Journal of Machine Learning Research 13 (2012) 227-253 Submitted 3/11; Revised 9/11; Published 2/12

Multi Kernel Learning with Online-Batch Optimization[*]

Francesco Orabona[†] FRANCESCO@ORABONA.COM
Toyota Technological Institute at Chicago
6045 South Kenwood Avenue
60637 Chicago, IL, USA

Luo Jie[†] LUOJIE@YAHOO-INC.COM
Yahoo! Labs
701 First Avenue
94089 Sunnyvale, CA, USA

Barbara Caputo BCAPUTO@IDIAP.CH
Idiap Research Institute
Centre du Parc, Rue Marconi 19
1920 Martigny, Switzerland

Editor: Francis Bach

Abstract

In recent years there has been a lot of interest in designing principled classification algorithms over multiple cues, based on the intuitive notion that using more features should lead to better performance. In the domain of kernel methods, a principled way to use multiple features is the Multi Kernel Learning (MKL) approach.

Here we present a MKL optimization algorithm based on stochastic gradient descent that has a guaranteed convergence rate. We directly solve the MKL problem in the primal formulation. By having a p-norm formulation of MKL, we introduce a parameter that controls the level of sparsity of the solution, while leading to an easier optimization problem. We prove theoretically and experimentally that 1) our algorithm has a faster convergence rate as the number of kernels grows; 2) the training complexity is linear in the number of training examples; 3) very few iterations are sufficient to reach good solutions. Experiments on standard benchmark databases support our claims.

Keywords: multiple kernel learning, learning kernels, online optimization, stochastic subgradient descent, convergence bounds, large scale

1. Introduction

In recent years there has been a lot of interest in designing principled classification algorithms over multiple cues, based on the intuitive notion that using more features should lead to better performance. Moreover, besides the purpose of decreasing the generalization error, practitioners are often interested in more flexible algorithms which can perform feature selection while training. This is for instance the case when a lot of features are available but among them noisy ones are hidden. Selecting the features also improves the interpretability of the decision function.

This has been translated into various algorithms, that dates back to the ’90s (Wolpert, 1992), based on a two-layers structure. There a classifier is trained for each cue and then their outputs

∗. A preliminary version of this paper appeared in Orabona et al. (2010).
†. Work done mainly while at Idiap Research Institute.

are combined by another classifier. This approach has been re-invented many times, with different flavors (Nilsback and Caputo, 2004; Sanderson and Paliwal, 2004; Jie et al., 2010a; Gehler and Nowozin, 2009b; Jin et al., 2010). In general, the two layers approaches use Cross-Validation (CV) methods to create the training set for the second layer (Wolpert, 1992; Gehler and Nowozin, 2009b). If the second layer is a linear classifier, these methods are equivalent to a linear combination of the single classifiers.

Focusing on the domain of the Support Vector Machines (SVM) (Cristianini and Shawe-Taylor, 2000), the use of multiple cues corresponds to the use of multiple kernels. Hence, instead of combining kernel classifiers, the focus of research has moved on how to build an optimal new kernel as a weighted combination of kernels.

A recent approach in this field is to use a two-stage procedure, in which the first stage finds the optimal weights to combine the kernels, using an improved definition of the kernel alignment (Cristianini et al., 2002) as a proxy of the generalization error, and a standard SVM as second stage (Cortes et al., 2010). This approach builds on the previous works on the maximization of the kernel alignment to combine kernels (Lanckriet et al., 2004). However in this approach, even if theoretically principled, the global optimality is not guaranteed, because the optimization process split in two phases.

A different approach with a joint optimization process is Multi Kernel Learning (MKL) (Lanckriet et al., 2004; Bach et al., 2004; Sonnenburg et al., 2006; Zien and Ong, 2007; Rakotomamonjy et al., 2008; Varma and Babu, 2009; Kloft et al., 2009). In MKL one solves a joint optimization problem while also learning the optimal weights for combining the kernels. MKL methods are theoretically founded, because they are based on the minimization of an upper bound of the generalization error (Kakade et al., 2009; Cortes et al., 2010), like in standard SVM. In most of these approaches the objective function is designed to impose sparsity on the weights of the combination using an l_1-norm constraint (Bach et al., 2004; Sonnenburg et al., 2006; Zien and Ong, 2007; Rakotomamonjy et al., 2008; Varma and Babu, 2009). However solving it is far more complex than training a single SVM classifier. In fact, the l_1 norm is not smooth, so it slows down the optimization process. The original MKL problem by Lanckriet et al. (2004) was cast as a semidefinite programming (SDP). SDP are known to have poor scalability, hence much of the subsequent research focused on devising more efficient optimization procedures. The first step towards practical MKL algorithms was to restrict the weights coefficients to be non-negative. In this way, it was possible to recast the problem as a much more efficient semi-infinite linear programming (SILP) (Sonnenburg et al., 2005; Rubinstein, 2005). This has allowed to solve the MKL problem with alternating optimization approaches (Sonnenburg et al., 2006; Rakotomamonjy et al., 2008; Xu et al., 2008; Nath et al., 2009), first optimizing over the kernel combination weights, with the current SVM solution fixed, then finding the SVM solution, given the current weights. One advantage of the alternating optimization approach is that it is possible to use existing efficient SVM solvers, such as Joachims (1999) and Chang and Lin (2001), for the SVM optimization step. On the other hand, for these algorithms, it is usually not possible to prove a bound on the maximum number of iterations needed, even if they are known to converge. In fact, to the best of our knowledge, none of the existing MKL algorithms provides theoretical guarantees on the convergence rate. For the same reason it is not possible to know the asymptotic computational complexity of these algorithms, and often these dependencies are estimated numerically for the specific implementation at hand. For example, multiclass MKL SILP algorithm (Sonnenburg et al., 2006; Zien and Ong, 2007) seems to depend polynomially on the number of training examples and number of classes with an exponent of ~ 2.4 and ~ 1.7 respectively. For the other algorithms these dependencies are not clear. Another disadvantage is that they need to solve the inner SVM problem till optimality. In fact, to guarantee convergence, the solution needs to be of a high enough precision so that the kernel weight gradient computation is accurate. On the other hand the learning process is usually stopped early, before reaching the optimal solution, based on the common assumption that it is enough to have an approximate solution of the optimization function. Considering the fact that the current MKL algorithms are solved based on their dual representation, this might mean being stopped far from the optimal solution (Hush et al., 2006), with unknown effects on the convergence.

An important point is that, very often, these approaches fail to improve much over the naive baseline of just summing all the kernels (Kloft et al., 2009). Recently, researchers start to realize that when the optimal Bayes classifier is not sparse, brutally imposing sparsity will hurt the generalization performance. Motivated by this, the l_p-norm constraint has been proposed (Kloft et al., 2009; Orabona et al., 2010; Vishwanathan et al., 2010), instead of l_1-norm constrain, to be able to tune the level of sparsity of the solution and to obtain an easier problem too. In particular Vishwanathan et al. (2010) derived the dual of a variation of the l_p MKL problem for $p > 1$, suited to be optimized with the popular Sequential Minimal Optimization algorithm (Platt, 1999). However even for their algorithm it is not clear how the convergence rate depends on p and how to generalize the algorithm to generic loss functions, such as the structured losses (Tsochantaridis et al., 2004). This limitation on the use of particular loss functions is common to all the recent MKL optimization algorithms. An alternative way to be able to tune the sparsity of the MKL solution, inspired by the elastic-net regularization, has been proposed by Tomioka and Suzuki (2010).

The main contribution of this paper is a new optimization algorithm to solve efficiently the l_p-MKL problem, with a guaranteed convergence rate to the optimal solution. We minimize it with a two-stage algorithm. The first one is an online initialization procedure that determines quickly the region of the space where the optimal solution lives. The second stage refines the solution found by the first stage, using a stochastic gradient descent algorithm. Bounds on the convergence rate are proved for the overall process. Notably different from the other methods, our algorithm solves the optimization problem directly in the primal formulation, in both stages. This allows us to use *any* convex loss function, as the multiclass loss proposed by Crammer and Singer (2002) or general structured losses (Tsochantaridis et al., 2004), without any change to the core of the algorithm. Using recent approaches in optimization theory, the algorithm takes advantage of the abundance of information to reduce the training time (Shalev-Shwartz and Srebro, 2008). In fact, we show that the presence of a large number of kernels helps the optimization process instead of hindering it, obtaining, theoretically and practically, a faster convergence rate with more kernels. Our algorithm has a training time that depends linearly on the number of training examples, with a convergence rate sub-linear in the number of features/kernels used, when a sparse solution is favored. At the same time, it achieves state-of-the-art performance on standard benchmark databases. We call this algorithm OBSCURE, Online-Batch Strongly Convex mUlti keRnel lEarning.

The rest of the paper presents the theory and the experimental results supporting our claims. Section 2 revises the basic definitions of Multi Kernel Learning and generalizes it to the l_p-norm formulation. Section 3 presents the OBSCURE algorithm and Section 4 shows our theoretical guarantees, while Section 5 reports experiments on categorization tasks. We conclude the paper with discussions and future works.

2. p-norm Multi Kernel Learning

In this section we first introduce formally the MKL framework and its notation, then its p-norm generalization.

2.1 Definitions

In the following we define some notations and we also introduce some concepts of convex analysis. For a more thorough introduction see, for example, Boyd and Vandenberghe (2004).

2.1.1 NOTATION

We indicate matrices and vectors with bold letters. We also introduce two notations that will help us to synthesize the following formulas. We indicate by $[w^j]_1^F := \left[w^1, w^2, \cdots, w^F\right]$, and with a bar, for example, $\bar{w}$, the vector formed by the concatenation of the F vectors w^j, hence $\bar{w} = [w^1, w^2, \cdots, w^F] = [w^j]_1^F$.

We consider closed convex functions $f : S \to \mathbb{R}$, where in the following S will always denote a proper subset of $\mathbb{X}$, an Euclidean vector space.[1] We will indicate the inner product between two vectors of $\mathbb{X}$, w and w', as $w \cdot w'$. Given a convex function $f : S \to \mathbb{R}$, its Fenchel conjugate $f^* : \mathbb{X} \to \mathbb{R}$ is defined as $f^*(u) = \sup_{v \in S}(v \cdot u - f(v))$. A generic norm of a vector $w \in \mathbb{X}$ is indicated by $\|w\|$, and its dual $\|\cdot\|_*$ is the norm defined as $\|y\|_* = \sup\{x \cdot y : \|x\| \leq 1\}$. A vector x is a subgradient of a function f at v, if $\forall u \in S$, $f(u) - f(v) \geq (u - v) \cdot x$. The differential set of f at v, indicated with $\partial f(v)$, is the set of all the subgradients of f at v. If f is convex and differentiable at v then $\partial f(v)$ consists of a single vector which is the gradient of f at v and is denoted by $\nabla f(v)$. A function $f : S \to \mathbb{R}$ is said to be λ-strongly convex with respect to a convex and differentiable function h iff for any $u, v \in S$ and any subgradient $\partial f(u)$, $f(v) \geq f(u) + \partial f(u) \cdot (v - u) + \lambda(h(v) - h(u) - (v - u) \cdot \nabla h(v))$, where the terms in parenthesis form the Bregman divergence between v and u of h.

2.1.2 BINARY AND MULTI-CLASS CLASSIFIERS

Let $\{x_i, y_i\}_{i=1}^N$, with $N \in \mathbb{N}$, $x_i \in \mathbb{X}$ and $y_i \in \mathbb{Y}$, be the training set. Consider a function $\phi(x) : \mathbb{X} \to \mathbb{H}$ that maps the samples into a high, possibly infinite, dimensional space. In the binary case $\mathbb{Y} = \{-1, 1\}$, and we use the standard setting to learn with kernels,[2] in which the prediction on a sample x is a function of the scalar product between an hyperplane w and the transformed sample $\phi(x)$. With multiple kernels, we will have F corresponding functions $\phi^j(\cdot)$, $i = 1, \cdots, F$, and F corresponding kernels $K^j(x, x')$ defined as $\phi^j(x) \cdot \phi^j(x')$.

For multiclass and structured classification $\mathbb{Y} = \{1, \ldots, M\}$, and we follow the common approach to use joint feature maps $\phi(x, y) : \mathbb{X} \times \mathbb{Y} \to \mathbb{H}$ (Tsochantaridis et al., 2004). Again, we will have F functions $\phi^j(\cdot, \cdot), i = 1, \cdots, F$, and F kernels $K^j((x, y), (x', y'))$ as $\phi^j(x, y) \cdot \phi^j(x', y')$. This definition includes the case of training M different hyperplanes, one for each class. Indeed $\phi^j(x, y)$ can be defined as

$$\phi^j(x, y) = [0, \cdots, 0, \underbrace{\phi'^j(x)}_{y}, 0, \cdots, 0],$$

1. We allow the functions to assume infinite values, as a way to restrict their domains to proper subsets of $\mathbb{X}$. However in the following the convex functions of interest will always be evaluated on vectors that belong to their domains.
2. For simplicity we will not use the bias here, but it can be easily added modifying the kernel definition.

where $\phi'^j(\cdot)$ is a transformation that depends only on the data. Similarly w will be composed by M blocks, $[w^1, \cdots, w^M]$. Hence, by construction, $w \cdot \phi^j(x,r) = w^r \cdot \phi'^j(x)$. According to the defined notation, $\phi(x,y) = [\phi^1(x,y), \cdots, \phi^F(x,y)]$. These definitions allow us to have a general notation for the binary and multiclass setting.

2.1.3 LOSS FUNCTION

In the following we consider convex Lipschitz loss functions. The most commonly used loss in binary classification is the hinge loss (Cristianini and Shawe-Taylor, 2000).

$$\ell^{HL}(w,x,y) = |1 - y\bar{w} \cdot \bar{\phi}(x)|_+,$$

where $|t|_+$ is defined as $\max(t,0)$. We also consider the following multi-class loss function (Crammer and Singer, 2002; Tsochantaridis et al., 2004), that will be used to specialize our results.

$$\ell^{MC}(w,x,y) = \max_{y' \neq y} |1 - \bar{w} \cdot (\bar{\phi}(x,y) - \bar{\phi}(x,y'))|_+ \ . \tag{1}$$

This loss function is convex and it upper bounds the multi-class misclassification loss.

2.1.4 NORMS AND DUAL NORMS

For $w \in \mathbb{R}^d$ and $p \geq 1$, we denote by $\|w\|_p$ the p-norm of w, that is, $\|w\|_p = (\sum_{i=1}^d |w_i|^p)^{1/p}$.

The dual norm of $\|\cdot\|_p$ is $\|\cdot\|_q$, where p and q satisfy $1/p + 1/q = 1$. In the following p and q will always satisfy this relation.

2.1.5 GROUP NORM

It is possible to define a $(2,p)$ *group norm* $\|\bar{w}\|_{2,p}^2$ on $\bar{w}$ as

$$\|\bar{w}\|_{2,p} := \left\| \left[\|w^1\|_2, \|w^2\|_2, \cdots, \|w^F\|_2 \right] \right\|_p,$$

that is the p-norm of the vector of F elements, formed by 2-norms of the vectors w^j. The dual norm of $\|\cdot\|_{2,p}$ is $\|\cdot\|_{2,q}$ (Kakade et al., 2009). These kind of norms have been used as *block regularization* in the LASSO literature (Yuan and Lin, 2006).

2.2 Multi Kernel Learning

The MKL optimization problem was first proposed by Bach et al. (2004) and extended to multiclass by Zien and Ong (2007). It can be written as

$$\begin{aligned} &\min_{w_j} \frac{\lambda}{2} \left(\sum_{j=1}^F \|w^j\|_2 \right)^2 + \frac{1}{N} \sum_{i=1}^N \xi_i \\ &\text{s.t.} \ \ \bar{w} \cdot (\bar{\phi}(x_i,y_i) - \bar{\phi}(x_i,y)) \geq 1 - \xi_i, \forall i, y \neq y_i \ . \end{aligned} \tag{2}$$

An equivalent formulation can be derived from the first one through a variational argument (Bach et al., 2004)

$$\begin{aligned}
\min_{w_j,\alpha_j \geq 0} \; & \frac{\lambda}{2}\left(\sum_{j=1}^{F}\frac{\|w^j\|_2}{\alpha_j}\right)^2 + \frac{1}{N}\sum_{i=1}^{N}\xi_i \\
\text{s.t.} \; & \bar{w}\cdot(\bar{\phi}(x_i,y_i)-\bar{\phi}(x_i,y)) \geq 1-\xi_i, \; \forall i, y \neq y_i \\
& \|\alpha\|_1^2 \leq 1 \, . \qquad (3)
\end{aligned}$$

This formulation has been used by Bach et al. (2004) and Sonnenburg et al. (2006), while the formulation proposed by Rakotomamonjy et al. (2008) is slightly different, although it can be proved to be equivalent. The reason to introduce this variational formulation is to use an alternating optimization strategy to efficiently solve the constrained minimization problem. However in the following we will show that it is possible to efficiently minimize directly the formulation in (2), or at least one variation of it.

We first rewrite (2) with group norms. Using the notation defined above, we have

$$\min_{\bar{w}} \; \frac{\lambda}{2}\|\bar{w}\|_{2,1}^2 + \frac{1}{N}\sum_{i=1}^{N}\ell^{MC}(\bar{w},x_i,y_i)\,, \qquad (4)$$

where $\bar{w} = [w_1, w_2, \cdots, w_F]$. The $(2,1)$ group norm is used to induce sparsity in the domain of the kernels. This means that the solution of the optimization problem will select a subset of the F kernels. However, even if sparsity can be desirable for specific applications, it could lead to a decrease in performance. Moreover the problem in (4) is not strongly convex (Kakade et al., 2009), so its optimization algorithm is rather complex and its rate of convergence is usually slow (Bach et al., 2004; Sonnenburg et al., 2006).

We generalize the optimization problem (4), using a generic group norm and a generic convex loss function

$$\min_{\bar{w}} \; \frac{\lambda}{2}\|\bar{w}\|_{2,p}^2 + \frac{1}{N}\sum_{i=1}^{N}\ell(\bar{w},x_i,y_i)\,, \qquad (5)$$

where $1 < p \leq 2$. We also define $f(\bar{w}) := \frac{\lambda}{2}\|\bar{w}\|_{2,p}^2 + \frac{1}{N}\sum_{i=1}^{N}\ell(\bar{w},x_i,y_i)$ and $\bar{w}^*$ equals to the optimal solution of (5), that is $\bar{w}^* = \arg\min_{\bar{w}} f(\bar{w})$. The additional parameter p allow us to decide the level of sparsity of the solution. Moreover this formulation has the advantage of being λ/q-strongly convex (Kakade et al., 2009), where λ is the regularization parameter in (5). Strong convexity is a key property to design fast batch and online algorithms: the more a problem is strongly convex the easier it is to optimize it (Shalev-Shwartz and Singer, 2007; Kakade et al., 2009). Many optimization problems are strongly convex, as the SVM objective function. When p tends to 1, the solution gets close to the sparse solution obtained by solving (2), but the strong convexity vanishes. Setting p equal to 2 corresponds to using the unweighted sum of the kernels. In the following we will show how to take advantage of the strong convexity to design a fast algorithm to solve (5), and how to have a good convergence rate even when the strong convexity is close to zero. Note that this formulation is similar to the one proposed by Kloft et al. (2009). Indeed, as for (2) and (3), using Lemma 26 in Micchelli and Pontil (2005) it is possible to prove that they are equivalent through a variational argument.

We have chosen to weight the regularization term by λ and divide the loss term by N, instead of the more common formulation with only the loss term weighted by a parameter C. This choice simplifies the math of our algorithm. However the two formulations are equivalent when setting $\lambda = \frac{1}{CN}$, hence a big value of C corresponds to a small value of λ.

3. The OBSCURE Algorithm

Our basic optimization tool is the framework developed in Shalev-Shwartz and Singer (2007); Shalev-Shwartz et al. (2007). It is a general framework to design and analyze stochastic sub-gradient descent algorithms for any strongly convex function. At each step the algorithm takes a random sample of the training set and calculates a sub-gradient of the objective function evaluated on the sample. Then it performs a sub-gradient descent step with decreasing learning rate, followed by a projection inside the space where the solution lives. The algorithm Pegasos, based on this framework, is among the state-of-art solvers for linear SVM (Shalev-Shwartz et al., 2007; Shalev-Shwartz and Srebro, 2008).

Given that the $(2, p)$ group norm is strongly convex, we could use this framework to design an efficient MKL algorithm. It would inherit all the properties of Pegasos (Shalev-Shwartz et al., 2007; Shalev-Shwartz and Srebro, 2008). In particular a Pegasos-like algorithm used to minimize (5) would have a convergence rate, and hence a training time, proportional to $\frac{q}{\lambda}$. Although in general this convergence rate can be quite good, it becomes slow when λ is small and/or p is close to 1. Moreover it is common knowledge that in many real-world problems (e.g., visual learning tasks) the best setting for λ is very small, or equivalently C is very big (the order of $10^2 - 10^3$). Notice that this is a general problem. The same problem also exists in the other SVM optimization algorithms such as SMO and similar approaches, as their training time also depends on the value of the parameter C (Hush et al., 2006).

Do et al. (2009) proposed a variation of the Pegasos algorithm called proximal projected sub-gradient descent. This formulation has a better convergence rate for small values of λ, while retaining the fast convergence rate for big values of λ. A drawback is that the algorithm needs to know in advance an upper bound on the norm of the optimal solution. Do et al. (2009) solve this problem with an algorithm that estimates this bound while training, but it gives a speed-up only when the norm of the optimal solution $\bar{w}^*$ is small. This is not the case in most of the MKL problems for categorization tasks.

Our OBSCURE algorithm takes the best of the two solutions. We first extend the framework of Do et al. (2009) to the generic non-Euclidean norms, to use it with the $(2, p)$ group norm. Then we solve the problem of the upper bound of the norm of the optimal solution using a new online algorithm. This is designed to take advantage of the characteristics of the MKL task and to quickly converge to a solution close to the optimal one. Hence OBSCURE is composed of two steps: the first step is a fast online algorithm (Algorithm 1), used to quickly estimate the region of the space where the optimal solution lives. The second step (Algorithm 2) starts from the approximate solution found by the first stage, and exploiting the information on the estimated region, it uses a stochastic proximal projected sub-gradient descent algorithm. We also found that, even if we cannot guarantee this theoretically, empirically in many cases the solution found by the first stage is extremely close to the optimal one. We will show this in the experiments of Section 5.6.

3.1 Efficient Implementation

The training time of OBSCURE is proportional to the number of steps required to converge to the optimal solution, that will be bounded in Theorem 3, multiplied by the complexity of each step. This

Algorithm 1 OBSCURE stage 1 (online)

```
1: Input: $q, \bar{\eta}$
2: Initialize: $\bar{\theta}_1 = 0, \bar{w}_1 = 0$
3: for $t = 1, 2, \ldots, T$ do
4:     Sample $(x_t, y_t)$ at random
5:     $\bar{z}_t = \partial \ell(\bar{w}_t, x_t, y_t)$
6:     $\bar{\theta}_{t+1} = \bar{\theta}_t - \eta \bar{z}_t$
7:     $w^j_{t+1} = \frac{1}{q} \left( \frac{\|\theta^j_{t+1}\|_2}{\|\bar{\theta}_{t+1}\|_{2,q}} \right)^{q-2} \theta^j_{t+1}, \ \forall j = 1, \cdots, F$
8: end for
9: return $\bar{\theta}_{T+1}, \bar{w}_{T+1}$
10: return $R = \sqrt{\|\bar{w}_{T+1}\|^2_{2,p} + \frac{2}{\lambda N} \sum_{i=1}^N \ell(\bar{w}_{T+1}, x_i, y_i)}$
```

Algorithm 2 OBSCURE stage 2 (stochastic optimization)

```
1: Input: $q$, $\bar{\theta}_1$, $\bar{w}_1$, $R$, $\lambda$
2: Initialize: $s_0 = 0$
3: for $t = 1, 2, \ldots, T$ do
4:     Sample $(x_t, y_t)$ at random
5:     $\bar{z}_t = \partial \ell(\bar{w}_t, x_t, y_t)$
6:     $d_t = \lambda t + s_{t-1}$
7:     $s_t = s_{t-1} + 0.5 \left( \sqrt{d_t^2 + q \frac{(\frac{\lambda}{q}\|\bar{\theta}_t\|_{2,q} + \|\bar{z}_t\|_{2,q})^2}{R^2}} - d_t \right)$
8:     $\eta_t = \frac{q}{\lambda t + s_t}$
9:     $\bar{\theta}_{t+\frac{1}{2}} = (1 - \frac{\lambda \eta_t}{q}) \bar{\theta}_t - \eta_t \bar{z}_t$
10:    $\bar{\theta}_{t+1} = \min \left( 1, \frac{qR}{\|\bar{\theta}_{t+\frac{1}{2}}\|_{2,q}} \right) \bar{\theta}_{t+\frac{1}{2}}$
11:    $w^j_{t+1} = \frac{1}{q} \left( \frac{\|\theta^j_{t+1}\|_2}{\|\bar{\theta}_{t+1}\|_{2,q}} \right)^{q-2} \theta^j_{t+1}, \ \forall j = 1, \cdots, F$
12: end for
```

in turn is dominated by the calculation of the gradient of the loss (line 5 in Algorithms 1 and 2). This complexity, for example, for the multiclass hinge loss ℓ^{MC} is $O(NFM)$, given that the number of support vectors is proportional to N. Note that this complexity is common to any other similar algorithm, and it can be reduced using methods like kernel caching (Chang and Lin, 2001).

Following (Shalev-Shwartz et al., 2007, Section 4), it is possible to use Mercer kernels without introducing explicitly the dual formulation of the optimization problem. In both algorithms, $\bar{\theta}_{t+1}$ can be written as a weighted linear summation of $\bar{\phi}(x_t, \cdot)$. For example, when using the multi-class loss function ℓ^{MC}, we have that $\bar{\theta}_{t+1} = -\sum_t \eta_t \bar{z}_t = \sum_t \eta_t (\bar{\phi}(x_t, y_t) - \bar{\phi}(x_t, \hat{y}_t))$. Therefore, the algorithm can easily store η_t, y_t, $\hat{y}_t$, and x_t instead of storing $\bar{\theta}_t$. Observing line 7 in Algorithm 1 and line 11 in the Algorithm 2, we have that at each round, w^j_{t+1} is proportional to θ^j_{t+1}, that is $w^j_{t+1} = \alpha^j_t \theta^j_{t+1}$. Hence $\bar{w}_{t+1}$ can also be represented using $\alpha^j_t \eta_t$, y_t, $\hat{y}_t$ and x_t. In prediction the dot product between $\bar{w}_t$ and $\bar{\phi}(x_t, \cdot)$ can be expressed as a sum of terms $\bar{w}^j_t \cdot \phi^j(x_t, \cdot)$, that can be calculated using the definition of the kernel.

Algorithm 3 Proximal projected sub-gradient descent

```
1: Input: $R$, $\sigma$, $w_1 \in S$
2: Initialize: $s_0 = 0$
3: for $t = 1, 2, \dots, T$ do
4:     Receive $g_t$
5:     $z_t = \partial g_t(w_t)$
6:     $s_t = s_{t-1} + \frac{\sqrt{(\alpha\sigma t + s_{t-1})^2 + \frac{\alpha L_t}{R^2}} - \alpha\sigma t - s_{t-1}}{2}$
7:     $\eta_t = \frac{1}{\sigma t + \frac{s_t}{\alpha}}$
8:     $w_{t+1} = \nabla h^*(\nabla h(w_t) - \eta_t z_t)$
9: end for
```

Another important speed-up can be obtained considering the nature of the updates of the second stage. If the optimal solution has a loss equal to zero or close to it, when the algorithm is close to convergence most of the updates will consist just of a scaling. Hence it is possible to cumulate the scalings in a variable, to perform the scaling of the coefficients just before an additive update must be performed, and to take it into account for each prediction. Moreover, when using the multiclass loss (1), each update touches only two classes at a time, so to minimize the number of scalings we can keep a vector of scaling coefficients, one for each class, instead of a single number. For more details on the implementation, we refer the reader to the MATLAB implementation of OBSCURE in DOGMA (Orabona, 2009).

4. Analysis

We now show the theorems that give a theoretical guarantee on the convergence rate of OBSCURE to the optimal solution of (5). The following lemma contains useful properties to prove the performance guarantees of Algorithm 1 and 2.

Lemma 1 *Let $B \in \mathbb{R}^+$, define $S = \{\bar{w} : \|\bar{w}\|_{2,p} \le B\}$. Let $h(\bar{w}) : S \to \mathbb{R}$ defined as $\frac{q}{2}\|\bar{w}\|_{2,p}^2$, define also* $\mathrm{Proj}(\bar{w}, B) = \min\left(1, \frac{B}{\|\bar{w}\|_{2,p}}\right)\bar{w}$*, then*

1. $\nabla h(\bar{w}) = q\left[\left(\frac{\|w^j\|_2}{\|\bar{w}\|_{2,p}}\right)^{p-2} w^j\right]_1^F, \ \forall \bar{w} \in S$

2. $\nabla h^*(\bar{\theta}) = \mathrm{Proj}\left(\frac{1}{q}\left[\left(\frac{\|\theta^j\|_2}{\|\theta\|_{2,q}}\right)^{q-2} \theta^j\right]_1^F, B\right)$

3. $\|\bar{w}\|_{2,p} = \frac{1}{q}\|\nabla h(\bar{w})\|_{2,q}, \ \forall \bar{w} \in S$

Proof All the proofs of these relations use the equality $1/p + 1/q = 1$. The first one can be obtained differentiating h. The second relation is obtained using Lemma 2 in Shalev-Shwartz and Singer (2007), through lengthy but straightforward calculations. The last one is obtained from the first one. ■

We now introduce Algorithm 3, that forms the basis for Algorithm 2, and a lemma that bounds its performance, that is a generalization of Theorem 1 in Do et al. (2009) to general norms, using the framework in Shalev-Shwartz and Singer (2007). Hence it can be seen as a particular case of the mirror descent algorithm (Beck and Teboulle, 2003), with an adaptive tuning of the learning rate.

Lemma 2 *Let $h(\cdot) = \frac{\alpha}{2}\|\cdot\|^2$ be a 1-strongly convex function w.r.t. a norm $\|\cdot\|$ over S. Assume that for all t, $g_t(\cdot)$ is a σ-strongly convex function w.r.t. $h(\cdot)$, and $\|z_t\|_* \le L_t$. Let $R \in \mathbb{R}^+$ such that $\|w - w'\| \le 2R$ for any $w, w' \in S$. Then for any $u \in S$, and for any sequence of non-negative $\xi_1, \dots, \xi_T$, Algorithm 3 achieves the following bound for all $T \ge 1$,*

$$\sum_{t=1}^{T} (g_t(w_t) - g_t(u)) \le \sum_{t=1}^{T} \left[4\xi_t R^2 + \frac{L_t^2}{\sigma t + \frac{\sum_{i=1}^t \xi_i}{\alpha}} \right] .$$

Proof Define $g'_t(w) = g_t(w) + \frac{s_t}{2}\|w - w_t\|^2$, where $w, w_t \in S$, and the value of s_t will be specified later. Using the assumptions of this Lemma, we have that g'_t is $(\sigma + \frac{s_t}{\alpha})$-strongly convex w.r.t. to h. Moreover we have that $\partial g'_t(w_t) = \partial g_t(w_t)$, because the gradient of the proximal regularization term is zero when evaluated at w_t (Do et al., 2009). Hence we can apply Theorem 1 from Shalev-Shwartz and Singer (2007) to have

$$\sum_{t=1}^{T} g_t(w_t) - \sum_{t=1}^{T} \left(g_t(u) + \frac{s_t}{2}\|u - w_t\|^2 \right) = \sum_{t=1}^{T} g'_t(w_t) - \sum_{t=1}^{T} g'_t(u) \le \frac{1}{2} \sum_{t=1}^{T} \frac{L_t^2}{\sigma t + \frac{\sum_{i=1}^t s_i}{\alpha}} .$$

Using the hypothesis of this Lemma we obtain

$$\sum_{t=1}^{T} g_t(w_t) - \sum_{t=1}^{T} g_t(u) \le \frac{1}{2} \sum_{t=1}^{T} \left(s_t \|u - w_t\|^2 + \frac{\alpha L_t^2}{\alpha\sigma t + \sum_{i=1}^t s_i} \right) \le \frac{1}{2} \sum_{t=1}^{T} \left(4 s_t R^2 + \frac{\alpha L_t^2}{\alpha\sigma t + \sum_{i=1}^t s_i} \right) .$$

Using the definition of s_t in the algorithm and Lemma 3.1 in Bartlett et al. (2008), we have

$$\sum_{t=1}^{T} g_t(w_t) - \sum_{t=1}^{T} g_t(u) \le \min_{\xi_1, \dots, \xi_T \ge 0} \sum_{t=1}^{T} \left(4\xi_t R^2 + \frac{\alpha L_t^2}{\alpha\sigma t + \sum_{i=1}^t \xi_i} \right) .$$

Hence these settings of s_t give us a bound that is only 2 times worse than the optimal one. ■

With this Lemma we can now design stochastic sub-gradient algorithms. In particular, setting $\|\cdot\|_{2,p}$ as the norm, $h(\bar{w}) = \frac{q}{2}\|\bar{w}\|_{2,p}^2$, and $g_t(\bar{w}) = \frac{\lambda}{q} h(\bar{w}) + \ell(\bar{w}, x_t, y_t)$, we obtain Algorithm 2 that solves the p-norm MKL problem in (5). The updates are done on the dual variables $\bar{\theta}_t$, in lines 9-10, and transformed into $\bar{w}_t$ in line 11, through a simple scaling. We can now prove the following bound on the convergence rate for Algorithm 2.

Theorem 3 *Suppose that $\|\partial\ell(\bar{w}, x_t, y_t)\|_{2,q} \le L$ and $\|\bar{w}^*\|_{2,p} \le R$, where $\bar{w}^*$ is the optimal solution of (5), that is $\bar{w}^* = \operatorname{argmin}_{\bar{w}} f(\bar{w})$. Let $1 < p \le 2$, and $c = \lambda R + L$, then in expectation over the choices of the random samples we have that, after T iterations of the 2nd stage of the OBSCURE algorithm, the difference between $f(\bar{w}_T)$ and $f(\bar{w}^*)$, is less than*

$$c\sqrt{q}\sqrt{1 + \log T} \min\left(\frac{c\sqrt{q}\sqrt{1 + \log T}}{\lambda T}, \frac{4R}{\sqrt{T}} \right) .$$

Proof Let $h(\bar{w}) : S \to \mathbb{R}$ defined as $\frac{q}{2}\|\bar{w}\|_{2,p}^2$, where $S = \{\bar{w} : \|\bar{w}\|_{2,p} \le R\}$. Define also $g_t(\bar{w}) = \frac{\lambda}{2}\|\bar{w}\|_{2,p}^2 + \ell(\bar{w}, x_t, y_t) = \frac{\lambda}{q}h(\bar{w}) + \ell(\bar{w}, x_t, y_t)$. Using Lemma 1 in Shalev-Shwartz and Singer (2007), we can see that these two functions satisfy the hypothesis of Lemma 1, with $\alpha = q$, $\sigma = \frac{\lambda}{q}$. It is easy to verify that $\bar{w}_{t+1}$ is equal to $\nabla h^*(\nabla h(\bar{w}_t) - \eta_t z_t)$. In fact, taking into account Properties 1-3 in Lemma 1 with with $B = R$, lines 9-11 in Algorithm 2 are equivalent to

$$\bar{w}_{t+1} = \nabla h^*(\theta_t - \eta_t z_t)$$
$$\bar{\theta}_{t+1} = \nabla h(\bar{w}_{t+1})\ .$$

We also have that

$$\|\partial g_t(\bar{w})\|_{2,q} \le \frac{\lambda}{q}\|\nabla h(\bar{w}_t)\|_{2,q} + \|\bar{z}_t\|_{2,q} = \lambda\|\bar{w}_t\|_{2,p} + \|\bar{z}_t\|_{2,q} \le c,$$

where the equality is due to Property 3 in Lemma 1. So we have

$$\sum_{t=1}^{T}(g_t(\bar{w}_t) - g_t(\bar{w}^*)) \le \min_{\xi_1,\cdots,\xi_T}\sum_{t=1}^{T}\left[4\xi_t R^2 + \frac{qc^2}{\lambda t + \sum_{i=1}^{t}\xi_i}\right]\ .$$

Reasoning as in Shalev-Shwartz et al. (2007), we divide by T, take the expectation on both sides. So we obtain that

$$\mathbb{E}[f(\bar{w}_T) - f(\bar{w}^*)] \le \min_{\xi_1,\cdots,\xi_T}\frac{1}{T}\sum_{t=1}^{T}\left[4\xi_t R^2 + \frac{qc^2}{\lambda t + \sum_{i=1}^{t}\xi_i}\right]\ .$$

Setting $\xi_i = \xi$, $i = 1, \ldots, T$, the last term in the last equation can be upper bounded by

$$A_T = \min_{\xi}\left[4\xi R^2 + \frac{1}{T}\sum_{t=1}^{T}\frac{qc^2}{t(\lambda + \xi)}\right]\ .$$

This term is smaller than any specific setting of ξ, in particular if we set ξ to 0, we have that $A_T \le \frac{qc^2(1+\log T)}{\lambda T}$. On the other hand setting optimally the expression over ξ and over-approximating we have that $A_T \le \frac{4cR\sqrt{q}\sqrt{1+\log T}}{\sqrt{T}}$. Taking the minimum of these two quantities we obtain the stated bound. ■

The most important thing to note is that the converge rate is independent from the number of samples, as in Pegasos (Shalev-Shwartz et al., 2007), and the relevant quantities on which it depends are λ and q. Given that for most of the losses, each iteration has a linear complexity in the number of samples, as stated in Section 3.1, the training time will be linearly proportional to the number of samples.

The parameter R is basically an upper bound on the norm of the optimal solution. In the next Section we show how to have a good estimate of R in an efficient way. The theorem first shows that a good estimate of R can speed-up the convergence of the algorithm. In particular if the first term is dominant, the convergence rate is $O(\frac{q\log T}{\lambda T})$. If the second term is predominant, the convergence rate is $O(\frac{R\sqrt{q\log T}}{\sqrt{T}})$, so it becomes independent from λ. The algorithm will always optimally interpolate between these two different rates of convergence. Note that R can also be set to the trivial upper bound of ∞. This would result in a standard Pegasos-like optimization. In fact, s_t would be equal to 0 in Algorithm 2, so the learning rate would be $\frac{1}{\lambda t}$ and the convergence rate would be $O(\frac{q \log T}{\lambda T})$. We will see in Section 5.3 that a tight estimate of R can improve dramatically the convergence rate. Another important point is that Algorithm 2 can start from any vector, while this is not possible in the Pegasos algorithm, where at the very first iteration the starting vector is multiplied by 0 (Shalev-Shwartz et al., 2007).

As said before, the rate of convergence depends on p, through q. A p close to 1 will result in a sparse solution, with a rate of at most $O(\frac{R\sqrt{q \log T}}{\sqrt{T}})$. However in the experiment section we show that the best performance is not always given by the most sparse solution.

This theorem and the pseudocode in Algorithm 2 allows us to design fast and efficient MKL algorithms for a wide range of convex losses. If we consider the multiclass loss ℓ^{MC} with normalized kernels, that is, $\|\phi^j(x_t, y_t)\|_2 \leq 1, \forall j = 1, \cdots, F,\ t = 1, \cdots, N$, we have that $L \leq \sqrt{2} F^{\frac{1}{q}}$. Instead, if we use the hinge loss ℓ^{HL} for binary classification, we have that $L \leq F^{\frac{1}{q}}$. Hence, in both cases, if $p < 2$, the convergence rate has a sublinear dependency on the number of kernels, F, and if the problem is linearly separable it can have a faster convergence rate using more kernels. We will explain this formally in the next section.

4.1 Initialization Through an Online Algorithm

In Theorem 3 we saw that if we have a good estimate of R, the convergence rate of the algorithm can be much faster. Moreover starting from a *good* solution could speed-up the algorithm even more.

We propose to initialize Algorithm 2 with Algorithm 1. It is the online version of problem (5) and it is derived using a recent result in Orabona and Crammer (2010). It is similar to the $2p$-norm matrix Perceptron of Cavallanti et al. (2008), but it overcomes the disadvantage of being used with the same kernel on each feature.

We can run it just for few iterations and then evaluate its norm and its loss. In Algorithm 1 R is then defined as

$$R := \sqrt{\|\bar{w}_{T+1}\|^2_{2,p} + \frac{2}{\lambda N} \sum_{i=1}^{N} \ell(\bar{w}_{T+1}, x_i, y_i)} \geq \sqrt{\|\bar{w}^*\|^2_{2,p} + \frac{2}{\lambda N} \sum_{i=1}^{N} \ell(\bar{w}^*, x_i, y_i)} \geq \|\bar{w}^*\|_{2,p},$$

where we remind that $\bar{w}^*$ is solution that minimizes (5), as defined in Section 2.2. So at any moment we can stop the algorithm and obtain an upper bound on $\|\bar{w}^*\|_{2,p}$. However if the problem is linearly separable we can prove that Algorithm 1 will converge in a finite number of updates. In fact, as in Cavallanti et al. (2008), for Algorithm 1 it is possible to prove a relative mistake bound. See also Jie et al. (2010b) and Jin et al. (2010) for similar algorithms for online MKL, with a different update rules and different mistake bounds.

Theorem 4 *Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of examples where $x_t \in \mathbb{X}$, $y \in \mathbb{Y}$. Suppose that the loss function ℓ has the following properties*

- $\|\partial \ell(\bar{w}, x, y)\|_{2,q} \leq L,\ \forall\, \bar{w} \in \mathbb{X},\ x_t \in \mathbb{X},\ y_t \in \mathbb{Y}$*;*
- $\ell(\bar{u}, x, y) \geq 1 + \bar{u} \cdot \partial \ell(\bar{w}, x, y),\ \forall\, \bar{u} \in \mathbb{X},\ \bar{w} \in \mathbb{X} : \ell(\bar{w}, x, y) > 0,\ x \in \mathbb{X},\ y \in \mathbb{Y}$*;*
- $\bar{w} \cdot \partial \ell(\bar{w}, x, y) \geq -1,\ \forall\, \bar{w} \in \mathbb{X},\ x \in \mathbb{X},\ y \in \mathbb{Y}$.

Denote by $\mathcal{U}$ the set of rounds in which there is an update, and by U its cardinality. Then, for any $\bar{u}$, the number of updates U of Algorithm 1 satisfies

$$U \leq q(2/\eta + L^2)\|\bar{u}\|_{2,p}^2 + \sum_{t\in\mathcal{U}} \ell(\bar{u}, x_t, y_t) + \|\bar{u}\|_{2,p}\sqrt{q(2/\eta+L^2)}\sqrt{\sum_{t\in\mathcal{U}} \ell(\bar{u}, x_t, y_t)}\ .$$

In particular, if the problem (5) is linearly separable by a hyperplane $\bar{v}$, then the Algorithm 1 will converge to a solution in a finite number of steps less than $q(2/\eta + L^2)\|\bar{v}\|_{2,p}^2$. In this case the returned value of R will be less than $(2+\eta L^2)\|\bar{v}\|_{2,p}$.

Proof The bound on the number of updates can be easily derived using a recent result in Orabona and Crammer (2010), that we report in Appendix for completeness. Let $h(\bar{w}) : \mathbb{X} \to \mathbb{R}$ defined as $\frac{q}{2}\|\bar{w}\|_{2,p}^2$. Notice that, differently from the proof of Theorem 3, here the domain of the function h is the entire Euclidean space $\mathbb{X}$. Using Property 2 in Lemma 1 with $B = \infty$, we have that line 7 in the algorithm's pseudo-code implies that $\bar{w}_t = \nabla h^*(\bar{\theta}_t)$. Using Lemma 5 in the Appendix, we have that

$$\begin{aligned} U &\leq \sum_{t\in\mathcal{U}} \ell(\bar{u}, x_t, y_t) + \sqrt{q}\|\bar{u}\|_{2,p}\sqrt{\sum_{t\in\mathcal{U}}\left(\|\bar{z}_t\|_{2,q}^2 - \frac{2\bar{w}_t\cdot\bar{z}_t}{\eta}\right)} \\ &\leq \sum_{t\in\mathcal{U}} \ell(\bar{u}, x_t, y_t) + \sqrt{q}\|\bar{u}\|_{2,p}\sqrt{U\left(L^2 + \frac{2}{\eta}\right)}\ . \end{aligned}$$

Solving for U and overapproximating we obtain the stated bound.

For the second part of the Theorem, using (Kakade et al., 2009, Corollary 19), we know that $h^*(\bar{w})$ is 1-smooth w.r.t. $\|\cdot\|_{2,q}$. Hence, we obtain

$$\begin{aligned} \|\bar{\theta}_{T+1}\|_{2,q}^2 &\leq \|\bar{\theta}_T\|_{2,q}^2 - 2q\eta\bar{w}_T\cdot\bar{z}_T + q\eta^2\|\bar{z}_T\|_{2,q}^2 \leq \eta^2 q \sum_{t\in\mathcal{U}}\left(\|\bar{z}_t\|_{2,q}^2 - \frac{2\bar{w}_t\cdot\bar{z}_t}{\eta}\right) \\ &\leq \eta^2 qU\left(\frac{2}{\eta} + L^2\right)\ . \end{aligned}$$

So we can write

$$\|\bar{\theta}_{T+1}\|_{2,q} \leq \eta\sqrt{qU(2/\eta+L^2)},$$

and using the bound of U and the hypothesis of linear separability, we have

$$\|\bar{\theta}_{T+1}\|_{2,q} \leq \eta\sqrt{q^2\|\bar{u}\|_{2,p}^2(2/\eta+L^2)^2} = q\|\bar{u}\|_{2,p}\left(2+\eta L^2\right)\ .$$

Using the relation $\|\bar{w}_t\|_{2,p} = \frac{1}{q}\|\bar{\theta}_t\|_{2,q}$, that holds for Property 2 in Lemma 1 with $B = \infty$, we have the stated bound on R. ■

From the theorem it is clear the role of η: a bigger value will speed up the convergence, but it will decrease the quality of the estimate of R. So η governs the trade-off between speed and precision of the first stage.

The multiclass loss ℓ^{MC} and the hinge loss ℓ^{HL} satisfy the conditions of the Theorem, and, as noted for Theorem 1 when p is close to 1, the dependency on the number of kernels in this theorem is strongly sublinear.

Note also that the separability assumption is far from being unrealistic in our setting. In fact the opposite is true: in the greater majority of the cases the problem will be linearly separable. This is due to the fact that in MKL to have separability it is sufficient that only one of the kernel induces a feature space where the problem is separable. So, for example, it is enough to have no repeated samples with different labels and at least one kernel that always produces kernel matrices with full rank, for example, the Gaussian kernel.

Moreover, under the separability assumption, if we increase the number of kernels, we have that $\|\bar{u}\|_{2,p}^2$ cannot increase, and in most of the cases it will decrease. In this case we expect Algorithm 1 to converge to a solution which has null loss on each training sample, in a finite number of steps that is almost independent on F and in some cases even *decreasing* while increasing F. The same consideration holds for the value of R returned by the algorithm, that can decrease when we increase the number of kernels. A smaller value of R will mean a faster convergence of the second stage. We will confirm this statement experimentally in Section 5.

5. Experiments

In this section, we study the behavior of OBSCURE in terms of classification accuracy, computational efficiency and scalability. We implemented our algorithm in MATLAB, in the DOGMA library (Orabona, 2009). We focus on the multiclass loss ℓ^{MC}, being it much harder to be optimized than the binary hinge loss ℓ^{HL}, especially in the MKL setting. Although our MATLAB implementation is not optimized for speed, it is already possible to observe the advantage of the low runtime complexity. This is particularly evident when training on data sets containing large numbers of categories and lots of training samples. Except in the synthetic experiment where we set $p = 1.0001$, in all the other experiments the parameter p is chosen from the set $\{1.01, 1.05, 1.10, 1.25, 1.50, 1.75, 2\}$. The regularization parameter λ is set through CV, as $\frac{1}{CN}$, where $C \in \{0.1, 1, 10, 100, 1000\}$.

We compare our algorithm with the binary SILP algorithm (Sonnenburg et al., 2006), the multi class MKL (MC-MKL) algorithm (Zien and Ong, 2007) and the p-norm MKL algorithm (Kloft et al., 2009), all of them implemented in the SHOGUN-0.9.2 toolbox.[3] For p-norm MKL, it is possible to convert from our p setting to the equivalent setting in Kloft et al. (2009) using $p_{\text{p-norm}} = p_{\text{OBSCURE}}/(2 - p_{\text{OBSCURE}})$. In our experiments, we will compare between OBSCURE and p-norm MKL using the equivalent p parameter. We also compare with the SimpleMKL algorithm[4] (Rakotomamonjy et al., 2008). To train with the unweighted sum of the kernels with an SVM, we use LIBSVM (Chang and Lin, 2001). The cost parameter is selected from the range $C \in \{0.1, 1, 10, 100, 1000\}$ for all the baseline methods. For all the binary classification algorithms, we use the 1-vs-All strategy for their multiple class extensions.

In the following we start by briefly introducing the data sets used in our experiments. Then we present a toy problem on a synthetic data which shows that it is more appropriate to use a multiclass loss instead of dividing the multiclass classification problem into several binary subproblems. We

3. Available at `http://www.shogun-toolbox.org`, implemented in C++.
4. Available at `http://asi.insa-rouen.fr/enseignants/~arakotom/code/mklindex.html`, implemented in MATLAB. SimpleMKL is more efficient than SILP when uses the same SVM solver (Rakotomamonjy et al., 2008). However, in practice, no efficient SimpleMKL implementation is available. SILP runs much faster compared to SimpleMKL, especially when the size of the problem grows. Moreover, the performance of SimpleMKL and SILP are the same because both algorithm solve an equivalent optimization problem (Rakotomamonjy et al., 2008). Therefore, we only use SILP algorithm as our l_1-norm MKL baseline in experiments whose size of training samples are large than 1000.

then study the convergence rate of OBSCURE and compare it with the original Pegasos algorithm (Shalev-Shwartz et al., 2007; Shalev-Shwartz and Srebro, 2008) as well as the p-norm MKL (Kloft et al., 2009) (Section 5.3). Following that, we study the behaviors of OBSCURE w.r.t different value of p and different number of input kernels (Sections 5.5 and 5.6). Finally we show that OBSCURE achieves state-of-art performance on a challenging image classification task with 102 different classes, and we show its scalability.

5.1 Data Sets

We first briefly introduce the data sets used in this section, and we describe how their kernel matrices are generated.

5.1.1 THE OXFORD FLOWER DATA SET (NILSBACK AND ZISSERMAN, 2006)

contains 17 different categories of flowers. Each class has 80 images with three predefined splits (train, validation and test). The authors also provide 7 precomputed distance matrices.[5] These distance matrices are transformed into kernel using $\exp(-\gamma^{-1}d)$, where γ is the average pairwise distance and d is the distance between two examples. It results in 7 different kernels.

5.1.2 THE PENDIGITS DATA SET (GÖNEN AND ALPAYDIN, 2010)

is on pen-based digit recognition (multiclass classification with 10 classes) and contains four different feature representations.[6] The data set is split into independent training and test sets with 7494 samples for training and 3498 samples for testing. We have generated 4 kernel matrices, one matrix for each feature, using an RBF kernel, $\exp(-\gamma^{-1}\|x_i - x_j\|^2)$. For each feature, γ is equal to the average of the squared pairwise distances between the examples.

5.1.3 THE KTH-IDOL2 DATA SET (PRONOBIS ET AL., 2010)

contains 24 image sequences acquired using a perspective camera mounted on two mobile robot platforms. These sequences were captured with the two robots moving in an indoor laboratory environment consisting of five different rooms under various weather and illumination conditions (sunny, cloudy, and night) and across a time span of six months. For experiments, we used the same setup described in Pronobis et al. (2010); Jie et al. (2010a). We considered the 12 sequences acquired by robot Dumbo, and divided them into training and test sets, where each training sequence has a corresponding one in the test sets captured under roughly similar conditions. In total, we considered twelve different permutations of training and test sets. The images were described using three visual descriptors and a geometric feature from the Laser Scan sensor, as in Jie et al. (2010a), which forms 4 kernels in total.

5.1.4 THE CALTECH-101 DATA SET (FEI-FEI ET AL., 2004)

is a standard benchmark data set for object categorization. In our experiments, we used the pre-computed features and kernels of Gehler and Nowozin (2009b) which the authors made available on their website,[7] with the same training and test split. This allows us to compare against them

5. Available at `www.robots.ox.ac.uk/~vgg/research/flowers/`.
6. Available at `http://mkl.ucsd.edu/dataset/pendigits`.
7. Available at `www.vision.ee.ethz.ch/~pgehler/projects/iccv09/`.

directly. Following that, we report results using all 102 classes of the Caltech-101 data set using five splits. There are five different image descriptors, namely, PHOG Shape Descriptors (PHOG) (Bosch et al., 2007), Appearance Descriptors (App) (Lowe, 2004), Region Covariance (RECOV) (Tuzel et al., 2007), Local Binary Patterns (LBP) (Ojala et al., 2002) and V1S+ (Pinto et al., 2008). All of them but the V1S+ features were computed in a spatial pyramid as proposed by Lazebnik et al. (2006), using several different setup of parameters. This generates several kernels (PHOG, 8 kernels; App, 16 kernels; RECOV, 3 kernels; LBP 3 kernels; V1S+, 1 kernels). We also compute a subwindow kernel, as proposed by Gehler and Nowozin (2009a). In addition to the 32 kernels, the products of the pyramid levels for each feature results in other 7 kernels, for a total of 39 different kernels For brevity, we omit the details of the features and kernels and refer to Gehler and Nowozin (2009a,b).

5.1.5 THE MNIST DATA SET (LECUN ET AL., 1998)

is a handwritten digits data set. It has a training set of 60,000 gray-scale 28x28 pixel digit images for training and 10,000 images for testing. We cut the original digit image into four square blocks (14×14) and obtained an input vector from each block. We used three kernels on each block: a linear kernel, a polynomial kernel and a RBF kernel, resulting in 12 kernels.

5.2 Multiclass Synthetic Data

Multiclass problems are often decomposed into several binary sub-problems using methods like 1-vs-All, however solving the multiclass learning problem jointly using a multiclass loss can yield much sparser solutions. Intuitively, when a l_1-norm is used to impose sparsity in the domain of kernels, different subsets of kernels can be selected for the different binary classification sub-problems. Therefore, the combined multiclass classifier might not obtain the desired properties of sparsity. Moreover, the confidence outputs of the binary classifiers may not lie in the same range, so it is not clear if the winner-takes-all hypothesis is the correct approach for combing them.

To prove our points, we have generated a 3-classes classification problem consisting of 300 samples, with 100 samples for each class. There are in total 4 different features, the kernel matrices corresponding to them are shown in Figure 1 (top). These features are generated in a way that Kernels 1–3 are useful only for distinguishing one class (class 3, class 1 and class 2, respectively) from the other two, while Kernel 4 can separate all the 3 classes. The corresponding kernel combination weights obtained by the SILP algorithm using the 1-vs-All extension and our multiclass OBSCURE are shown in Figure 1 (bottom). It can be observed that each of the binary SILP classifiers pick two kernels. OBSCURE selects only the 4th kernel, achieving a much sparser solution.

5.3 Comparison with $\frac{1}{t}$ Learning Rate

We have compared OBSCURE with a simple one-stage version that uses a $\frac{1}{t}$ learning rate. This can be obtained setting $s_t = 0,\ \forall t$, in Algorithm 2. It can be considered as a straightforward extension of the original Pegasos algorithm (Shalev-Shwartz et al., 2007; Shalev-Shwartz and Srebro, 2008) to the MKL problem of (5), so we denote it Pegasos-MKL.

We first compare the running time performance between OBSCURE and Pegasos-MKL on the Oxford flowers data set. Their generalization performance on the testing data (Figure 2(Top, left)) as well as the value of the objective function (Figure 2(Top, right)) are shown in Figure 2. In the same Figure, we also present the results obtained using SILP, SimpleMKL, p-norm MKL and MC-MKL.

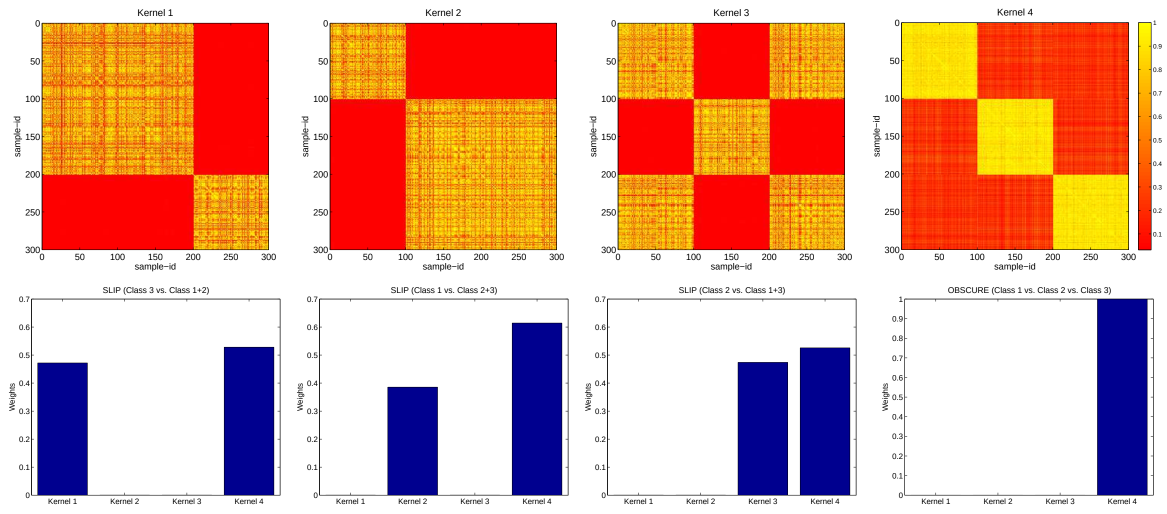

Figure 1: (top) Kernel matrices of the 3-classes synthetic experiments correspond to 4 different features. Sample 1–100, 101–200 and 201–300 are from class 1, 2 and 3 respectively. (bottom) Corresponding kernel combination weights, normalized to have sum equal to 1, obtained by SILP (binary) and by OBSCURE (last figure).

We see that OBSCURE converges much faster compared to Pegasos-MKL. This proves that, as stated in Theorem 3, OBSCURE has a better convergence rate than Pegasos-MKL, as well as faster running time than SILP and SimpleMKL. Note that all the feature combination methods achieve similar results on this data set.

Similar results are shown in Figure 2(Bottom, left) and (Bottom, right) on the Pendigits data sets.

5.4 Comparison with p-norm MKL and Other Baselines

We compare OBSCURE with p-norm MKL (Kloft et al., 2009). Figure 3 reports the results obtained by both algorithms for varying values of p on the Pendigits data set. We can see that all the algorithms (OBSCURE, SILP and p-norm MKL) are order of magnitudes faster than MC-MKL. OBSCURE and p-norm MKL achieve similar performance, but OBSCURE achieve optimal performance in a training time much faster (10^1 and 10^2). The performance of SILP and p-norm MKL are quite close, and their classification rate seems to be more stable on this data set. The difference between OBSCURE and p-norm MKL may be due to the different types of multi class extension they use.

5.5 Experiments with Different Values of p

This experiment aims at showing the behavior of OBSCURE for varying value of p. We consider $p \in (1,2]$, and train OBSCURE on the KTH-IDOL2 and Caltech-101. The results for the two data sets are shown in Figure 4 (top).

For the IDOL2 data set (Figure 4 (top, left)), the best performance is achieved when p is large, which corresponds to give all the kernels similar weights in the decision. On the contrary, a sparse

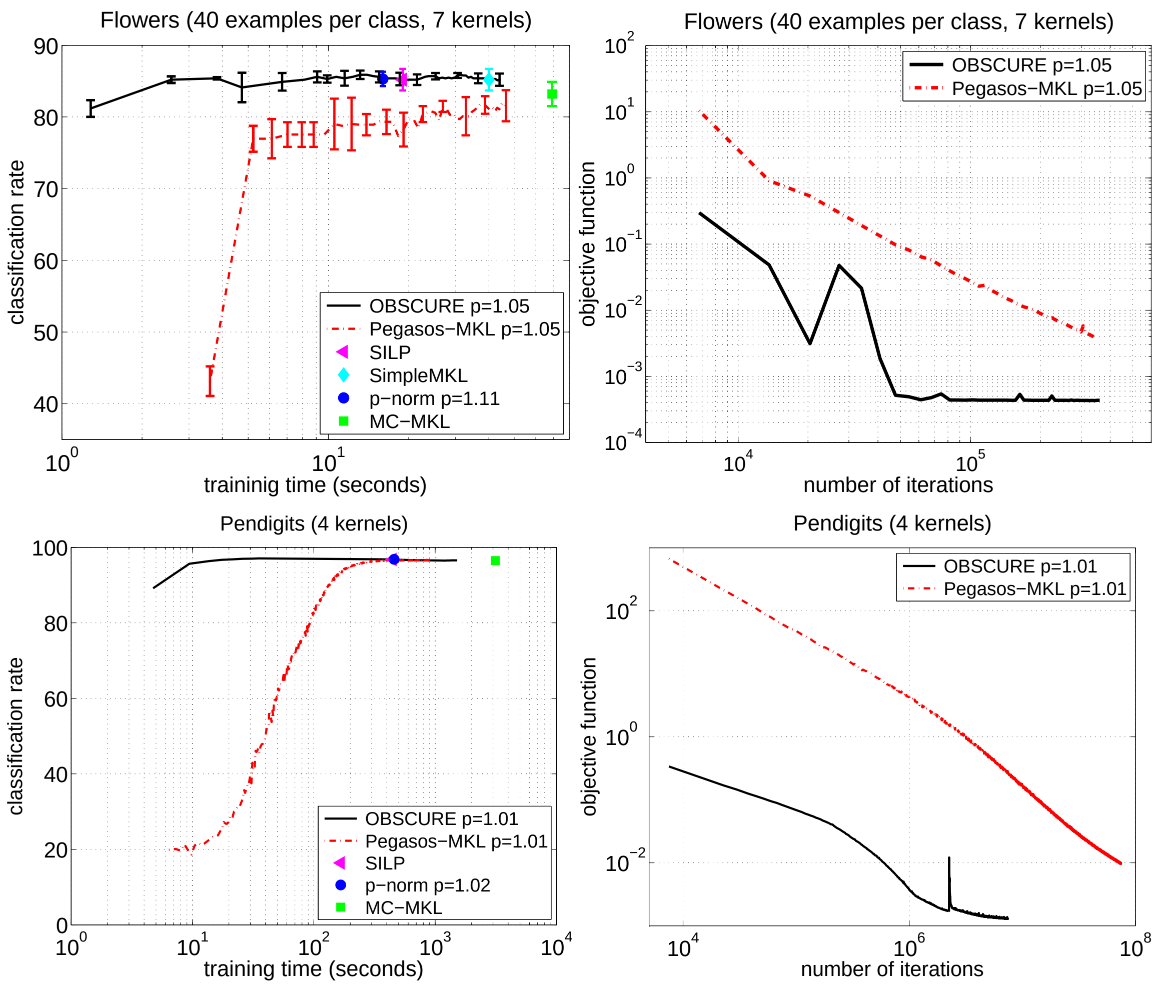

Figure 2: Comparison of running time performance (Left) and objective function value (Right) on the Oxford flowers data set (Top) and Pendigits data set (Bottom).

solution achieves lower accuracy. It indicates that all the kernels carry discriminative information, and excluding some of them can decrease the performance.

For the Caltech-101 data set (Figure 4 (top, right)), following Gehler and Nowozin (2009b), we consider four PHOG (Bosch et al., 2007) kernels computed at different spatial pyramid level. It can be observed that by adjusting p it is possible to improve the performance—sparser solutions (i.e., when p tends to 1) achieve higher accuracy compared to non-sparse solutions (when p tends to 2). However, the optimal p here is 1.10. In other words the optimal performance is achieved for a setting of p different from 1 or 2, fully justifying the presence of this parameter.

Furthermore, Figure 4 (bottom) shows the running time of OBSCURE using the same four kernels, with varying values of p. The dashed lines in the figure correspond to the results obtained by the first online stage of the OBSCURE algorithm. It can be observed that the online stage of OBSCURE converges faster when p is large, and this is consistent with Theorem 4. The online step

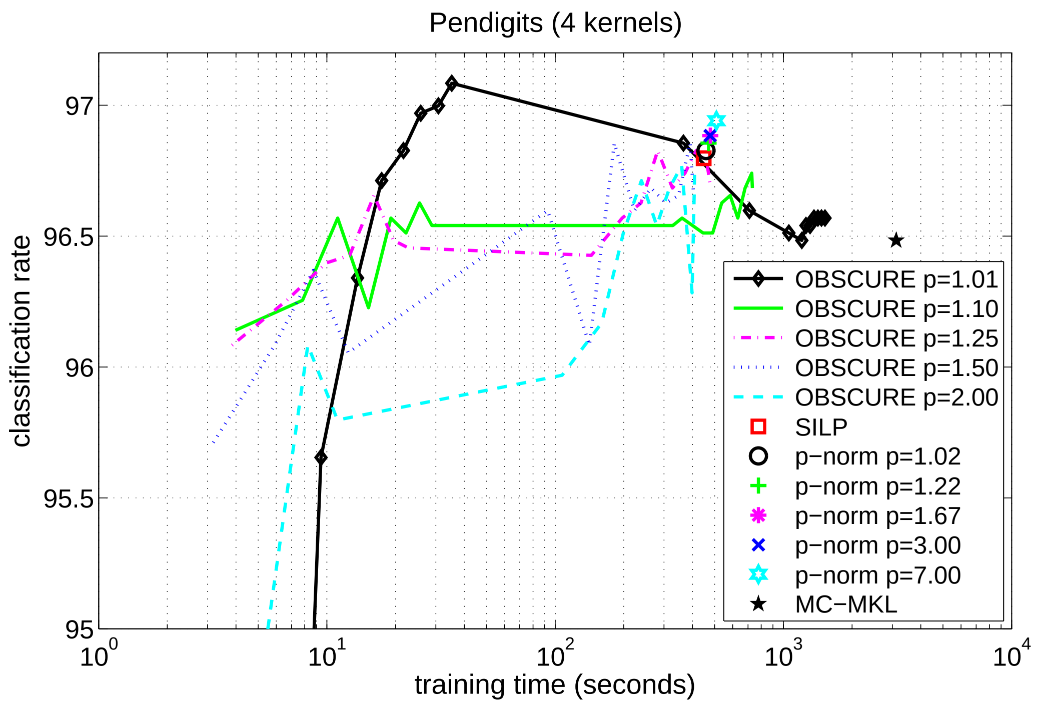

Figure 3: (Best viewed in color.) Comparison of OBSCURE and p-norm MKL with varying value of p on Pendigits.

of OBSCURE converges in a training time orders of magnitudes faster (10^1 to 10^3) compared to the full training stage, and in certain cases ($p \leq 1.10$) it can also achieve a performance close to the optimal solution.

5.6 Experiments on Different Number of Kernels

Figure 5 (left) reports the behavior of OBSCURE for different numbers of input kernels. It shows that the algorithm achieves a given accuracy in less iterations when more kernels are given. The dashed line in the figure again corresponds to the results obtained by the first online stage of the OBSCURE algorithm. Figure 5 (right) shows the number of iterations to converge of the online step, proving that the convergence rate improves when there are more kernels, as stated in Theorem 4.

5.7 Multiclass Image Classification

In this experiment, we use the Caltech-101 data set with all the 39 kernels, and the results are shown in Figure 6. The best results for OBSCURE were obtained when p is at the smallest value (1.01). This is probably because among these 39 kernels many were redundant or not discriminative enough. For example, the worst single kernel achieves only an accuracy of $13.5\% \pm 0.6$ when trained using 30 images per category, while the best single kernel achieves $69.4\% \pm 0.4$. Thus, sparser solutions are to be favored. The results support our claim in Section 5.2 that multiclass loss function is more suitable for this type of problem, as all the methods that use the multiclass loss

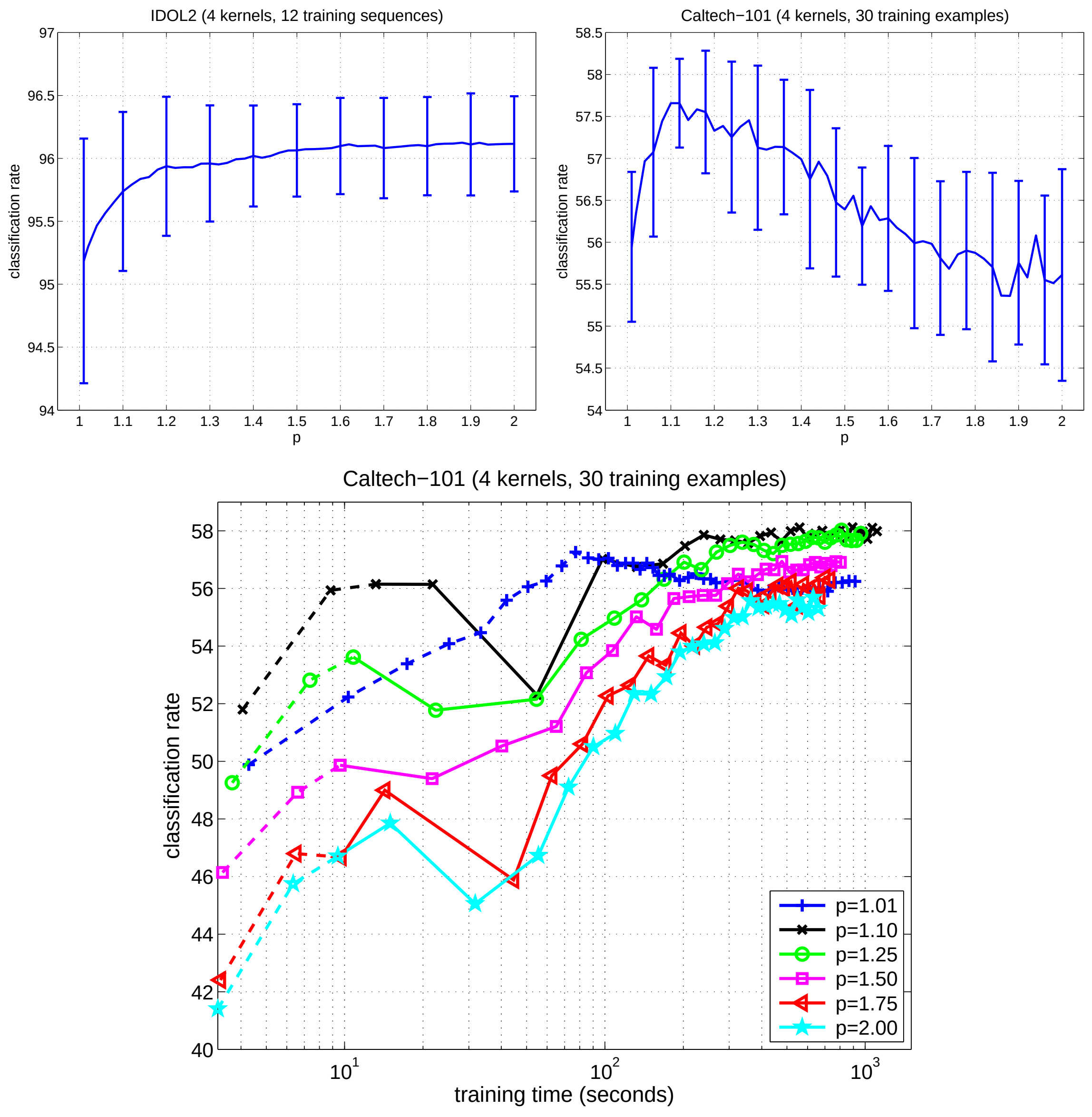

Figure 4: Behaviors of the OBSCURE algorithm w.r.t. p: (top, left) the effect of different values of p on the IDOL2 data set and (top, right) on the Caltech-101 data set using 4 PHOG (Bosch et al., 2007) kernels; (bottom) running time for different values of p on Caltech-101 data set.

outperform SILP and p-norm MKL (p=1.02) using 1-vs-All strategy. MC-MKL is computationally infeasible for 30 sample per category. Its significant gap from OBSCURE seems to indicate that it stops before converging to the optimal solution. Figure 6 (left) reports the training time for different algorithms. Again, OBSCURE reaches optimal solution much faster than the other three baseline algorithms which are implemented in C++. Figure 6 (right) reports the results obtained

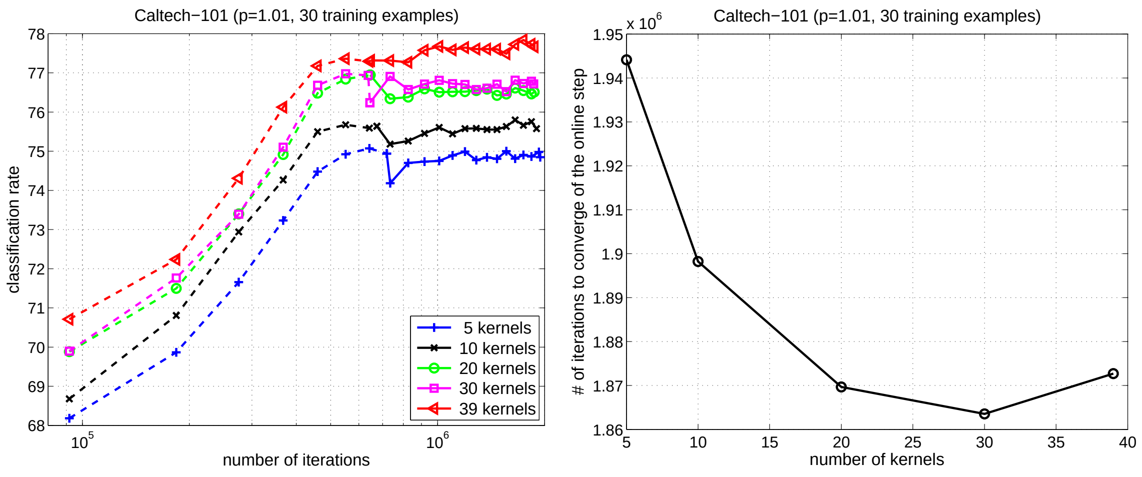

Figure 5: Behaviors of the OBSCURE algorithm w.r.t. the number of kernels: (left) the effect of different number of kernels randomly sampled from the 39 kernels; (right) number of iterations to converge of the online stage.

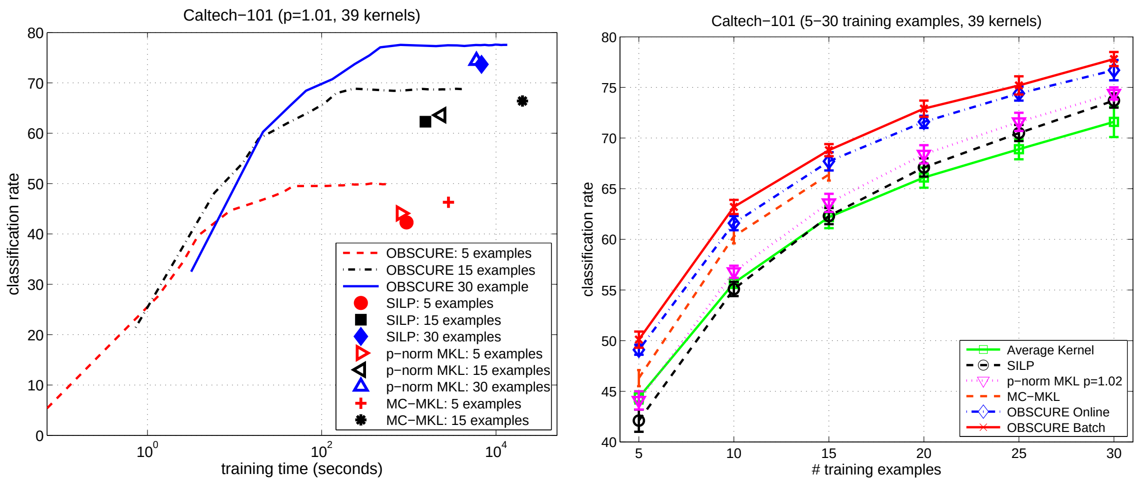

Figure 6: Performance comparison on Caltech-101 using different MKL methods.

using different combination methods for varying size of training samples. It is also interesting to note the performance of the solution generated by the online step of OBSCURE, denoted by "OBSCURE Online", that is very close to the performance of the full training stage, as already noted above.

5.8 Scalability

In this section, we report the experiments on the MNIST data set using varying sizes of training samples. Figure 7 shows the generalization performance on the test set achieved by OBSCURE over time, for various training size. We see that OBSCURE quickly produces solutions with good

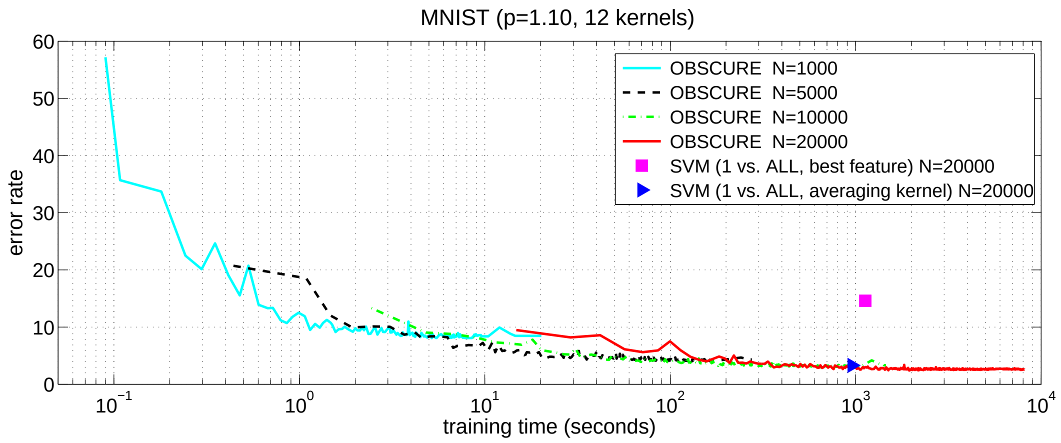

Figure 7: The generalization performance of MNIST data set over different size of training samples.

performance. The performance of the SVM trained using the unweighted sum of the kernels and the best kernel are also plotted. Notice that in the figure we only show the results of up to 20,000 training samples for the sake of comparison, otherwise we could not cache all the 12 kernels in memory. However, by computing the kernel "*on the fly*" we are able to solve the MKL problem using the full 60,000 examples: OBSCURE obtains 1.95% error rate after 10 epochs, which is 0.45% lower compared to the results obtained by OBSCURE with 20,000 training samples after 500 epochs.

6. Conclusions and Discussion

This paper presents OBSCURE, a novel and efficient algorithm for solving p-norm MKL. It uses a hybrid two-stages online-batch approach, optimizing the objective function directly in the primal with a stochastic sub-gradient descent method. Our minimization method allows us to prove convergence rate bounds, proving that the number of iterations required to converge is independent of the number of training samples, and, when a sparse solution is induced, is sub-linear in the number of kernels. Moreover we show that OBSCURE has a faster convergence rate as the number of kernels grows.

Our approach is general, so it can be used with any kind of convex losses, from binary losses to structure output prediction (Tsochantaridis et al., 2004), and even to regression losses.

Experiments show that OBSCURE achieves state-of-art performance on the hard problem of multiclass MKL, with smaller running times than other MKL algorithms. Furthermore, the solution found by the online stage is often very close to the optimal one, while being computed several orders of magnitude faster.

Acknowledgments

The kernel matrixes of Caltech-101 were kindly provided by Peter Gehler, who we also thank for his useful comments. This work was sponsored by the EU project DIRAC IST-027787. FO also acknowledges partial support by the PASCAL2 NoE FP7-216886.

Appendix A.

The following algorithm and Lemma can in found in Orabona and Crammer (2010), stated for the binary case. Here we state them for generic convex losses and report them here for completeness.

Algorithm 4 Prediction algorithm

```
1: Input: A series of strongly convex functions $h_1, \dots, h_T$.
2: Initialize: $\theta_1 = 0$
3: for $t = 1, 2, \dots, T$ do
4:     Receive $x_t$
5:     Set $w_t = \nabla h_t^*(\theta_t)$
6:     $z_t = \partial \ell_t(w_t)$
7:     $\theta_{t+1} = \theta_t - \eta_t z_t$
8: end for
```

Lemma 5 *Let $h_t, t = 1, \dots, T$ be β_t-strongly convex functions with respect to the norms $\|\cdot\|_{h_1}, \dots, \|\cdot\|_{h_T}$ over a set S and let $\|\cdot\|_{h_i^*}$ be the respective dual norms. Let $h_0(0) = 0$, and $x_1, \dots, x_T$ be an arbitrary sequence of vectors in $\mathbb{R}^d$. Assume that algorithm in Algorithm 4 is run on this sequence with the functions h_i. If $h_T(\lambda u) \le \lambda^2 h_T(u)$, and ℓ satisfies*

$$\ell(u, x_t, y_t) \ge 1 + u^\top \partial \ell_t(w_t),\ \forall u \in S, w_t : \ell_t(w_t) > 0,$$

then for any $u \in S$, and any $\lambda > 0$ we have

$$\sum_{t=1}^{T} \eta_t \le L + \lambda h_T(u) + \frac{1}{\lambda}\left(D + \sum_{t=1}^{T}\left(\frac{\eta_t^2}{2\beta_t}\|z_t\|_{h_t^*}^2 - \eta_t w_t^\top z_t\right)\right),$$

where $L = \sum_{t \in \mathcal{M} \cup \mathcal{U}} \eta_t \ell_t(u)$, and $D = \sum_{t=1}^{T}(h_t^*(\theta_t) - h_{t-1}^*(\theta_t))$. In particular, choosing the optimal λ, we obtain

$$\sum_{t=1}^{T} \eta_t \le L + \sqrt{2h_T(u)}\sqrt{2D + \sum_{t=1}^{T}\left(\frac{\eta_t^2}{\beta_t}\|z_t\|_{h_t^*}^2 - 2\eta_t w_t^\top z_t\right)}\ .$$

References

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO, algorithm. In *Proc. of the International Conference on Machine Learning*, 2004.

P. Bartlett, E. Hazan, and A. Rakhlin. Adaptive online gradient descent. In *Advances in Neural Information Processing Systems 20*, pages 65–72. MIT Press, Cambridge, MA, 2008.

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proc. of the 6th ACM International Conference on Image and Video Retrieval*, pages 401–408. ACM, July 2007.

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

G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear algorithms for online multitask classification. In *Proc. of the 21st Conference on Learning Theory*, 2008.

C. C. Chang and C. J. Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at `www.csie.ntu.edu.tw/~cjlin/libsvm`.

C. Cortes, M. Mohri, and A. Rostamizadeh. Two-stage learning kernel algorithms. In *Proc. of the 27th International Conference on Machine Learning*, 2010.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. On kernel-target alignment. In *Advances in Neural Information Processing Systems 14*, volume 14, pages 367–373, 2002.

C. B. Do, Q. V. Le, and Chuan-Sheng Foo. Proximal regularization for online and batch learning. In *Proc. of the International Conference on Machine Learning*, 2009.

L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2004.

P. Gehler and S. Nowozin. Let the kernel figure it out: Principled learning of pre-processing for kernel classifiers. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009a.

P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *Proc. of the International Conference on Computer Vision*, 2009b.

M. Gönen and E. Alpaydin. Cost-conscious multiple kernel learning. *Pattern Recognition Letters*, 31:959–965, July 2010.

D. Hush, P. Kelly, C. Scovel, and I. Steinwart. QP algorithms with guaranteed accuracy and run time for support vector machines. *Journal of Machine Learning Research*, 7:733–769, 2006.

L. Jie, F. Orabona, and B. Caputo. An online framework for learning novel concepts over multiple cues. In H. Zha, R. Taniguchi, and S. J. Maybank, editors, *Computer Vision - ACCV 2009, 9th Asian Conference on Computer Vision, Xi'an, China, September 23-27, 2009, Revised Selected Papers, Part I*, volume 5994 of *Lecture Notes in Computer Science*, Berlin / Heidelberg, 2010a. Springer.

L. Jie, F. Orabona, M. Fornoni, B. Caputo, and N. Cesa-Bianchi. OM-2: An online multi-class multi-kernel learning algorithm. In *4th IEEE Online Learning for Computer Vision Workshop (in CVPR10)*. IEEE Computer Society, June 2010b.

R. Jin, S. C. H. Hoi, and T. Yang. Online multiple kernel learning: Algorithms and mistake bounds. In *Proc. of the 21st International Conference on Algorithmic Learning Theory*, pages 390–404, 2010.

T. Joachims. Making large-scale SVM learning practical. In *Advances in Kernel Methods – Support Vector Learning*, pages 169–185. MIT Press, 1999.

S. Kakade, S. Shalev-Shwartz, and A. Tewari. On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization. Technical report, TTI, 2009.

M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Müller, and A. Zien. Efficient and accurate ℓ_p-norm multiple kernel learning. In *Advances in Neural Information Processing Systems 22*, pages 997–1005, 2009.

G. Lanckriet, N. Cristianini, P. Bartlett, and L. E. Ghaoui. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

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

J. S. Nath, G. Dinesh, S. Ramanand, C. Bhattacharyya, A. Ben-Tal, and K. R. Ramakrishnan. On the algorithmics and applications of a mixed-norm based kernel learning formulation. In *Advances in Neural Information Processing Systems 22*, pages 844–852, 2009.

M. E. Nilsback and B. Caputo. Cue integration through discriminative accumulation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

T. Ojala, M. Pietikaäinen, and T. Maäenpaäää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

F. Orabona. *DOGMA: a MATLAB toolbox for Online Learning*, 2009. Software available at `http://dogma.sourceforge.net`.

F. Orabona and K. Crammer. New adaptive algorithms for online classification. In *Advances in Neural Information Processing Systems*, December 2010.

F. Orabona, L. Jie, and B. Caputo. Online-batch strongly convex multi kernel learning. In *Proc. of the 23rd IEEE Conference on Computer Vision and Pattern Recognition*, June 2010.

N. Pinto, D. D. Cox, and J. J. Dicarlo. Why is real-world visual object recognition hard? *PLoS Computational Biology*, 4(1), January 2008.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods – Support Vector Learning*, pages 185–208. MIT Press, 1999.

A. Pronobis, J. Luo, and B. Caputo. The more you learn, the less you store: Memory-controlled incremental SVM for visual place recognition. *Image and Vision Computing (IMAVIS), Special Issue on Online Pattern Recognition and Machine Learning Techniques for Computer-Vision: Theory and Applications*, 28(7):1080–1097, July 2010.

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

E. Rubinstein. Support vector machines via advanced optimization techniques. Technical report, Masters thesis, Faculty of Electrical Engineering, Technion, Nov 2005.

C. Sanderson and K. K. Paliwal. Identity verification using speech and face information. *Digital Signal Processing*, 14(5):449–480, 2004.

S. Shalev-Shwartz and Y. Singer. Logarithmic regret algorithms for strongly convex repeated games. Technical Report 2007-42, The Hebrew University, 2007.

S. Shalev-Shwartz and N. Srebro. SVM, optimization: inverse dependence on training set size. In *Proc. of the International Conference on Machine Learning*, 2008.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proc. of the International Conference on Machine Learning*, 2007.

S. Sonnenburg, G. Rätsch, and C. Schäfer. Learning interpretable SVMs for biological sequence classification. In *Research in Computational Molecular Biology, 9th Annual International Conference, RECOMB 2005, Cambridge, MA, USA, May 14-18, 2005, Proceedings*, volume 3500 of *Lecture Notes in Computer Science*, pages 389–407. Springer, 2005.

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

R. Tomioka and T. Suzuki. Sparsity-accuracy trade-off in MKL, 2010. URL `http://arxiv.org/abs/1001.2615`.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. of the International Conference on Machine Learning*, 2004.

O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on Riemannian manifolds. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *Proc. of the International Conference on Machine Learning*, 2009.

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*, December 2010.

D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

Z. Xu, R. Jin, I. King, and M. R. Lyu. An extended level method for efficient multiple kernel learning. In *Advances in Neural Information Processing Systems 21*, pages 1825–1832, 2008.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society*, 68:49–67, 2006.

A. Zien and C. S. Ong. Multiclass multiple kernel learning. In *Proc. of the International Conference on Machine Learning*, 2007.