$ and $Dv(u^0) = D_u f(x^0, u^0)$.*