Online-Batch Strongly Convex Multi Kernel Learning

Francesco Orabona[*]
Università degli Studi di Milano
orabona@dsi.unimi.it

Luo Jie
Idiap Research Institute
EPF Lausanne
jluo@idiap.ch

Barbara Caputo
Idiap Research Institute
bcaputo@idiap.ch

Abstract

Several object categorization algorithms use kernel methods over multiple cues, as they offer a principled approach to combine multiple cues, and to obtain state-of-the-art performance. A general drawback of these strategies is the high computational cost during training, that prevents their application to large-scale problems. They also do not provide theoretical guarantees on their convergence rate.

Here we present a Multiclass Multi Kernel Learning (MKL) algorithm that obtains state-of-the-art performance in a considerably lower training time. We generalize the standard MKL formulation to introduce a parameter that allows us to decide the level of sparsity of the solution. Thanks to this new setting, we can directly solve the problem in the primal formulation. We prove theoretically and experimentally that 1) our algorithm has a faster convergence rate as the number of kernels grow; 2) the training complexity is linear in the number of training examples; 3) very few iterations are enough to reach good solutions. Experiments on three standard benchmark databases support our claims.

1. Introduction

Categorization is one of the most challenging problems in computer vision today. Object categories present a wide visual variability within each class. This, coupled with robustness issues (*e.g.* changes in illumination, occlusion, clutter), makes it unclear how to build general models suitable for all categories. Because of this, a dominant approach is to learn instead what distinguishes them, by using highly discriminative and robust features combined with machine learning techniques [9, 10, 14, 17, 25, 26]. In particular this has been recently translated into Support Vector Machine (SVM) based classifiers combined with kernels over multiple cues [3, 9, 10, 14, 25, 26]. Results obtained by these methods on various benchmark databases represent the current state-of-the-art in object categorization. Among them, Multi Kernel Learning (MKL) approaches have attracted considerable attention [9, 14, 25]. However most emphasis has been put so far on their accuracy, and recent findings seem to indicate that current MKL algorithms do not improve much over the naive baseline of averaging all the kernels [10].

Almost every interesting categorization problems have more than two classes, and most of the MKL algorithms [13, 19, 23] solves the multiclass problem by decomposing it into multiple independent binary classification tasks (except [28]). However, recent evidence [10] seems to suggest that a principled multiclass formulation (such as those in [10, 24, 28]) achieves better performance, at least on sparse problems using l_1 regularization. Moreover, to our knowledge, none of the MKL algorithms [13, 19, 23] provides theoretical guarantees on their convergence rate. In practice, 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 current MKL algorithms are solved based on their dual representation, this might mean being stopped far from the optimal solution [11]. Last but not least, scalability is also very important for many real world applications.

The contribution of this paper is a Multiclass MKL algorithm that has a guaranteed and fast convergence rate to the optimal solution. We also generalize the MKL learning problem, adding a parameter to tune the level of sparsity in the kernel domain. We show experimentally that aiming at sparsity, as in the original MKL formulation, is not always the optimal strategy. 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 kernels used. At the same time, it achieves state-of-the-art performance on standard benchmark databases. The algorithm is based on a stochastic sub-gradient descent algorithm in the primal objective formulation. Minimizing the primal objective function directly results in a convergence rate that is faster and provable, rather than optimizing the dual objective. We show that by optimizing the primal objective function directly, we are able to solve the multiclass formulation effi-

[*] Work done while at Idiap Research Institute, Martigny, Switzerland

ciently, with a running time which is linear to the number of classes. We can stop the algorithm after few iterations, while still retaining a performance close to the optimal one. We call this algorithm OBSCURE, Online-Batch Strongly Convex mUlti keRnel lEarning.

1.1. Multiple Cues and Kernels

Consider the task of image classification with M classes, F different cues and N training instances $\{\boldsymbol{x}_i\}_{i=1}^N$ drawn from an unknown fixed probability distribution. We want to learn a score function $s(\cdot,\cdot)$ that best predicts the class $\hat{y}$ for any future sample $\boldsymbol{x}$ drawn from the same distribution, where the predicted class is the one with the highest score

$$\hat{y}(\boldsymbol{x}) = \operatorname*{argmax}_{y \in \mathbb{Y}} s(\boldsymbol{x}, y) \, . \tag{1}$$

This score function should be learned using all the F different cues, to gain robustness and performance.

Some of the methods addressing this task are based on a two-layers structure [10, 17]. A classifier is trained for each cue and then their outputs are combined by another classifier. Even if this strategy has recently received attention in the computer vision community, this kind of approach is the oldest and dates back to the seminal work of Wolpert [27]. They use Cross-Validation (CV) methods to create the training set for the second layer [10, 27]. Hence they have a runtime of about K+1 times the training of a single classifier, such as support vector machine (SVM), where K is the number of folds of the CV. This method is currently considered the state-of-art method for image classification tasks [10].

Another interesting strategy uses a one-layer architecture, such as the MKL [15, 19, 23, 25, 28]. Using the theory of *kernels*, one solves a joint optimization problem while also learning the optimal weights for combining the kernels, with each cue corresponding to a kernel. The optimization problem is similar in all these approaches. This approach is theoretically founded, plus it consists of a unique optimization problem. However solving it is more complex than training, *e.g.*, a single SVM classifier. Another issue is that current MKL approaches do not scale well to the number of training examples and number of classes. For example, the SILP algorithm [23, 28] depends 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.

From a theoretical point of view, if we consider a two-layers architecture with the first layer composed by kernel classifiers, and a linear classifier in the second stage, the two approaches are very similar. In both cases the final prediction function is written as

$$\hat{y}(\boldsymbol{x}) = \operatorname*{argmax}_{y \in \mathbb{Y}} \sum_{j=1}^{F} \beta_y^j s^j(\boldsymbol{x}, y), \tag{2}$$

where β_y^j are the weights learned by the one-layer or two-layers framework, and s^j is the score function for each kernel. Therefore the two formulations are essentially equivalent, with differences given only by the specific training procedures used. In both methods a regularizer that favors the selection of only a subset of the kernels is used [1, 10, 23, 25].

The main contribution of this paper is showing that the one-layer formulation, beside being more principled, can also achieve a comparable performance and a considerably lower training time than state-of-the-art two-layers architectures. We propose a p-norm version of the standard MKL algorithm, and we minimize it with a two stages 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. Differently from the other methods, our algorithm solves the optimization problem directly in the primal formulation, in both stages. Using recent approaches in optimization theory, the algorithm takes advantage of the abundance of information to reduce the training time [22]. 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.

The rest of the paper presents the theory and the experimental results supporting our claims. Section 2 revises the basic definitions of MKL and generalizes it to p-norm formulation. Section 3 presents the theory and algorithm of OBSCURE, while Section 4 reports experiments on categorization tasks.

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

Notations. Let $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$, with $N \in \mathbb{N}$, $\boldsymbol{x}_i \in \mathbb{X}$ and $y_i \in \mathbb{Y} = \{1, \cdots, M\}, M > 2$, be the training set. We indicate matrix and vectors with bold letters. A bar, *e.g.* $\bar{\boldsymbol{w}}$, denotes the vector formed by the concatenation of the F vectors $\boldsymbol{w}^j$, hence $\bar{\boldsymbol{w}} = [\boldsymbol{w}^1, \boldsymbol{w}^2, \cdots, \boldsymbol{w}^F]$.

Multi-class Classifier. A common approach to multiclass classification is to use joint feature maps $\phi(\boldsymbol{x}, y)$ on data $\mathbb{X}$ and labels $\mathbb{Y}$ [24]. The function s^j will be defined as

$$s^j(\boldsymbol{x}, y) = \boldsymbol{w}^j \cdot \phi^j(\boldsymbol{x}, y), \tag{3}$$

where $\boldsymbol{w}^j$ is a hyperplane[1]. The functions $\phi^j(\boldsymbol{x}, y)$ map the samples into a high, possibly infinite, dimensional

[1] For simplicity we will not use the bias, it can be easily added by modifying the kernel definition.

space. With multiple cues, we will have F functions $\phi^j(\cdot,\cdot), j = 1, \cdots, F$. This will also define F kernels $K^j((\boldsymbol{x}, y), (\boldsymbol{x}', y'))$ as $\phi^j(\boldsymbol{x}, y) \cdot \phi^j(\boldsymbol{x}', y')$. This definition includes the case of training M different hyperplanes, one for each class. Indeed $\phi^j(\boldsymbol{x}, y)$ can be defined as

$$\phi^j(\boldsymbol{x}, y) = [\mathbf{0}, \cdots, \mathbf{0}, \underbrace{\phi'^j(\boldsymbol{x})}_{y}, \mathbf{0}, \cdots, \mathbf{0}], \tag{4}$$

where $\phi'^j(\cdot)$ is a transformation that depends only on data. Similarly $\boldsymbol{w}$ will be composed by M blocks, $[\boldsymbol{w}^1, \cdots, \boldsymbol{w}^M]$. Hence, by construction, $\boldsymbol{w} \cdot \phi^j(\boldsymbol{x}, r) = \boldsymbol{w}^r \cdot \phi'^j(\boldsymbol{x})$. According to the defined notation, $\bar\phi(x, y) = [\phi^1(x, y), \cdots, \phi^F(x, y)]$.

Loss Function. We define a multi-class loss function [24]

$$\ell(\boldsymbol{w}, \boldsymbol{x}, y) = \max_{y' \neq y} |1 - \bar{\boldsymbol{w}} \cdot (\bar\phi(\boldsymbol{x}, y) - \bar\phi(\boldsymbol{x}, y'))|_+, \tag{5}$$

where $|t|_+$ is $\max(t, 0)$. This loss function is convex and it upper bounds the multi-class misclassification loss.

Norms and dual norms. A generic norm of a vector $\boldsymbol{w}$ is indicated by $\|\boldsymbol{w}\|$, its *dual norm* is indicated by $\|\boldsymbol{w}\|_*$. For $\boldsymbol{w} \in \mathbb{R}^d$ and $p \geq 1$, we denote by $\|\boldsymbol{w}\|_p$ the p-norm of $\boldsymbol{w}$, *i.e.*, $\|\boldsymbol{w}\|_p = (\sum_{i=1}^d |w_i|^p)^{1/p}$. 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.

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

$$\|\bar{\boldsymbol{w}}\|_{2,p} := \left\| \left[\|\boldsymbol{w}^1\|_2, \|\boldsymbol{w}^2\|_2, \cdots, \|\boldsymbol{w}^F\|_2 \right] \right\|_p, \tag{6}$$

that is the p-norm of the vector of F elements, formed by 2-norms of the vectors $\boldsymbol{w}^j$. The dual norm of $\|\cdot\|_{2,p}$ is $\|\cdot\|_{2,q}$ [12].

2.2. Multi Kernel Learning

The MKL optimization problem was first proposed in [1] and extended to multiclass in [28]. It can be written as

$$\begin{aligned} &\min_{\boldsymbol{w}_j} \ \frac{\lambda}{2} \left(\sum_{j=1}^F \|\boldsymbol{w}^j\|_2 \right)^2 + \frac{1}{N} \sum_{i=1}^N \xi_i \\ &\text{s.t.} \ \ \bar{\boldsymbol{w}} \cdot (\bar\phi(\boldsymbol{x}_i, y_i) - \bar\phi(\boldsymbol{x}_i, y)) \geq 1 - \xi_i, \forall i, y \neq y_i \, . \end{aligned} \tag{7}$$

This same formulation is used in [1, 23], while in [19] the proposed formulation is slightly different, although it is proved to be equivalent. Note that we 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. Our choice greatly simplifies the math of our algorithm. The two formulations are fully equivalent when setting $\lambda = \frac{1}{CN}$.

We will now generalize this formulation to group-norms. Using the notation defined above, we can rewrite (7) as

$$\min_{\bar{\boldsymbol{w}}} \ \frac{\lambda}{2} \|\bar{\boldsymbol{w}}\|_{2,1}^2 + \frac{1}{N} \sum_{i=1}^N \ell(\bar{\boldsymbol{w}}, \boldsymbol{x}_i, y_i) \, , \tag{8}$$

where $\bar{\boldsymbol{w}} = [\boldsymbol{w}^1, \boldsymbol{w}^2, \cdots, \boldsymbol{w}^F]$. 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 bring to a decrease in performance. Moreover the problem in (8) is not strongly convex [12], so its optimization algorithm is rather complex and its rate of convergence is usually slow [1, 23].

We propose to generalize the optimization problem, using a generic group norm

$$\min_{\bar{\boldsymbol{w}}} \ \frac{\lambda}{2} \|\bar{\boldsymbol{w}}\|_{2,p}^2 + \frac{1}{N} \sum_{i=1}^N \ell(\bar{\boldsymbol{w}}, \boldsymbol{x}_i, y_i) \, , \tag{9}$$

where $1 < p \leq 2$. We define $f(\bar{\boldsymbol{w}}) = \frac{\lambda}{2} \|\bar{\boldsymbol{w}}\|_{2,p}^2 + \frac{1}{N} \sum_{i=1}^N \ell(\bar{\boldsymbol{w}}, \boldsymbol{x}_i, y_i)$ and $\bar{\boldsymbol{w}}^*$ equals to the optimal solution of (9), $\bar{\boldsymbol{w}}^* = \arg\min_{\bar{\boldsymbol{w}}} f(\bar{\boldsymbol{w}})$. The added parameter p will allow us to decide the level of sparsity of the solution. In fact it is known that the 1-norm favors sparsity, and here the 1-norm favors a solution in which only few hyperplanes have a norm different from zero. Moreover this new formulation has the advantage of being λ/q-strongly convex [12]. 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 [12, 20]. 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 solving the problem in (7), but the strong convexity decreases. When p equals to 2, it is equivalent to using a single kernel equal to the sum of all the kernels. Recently a different p-norm MKL problem has been also proposed in [13], that allows non-sparse solutions. However, their algorithm did not take advantage of nice properties of the strong convexity for the optimization process. In the next section, we will show how to use the strong convexity to design a fast algorithm to solve (9).

3. The OBSCURE Algorithm

Our basic optimization tool is the framework developed in [20, 21]. 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 of the solution inside the space where the solution lives. The algorithm Pegasos,

Algorithm 1 OBSCURE stage 1 (online)

```
Input: $q$, $\eta$
Initialize: $\bar{\boldsymbol{\theta}}_1 = \mathbf{0}, \bar{\boldsymbol{w}}_1 = \mathbf{0}$
for $t = 1, 2, \ldots, T$ do
    Sample at random $(\boldsymbol{x}_t, y_t)$
    $\hat{y}_t = \operatorname*{argmax}_{y \neq y_t} \bar{\boldsymbol{w}}_t \cdot \bar{\phi}(\boldsymbol{x}_t, y)$
    $\bar{\boldsymbol{z}}_t = \bar{\phi}(\boldsymbol{x}_t, y_t) - \bar{\phi}(\boldsymbol{x}_t, \hat{y}_t)$
    if $\ell(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t) > 0$ then $\bar{\boldsymbol{\theta}}_{t+1} = \bar{\boldsymbol{\theta}}_t + \eta \bar{\boldsymbol{z}}_t$
    else $\bar{\boldsymbol{\theta}}_{t+1} = \bar{\boldsymbol{\theta}}_t$
    $\boldsymbol{w}^j_{t+1} = \frac{1}{q}\left(\frac{\|\boldsymbol{\theta}^j_{t+1}\|_2}{\|\bar{\boldsymbol{\theta}}_{t+1}\|_{2,q}}\right)^{q-2} \boldsymbol{\theta}^j_{t+1}, \ \forall j = 1, \cdots, F$
end for
return $\bar{\boldsymbol{\theta}}_{T+1}, \bar{\boldsymbol{w}}_{T+1}$
return $R = \sqrt{\|\bar{\boldsymbol{w}}_{T+1}\|^2_{2,p} + \frac{2}{\lambda N}\sum_{i=1}^N \ell\left(\bar{\boldsymbol{w}}_{T+1}, \boldsymbol{x}_i, y_i\right)}$
```

Algorithm 2 OBSCURE stage 2 (batch)

```
Input: $q$, $\bar{\boldsymbol{\theta}}_1$, $\bar{\boldsymbol{w}}_1$, $R$, $\lambda$
Initialize: $s_0 = 0$
for $t = 1, 2, \ldots, T$ do
    Sample at random $(\boldsymbol{x}_t, y_t)$
    $\hat{y}_t = \operatorname*{argmax}_{y \neq y_t} \bar{\boldsymbol{w}}_t \cdot \bar{\phi}(\boldsymbol{x}_t, y)$
    if $\ell(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t) > 0$ then $\bar{\boldsymbol{z}}_t = \bar{\phi}(\boldsymbol{x}_t, y_t) - \bar{\phi}(\boldsymbol{x}_t, \hat{y}_t)$
    else $\bar{\boldsymbol{z}}_t = \mathbf{0}$
    $d_t = \lambda t + s_{t-1}$
    $s_t = s_{t-1} + 0.5\left(\sqrt{d_t^2 + q\frac{(\frac{\lambda}{q}\|\bar{\boldsymbol{\theta}}_t\|_{2,q} + \|\bar{\boldsymbol{z}}_t\|_{2,q})^2}{R^2}} - d_t\right)$
    $\eta_t = \frac{q}{\lambda t + s_t}$
    $\bar{\boldsymbol{\theta}}_{t+\frac{1}{2}} = (1 - \frac{\lambda \eta_t}{q})\bar{\boldsymbol{\theta}}_t + \eta_t \bar{\boldsymbol{z}}_t$
    $\bar{\boldsymbol{\theta}}_{t+1} = \min\left(1, qR/\|\bar{\boldsymbol{\theta}}_{t+\frac{1}{2}}\|_{2,q}\right)\bar{\boldsymbol{\theta}}_{t+\frac{1}{2}}$
    $\boldsymbol{w}^j_{t+1} = \frac{1}{q}\left(\frac{\|\boldsymbol{\theta}^j_{t+1}\|_2}{\|\bar{\boldsymbol{\theta}}_{t+1}\|_{2,q}}\right)^{q-2} \boldsymbol{\theta}^j_{t+1}, \ \forall j = 1, \cdots, F$
end for
```

based on this framework, is the current state-of-art solver for linear SVM [21, 22].

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 [21, 22]. In particular the convergence rate, and hence the training time, would be proportional to $\frac{q}{\lambda}$. Although in general this convergence rate can be quite good, it becomes slow when λ is small and/or q is big. Moreover it is common knowledge that in many real-world problems, particularly in visual learning tasks, the best setting for λ is very small, or equivalently C is 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 [11], as their training time also depends on the value of the parameter C.

Do *et al*. [7] 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. In [7] the authors proposed an algorithm that estimates this bound while training, but it gives a speed-up only when the norm of the optimal solution $\bar{\boldsymbol{w}}^*$ 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 [7] to the generic non-Euclidean norms. Then we solve the problem of the upper bound of the norm of the optimal solution using an new online algorithm. This takes advantage of the characteristic of the MKL task and quickly converges to a solution close to the optimal one. Hence OBSCURE is composed by 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.

The following theorem[2] gives a theoretical guarantee on the convergence rate of OBSCURE to the solution of (9).

Theorem 1. *Suppose that* $\|\phi^j(\boldsymbol{x}_t, y_t)\|_2 \leq 1, \forall j = 1, \cdots, F,\ t = 1, \cdots, N$. *Let* $1 < p \leq 2$, $\delta \in (0, 1)$, R *the value returned by the first stage, and* $c = \sqrt{2}F^{1/q} + \lambda R$. *Then with probability at least* $1 - \delta$ *over the choices of the random samples we have that after* T *iterations of the 2nd stage of the OBSCURE algorithm, the difference between* $f(\bar{\boldsymbol{w}}_T)$ *and the optimal solution of* (9), $f(\bar{\boldsymbol{w}}^*)$, *is less than*

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

Moreover if the problem is linearly separable by a hyperplane $\bar{\boldsymbol{u}}$ *and the first stage is run until convergence,* R *is less than* $2(1 + \eta F^{\frac{2}{q}})\|\bar{\boldsymbol{u}}\|_{2,p}$.

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 $\mathcal{O}(\frac{q \log T}{\lambda T})$. If the second term is predominant, the convergence rate is $\mathcal{O}(\frac{R\sqrt{q \log T}}{\sqrt{T}})$, so it becomes independent from λ (*i.e*. independent from C). The algorithm will always optimally interpolate between these two different rates of convergence. As said before, the rate of convergence depends on p, through q. When p tends to 1, the solution tends to the sparse one of (7), with a worst rate. However in the experiment section we show that the best performance is not always given by the sparsest solution. Moreover Theorem 1

[2] For an extended version of this paper with proofs, see http://francesco.orabona.com/papers/obscure-proofs.pdf.

also shows that, when p is close to 1, 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 Section 3.2.

The training time of OBSCURE is proportional to the number of steps given by Theorem 1 multiplied by the complexity of each step. This in turn is dominated by the prediction (line 5 in Algorithms 1 and 2), that has complexity $\mathcal{O}(NFM)$. Note that this complexity is common to any other similar algorithm, and it can be reduced using methods like kernel caching [6].

In the following we introduce the necessary mathematical tools to be able to derive OBSCURE and its theorem.

3.1. Batch p-norm MKL

We first state a Lemma that is a generalization of Theorem 1 in [7] to general norms, using the framework in [20]. We need two additional definitions. Given a convex function $f : S \rightarrow \mathbb{R}$, its Fenchel conjugate $f^* : S \rightarrow \mathbb{R}$ is defined as $f^*(\boldsymbol{u}) = \sup_{\boldsymbol{v} \in S}(\boldsymbol{v} \cdot \boldsymbol{u} - f(\boldsymbol{v}))$. A vector $\boldsymbol{x}$ is a sub-gradient of a function f at $\boldsymbol{v}$, indicated with $\partial f(\boldsymbol{v})$, if $\forall \boldsymbol{u} \in S, f(\boldsymbol{u}) - f(\boldsymbol{v}) \geq (\boldsymbol{u} - \boldsymbol{v}) \cdot \boldsymbol{x}$.

Lemma 1. *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 $\|\boldsymbol{z}_t\|_* \leq L_t$. Then for any $\boldsymbol{u}$: $\|\boldsymbol{u} - \boldsymbol{w}_t\| \leq 2R$, and for any sequence of non-negative $\xi_1, \ldots, \xi_T$, Algorithm 3 achieves the following bound for all $T \geq 1$,*

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

With this Lemma we can now design stochastic sub-gradient algorithms. In particular, setting $\|\cdot\|_{2,p}$ as norm, $h(\bar{\boldsymbol{w}}) = \frac{q}{2}\|\bar{\boldsymbol{w}}\|_{2,p}^2$, and $g_t(\bar{\boldsymbol{w}}) = \frac{\lambda}{q}h(\bar{\boldsymbol{w}}) + \ell(\bar{\boldsymbol{w}}, \boldsymbol{x}_t, y_t)$, we obtain Algorithm 2 that solves the p-norm MKL problem in (9). In particular lines 6-7 correspond to the calculation of the sub-gradient of the multiclass loss function (5). The updates are done on the dual variables $\bar{\boldsymbol{\theta}}_t$, in lines 11-12, that are transformed into $\bar{\boldsymbol{w}}_t$ in line 13.

Note also 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 [21]. The parameter R is basically an upper bound on the norm of the optimal solution, *i.e.* $R \geq \|\bar{\boldsymbol{w}}^*\|_{2,p}$. In the next Section we show how to initialize this algorithm and to calculate R in an efficient way.

3.2. Initialization through an online algorithm

In Theorem 1 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.

Algorithm 3 Proximal projected sub-gradient descent

```
Input: $R$, $\sigma$, $\boldsymbol{w}_1 \in S$
Initialize: $s_0 = 0$
for $t = 1, 2, \ldots, T$ do
    Receive $g_t$
    $\boldsymbol{z}_t = \partial g_t(\boldsymbol{w}_t)$
    $s_t = s_{t-1} + \frac{\sqrt{(\alpha\sigma t + s_{t-1})^2 + \frac{\alpha L_t}{R^2}} - (\alpha\sigma t + s_{t-1})}{2}$
    $\eta_t = (\sigma t + s_t/\alpha)^{-1}$
    $\boldsymbol{w}_{t+1} = \nabla h^*(\nabla h(\boldsymbol{w}_t) - \eta_t \boldsymbol{z}_t)$
end for
```

We propose to initialize Algorithm 2 with an online algorithm. Algorithm 1 is the online version of problem (9) and it is derived using Corollary 7 in [12]. It is similar to the $2p$-norm matrix Perceptron in [4], but it overcomes the disadvantage of being used with the same kernel on each feature. As in [4], for Algorithm 1 it is possible to prove a relative mistake bound. We omit the details for lack of space, a future longer version of this work will include it.

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

$$\begin{aligned} R &:= \sqrt{\|\bar{\boldsymbol{w}}_{T+1}\|_{2,p}^2 + \tfrac{2}{\lambda N} \textstyle\sum_{i=1}^{N} \ell\left(\bar{\boldsymbol{w}}_{T+1}, \boldsymbol{x}_i, y_i\right)} \\ &\geq \sqrt{\|\bar{\boldsymbol{w}}^*\|_{2,p}^2 + \tfrac{2}{\lambda N} \textstyle\sum_{i=1}^{N} \ell\left(\bar{\boldsymbol{w}}^*, \boldsymbol{x}_i, y_i\right)} \geq \|\bar{\boldsymbol{w}}^*\|_{2,p} . \end{aligned} \tag{10}$$

So at any moment we can stop the algorithm and obtain an upper bound on $\|\bar{\boldsymbol{w}}^*\|_{2,p}$.

If the dimension of the space induced by the F kernles is big enough, it is very likely that the classification problem is linearly separable. When this is the case, we can prove that Algorithm 1 will converge to a solution which has null loss on each training sample, in a finite number of steps. More specifically we can state the following Theorem.

Theorem 2. *Suppose that $\|\phi^j(\boldsymbol{x}_t, y_t)\|_2 \leq 1, \forall j = 1, \cdots, F$, $t = 1, \cdots, N$, and $1 < p \leq 2$. If the problem (9) is linearly separable by a hyperplane $\bar{\boldsymbol{u}}$, then the Algorithm 1 will converge to a solution in a finite number of steps less than $2q(1/\eta + F^{\frac{2}{q}})\|\bar{\boldsymbol{u}}\|_{2,p}^2$. Moreover the returned value of R will be less than $2(1+\eta F^{\frac{2}{q}})\|\bar{\boldsymbol{u}}\|_{2,p}$.*

From the theorem it is clear the role of η: a bigger value will speed up the convergence, but will decrease the quality of the estimate of R. So η governs the trade-off between speed and precision of the first stage. If p is close to 1, the dependency on the number of kernels in this theorem is strongly sublinear, moreover increasing the number of kernels to $F' > F$, we have that $\|\bar{\boldsymbol{u}}\|_{2,p}^2$ will most likely decrease or remain constant. This means that we expect Algorithm 1 to converge, in a number of steps that is almost independent on F and in some cases even *decreasing* in 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 4.

4. Experiments

In this section we test OBSCURE on the Oxford flowers [18], Caltech-101 [8] and MNIST [16] datasets. Although our MATLAB implementation of the algorithm[3] 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 datasets containing large numbers of categories and lots of training samples. In all our experiments, the parameter η is fixed at 2, and 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 $C \in \{1, 10, 100, 1000\}$.

4.1. Oxford flowers

The Oxford flowers dataset [18] contains 17 different categories of flowers. Each class has 80 images with three predefined splits (train, validation and test). The authors also provide seven precomputed distance matrices[4]. These distance matrices are transformed into kernel using $\exp(-\gamma^{-1} \cdot d)$, where γ is the mean of the pairwise distances and d is the distance between two examples. We used a value of p equal to 1.05, found through CV.

We have implemented an extended version of the original Pegasos algorithm [21, 22] for problem (9). We first compare the running time performance between OBSCURE and Pegasos. Their generalization performance on the testing data (Figure 1(Left)) as well as the value of the objective function (Figure 1(Right)) are shown in Figure 1. In the same Figure, we also present the results obtained using other combination methods: SILP [23], SimpleMKL [19] and LP-β [10]. The cost parameter is selected from the range $C \in \{1, 10, 100, 1000\}$ for MKL methods. We see that OBSCURE converges much faster compared to Pegasos. This proves that, as stated in Theorem 1, OBSCURE has a better convergence rate than Pegasos. All the feature combination methods achieve similar results on this dataset. LP-β is order of magnitudes faster as it uses an efficient standard SVM solver [6].

4.2. Caltech-101 datasets

The Caltech-101 [8] dataset is a standard benchmark dataset for object categorization. Here we followed the experimental setup originally proposed and widely used in the literature. In our experiments, we used the pre-computed features and kernels of [10], with the same training and test split[5]. This allows us to compare against them directly. Following that, we report results using all 102 classes of the Caltech-101 dataset using five splits. There are five different image descriptors, using different setup of parameters and computed at different scales. It results in a total of 39 kernels. Note that, as they are derived from 5 features only, some of them might be redundant. For brevity, we omit the details of the features and kernels that can be found in [10].

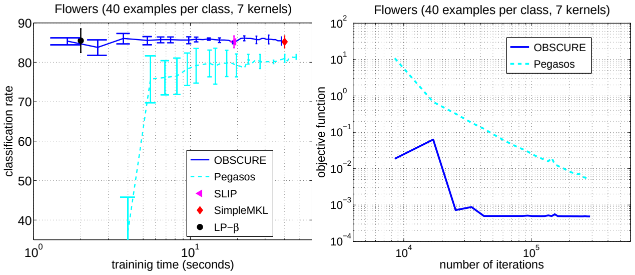

Figure 1. Comparison of performance on Oxford flowers dataset.

Figure 2 shows the behavior of our algorithm using different values of the parameter p (Figure 2(Left)), different number of kernels (Figure 2(middle)) and the running time under different size of training examples (Figure 2(right)). The dashed line in Figure 2(left & middle) corresponds to the results obtained by the first online stage of the OBSCURE algorithm. It can be observed from the figures that:

a). [Figure 2(Left)] The online step of OBSCURE achieves a performance close to the optimal solution in a training time order of magnitudes faster (10^1 to 10^3). When p is large (*i.e.* q is small) the online stage converges even faster. This is consistent with Theorem 2.

b). [Figure 2(Left)] By changing p, it is possible to improve performance. As stated before, when p tends to 1, the solution gets close to the sparse solution. In particular here 3 $\|\boldsymbol{w}^j\|_2$ (out of 4) approach 0. When p equals 2, we obtain a dense solution, that corresponds to use the sum of all the kernels. Although some of the kernels may contain redundant information, all of them may be informative for classification. Thus imposing sparsity on them does not always help increasing performance. Hence the optimal p here is $1.10 - 1.25$.

c). [Figure 2(Middle)] OBSCURE has a better converges rate when there are more kernels, as stated in Theorem 2. That is, the algorithm achieves a given accuracy in less iterations when more kernels are given.

d). [Figure 2(Right)] We can see that the algorithm converges quite fast to the optimal solution. Using 15 examples per class, the run time is similar to the runtime of LP-β (about 24 mins). When the number of training

[3] Code available at http://dogma.sourceforge.net/

[4] www.robots.ox.ac.uk/~vgg/research/flowers/

[5] www.vision.ee.ethz.ch/~pgehler/projects/iccv09/

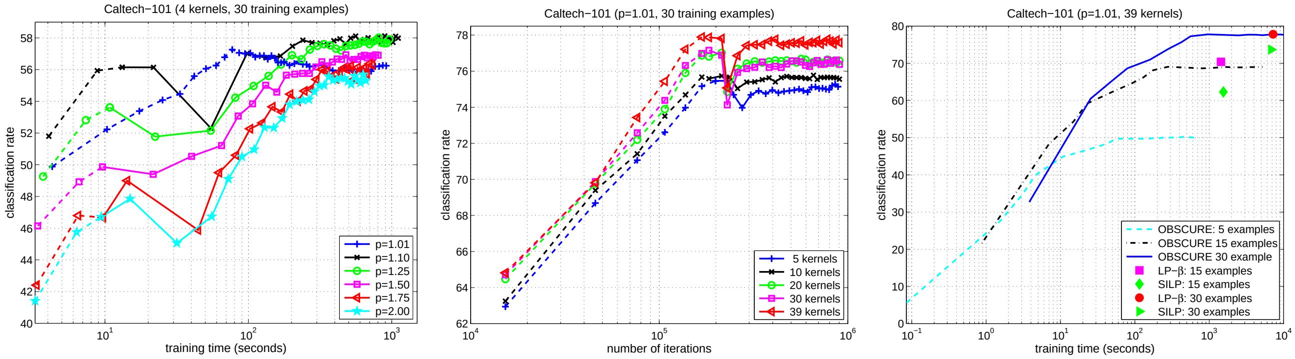

Figure 2. Behaviors of the OBSCURE algorithm on Caltech-101 dataset: (Left) the effect of different value of p, using four PHOG [3] kernels computed at different spatial pyramid level, as similar experiment performed in [10]; (Middle) the effect of different number of kernels randomly sampled from the 39 kernels; (Right) running time for different number of training examples using all the 39 kernels.

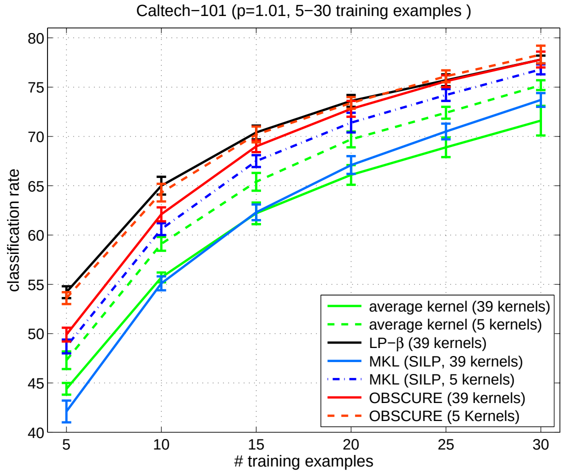

Figure 3. Performance comparison on Caltech-101 using different combination methods.

examples increases to 30, our algorithm has an advantage over LP-β, that takes about 121 mins.

In Figure 3 we report the results obtained using different combination methods. 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. We see also that our method achieves performance comparable to the state-of-art (LP-β, [10]), and outperforms the other MKL (SILP) methods. One possible reason may be the one-vs-all multiclass extension used in the MKL algorithm. The sparse MKL algorithm may choose different subset of kernels in different independent binary classification tasks, which may introduce a bias on some classes in the final decision process. However, note that although our algorithm obtains a solution close to the sparse one, it will never reach a completely sparse solution. This may be one of the reasons for the gap in performance between OBSCURE and LP-β [10]. However, this may not be critical, since usually in practice all used features/kernels are informative. Non-informative/duplicate features are unlikely to be included in a real system. We did a simple test by selecting five kernels from the five different families of features [10] which achieve low leave-one-out (LOO) error using 30 training examples per class. It can be done automatically using LS-SVM, which has a closed form solution for LOO error estimation [5]. The results as well as the performance of the averaging of these five kernels are also shown in Figure 3. We see that the algorithm improves slightly over the previous one. This suggests that OBSCURE as well as SILP, when provided with discriminative features, could increase performance even further. It also seems to indicate that there is a margin to improve the regularization used in MKL methods, as currently more kernels do not necessarily transform into better accuracy.

4.3. MNIST

In the last experiment we use the MNIST [16] dataset of handwritten digits. The dataset 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. Figure 4 shows the generalization performance on the test set achieved by OBSCURE over time, for various sizes of training set. We see that OBSCURE quickly converges to the best performance. It also shows that the time to reach the optimum is approximately linear in the number of training samples. The SVM performance using averaging kernel and the best kernel is 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

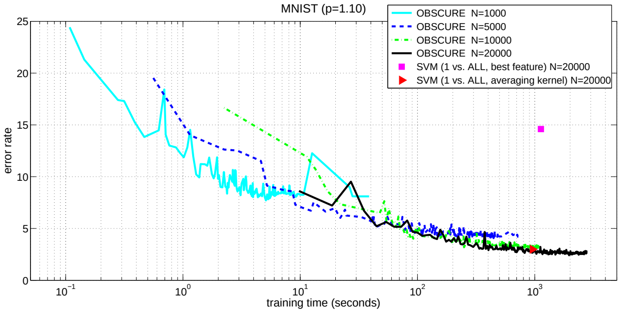

Figure 4. The generalization performance of MNIST dataset over different size of training samples.

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 efficiently.

5. 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 subgradient descent method. Experiments show that OBSCURE achieves state-of-art performance on multiclass classification problems. Furthermore, the solution found by the online stage is close to the optimal one for various tasks, while being computed several orders of magnitude faster. Our approach is general, hence it can be applied to any other algorithm with a strongly convex regularizer [12]. For example the framework can be very easily extended to solve other problems such as *structure output prediction* [24], to have an MKL algorithm for structured output.

OBSCURE has a faster convergence rate as the number of cues/kernels grows. Thus we expect to achieve better performance with more discriminative features. A simple feature selection technique such as cross-validation could already be beneficial. On the other hand, our results show that non-sparse models might get better performance (in the sense of accuracy and speed). This is in agreement with recent findings in [13]. As a last remark, we notice that the disadvantageous results of MKL methods, reported in [10], may be because those algorithms does not have a proper multiple class formulation for the object categorization problems. By using our method, MKL can still be an efficient machine learning tool for cue combination tasks.

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.

References

[1] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO, algorithm. In *Proc. ICML*, 2004. 2, 3

[2] P. Bartlett, E. Hazan, and A. Rakhlin. Adaptive online gradient descent. In *Proc. NIPS 20*, 2008. 10

[3] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proc. CIVR*, 2007. 1, 7

[4] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear algorithms for online multitask classification. In *Proc. COLT*, 2008. 5

[5] G. Cawley. Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs. In *Proc. IJCNN*, 2006. 7

[6] 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`. 5, 6

[7] C. B. Do, Q. V. Le, and C.-S. Foo. Proximal regularization for online and batch learning. In *Proc. ICML*, 2009. 4, 5, 9

[8] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE PAMI*, 28(4):594–611, 2004. 6

[9] P. Gehler and S. Nowozin. Let the kernel figure it out: Principled learning of pre-processing for kernel classifiers. In *Proc. CVPR*, 2009. 1

[10] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *Proc. ICCV*, 2009. 1, 2, 6, 7, 8

[11] D. Hush, P. Kelly, C. Scovel, and I. Steinwart. QP algorithms with guaranteed accuracy and run time for support vector machines. *JMLR*, 7, 2006. 1, 4

[12] 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. 3, 5, 8, 9, 10

[13] M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Müller, and A. Zien. Efficient and accurate lp-norm multiple kernel learning. In *Proc. NIPS*. 2009. 1, 3, 8

[14] A. Kumar and C. Sminchisescu. Support kernel machines for object recognition. In *Proc. ICCV*, 2007. 1

[15] G. Lanckriet, N. Cristianini, P. Bartlett, and L. E. Ghaoui. Learning the kernel matrix with semidefinite programming. *JMLR*, 5, 2004. 2

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998. 6, 7

[17] M. E. Nilsback and B. Caputo. Cue integration through discriminative accumulation. In *Proc. CVPR*, 2004. 1, 2

[18] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Proc. CVPR*, 2006. 6

[19] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. SimpleMKL,. *JMLR*, 9:2491–2521, November 2008. 1, 2, 3, 6

[20] S. Shalev-Shwartz and Y. Singer. Logarithmic regret algorithms for strongly convex repeated games. Technical Report 2007-42, The Hebrew University, 2007. 3, 5, 9

[21] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proc. ICML*, 2007. 3, 4, 5, 6, 9

[22] S. Shalev-Shwartz and N. Srebro. SVM optimization: inverse dependence on training set size. In *Proc. ICML*, 2008. 2, 4, 6

[23] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *JMLR*, 7, 2006. 1, 2, 3, 6

[24] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. ICML*, 2004. 1, 2, 3, 8

[25] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *Proc. ICCV*, 2007. 1, 2

[26] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proc. ICCV*, 2009. 1

[27] D. Wolpert. Stacked generalization. *Neural Networks*, 5(2), 1992. 2

[28] A. Zien and C. S. Ong. Multiclass multiple kernel learning. In *Proc. ICML*, 2007. 1, 2, 3

A. Appendix

In this appendix we report the proofs of the theorems in the paper.

A.1. Preliminary definitions and notations

We start by introducing some basic notions of convex analysis and a lemma contains the necessary relations to derive the update rules in Algorithm 1 and 2.

Given a convex function $f : S \to \mathbb{R}$, its Fenchel conjugate $f^* : S \to \mathbb{R}$ is defined as $f^*(\boldsymbol{u}) = \sup_{\boldsymbol{v} \in S}\big(\boldsymbol{v} \cdot \boldsymbol{u} - f(\boldsymbol{v})\big)$. A differentiable function $f : S \to \mathbb{R}$ is said to be λ-smooth with respect to a norm $\|\cdot\|$ iff for any $\boldsymbol{u}, \boldsymbol{v} \in S$, $f(\boldsymbol{u} + \boldsymbol{v}) \le f(\boldsymbol{u}) + \nabla f(\boldsymbol{u}) \cdot \boldsymbol{v} + \frac{\lambda}{2}\|\boldsymbol{v}\|^2$. We also introduce a notation that will help us to synthesize the following formulas. We indicate by $[w^j]_1^F := \left[w^1, w^2, \cdots, w^F\right]$.

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

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

Proof. The first relation can be found in the proof of Theorem 20 in [12]. The second can be obtained differentiating h. The last relation is obtained using Lemma 2 in [20]. □

Now we can derive the following Corollary, that allows us to compute all the quantities involved in Algorithm 1 and 2.

Corollary 1. *The following relations holds for Algorithm 1 and 2, for any t:*

- $\bar{\boldsymbol{\theta}}_t = q\left[\left(\frac{\|\boldsymbol{w}_t^j\|_2}{\|\bar{\boldsymbol{w}}_t\|_{2,p}}\right)^{p-2}\boldsymbol{w}_t^j\right]_1^F = \frac{q}{2}\nabla\|\bar{\boldsymbol{w}}_t\|_{2,p}^2$
- $\bar{\boldsymbol{w}}_t = \frac{1}{q}\left[\left(\frac{\|\boldsymbol{\theta}_t^j\|_2}{\|\bar{\boldsymbol{\theta}}_t\|_{2,q}}\right)^{q-2}\boldsymbol{\theta}_t^j\right]_1^F$
- $\|\bar{\boldsymbol{w}}_t\|_{2,p} = \frac{1}{q}\|\bar{\boldsymbol{\theta}}_t\|_{2,q}$
- $\left\|\partial(\frac{\lambda}{2}\|\bar{\boldsymbol{w}}_t\|_{2,p}^2 + \ell(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t))\right\|_{2,q}$
  $\le \frac{\lambda}{q}\|\bar{\boldsymbol{\theta}}_t\|_{2,q} + \|\bar{\boldsymbol{z}}_t\|_{2,q}$
- $\ell(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t) > 0$
  $\Rightarrow \|\bar{\boldsymbol{z}}_t\|_{2,q} = \|\bar{\phi}(\boldsymbol{x}_t, y_t) - \bar{\phi}(\boldsymbol{x}_t, \hat{y}_t)\|_{2,q}$
  $\le \sqrt{2\max_{i,j} K^j(\boldsymbol{x}_i, \boldsymbol{x}_i)}F^{1/q}$

A.2. Proof of Theorem 1

Proof. Define $g_t(\bar{\boldsymbol{w}}) = \frac{\lambda}{2}\|\bar{\boldsymbol{w}}\|_{2,p}^2 + \ell(\bar{\boldsymbol{w}}, \boldsymbol{x}_t, y_t)$ and $h(\bar{\boldsymbol{w}}) = \frac{q}{2}\|\bar{\boldsymbol{w}}\|_{2,p}^2$. Using Lemma 1 in [20], we can see that these two functions satisfy the hypothesis of Lemma 1, with $\alpha = q$, $\sigma = \frac{\lambda}{q}$. So we have

$$\sum_{t=1}^{T}\left(g_t(\bar{\boldsymbol{w}}_t) - g_t(\bar{\boldsymbol{w}}^*)\right) \tag{11}$$

$$\le \min_{\xi_1,\cdots,\xi_T}\sum_{t=1}^{T}\left[4\xi_t R^2 + \frac{qc^2}{\lambda t + \sum_{i=1}^{t}\xi_i}\right] \tag{12}$$

Reasoning as in [21], we divide by T, take the expection on both side and use the Markov's inequality. So we obtain that we probability at least $1 - \delta$

$$f(\bar{\boldsymbol{w}}_T) - f(\bar{\boldsymbol{w}}^*) \tag{13}$$

$$\le \min_{\xi_1,\cdots,\xi_T}\frac{1}{\delta T}\sum_{t=1}^{T}\left[4\xi_t R^2 + \frac{qc^2}{\lambda t + \sum_{i=1}^{t}\xi_i}\right]. \tag{14}$$

Setting all the ξ_i to the same value ξ, the last term in the last equation can be upper bounded by

$$A_T = \min_{\xi}\frac{1}{\delta}\left[4\xi R^2 + \frac{1}{T}\sum_{t=1}^{T}\frac{qc^2}{t(\lambda + \xi)}\right] \tag{15}$$

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

A.3. Proof of Lemma 1

Proof. Define $g_t'(\boldsymbol{w}) = g_t(\boldsymbol{w}) + \frac{\xi_t}{2}\|\boldsymbol{w} - \boldsymbol{w}_t\|^2$. Using the assumptions of this Lemma, we have that g_t' is $(\sigma + \frac{\xi_t}{\alpha})$-strongly convex w.r.t. to h. Moreover we have that $\partial g_t'(\boldsymbol{w}_t) = \partial g_t(\boldsymbol{w}_t) = \sigma\nabla h(\boldsymbol{w}) + \partial f_t(\boldsymbol{w})$, because the gradient of the proximal regularization term is zero when evaluated at $\boldsymbol{w}_t$ [7]. Hence we can apply Theorem 1 from [20] to have

$$\sum_{t=1}^{T} g_t(\boldsymbol{w}_t) - \sum_{t=1}^{T}\left(g_t(\boldsymbol{u}) + \frac{\xi_t}{2}\|\boldsymbol{u} - \boldsymbol{w}_t\|^2\right) \tag{16}$$

$$= \sum_{t=1}^{T} g_t'(\boldsymbol{w}_t) - \sum_{t=1}^{T} g_t'(\boldsymbol{u}) \tag{17}$$

$$\le \frac{1}{2}\sum_{t=1}^{T}\frac{L_t^2}{\sigma t + \frac{\sum_{i=1}^{t}\xi_i}{\alpha}}. \tag{18}$$

Using the hypothesis of this Lemma we obtain

$$\sum_{t=1}^{T} g_t(\boldsymbol{w}_t) - \sum_{t=1}^{T} g_t(\boldsymbol{u}) \tag{19}$$

$$\leq \frac{1}{2}\sum_{t=1}^{T}\left(\xi_t\|\boldsymbol{u}-\boldsymbol{w}_t\|^2 + \frac{\alpha L_t^2}{\alpha\sigma t + \sum_{i=1}^{t}\xi_i}\right) \tag{20}$$

$$\leq \frac{1}{2}\sum_{t=1}^{T}\left(4\xi_t R^2 + \frac{\alpha L_t^2}{\alpha\sigma t + \sum_{i=1}^{t}\xi_i}\right) . \tag{21}$$

Using the definition of ξ_t in the algorithm and Lemma 3.1 in [2], we have the bound. □

A.4. Proof of Theorem 2

Proof. The proof is based on an adaptation of a result from [12].

Denote by $\mathcal{U}$ the set of rounds in which there is an update, and by U its cardinality. We proceed by bounding the quantity $\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}}$ from below and from above. Define $h(\boldsymbol{v}) = \frac{q}{2}\|\boldsymbol{v}\|_{2,p}^2$. From [12, Corollary 19], we know that h is 1-smooth w.r.t. $\|\cdot\|_{2,q}$. Moreover, line 9 in the algorithm's pseudo-code implies that $\bar{\boldsymbol{w}}_t = \nabla h^*(\bar{\boldsymbol{\theta}}_t) = \nabla h^*\big(\sum_{i=1}^{t-1}\eta\bar{\boldsymbol{z}}_i\big)$. Hence, we obtain

$$\|\bar{\boldsymbol{\theta}}_{T+1}\|_{2,q}^2 \leq \|\bar{\boldsymbol{\theta}}_T\|_{2,q}^2 + 2q\eta\bar{\boldsymbol{w}}_T \cdot \bar{\boldsymbol{z}}_T + q\eta^2\|\bar{\boldsymbol{z}}_T\|_{2,q}^2 \tag{22}$$

$$\leq q\sum_{t\in\mathcal{U}}\big(2\eta\bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t + \eta^2\|\bar{\boldsymbol{z}}_t\|_{2,q}^2\big) . \tag{23}$$

Using the convex inequality for norms we then get

$$\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}} \leq \|\bar{\boldsymbol{\theta}}_{T+1}\|_{2,q}\,\|\bar{\boldsymbol{u}}\|_{2,p} \tag{24}$$

$$\leq \|\bar{\boldsymbol{u}}\|_{2,p}\sqrt{q\sum_{t\in\mathcal{U}}\big(2\eta\bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t + \eta^2\|\bar{\boldsymbol{z}}_t\|_{2,q}^2\big)} . \tag{25}$$

We can further bound the last term by considering that when an update is performed $\bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t$ is less than 1. Using that $\|\bar{\boldsymbol{z}}_t\|_{2,q}^2 \leq 2F^{2/q}$ we can further upper bound $\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}}$ as follows

$$\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}} \leq \|\bar{\boldsymbol{u}}\|_{2,p}\eta\sqrt{2q(1/\eta + F^{2/q})U} . \tag{26}$$

For the lower bound we have that

$$\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}} = \sum_{t\in\mathcal{U}}\eta\bar{\boldsymbol{u}} \cdot \bar{\boldsymbol{z}}_t \tag{27}$$

$$= \sum_{t\in\mathcal{U}}\eta\bar{\boldsymbol{u}} \cdot \big(\bar{\phi}(\boldsymbol{x}_t, y_t) - \bar{\phi}(\boldsymbol{x}_t, \hat{y}_t)\big)$$

$$\geq \sum_{t\in\mathcal{U}}\eta\big(1 - \ell(\bar{\boldsymbol{u}}, \boldsymbol{x}_t, y_t)\big) \tag{28}$$

$$\geq \eta U, \tag{29}$$

where in the last step we used the fact that the problem is linearly separable by $\bar{\boldsymbol{u}}$. Combining this last inequality with (26), solving the inequality for U, we obtain

$$U \leq 2q\|\bar{\boldsymbol{u}}\|_{2,p}^2(1/\eta + F^{2/q}) . \tag{30}$$

For the second part of the theorem, using fifth result in Corollary 1 in (22) we have

$$\|\bar{\boldsymbol{\theta}}_{T+1}\|_{2,q} \leq \eta\sqrt{q2U(1/\eta + F^{2/q})}, \tag{31}$$

and using (30), we have

$$\|\bar{\boldsymbol{\theta}}_{T+1}\|_{2,q} \leq \eta\sqrt{4q^2\|\bar{\boldsymbol{u}}\|_{2,p}^2(1/\eta + F^{2/q})^2} \tag{32}$$

$$= 2q\|\bar{\boldsymbol{u}}\|_{2,p}(1 + \eta F^{2/q}) . \tag{33}$$

Using the relation $\|\bar{\boldsymbol{w}}_t\|_{2,p} = \frac{1}{q}\|\bar{\boldsymbol{\theta}}_t\|_{2,q}$, that holds for any t, we have the stated bound.

□