IDIAP RESEARCH REPORT

OM-2: AN ONLINE MULTI-CLASS MULTI-KERNEL LEARNING ALGORITHM

Jie Luo Francesco Orabona Marco Fornoni
Barbara Caputo Nicolo Cesa-Bianchi

Idiap-RR-06-2010

APRIL 2010

Centre du Parc, Rue Marconi 19, P.O. Box 592, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

OM-2: An Online Multi-class Multi-kernel Learning Algorithm

Luo Jie*† , Francesco Orabona‡ , Marco Fornoni*† , Barbara Caputo* , Nicolò Cesa-Bianchi‡
*Idiap Research Institute, 1920 Martigny, Switzerland
†Swiss Federal Institute of Technology in Lausanne (EPFL), 1015 Lausanne, Switzerland
‡DSI, Università degli Studi di Milano, 20135 Milan, Italy
jluo@idiap.ch, francesco@orabona.com, mfornoni@idiap.ch
bcaputo@idiap.ch, cesa-bianchi@dsi.unimi.it

Abstract

Efficient learning from massive amounts of information is a hot topic in computer vision. Available training sets contain many examples with several visual descriptors, a setting in which current batch approaches are typically slow and does not scale well. In this work we introduce a theoretically motivated and efficient online learning algorithm for the Multi Kernel Learning (MKL) problem. For this algorithm we prove a theoretical bound on the number of multiclass mistakes made on any arbitrary data sequence. Moreover, we empirically show that its performance is on par, or better, than standard batch MKL (e.g. SILP, SimpleMKL) algorithms.

1. Introduction

A common trend in computer vision research is the exploitation of massive amounts of information to achieve a performance increase. In particular, learning from huge collections of labeled images collected from the Web, and using multiple discriminative features, has led to a significant boost of performance on problems considered very hard for many years (for example, see [8, 11, 27] and references therein). Yet, current classification *batch* algorithms, such as Support Vector Machines (SVM), do not scale well with the training set size. For this reason, researchers are constantly looking for new ways of learning efficiently from large-scale datasets.

A further issue hampering the application of batch learning to vision is that many computer vision problems are intrinsically *sequential*; *e.g.*, applications with "human in the loop" in robotic vision [10, 16] and computer vision [14]. In these problems the system is primed on a small dataset, and then keeps updating its current model as more data are acquired (often through an interaction with the user). This is in contrast with batch algorithms, where updating the model often means re-training from scratch.

A framework for learning algorithms designed with a clear focus on incremental learning and scalability is *online learning* [4]. The algorithms developed in the online learning framework are designed so that their current model can be efficiently updated after each sample is received. The time needed for each update is small, typically at most *linear* in the number of observed examples. In online learning there is no distinction between a training and a testing phase: learning proceeds sequentially and the knowledge is continuously exploited and updated. More specifically, at each time step the system is presented with a new input vector and uses the current model to predict the associated label. The correct label is then revealed, and the system can update the model based on the received feedback. The goal is to minimize the number of prediction mistakes, regardless of the mechanism generating the samples. For this reason the online learning framework is suitable to model situations in which that i.i.d. assumption on the samples is not realistic. Yet, in case the data are really i.i.d., it is still possible to have algorithms with the same (or better) statistical guarantees as the batch algorithms. In this sense, the online learning scenario is both harder and more general than the batch one.

Online learning algorithms have been successfully applied to many computer vision tasks where the amount of data is so big [5, 13] that standard batch algorithms would have unacceptably long training times. On the other hand, the practical performance of online learning algorithms is usually lower compared to batch methods. Ideally, we would like to get best of both worlds, *i.e.*, high performance and fast computation w.r.t. sample size.

The current state-of-the-art methods for object categorization are based on the combination of multiple discriminative and robust features via machine learning techniques [11, 18, 26, 27]. Among these approaches, Multi Kernel Learning (MKL) [18, 26, 27] has attracted considerable attention. However, even if many attempts have been

made to speed up the training process [21, and references therein], this approach is still slow and does not scale to big datasets. Some recent work has tried to address this problem using online learning method. Cavallanti et. al. [3] proposed a p-norm multiview Perceptron. However, they assume that all the cues live in the same space, meaning that same kernel must be used on all the cues. Jie et. al. [15] present a wrapper algorithm using a two-layer structure, which can use most of the known online learning methods as base algorithms. The specific version proposed in [15] has a bounded memory complexity, but there is neither mistake bound nor convergence guarantee.

In this paper, we first introduce a generalization of the multi-class MKL learning problem that allows to tune the level of sparsity of the solution in the domain of the kernels. This enable us to design a theoretically motivated online learning algorithm, which we call OM-2 for Online Multiclass Multi-kernel, with a guaranteed mistake bound on any individual data sequence. OM-2 achieves state-of-the-art performance for online algorithms on standard benchmark databases. We also show that in the traditional train/test setup, OM-2 achieves a performance close to MKL batch algorithms while requiring an update time linear in the number of past examples.

In the next section we describe the online learning framework and the building blocks that we use in our online MKL architecture (Section 2 and 3). Section 4 describes our experimental findings. Section 5 contains the conclusions.

2. Preliminaries

In this section we first formally define the multiclass online learning problem and the p-norm MKL framework. We start by introducing some basic notions of classification and convex analysis.

In the following bold letters are used to denote the elements of a linear space (*e.g.*, vectors $\boldsymbol{x}$ in $\mathbb{R}^d$). Let $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T)$ be a sequence of examples (instances-label pairs), with $(\boldsymbol{x}_t, y_t) \in \mathbb{X} \times \mathbb{Y}$, where $y_t \in \mathbb{Y} = \{1, \ldots, M\}$ and $M \geq 2$.

Multi-class classification. We consider multiclass classifiers of the form

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

where $s(\boldsymbol{x}, y)$ is the value of the score function when instance $\boldsymbol{x}$ is assigned to class y. Hence, the predicted class $\hat{y}(\boldsymbol{x})$ in (1) is the class achieving the highest score.

We use linear[1] score functions $s(\boldsymbol{x}, y) = \boldsymbol{w} \cdot \phi(\boldsymbol{x}, y)$, where $\boldsymbol{w}$ are the linear parameters and ϕ is a kernel feature map jointly defined on $\mathbb{X} \times \mathbb{Y}$ —see, *e.g.*, [10]. The function ϕ maps the examples into a reproducing kernel Hilbert space (of arbitrarily high dimensionality) whose inner product is implemented by the associated kernel, $K\big((\boldsymbol{x}, y), (\boldsymbol{x}', y')\big) = \phi(\boldsymbol{x}, y) \cdot \phi(\boldsymbol{x}', y')$. With multiple cues, we have F feature maps $\phi^1, \ldots, \phi^F$ and F kernels $K^1, \ldots, K^F$, where $K^i\big((\boldsymbol{x}, y), (\boldsymbol{x}', y')\big) = \phi^i(\boldsymbol{x}, y) \cdot \phi^i(\boldsymbol{x}', y')$ for each $i = 1, \ldots, F$. In our multiclass setup, for each $y = 1, \ldots, M$ we define

$$\phi^j(\boldsymbol{x}, y) = \big(\mathbf{0}, \ldots, \mathbf{0}, \underbrace{\psi^j(\boldsymbol{x})}_{y}, \mathbf{0}, \ldots, \mathbf{0}\big) \tag{2}$$

where $\psi^j(\cdot)$ is a label-independent feature map. Similarly, $\boldsymbol{w}$ consists of M blocks, $\big(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M\big)$. Hence, by construction, $\boldsymbol{w} \cdot \phi^j(\boldsymbol{x}, y) = \boldsymbol{w}_y \cdot \psi^j(\boldsymbol{x})$.

Loss Function. We introduce the following "bar" notation $\bar{\phi}(x, y) = \big(\phi^1(x, y), \ldots, \phi^F(x, y)\big)$ and, similarly, $\bar{\boldsymbol{w}} = \big(\boldsymbol{w}^1, \boldsymbol{w}^2, \ldots, \boldsymbol{w}^F\big)$, where each $\boldsymbol{w}^j$ is a M-block vector $\big(\boldsymbol{w}^j_1, \ldots, \boldsymbol{w}^j_M\big)$. Consider the following multiclass loss function [10]

$$\ell\left(\bar{\boldsymbol{w}}, \boldsymbol{x}, y\right) = \max_{y' \neq y} \left|1 - \bar{\boldsymbol{w}} \cdot (\bar{\phi}(\boldsymbol{x}, y) - \bar{\phi}(\boldsymbol{x}, y'))\right|_+ \tag{3}$$

where $|t|_+ = \max\{t, 0\}$. This loss function is convex and upper bounds the multiclass misclassification loss.

p-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 $\boldsymbol{w}$, where $\|\boldsymbol{w}\|_p = \left(\sum_{i=1}^d |w_i|^p\right)^{1/p}$. The *dual norm* of $\|\cdot\|_p$ is $\|\cdot\|_q$, where p and q are dual coefficients; *i.e.*, they satisfy $1/p + 1/q = 1$. In the following, p and q always denote a pair of dual coefficients.

Group Norm. We can define the $(2, p)$-*group norm* $\|\bar{\boldsymbol{w}}\|_{2,p}$ of $\bar{\boldsymbol{w}}$ as

$$\|\bar{\boldsymbol{w}}\|_{2,p} := \Big\| \big(\|\boldsymbol{w}^1\|_2, \|\boldsymbol{w}^2\|_2, \ldots, \|\boldsymbol{w}^F\|_2\big) \Big\|_p . \tag{4}$$

This is the p-norm of the vector whose elements are the 2-norms of the vectors $\boldsymbol{w}^1, \ldots, \boldsymbol{w}^F$. Note that the dual norm of $\|\cdot\|_{2,p}$ is $\|\cdot\|_{2,q}$ [17].

Sub-gradients and Fenchel conjugate. Given a convex function $h : \mathbb{X} \to \mathbb{R}$, its sub-gradient $\partial h(\boldsymbol{v})$ at $\boldsymbol{v}$ satisfies: $\forall \boldsymbol{u} \in \mathbb{X}, h(\boldsymbol{u}) - h(\boldsymbol{v}) \geq (\boldsymbol{u} - \boldsymbol{v}) \cdot \partial h(\boldsymbol{v})$. The Fenchel conjugate of h, $h^* : S \to \mathbb{R}$, is defined as $h^*(\boldsymbol{u}) = \sup_{\boldsymbol{v} \in S}\big(\boldsymbol{v} \cdot \boldsymbol{u} - h(\boldsymbol{v})\big)$.

2.1. Multi Kernel Learning (MKL)

The MKL optimization problem was first proposed in [1] and then extended to multiclass classification in [28]. Using

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

the group norm notation introduced above, we can define the MKL [1] problem as

$$\min_{\bar{\boldsymbol{w}}} \quad \frac{\lambda}{2}\|\bar{\boldsymbol{w}}\|_{2,1}^2 + \sum_{t=1}^{T} \ell\left(\bar{\boldsymbol{w}}, \boldsymbol{x}_t, y_t\right) \tag{5}$$

where, as usual, $\bar{\boldsymbol{w}} = \left(\boldsymbol{w}^1, \boldsymbol{w}^2, \ldots, \boldsymbol{w}^F\right)$. The same formulation is used in [25] —see also [21] for equivalent formulations. The $(2,1)$ group norm induces sparsity in the domain of the kernels. This means that the solution of (5) tends to depend on a small subset of the F kernels.

Even if enforcing sparsity is desirable in many applications, in some cases it could lead to a decrease in performance. For this reason, problem (5) has been recently generalized to p-norms in [19]. This allows to finely tune the level of sparsity needed for the task at hand. In fact, the authors show that in computational biology problems MKL achieves higher accuracies using $(2,p)$ group norms with $p > 1$ [19]. In this paper we propose to consider the p-norm problem in the following formulation

$$\min_{\bar{\boldsymbol{w}}} \quad \frac{\lambda}{2}\|\bar{\boldsymbol{w}}\|_{2,p}^2 + \sum_{t=1}^{T} \ell\left(\bar{\boldsymbol{w}}, \boldsymbol{x}_t, y_t\right) \tag{6}$$

where $1 < p \leq 2$. This directly generalizes (5) and is different from the one proposed in [19]. The advantage of (6) is that the objective function is λ/q-strongly convex [17]. Intuitively, strong convexity measures the minimum curvature of a function. If a problem has a strongly convex objective function, then it is easier to optimize it. Therefore strong convexity is a key property for the design of fast online and batch algorithms [23, 17]. The added parameter p allow us to decide the level of sparsity of the solution. 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. When p tends to 1, the solution of (6) gets close to the sparse solution obtained by solving (5), but the strong convexity vanishes. Setting p equals to 2 corresponds to using the unweighted sum of the kernels.

2.2. Online Learning

An online algorithm learns by observing the training examples $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_T, y_T)$ in a sequential manner. At each round $t = 1, 2, \ldots, T$, the algorithm receives the next instance $\boldsymbol{x}_t$ and makes a prediction $\hat{y}_t$. Once the algorithm has made its prediction, it observes the correct label y_t. Based on the label information the algorithm may update its current model, thus improving the chances of making accurate predictions in the subsequent rounds. The quality of the predictions is measured by a given loss function, and the algorithm's goal is to minimize the cumulative loss suffered along the sequence of observations.

Note that, contrary to the standard batch learning framework, in online learning there are no stochastic assumption on the data-generating process. In other words, the algorithm's performance is measured on each individual sequence of observations, rather than only on those sequences that are typical with respect to some underlying probability distribution. For this reason the performance of online algorithms is usually compared to that of the fixed classifier performing best on the entire observed data sequence.

An approach to solve the online learning problem is to use the "Follow the Regularized Leader" framework [22, 17]. This corresponds to using at each step the solution of the optimization problem

$$\bar{\boldsymbol{w}}_{t+1} = \operatorname*{argmin}_{\bar{\boldsymbol{w}}} \; h(\bar{\boldsymbol{w}}) + \bar{\boldsymbol{w}} \cdot \eta \sum_{i=1}^{t} \partial\ell(\bar{\boldsymbol{w}}_i, \boldsymbol{x}_i, y_i) \tag{7}$$

where $h(\bar{\boldsymbol{w}})$ is the regularizer and $\eta > 0$ is a parameter. Intuitively, this amounts to solving at each step a problem similar to (6), the difference being that the loss function has been replaced by its subgradient. The linearization of the loss function through the subgradient provides an efficient closed formula update and allows to prove regret bounds [22, 17]. The solution of the above minimization problem gives an update of the form

$$\bar{\boldsymbol{w}}_{t+1} = \nabla h^* \left(-\eta \sum_{i=1}^{t} \partial\ell(\bar{\boldsymbol{w}}_i, \boldsymbol{x}_i, y_i) \right) \tag{8}$$

where h^* is the Fenchel conjugate of h.

Next, we propose a variant of "Follow the Regularized Leader" in which the parameter $\eta = \eta_t$ is changed at each time step. We prove that the cumulative number of mistakes made on any sequence of T observations is roughly equal to the optimum value of the MKL problem (6).

3. Algorithm

In this Section we describe the OM-2 algorithm (pseudocode in Algorithm 1) and analyze its performance on any arbitrary sequence of observations. It is important to point out that even if we solve the problem in the primal, we do use nonlinear kernels without computing the nonlinear mapping $\bar{\phi}(\boldsymbol{x}, y)$ explicitly. In other words, OM-2 can be used with any kernel function satisfying Mercers conditions. We briefly outline the approach for implementing it in Section 3.1.

The design and analysis of this algorithm is based on the framework and machinery developed in [17]. It is similar to the p-norm matrix Perceptron of [3], but it overcomes the disadvantage of using the same kernel on each feature. The regularizer of our online MKL problem is defined by $h(\bar{\boldsymbol{w}}) = \frac{q}{2}\|\bar{\boldsymbol{w}}\|_{2,p}^2$. It can be verified that

$$h^*(\bar{\boldsymbol{\theta}}) = \frac{1}{2q}\|\bar{\boldsymbol{\theta}}\|_{2,q}^2$$

$$\nabla h^*(\boldsymbol{\theta}^j) = \frac{1}{q}\left(\frac{\|\boldsymbol{\theta}^j\|_2}{\|\bar{\boldsymbol{\theta}}\|_{2,q}}\right)^{q-2} \boldsymbol{\theta}^j, \quad \forall j = 1, \ldots, F.$$

Similarly to [3], we prove a mistake bound for Algorithm 1. We say that the algorithm makes a "mistake" each time the prediction $\tilde{y}$ is different from the true class y_t. Algorithm OM-2 updates the weight $\bar{\boldsymbol{w}}_t$ not only at each mistake, but also when the prediction is correct and the multiclass loss $\ell(\bar{\boldsymbol{w}}_t, x_t, y_t)$, is greater than 0. In the following, we show that this aggressive update strategy improves the theoretical performance of the algorithm.

We denote by I the set of rounds in which there is no mistake but there is an update; that is,

$$I = \left\{t \,:\, 0 < \bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t < 1\right\}.$$

Theorem 1. *Let $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T)$ be a sequence of examples where $\boldsymbol{x}_t \in \mathbb{X}$, $y \in \mathbb{Y}$ and such that $\left\|\phi^j(\boldsymbol{x}_t, y_t)\right\|_2 \leq 1$ for $j = 1, \ldots, F$ and $t = 1, \ldots, T$. Then, for any $\bar{\boldsymbol{u}}$, the number of mistakes M of Algorithm OM-2 satisfies*

$$M \leq C + L + \sqrt{LC} - \sum_{t \in I} \eta_t$$

where $C = 2qF^{2/q}\|\bar{\boldsymbol{u}}\|^2_{2,p}$ and $L = \sum_{t=1}^{T} \ell(\bar{\boldsymbol{u}}, \boldsymbol{x}_t, y_t)$.

Proof. Proofs are deferred to Appendix A. □

Apart from the negative term $-\sum_{t \in I} \eta_t$, the bound has the standard form of mistake bounds for online learning multiclass algorithms that only update on mistakes [10]. The presence of this negative term theoretically motivates the use of updates in margin error rounds. Note also that when the problem is linearly separable, the algorithm converges to a solution that has training error equal to zero. Moreover, when p goes to 1, the term C in the bound has a strongly sublinear dependence on the number of kernels, but unfortunately q goes to infinity. Similarly to [12], we trade-off these two terms to obtain a logarithmic dependence on the number kernels. In particular, we obtain the following corollary.

Corollary 1. *Under the hypotheses of Theorem 1, if $p = \frac{2\ln(F)}{2\ln(F)-1}$ then for any $\bar{\boldsymbol{u}}$, the number of mistakes M of Algorithm OM-2 is less than*

$$C + L + \sqrt{CL} - \sum_{t \in I} \eta_t$$

where $C = 4e\ln(F)\|\bar{\boldsymbol{u}}\|^2_{2,1}$ and $L = \sum_{t=1}^{T} \ell(\bar{\boldsymbol{u}}, \boldsymbol{x}_t, y_t)$.

Suppose that only one kernel is useful for the classification task at hand while the others $F-1$ are irrelevant. With the optimal tuning of p the price we pay in the bound for not knowing which kernel is the right one is just logarithmic in their number F.

Algorithm 1 OM-2

1: **Input:** q
2: **Initialize:** $\bar{\boldsymbol{\theta}}_1 = \mathbf{0}, \bar{\boldsymbol{w}}_1 = \mathbf{0}$
3: **for** $t = 1, 2, \ldots, T$ **do**
4: Receive new instance $\boldsymbol{x}_t$
5: Predict $\hat{y}_t = \operatorname*{argmax}_{y=1,\ldots,M} \bar{\boldsymbol{w}}_t \cdot \bar{\phi}(\boldsymbol{x}_t, y)$
6: Receive label y_t
7: $\bar{\boldsymbol{z}}_t = \bar{\phi}(\boldsymbol{x}_t, y_t) - \bar{\phi}(\boldsymbol{x}_t, \hat{y}_t)$
8: **if** $\ell(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t) > 0$ **then** $\eta_t = \min\left\{1 - \frac{2\bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t}{\|\bar{\boldsymbol{z}}_t\|^2_{2,q}}, 1\right\}$
9: **else** $\eta_t = 0$
10: $\bar{\boldsymbol{\theta}}_{t+1} = \bar{\boldsymbol{\theta}}_t + \eta_t \bar{\boldsymbol{z}}_t$
11: $\boldsymbol{w}^j_{t+1} = \frac{1}{q}\left(\frac{\|\boldsymbol{\theta}^j_{t+1}\|_2}{\|\bar{\boldsymbol{\theta}}_{t+1}\|_{2,q}}\right)^{q-2} \boldsymbol{\theta}^j_{t+1}, \quad \forall j = 1, \cdots, F$
12: **end for**

3.1. Implementation

Following [24, Section 4], it is possible to use Mercer kernels without introducing explicitly the dual formulation of the optimization problem. As $\bar{\boldsymbol{\theta}}_1$ is initialized to the zero vector, $\bar{\boldsymbol{\theta}}_{t+1}$ can be written as $\bar{\boldsymbol{\theta}}_{t+1} = \sum_t \eta_t \bar{\boldsymbol{z}}_t = \sum_t \eta_t(\bar{\phi}(\boldsymbol{x}_t, y_t) - \bar{\phi}(\boldsymbol{x}_t, \hat{y}_t))$. Therefore, the algorithm can easily store η_t, y_t, $\hat{y}_t$, and $\boldsymbol{x}_t$ instead of storing $\bar{\boldsymbol{\theta}}_t$. Observing line 11 in the algorithm, we have that at each round, $\boldsymbol{w}^j_{t+1}$ is proportional to $\boldsymbol{\theta}^j_{t+1}$, that is $\boldsymbol{w}^j_{t+1} = \alpha^j_t \boldsymbol{\theta}^j_{t+1}$. Hence $\bar{\boldsymbol{w}}_{t+1}$ can also be represented using $\alpha^j_t \eta_t$, y_t, $\hat{y}_t$ and $\boldsymbol{x}_t$. During prediction, the dot product between $\bar{\boldsymbol{w}}_{t+1}$ and $\bar{\phi}(\boldsymbol{x}_{t+1}, y_{t+1})$ can be expressed as a sum of terms $\bar{\boldsymbol{w}}^j_{t+1} \cdot \phi^j(\boldsymbol{x}_{t+1}, y_{t+1})$, that can be calculated as

$$\begin{aligned}
&\boldsymbol{w}^j_{t+1} \cdot \phi^j(\boldsymbol{x}_{t+1}, y_{t+1}) \\
&= \alpha^j_{t+1} \sum_t \eta_t \big(\phi^j(\boldsymbol{x}_t, y_t) \cdot \phi^j(\boldsymbol{x}_{t+1}, y_{t+1}) \\
&\quad - \phi^j(\boldsymbol{x}_t, \hat{y}_t) \cdot \phi^j(\boldsymbol{x}_{t+1}, y_{t+1}) \big) \\
&= \alpha^j_{t+1} \sum_t \eta_t \Big(K^j\big((\boldsymbol{x}_t, y_t), (\boldsymbol{x}_{t+1}, y_{t+1})\big) \\
&\quad - K^j\big((\boldsymbol{x}_t, \hat{y}_t), (\boldsymbol{x}_{t+1}, y_{t+1})\big) \Big) .
\end{aligned}$$

Note that the time spent by the OM-2 algorithm in each step is dominated by the prediction (line 5). The time required to execute line 5 is $\mathcal{O}(S_t F M)$, where S_t is the number of updates made before the t-th iteration.

4. Experiments

In this section we present an experimental evaluation of our approach in two different scenarios, corresponding to two publicly available databases. The first scenario is about learning new object categories, and the experiments were conducted on the Caltech-101 dataset [9]. The second scenario is about place recognition for a mobile robot, using the IDOL2 dataset [16].

We compared the performance of our algorithm to the OMCL algorithm [15], to the Passive-Aggressive algorithm (PA-I) [7] using the average kernel and the single best feature, and to a batch MKL algorithm [25][2]. We determined all of our online learning parameters via cross-validation. The features and kernel parameters were obtained directly from the authors who made this information publicly available [15, 11].

In both experiments we reported the average online training error. This is the number of prediction mistakes the algorithm makes on a given input sequence normalized by the length of the sequence, which is a standard performance measure for online learning algorithms. We also tested the generalization performance of our algorithms on a separate test set, which is similar to the classic batch learning setup. However, it is known that the generalization performance of an online algorithm trained with only one pass (epoch) is typically inferior to batch algorithms, especially when the number of training instances is small. Hence we cycled through the training samples several times (epochs), and we reported the performance on a separate test set as a function of the number of epochs. This is similar to a stochastic gradient descent optimization [2]. Experiments show that OM-2 achieves better performance than the other online learning methods, and comparable performance as the batch MKL method at much lower computational cost. The MATLAB implementation of our algorithm is available within the DOGMA library [20], which we also used for implementation of the baseline online algorithms.

4.1. Object Categorization: Caltech-101

The Caltech-101 [9] dataset is a standard benchmark dataset for object categorization. In our experiments we used the pre-computed features as well as the train/test splits of [11], which the authors made available on their website[3]. There are eight different image descriptors. Using different setup of parameters and different pyramid scales we end up with a total of 48 kernel matrices. For brevity, we omit the details of the features and kernels. These can be found in [11] and its project website.

We report results using all 102 classes and 30 training images per category over five different splits. For each split, the experiments were performed over 10 different permutations of the training images. Figure 1 (Left) shows the average online training error rate using different online learning algorithms as a function of the number of training examples. Figure 1 (Right) reports the classification performance obtained using different online learning algorithms and the batch MKL algorithm. It can be observed from the plots that the OM-2 algorithm achieves better performance on both training and test phase compared to the other online learning algorithms. The best results were obtained when p is small (1.01 in our setup). This is probably because among these 48 kernels (computed from eight different image descriptors) many of them were redundant. Thus a sparse solution is favored. After about 80 epochs, the OM-2 achieve comparable results as the batch MKL algorithm at a relative low computational time. Although our MATLAB implementation is not optimized for speed, training on Caltech-101 dataset using 30 examples per class and 48 kernels takes about 45 mins[4] (150 epochs, 3060×150 iterations). While for MKL the training time using the SILP implementation is more than 2 hours. The performance advantage of OM-2 over SILP is due the fact that OM-2 is based on a native multiclass formulation (see Section 2), while SILP solves the multiclass problem by decomposing it into multiple independent binary classification tasks. Thanks to the online learning framework, we could solve the new MKL formulation using the multi-class loss function efficiently. This is in line with similar observations in [11].

4.2. Place Recognition: IDOL-2

We also performed a series of experiments on the IDOL2 database [16], which 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. This dataset is ideal for testing online learning algorithm: the algorithm has to incrementally update the model, so to adapt to the variations captured in the dataset. For experiments, we used the same setup described in [16, 15]. 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 [15].

Figure 2 reports the average online training and recognition error rate on the test set. In contrast to the previous experiments, here the best performance is obtained at $p = 2$, which means that there are no redundant features and all of them are discriminative for the given task. OM-2 and PA-I using average kernel achieve better performance com-

[2] To the best of our knowledge, SILP [25] is the most efficient MKL implementation publicly available. Although SimpleMKL [21] is found to be more efficient, a good implementation for large problems is not available.

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

[4] Notice that this time is measured using precomputed kernel matrix as inputs. In practice, if not all the data points are available from the beginning, the kernel value could be computed "on the fly" and the total computation time of several epochs could be reduced using methods like kernel caching.

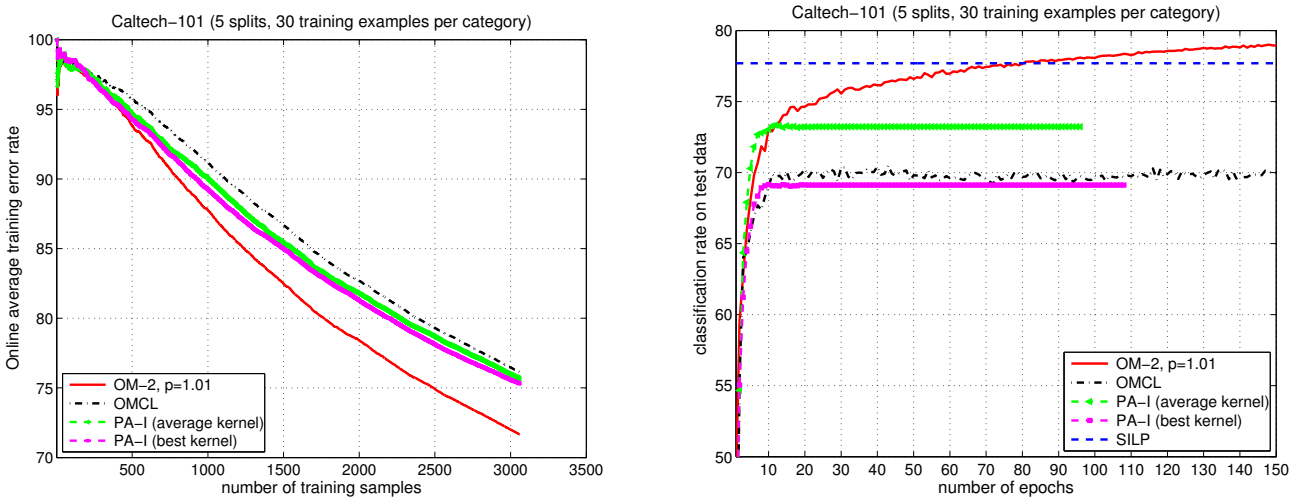

Figure 1. Performance of different online learning algorithms on Caltech-101 dataset as a function of the number of training examples: (left) average online training error rate; (right) classification rate on the test set.

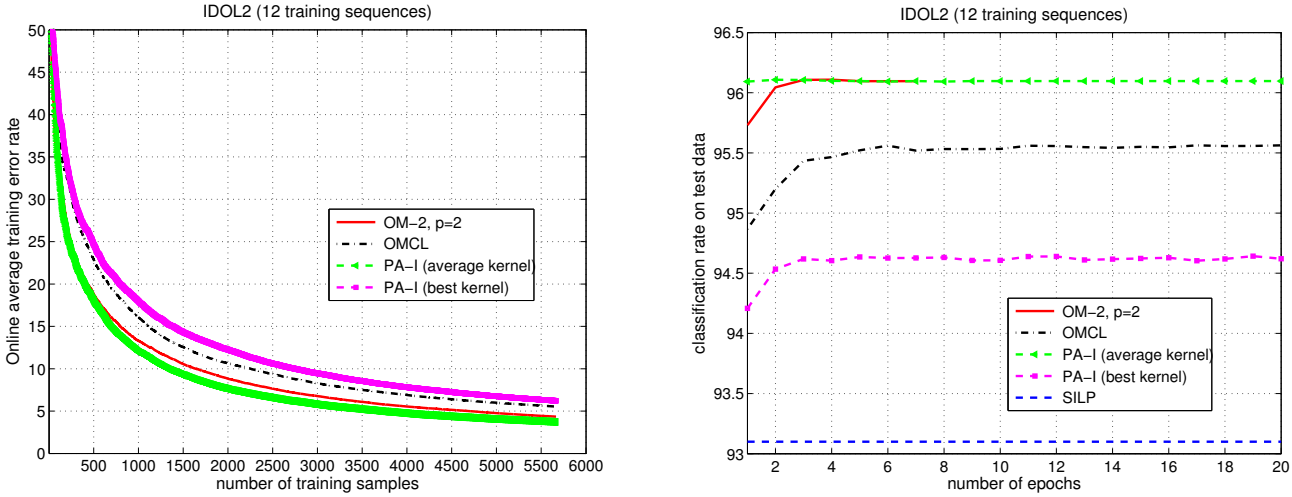

Figure 2. Performance of different online learning algorithms on IDOL2 dataset as a function of the number of training examples: (left) average online training error rate; (right) classification rate on the test set.

pared to the other online learning algorithms. Surprisingly, the performance of batch MKL is worse than all the other online feature combination method, including the performance of the single best feature. Hence, in agreement with recent findings [6], promoting sparsity hurts performance when the problem is not sparse at all.

5. Conclusion

This paper presents a theoretically motivated online learning algorithm for solving the multiclass MKL problem. The algorithm is based on a new p-norm formulation of the MKL problem that allows us to tune the level of sparsity in order to obtain always nearly optimal performance. The time to update the model of OM-2 is linear in the number of features, classes and training examples. Therefore, our algorithm scales well to large problems. Experiments show that our algorithm achieves better performance compared to common baselines. Using enough epochs of training, our approach achieves the same generalization performance, or even better, than batch MKL, with a considerably lower training time.

Acknowledgments

The Caltech-101 kernel matrices were kindly provided by Peter Gehler. LJ, MF and BC were supported by the EU project DIRAC IST-027787. FO and NCB gratefully acknowledge partial support by the PASCAL2 Network of Excellence under EC grant no. 216886. This publication only reflects the authors' views. Part of this work was done while FO was still at Idiap Research Institute.

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] L. Bottou and Y. L. Cun. On-line learning for very large dataset. *Applied Stochastic Models in Business and Industry*, 21(2):136–151, 2005. 5

[3] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear algorithms for online multitask classification. In *Proc. COLT*, 2008. 2, 3, 4

[4] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006. 1

[5] G. Chechik, V. Sharma, U. Shalit, , and S. Bengio. An online algorithm for large scale image similarity learning. In *Proc. NIPS*. 2009. 1

[6] C. Cortes, A. Gretton, G. Lanckriet, M. Mohri, and A. Rostamizadeh. Proceedings of the NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels, 2009. 6

[7] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *JMLR*, 7:551–585, 2006. 5

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/. 1

[9] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE PAMI*, 28(4):594–611, 2004. 4, 5

[10] M. Fink, S. Shalev-Shwartz, Y. Singer, and S. Ullman. Online multiclass learning by interclass hypothesis sharing. In *Proc. ICML*, pages 313–320, 2006. 1, 2, 4

[11] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *Proc. ICCV*, 2009. 1, 5

[12] C. Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53(3):265–299, 2003. 4

[13] D. Grangier and S. Bengio. A discriminative kernel-based approach to rank images from text queries. *IEEE PAMI*, 30(8), 2008. 1

[14] P. Jain and A. Kapoor. Active learning for large multi-class problems. In *Proc. CVPR*, 2009. 1

[15] L. Jie, F. Orabona, and B. Caputo. An online framework for learning novel concepts over multiple cues. In *Proc. ACCV*, 2009. 2, 5

[16] L. Jie, A. Pronobis, B. Caputo, and P. Jensfelt. Incremental learning for place recognition in dynamic environments. In *Proc. IROS*, 2007. 1, 4, 5

[17] 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. 2, 3, 8

[18] A. Kembhavi, B. Siddiquie, R. Miezianko, S. McCloskey, and L. S. Davis. Incremental multiple kernel learning for object recognition. In *Proc. ICCV*, 2009. 1

[19] 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. 3

[20] F. Orabona. *DOGMA: a MATLAB toolbox for Online Learning*, 2009. Software available at `http://dogma.sourceforge.net`. 5

[21] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. SimpleMKL,. *JMLR*, 9:2491–2521, November 2008. 2, 3, 5

[22] S. Shalev-Shwartz. Online learning: Theory, algorithms, and applications, 2007. PhD thesis. 3

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

[24] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos:primal estimated sub-gradient solver for svm. In *Proc. ICML*, 2007. 4

[25] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *JMLR*, 2006. 3, 5

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

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

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

A. Appendix

Proof of Theorem 1. The proof is based on an adaptation of a result from [17] for the "Follow the Regularized Leader" algorithm. We start by introducing a few basic notions of convex analysis. A differentiable function $f : S \to \mathbb{R}$ is said to be λ-smooth with respect to a norm $\|\cdot\|$ iff for any $\boldsymbol{u}, \boldsymbol{v} \in S$, $f(\boldsymbol{u}+\boldsymbol{v}) \leq f(\boldsymbol{u}) + \nabla f(\boldsymbol{u}) \cdot \boldsymbol{v} + \frac{\lambda}{2} \|\boldsymbol{v}\|^2$. Note that f is λ-smooth w.r.t. a norm iff f is $(1/\lambda)$-strongly convex w.r.t. the corresponding dual norm [17].

We now proceed by bounding the quantity $\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}}$ from below and from above. From [17, Corollary 19], we know that $h(\bar{\boldsymbol{w}})$ is 1-smooth w.r.t. $\|\cdot\|_{2,q}$. Moreover, line 11 in the algorithm's pseudo-code implies that $\bar{\boldsymbol{w}}_t = \nabla h^*(\bar{\boldsymbol{\theta}}_t) = \nabla h^*\big(\sum_{i=1}^{t-1} \eta_i \bar{\boldsymbol{z}}_i\big)$. Hence, we obtain

$$\begin{aligned}\|\bar{\boldsymbol{\theta}}_{T+1}\|_{2,q}^2 &\leq \|\bar{\boldsymbol{\theta}}_T\|_{2,q}^2 + 2q\eta_T \bar{\boldsymbol{w}}_T \cdot \bar{\boldsymbol{z}}_T + q\eta_T^2 \|\bar{\boldsymbol{z}}_T\|_{2,q}^2 \\ &\leq q \sum_{t=1}^{T} \big(2\eta_t \bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t + \eta_t^2 \|\bar{\boldsymbol{z}}_t\|_{2,q}^2\big) \,.\end{aligned}$$

Let $R_t = 2\eta_t \bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t + \eta_t^2 \|\bar{\boldsymbol{z}}_t\|_{2,q}^2$. Using the convex inequality for norms we then get

$$\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}} \leq \|\bar{\boldsymbol{\theta}}_{T+1}\|_{2,q} \, \|\bar{\boldsymbol{u}}\|_{2,p} \leq \|\bar{\boldsymbol{u}}\|_{2,p} \sqrt{q \sum_{i=1}^{T} R_t} \,.$$

We can further bound the last term by considering separately the steps t when $\bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t \leq 0$ (a mistake occurs), and the steps when $0 \leq \bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t \leq 1$ (a margin error occurs). Using that $\|\bar{\boldsymbol{z}}_t\|_{2,q}^2 \leq 2F^{2/q}$ given $\big\|\phi^j(\boldsymbol{x}_t, y_t)\big\|_2 \leq 1$, we have that $\eta_t = 1$ at mistakes, and we can further upper bound $\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}}$ as follows

$$\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}} \leq \|\bar{\boldsymbol{u}}\|_{2,p} \sqrt{2qF^{2/q}M + q \sum_{t \in I} R_t} \,. \tag{9}$$

For the lower bound we have that

$$\begin{aligned}\bar{\boldsymbol{\theta}}_{T+1} \cdot \bar{\boldsymbol{u}} &= \sum_{t=1}^{T} \eta_t \bar{\boldsymbol{u}} \cdot \bar{\boldsymbol{z}}_t \\ &= \sum_{t=1}^{T} \eta_t \bar{\boldsymbol{u}} \cdot \big(\bar{\phi}(\boldsymbol{x}_t, y_t) - \bar{\phi}(\boldsymbol{x}_t, \hat{y}_t)\big) \\ &\geq \sum_{t=1}^{T} \eta_t \big(1 - \ell(\bar{\boldsymbol{u}}, \boldsymbol{x}_t, y_t)\big) \\ &\geq M + \sum_{t \in I} \eta_t - L \,.\end{aligned}$$

Combining this last inequality with (9), solving the inequality for M, and using the definition of C, we obtain

$$M \leq \frac{C}{2} + L - \sum_{t \in I} \eta_t + \frac{C}{2} \sqrt{1 + \frac{4}{C}(A + L)}$$

where

$$A = \sum_{t \in I} \frac{R_t - 2\eta_t F^{2/q}}{2F^{2/q}} \,.$$

Now, observing that η_t has been chosen so that $A \leq 0$ and overapproximating we obtain the stated bound. □