Detection and Identification of Rare Audiovisual Cues

Insperata accident magis saepe quam quae speres.
(Things you do not expect happen more often than
things you do expect) Plautus (ca 200 (B.C.))

Project no. 027787

# DIRAC

## Detection and Identification of Rare Audio-visual Cues

Integrated Project
IST – Priority 2

DELIVERABLE NO: D5.11
Discriminative methods for multi-cue classification
Updated Version for M48

Date of deliverable: 31.12.2009
Actual submission date: 25.01.2010

Start date of project: 01.01.2006 Duration: 60 months

Organization name of lead contractor for this deliverable: **IDIAP Research Institute**

Revision [0]

| **Project co-funded by the European Commission within the Sixth Framework Program (2002-2006)** | | |
|---|---|---|
| **Dissemination Level** | | |
| PU | Public | |
| PP | Restricted to other program participants (including the Commission Services) | |
| RE | Restricted to a group specified by the consortium (including the Commission Services) | X |
| CO | Confidential, only for members of the consortium (including the Commission Services) | |

# D5.11 Discriminative methods for multi-cues classification

Idiap Research Institute (Idiap)

## Abstract:

Detecting incongruent audio-visual events requires robust models of generic and specific classes. Furthermore, after detection action should follow. This requires multi cue online learning algorithms.

We propose two classification algorithms that consider multi-cues inputs. Both algorithms are motivated from the recent development in online learning literatures, thus they shared the simplicity and speed of the online learning algorithms. The first algorithm tackles the problem of learning under limited computational resources in a teacher-student scenario, over multiple visual cues. For each separate cue, we train an online learning algorithm that sacrifices performance in favor of bounded memory growth and fast update of the solution. We then recover back performance by using multiple cues in the online setting. As in standard online learning setups, the learning takes place in rounds. On each round, a new hypothesis is estimated as a function of the previous one.

The second algorithm is a Multi Kernel Learning algorithm that obtains state-of-art performance in a considerably lower training time. We prove theoretically and experimentally that 1) our algorithm has a faster convergence rate as the number of kernels grows; 2) the training complexity is linear in the number of training examples; 3) very few iterations are enough to reach good solutions.

# 1. Introduction

In order to identify incongruent event (Weinshall, et. al., 2008), we need robust models of the worlds, which could successfully categorize the known sensory inputs. However, categorization is still one of the most challenging open problems in artificial perception research, especially in computer vision today. One the other hand, robust classifiers with high accuracy is an essential component when we design novelty detection algorithms. They help us to analyze the performance of new algorithms, where the errors come from, which finally lead to better algorithms.

Object categories present a wide visual variability. This, coupled with scalability over hundreds of classes and 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 sophisticated machine learning techniques (Nilsback and Caputo, 2004; Lin et. al., 2007; Kumar and Sminchisescu, 2007; Varma and Ray, 2007; Gehler and Nowozin, 2009; Kembhavi et. al., 2009; Vedldi et. al., 2009). The intuitive notion that using more features leads to better performance has been recently translated into discriminative classifiers combined with kernels over multiple cues (Bosch et. al., 2007; Gehler and Nowozin, 2009; Kumar and Sminchisescu, 2007; Varma and Ray, 2007; Kembhavi et. al., 2009; Vedldi et. al., 2009). Results obtained by these methods on various benchmark databases represent the current state-of-the-art in object categorization. However, most of the emphasis so far has been put on the accuracy of these methods. They are usually more computational expensive compared to method using single cue, which prevents their application to large-scale problems. Moreover, these method are batch method, which cannot be directly applied in problems where the data is sequential. Many related application scenario of the DIRAC project are intrinsically sequential. In these problems, the system finds something unknown (Weinshall, et. al., 2008) in the scene, and it cannot wait to collect enough data before building a model for the new concept, as it is expected to interact continuously with the environment. Under this setup, the system starts learning from impoverished data sets and keeps updating its solution as more data is acquired.

Motivated from the aforementioned problems, we have designed two new discriminative multi-cues classification algorithms. Both algorithms are motivated from the recent development in online learning literatures, thus they shared the simplicity and speed of that theoretical framework.

The first algorithm is called OMCL, Online Multi-Cues Learning Algorithm. It is a wrapper algorithm, which could plug-in most of the multi-class online learning algorithms in the machine learning literatures. For each separate cue, we train an online learning algorithm that sacrifices performance in favor of bounded memory growth and fast update of the solution. We then recover back performance by using multiple cues in the online setting. As in standard online learning setups, learning takes place in rounds and the algorithm only sees each data sample once. On each round, a new hypothesis is estimated as a function of the previous one. The algorithm will preserve most of the theoretical guarantee of its base classifiers. That is, it will have a mistake and convergence bound. When we use the Projectron++ (Orabona et. al., 2008) algorithm as base classifier, we will have a bounded memory complexity. A preliminary version of this algorithm has been published in the Proceeding of ACCV 2009 (Jie et. al., 2009).

The second algorithm is called OBSCURE, Online-Batch Strongly Convex mUlti keRnel lEarning algorithm. It is a Multiple Kernel Learning (MKL) algorithm that has a guaranteed and fast convergence rate to the optimal solution. Our algorithm has a training time that

depends linearly on the number of training examples, with a convergence rate sub-linear in the number of features/kernels used. 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. Furthermore, we show that by optimizing the primal objective function directly, we can stop the algorithm after a few iterations, while still retaining a performance close to the optimal one. A preliminary version of this work has been submitted to CVPR 2010.

# An online framework for learning novel concepts over multiple cues

Luo Jie[1,2], Francesco Orabona[1], and Barbara Caputo[1]

[1] Idiap Research Institute, Centre du Parc, Martigny, Switzerland
[2] École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
{jluo,forabona,bcaputo}@idiap.ch

**Abstract.** We propose an online learning algorithm to tackle the problem of learning under limited computational resources in a teacher-student scenario, over multiple visual cues. For each separate cue, we train an online learning algorithm that sacrifices performance in favor of bounded memory growth and fast update of the solution. We then recover back performance by using multiple cues in the online setting. To this end, we use a two-layers structure. In the first layer, we use a budget online learning algorithm for each single cue. Thus, each classifier provides confidence interpretations for target categories. On top of these classifiers, a linear online learning algorithm is added to learn the combination of these cues. As in standard online learning setups, the learning takes place in rounds. On each round, a new hypothesis is estimated as a function of the previous one. We test our algorithm on two student-teacher experimental scenarios and in both cases results show that the algorithm learns the new concepts in real time and generalizes well.

## 1 Introduction

There are many computer vision problems that are intrinsically *sequential*. In these problems the system starts learning from impoverished data sets and keeps updating its solution as more data is acquired. Therefore the system must be able to continuously learn new concepts, as they appear in the incoming data. This is a very frequent scenario for robots in home settings, where it is very likely to see something unknown [1] in a familiar scene. In such situations the robot cannot wait to collect enough data before building a model for the new concept, as it is expected to interact continuously with the environment. Limited space and computing power may also constrain the algorithm from being actually implemented, considering that the stream of training data can be theoretically infinite. Still, most of the used algorithms for computer vision are intrinsically *batch*, that is they produce a solution only after having seen enough training data. Moreover they are not designed to be updated often, because most of the time updating the solution is possible only through a complete re-training.

A different approach is the *online learning* framework [2]. This framework is motivated by a teacher-student scenario, that is when a new concept is presented to the machine, the machine (*student*) can ask the user (*teacher*) to provide a

label. This scenario would correspond to the case of a user explaining to the robot a detected source of novelty in a scene. The algorithms developed in the online learning framework are intrinsically designed to be updated after each sample is received. Hence the computational complexity per update is extremely low, but their performance is usually lower than similar batch algorithms.

Ideally, we would like to have an online method with performance as high as batch based algorithms, with fast learning and bounded memory usage. Existing online algorithms (see for example [3, 4]) fail to satisfy all these conditions. The mainstream approaches attempt to keep the same performance of batch based methods while retaining either fast learning or bounded memory growth but not both. On the other hand, multiple-cues/sources inputs guarantee diverse and information-rich sensory data. They make it possible to achieve higher and robust performance in varied, unconstrained settings. However, when using multiple inputs, the expansion of the input space and memory requirements is linearly proportional to the number of inputs as well as the computational time, for both the training and test phase.

Some recent works in online learning applied to computer vision include: Monteleoni and Kääriäinen [5] present two active learning algorithms in the online classification setting and test it on an OCR application; Fink et. al. [6] who describe a general framework for online learning and present preliminary results on office images and face recognition. Grangier and Bengio [7] propose a Passive-Aggressive algorithm [3] for Image Retrieval, which takes advantage of the efficient online learning algorithms. On the multi-cues literature, a recently and influential proposed approach to combine cues in the batch setting is to learn the weights of the weighted sum of kernels [8]. Even if many attempts have been done to speed up the training process [9, and references therein], this approach is still slow and does not scale well to big datasets. Moreover these methods are intrinsically non-incremental, hence they cannot be used in a sequential learning setting. A theoretically motivated method for online learning over multiple cues has been proposed in [10], however they assume that all the cues live in the same space, meaning that the same kernel must be used on all the cues.

In this work we tackle the problem of learning from data using an online learning algorithm over multiple visual cues. By combining online learning with multiple cues, we manage to get the best of both worlds, i.e. high performance, bounded memory growth and fast learning time. The proposed algorithm is tested on two experimental scenarios: the first is place recognition which simulates the student-teacher scenario where the robot is shown an indoor environment composed of several rooms (this is the kitchen, this is the corridor, etc), and later it is supposed to localize and navigate to perform assigned tasks. The second is object categorization which simulates the student-teacher scenario where the autonomous agent is presented a collection of new objects. For both scenarios, results show that the algorithm learns the new concepts in real time and generalizes well to new concepts.

In the next section we describe the online learning framework and the building blocks that we will use in our online multi-cues architecture (Section 2-3).

Section 4 describes our experimental findings. Finally, we conclude the paper with a summary and a discussion on possible future research.

## 2 Online Learning

Online learning is a process of continuous updating and exploitation of the internal knowledge. It can also be thought of as learning in a teacher-student scenario. The teacher shows an instance to the student who predicts its label. Then the teacher gives feedback to the student. An example of this would be a robot which navigates in a closed environment, learning to recognize each room from its own sensory inputs. Moreover, to gain robustness and increase the classification performance, we argue for the need of learning using multiple cues. Hence our goal is to design an online learning algorithm for learning over multiple features from the same sensor, or data from multiple sensors, which is able to take advantage of the diverse and information-rich inputs and to achieve more robust results than systems using only a single cue. In the following we will introduce the online learning framework and we will explain how to extend it to multiple cues. Due to space limitations, this is a very quick account of the online learning framework — the interested readers are referred to [2] for a comprehensive introduction.

### 2.1 Starting from Kernel Perceptron

In online setting, the learning takes place in rounds. The online algorithm learns the mapping $f : \mathcal{X} \to \mathbb{R}$ based on a sequence of examples $\{\mathbf{x}_t, y_t\}_{t=1}^{l}$, with instance $\mathbf{x}_t \in \mathcal{X}$ and label $y_i \in \{-1, 1\}$. We denote the hypothesis estimated after the $t$-th round by $f_t$. At each round $t$, the algorithm receives a new instance $\mathbf{x}_t$, then it predicts a label $\hat{y}_t$ by using the current function, $\hat{y}_t = \text{sign}(f_t(\mathbf{x}_t))$, where we could interpret $|f(\mathbf{x})|$ as the confidence in the prediction. Then, the correct label $y_t$ is revealed. The algorithm changes its internal model everytime it makes a mistake or the confidence on the prediction is too low. Here, we denote the set of all attainable hypotheses by $\mathcal{H}$. In this paper we assume that $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS) with a positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ implementing the inner product which satisfies the reproducing property, $f(\mathbf{x}) = \langle k(\mathbf{x}, \cdot), f(\cdot) \rangle$.

Perhaps the most well known online learning algorithm is Rosenblatt's Perceptron algorithm [11]. On the $t$-th round the instance $\mathbf{x}_t$ is given, and the algorithm makes a prediction $\hat{y}_t$. Then the true label is revealed: if there is a prediction mistake, i.e. $\hat{y}_t \neq y_t$, it updates the hypothesis, $f_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$, namely it stores $\mathbf{x}_t$ in the solution. Otherwise the hypothesis is left intact, $f_t = f_{t-1}$. Given the nature of the update, the hypothesis $f_t$ can be written as a kernel expansion [12], $f_t(\mathbf{x}) = \sum_{i \in \mathcal{S}_t} \alpha_i k(\mathbf{x}_i, \mathbf{x})$. The subset of instances used to construct the function is called the *support set*. Although the Perceptron is a very simple algorithm, it has been shown to produce very good results. Several other algorithms (see Passive-Aggressive [3] and the references therein) can be seen as belonging to the Perceptron algorithm family. However, given that they update

each time there is an error, if the problem is not linearly separable, they will never stop adding new instances to the support set. This will eventually lead to a memory explosion. As we aim to use the algorithm in applications where data must be acquired continuously in time, a Perceptron algorithm cannot be used as it is. Hence we will use as a basic component of our architecture the Projectron++ algorithm [13].

### 2.2 The Projectron++ Algorithm

The Projectron++ [13] algorithm is a Perceptron-like algorithm bounded in space and time complexity. It has a better mistake bound than Perceptron. The core idea of the algorithm comes from the work of Downs et. al. [14] on simplifying Support Vector Machine solutions. Hence, instead of updating the hypothesis every time a prediction mistake is made, or when the prediction is correct with low confidence[3], the Projectron++ first checks if the update can be expressed as a linear combination of vectors in the support set, i.e. $k(\mathbf{x}_t, \cdot) = \sum_{i=1}^{t-1} d_i k(\mathbf{x}_i, \cdot) = P_{t-1}(k(\mathbf{x}_t, \cdot))$, where $P_{t-1}(\cdot)$ is the projection operator. The concept of linear independence can be approximated and tuned by a parameter $\eta$ that measures the quality of the approximation. If the instance can be approximated within an error $\eta$, it is not added to the support set but the coefficients in the old hypothesis are changed to reflect the addition of the instance. If the instance and the support set are linearly independent, the instance is added to the set, as Perceptron. We refer the reader to [13] for a detailed analysis.

## 3 Online Multi-Cues Learning Algorithm

In this section we describe our algorithm for learning over multiple cues. We adapt the idea of *high-level* integration from the information fusion community (see [15] for a comprehensive survey), and design our algorithm with a two-layers structure. The first layer is composed of different Projectrons++, one for each cue. The second layer learns online a weighted combination of the classifiers of the first layer, hence we interpret the output of the Projetrons++ on the first layer as confidence measures of the different cues.

A lot of work has been done on how to select the best algorithm from a pool of prediction algorithms, such as the Weighted Majority algorithm [16]. However, they usually assume black-box classifiers. Here we want to learn the best combination of classifiers, not just picking the best one. Therefore we train *both* the single cue classifiers and the weighted combination with online algorithms. In the rest of this section, we will describe our algorithm in the binary setup. For multi-class problems, the algorithm is extended using the multi-class extension method presented in [17]; we omit the detailed derivation for lack of space.

Suppose we have $N$ cues of the same data $\mathbf{X} = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N\}$. Each cue is described by a feature vector $\mathbf{x}^i \in \mathbb{R}^{m_i}$, where $\mathbf{x}^i$ could be the feature vector

[3] that is when $0 < y_t f_{t-1}(\mathbf{x}_t) < 1$

**Algorithm 1** OMCL Algorithm

**Input:** Projectron++ parameter $\eta \geq 0$; Passive-aggressive parameter $C > 0$.
**Initialize:** $f_0^i = \mathbf{0}, \mathcal{S}_0^i = \emptyset$, where $i = 1, 2, \ldots, N$; $\omega_0 = \mathbf{0}$.
**for** $t = 1, 2, \ldots, T$ **do**
  Receive new instance $\mathbf{x}_t^i$, where i=1, 2, ..., N
  Predict $\hat{h}_t^i = f_{t-1}^i(\mathbf{x}_t^i)$, where i=1, 2, ..., N
  Predict $\hat{y}_t = \text{sign}(\omega_\mathbf{t} \cdot \hat{\mathbf{h}}_\mathbf{t})$
  Receive label $y_t$
  **for** $i = 1, 2, \ldots, N$ **do**
    Loss: $l1_t^i = max(0, 1 - y_t \cdot \hat{h}_t^i)$
    **if** $l_t^i > 0$ **then**
      Compute projection error $\Delta$
      **if** $y_t = \hat{h}_t^i$ or $\Delta \leq \eta$ **then**
        Projection update: $f_t^i = f_{t-1}^i + \alpha_t^i y_t P_{t-1}^i(k(\mathbf{x}_t^i, \cdot))$
      **else**
        Normal update: $f_t^i = f_{t-1}^i + y_t k(\mathbf{x}_t^i, \cdot)$
      **end if**
      Update hypothesis: $\tilde{\mathbf{h}}_t^i = f_t^i(\mathbf{x}_t^i)$
    **end if**
  **end for**
  Loss: $l2_t = max(0, 1 - y_t \omega_\mathbf{t} \cdot \tilde{\mathbf{h}}_\mathbf{t})$
  Set: $\tau_t = min(C, \frac{l2_t}{\|\tilde{\mathbf{h}}_t\|^2})$
  Update: $\omega_{\mathbf{t+1}} = \omega_\mathbf{t} + \tau_t y_t \tilde{\mathbf{h}}_\mathbf{t}$
**end for**

associated with one feature descriptor or one input sensor. Suppose also we are given a sequence of data $\{\mathbf{X}_t, y_t\}_{t=1}^l$, where $y_t \in \{-1, 1\}$. On round $t$, in the first layer, $N$ Projectrons++ [13] learn the mapping for each views: $f_t^i : \mathbf{x}_t^i \rightarrow h_t^i$, where $h_t^i \in \mathbb{R}$, for $i = 1, 2, \ldots, N$. On top of the Projectron++ classifiers, a linear Passive-Aggressive [17] algorithm is used to learn a linear weighted combination of the confidence outputs of the classifiers: $\omega_t : \mathbf{h}_t \rightarrow \mathbb{R}$. The prediction of the algorithm is $\hat{y}_t = \text{sign}(\omega_t \cdot \mathbf{h}_t)$.

After each update of the first layer, we also update the confidences on the instances before passing them to the second layer. We denote by $\tilde{h}_t^i$ the confidence of the updated hypotheses and denote by $\hat{h}_t^i$ the confidence predictions of the Projection++ classifiers before knowing the true label. Hence, instead of updating the second layer based on $\hat{h}_t^i$, the linear Passive-Aggressive algorithm on top considers the new updated confidence $\tilde{h}_t^i$. This modified updating rule prevents the errors propagating from the first layer to the second layer, and in preliminary experiments it has shown to be less likely prone to over-fitting. We call this algorithm OMCL (Online Multi-Cue Learning, see Algorithm 1).

### 3.1 Online to Batch Conversion

Online algorithms are meant to be constantly used in teacher-student scenarios. Hence the update process will never stop. However it is possible to transform

them to batch algorithms, that is to stop the training and to test the current hypothesis on a separate test set. It is known that when an online algorithm stops the last hypothesis found can have an extremely high generalization error. This is due to the fact that the online algorithms are not converging to a fixed solution, but they are constantly trying to "track" the best solution. If the samples are Independent & Identically Distributed (IID), to obtain a good batch solution one can use the *average* of all the solutions found, instead of the last one. This choice gives also theoretical guarantees on the generalization error [18].

Our system produces two different hyperplanes at each round, one for each layer. In principle we could simply average each hyperplane separately, but this would break the IID assumption of the inputs for the second layer. So we propose an alternative method: given that the entire system is linear, it can be viewed as producing only one hyperplane at each round, that is the product of the two hyperplanes. Hence we average this unique hyperplane and in the testing phase we predict with the formula: $\text{sign}\left(\frac{1}{T}\sum_{t=1}^{T}\omega_{\mathbf{t}}\cdot\hat{\mathbf{h}}_{\mathbf{t}}\right)$.

Note that, as $f_t$ can be written as a kernel expansion [12], the averaging does not imply any additional computational cost, but just an update of the coefficients of the expansion. We use this approach as it guarantees a theoretical bound and it was also found to perform better in practice.

## 4 Experiments and Results

In this section, we present an experimental evaluation of our approach on two different scenarios, corresponding to two publicly available databases. The first scenario is about place recognition for a mobile robot, and the experiments were conducted on the IDOL2 dataset [19]. The second scenario is about learning new object categories, and the experiments were conducted on the ETH80 dataset [20]. Both experiments can be considered as a teacher-student scenario, where the system is taught to recognize rooms (or objects) by a human tutor. Therefore the robot has to learn the concepts in real time, and generalize well to new instances. For all experiments, we compared the performance and the memory requirements to the standard Perceptron algorithm by replacing the Projectron++ algorithm in our framework. We also compared our algorithm to two different cues combination algorithms: the "flat" data structure and the majority voting algorithm. The "flat" structure is simply a concatenation of all the features of different cues into a long feature vector, and we trained a Projectron++ classifier for it. The majority voting algorithm predicts the label by choosing the class which receives the highest number of votes. As for a majority voting algorithm in multi-class case, the number of experts (number of cues in our experiments) required by the algorithm which guarantees a unique solution will grow exponentially with the number of classes. Although it does not happen very often in practice, we show that sometimes two or more classes receive an equal number of votes, especially when the number of cues is relatively small compared to the number of classes. We determined all of our online learning and kernel parameters via cross-validation. Our implementation of the proposed

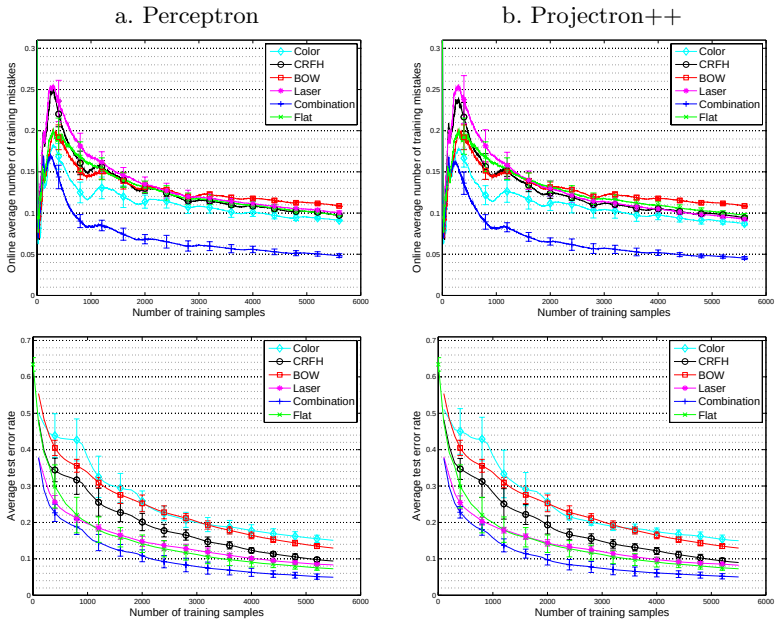


**Fig. 1.** Average online training error rate and recognition error rate for the test set on IDOL2 dataset as a function of the number of training samples.

algorithm uses the DOGMA package [21]; the source code is available within the same package.

### 4.1 First Scenario: Place Recognition

We performed the first series of experiments on the IDOL2 database [19], 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 (one-person office (OO), corridor (CR), two-person office (TO), kitchen (KT) and Printer Area (PA)); they were acquired under various weather and illumination conditions (sunny, cloudy, and night) and across a time span of six months. We considered the scenario where our 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 the original paper [19] (Section V, Part B). We considered the 12 sequences acquired by robot Dumbo, and divided them into training and testing sets, where each training sequence has a corresponding one in the test sets captured under roughly similar conditions. Similarly, each sequence was divided into five subsequences. Learning is done in chronological order, i.e. how the images were captured during acquisition of the dataset. During testing, all the sequences in the test set were taken into account. In total, we considered six different permutations of training and testing sets.

| cue | OO | CR | TO | KT | PA | ALL |
|---|---|---|---|---|---|---|
| Color | 28.4 | 9.5 | 8.1 | 9.9 | 31.4 | 17.4 |
| CRFH | 14.2 | 4.1 | 15.4 | 15.4 | 11.1 | 12.0 |
| BOW | 21.5 | 6.9 | 17.5 | 11.4 | 8.4 | 13.1 |
| Laser | 7.6 | 3.7 | 8.5 | 10.7 | 12.9 | 8.7 |
| OLMC | 7.6 | 2.5 | 1.9 | 6.7 | 13.3 | 6.4 |
| Optimal | 4.9 | 3.0 | 3.7 | 4.0 | 3.9 | 3.9 |
| Flat | 5.7 | 2.7 | 7.5 | 8.5 | 11.8 | 7.2 |
| Vote | 11.7 (6%) | 3.3 (2%) | 5.3 (3%) | 5.2 (3%) | 9.1 (7%) | 6.9 (4%) |

Table I: Place recognition error rate using different cues after the last training round. Each room is considered separately during testing, and it contributes equally to the overall results as an average. It shows that the OLMC algorithm achieves better performance than that of using each single cue. For the "vote" algorithm, the percentage of test data which have two or more classes receive equal number of votes is reported in the bracket below the error rate. Hence the algorithm can not make a definite prediction. Therefore we considered that the algorithm made a prediction error.

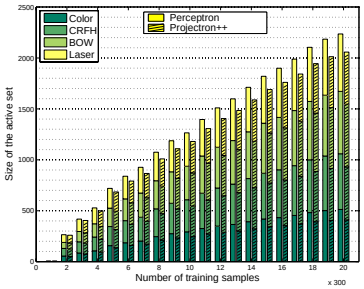


**Fig. 2.** Average size of support set for different algorithms on IDOL2 dataset as a function of the number of training samples.

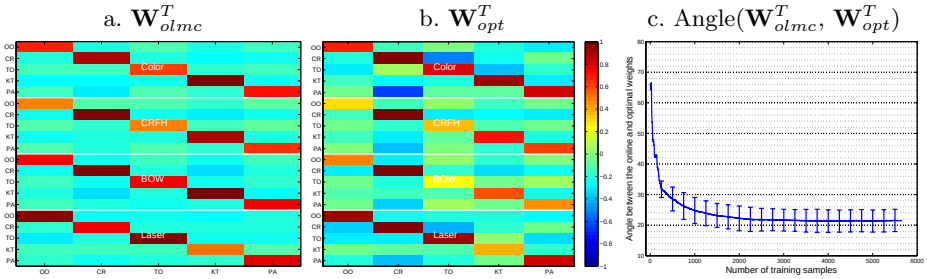


**Fig. 3.** Visualization of the average normalized weights for combining the confidence outputs of the Projectron++ classifiers: a. the weights obtained by our algorithm at last training round; b. the weights obtained by solving the above optimization problem. c. the angles between the two vectors $\mathbf{W}_{opt}$ and $\mathbf{W}_{olmc}$ as function of the number of training samples.

The images were described using three image descriptors, namely, CRFH [22] (Gaussian derivative along $x$ & $y$ direction), Bag-of-Words [23] (SIFT, 300 visual words) and RGB color histogram. In addition, we also used a simple geometric feature from the Laser Scan sensor [24].

Fig. 1 reports the average online training and recognition error rate on the test set, where the average online training error rate is the number of prediction mistakes the algorithm makes on a given input sequence normalized by the length of the sequence. Fig. 2 shows the size of the support sets as a function of the number of training samples. Here, the size of the support set of Projectron++ is close to that of Perceptron. This is because the support set of Perceptron is already very compact. Since the online training error rate is low, both algorithms do not update very frequently. In Table I we summarize the results using each cue after finishing the last training round. We see that our algorithm outperforms

both the "flat" data structure and the majority vote algorithm. The majority vote algorithm could not make a definite prediction on approximately 4% of the test data, because there are two or more classes which received an equal number of votes.

Moreover, we would like to see what is the difference in performance between the learned linear weights for combining confidence outputs of the Projectron++ classifiers and the optimal linear weighted solution. In another words, what is the best performance a linear weighted combination rule can achieve? We obtained an optimal combination weights, denoted as $\mathbf{W}_{opt}$, by solving a convex optimization program (see Appendix A) on the confidence outputs of the Projectron++ classifiers on the test set. We reported the result in Table I, which shows that our algorithms achieve performance similar to that of the optimal solution. We also visualized the average normalized weights for both the optimal solution and the weights obtained by our algorithm at the last learning round in Fig. 3a&b. From these figures, we can see that the weights on the diagonal of the matrix, which corresponds to the multi-class classifiers' confidence interpretations on the same target category, have highest values. Fig. 3c reports the average angle between the two vectors $\mathbf{W}_{opt}$ and $\mathbf{W}_{olmc}$, which is the weights obtained by our algorithms during the online learning process. We can see that the angle between these two vectors gradually converges to a low value.

### 4.2 Second Scenario: Object Categorization

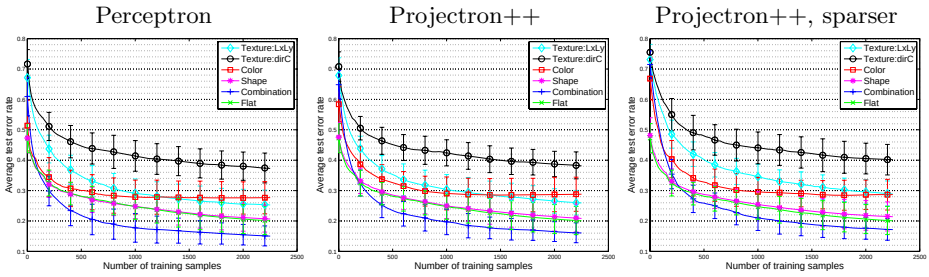


**Fig. 4.** Average online recognition error rate for the categorization of never seen objects on ETH-80 dataset as a function of the number of training samples. We see that all algorithms achieve roughly similar performance (Projectron++ is slightly better), the Perceptron converges earlier than the Projectron++ algorithms.

We tested the algorithm on the ETH-80 objects dataset [20]. The ETH-80 dataset consists of 80 objects from eight different categories (apple, tomato, pear, toy-cows, toy-horses, toy-dogs, toy-cars and cups). Each category contains 10 objects with 41 views per object, spaced equally over the viewing hemisphere, for a total of 3280 images. We use four image descriptors: one color feature (RGB color histogram), two texture descriptors (Composed Receptive Field Histogram

| Cues | apple | car | cow | cup | dog | horse | pear | tomato | all |
|---|---|---|---|---|---|---|---|---|---|
| $L_xL_y$ | 26.3 | 4.1 | 52.4 | 29.3 | 29.2 | 49.4 | 7.0 | 9.6 | 25.9 |
| $DirC$ | 25.0 | 21.9 | 60.7 | 43.3 | 65.9 | 55.7 | 29.8 | 3.2 | 38.2 |
| Color | 38.9 | 18.8 | 15.5 | 16.2 | 49.3 | 57.7 | 33.0 | 1.2 | 28.8 |
| Shape | 30.4 | 1.3 | 28.6 | 2.5 | 33.3 | 28.9 | 3.4 | 37.8 | 20.8 |
| OLMC | 11.1 | 1.5 | 17.4 | 13.7 | 34.1 | 44.2 | 5.0 | 1.1 | 16.0 |
| Flat | 29.3 | 1.5 | 28.2 | 2.0 | 32.1 | 28.0 | 3.3 | 35.1 | 20.0 |
| Vote | 24.2 (19%) | 1.3 (1%) | 30.1 (15%) | 11.3 (9%) | 34.0 (18%) | 41.2 (20%) | 3.6 (3%) | 6.1 (5%) | 19.0 (11%) |

Table II: Categorization error rate for different objects using different cues after finishing the last training round. We could see that our algorithm outperforms the "Flat" structure, the "Vote" algorithm and the case when using each cue alone. It also shows that some cues are very descriptive of certain objects, but not of the others. For example, the color feature achieves almost perfect performance on tomato, but its performance on other objects is low. It also supports our motivation on designing multi-cues algorithms. For the "vote" algorithm, the percentage of test data which have tow or more classes receives equal number of votes is reported in the bracket.

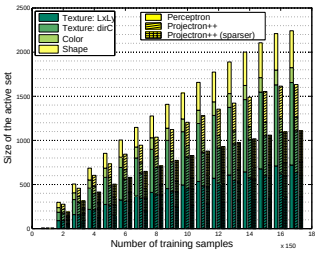


**Fig. 5.** Average size of support set for different algorithms on ETH80 dataset as a function of the number of training samples.

(CRFH) [22] with two different kinds of filters: Gaussian derivative $L_xL_y$ and gradient direction ($DirC$) and a global shape feature (centered masks). We randomly selected 7 out of the 10 objects for each category as training set, and the rest of them as test set. All the experiments were performed over 20 different permutations of the training set.

We first show the behavior of the algorithms over time. In Fig. 4, we show the average recognition error on never seen objects as a function of the number of learned samples. In the experiments, we used two different setting of $\eta$ parameters, labeled as Projectron++ and Projectron++, sparser. The growth of the support set as a function of the number of samples is depicted in Fig. 5. We see that the Projectron++ algorithm obtain similar performance as the Perceptron algorithm with less than 3/4 (Projectron++) and 1/2 (Projectron++, sparser) of the size of the support set. Finally, in Table II we summarize the error rate using different cues for each category after finishing the last training round (Projectron++).

## 5 Discussion and Conclusions

We presented an online method for learning from multi-cues/sources inputs. Through experiments on two image datasets, representative of two student-teacher scenarios for autonomous systems, we showed that our algorithm is able to learn a linear weighted combination of the marginal output of classifiers on each sources, and that this method outperforms the case when we use each cue alone. Moreover, it achieves performance comparable to the batch performance [19, 20] with a much lower memory and computational cost. We also showed that the budget Projectron++ algorithm had the advantage of reducing the support set without removing or scaling instances in the set. This keeps performance high, while reducing the problem of the expansion of the input space and memory requirement when using multiple inputs. Thanks to the robustness gained by using multiple cues, the algorithm could reduce more the support set

(e.g. Projectron++, sparser, see Fig. 4c & Fig. 5) without any significant loss in performance. This trade-off would be a potentially useful function for applications working in a highly dynamic environment and with limited memory resources, particularly for systems equipped with multiple sensors. Thanks to the efficiency of the learning algorithms, both learning and predicting could be done in real time with our Matlab implementation on most computer hardwares which run Matlab software.

In the future, we would like to explore theoretical properties of our algorithm. It is natural to extend our algorithms to the active learning setup [5] to reduce the effort of data labeling. Meanwhile, it would be interesting to explore the properties of co-training the classifier using the messages passed from some other classifiers with high confidence on predicting certain cues.

**Acknowledgments** We thank Andrzej Pronobis for sharing the laser sensor features. This work was sponsored by the EU project DIRAC IST-027787.

## References

1. Weinshall, D., Hermansky, H., Zweig, A., Jie, L., Jimison, H., Ohl, F., Pavel, M.: Beyond novelty detection: Incongruent events, when general and specific classifiers disagree. In: Proc. NIPS'08
2. Cesa-Bianchi, N., Lugosi, G.: Prediction, learning, and games. Cambridge University Press (2006)
3. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research **7** (2006) 551–585
4. Dekel, O., Shalev-Shwartz, S., Singer, Y.: The Forgetron: A kernel-based Perceptron on a budget. SIAM Journal on Computing **37** (2007) 1342–1372
5. Monteleoni, C., Kääriäinen, M.: Practical online active learning for classication. In: Proc. CVPR'07, Online Learning for Classification Workshop
6. Fink, M., Shalev-Shwartz, S. Singer, Y., Ullman, S.: Online multiclass learning by interclass hypothesis sharing. In: Proc. ICML'06
7. Grangier, D., Bengio, S.: A discriminative kernel-based approach to rank images from text queries. IEEE Transactions on Pattern Analysis and Machine Intelligence **30**(8) (2008) 1371–1384
8. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., El Ghaoui, L., Jordan, M.I.: Learning the kernel matrix with semideinite programming. Journal of Machine Learning Research **5** (2004) 27–72
9. Rakotomamonjy, A., Bach, F., Grandvalet, Y., Canu, S.: SimpleMKL. Journal of Machine Learning Research **9** (2008) 2491–2521
10. Cavallanti, G., Cesa-Bianchi, N., Gentile, C.: Linear algorithms for online multitask classification. In: Proc. COLT'08
11. Rosenblatt, F.: The Perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review **65** (1958) 386–407
12. Schölkopf, B., Herbrich, R., Smola, A., Williamson, R.: A generalized representer theorem. In: Proc. COLT'00
13. Orabona, F., Keshet, J., Caputo, B.: The Projectron: a bounded kernel-based Perceptron. In: Proc. ICML'08

14. Downs, T., Gates, K., Masters, A.: Exact simplification of support vectors solutions. Journal of Machine Learning Research **2** (2001) 293–297
15. Sanderson, C., Paliwal, K.K.: Identity verification using speech and face information. Digital Signal Processing **14**(5) (2004) 449–480
16. Littlestone, N., Warmuth, M.: Weighted majority algorithm. In: IEEE Symposium on Foundations of Computer Science. (1989)
17. Crammer, K., Singer, Y.: Ultraconservative online algorithms for multiclass problems. Journal of Machine Learning Research **3** (2003) 951–991
18. Cesa-Bianchi, N., Conconi, A., Gentile, C.: On the generalization ability of on-line learning algorithms. IEEE Trans. on Information Theory **50**(9) (2004) 2050–2057
19. Luo, J., Pronobis, A., Caputo, B., Jensfelt, P.: Incremental learning for place recognition in dynamic environments. In: Proc. IROS'07
20. Leibe, B., Schiele, B.: Analyzing appearance and contour based methods for object categorization. In: Proc. CVPR'03
21. Orabona, F.: DOGMA: a MATLAB toolbox for Online Learning. (2009) Software available at `http://dogma.sourceforge.net`.
22. Linde, O., Lindeberg, T.: Object recognition using composed receptive field histograms of higher dimensionality. In: Proc. ICPR'04
23. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: Proc. ICCV'03
24. Mozos, O.M., Stachniss, C., Burgard, W.: Supervised learning of places from range data using adaboost. In: Proc. ICRA'05

## Appendix A

Let $\{\hat{\mathbf{h}}_i, y_i\}_{i=1}^{l}$ be the confidence outputs of the Projectron++ classifiers on the test set of $l$ instances, where each sample $\hat{\mathbf{h}}_i$ is drawn from a domain $\mathbb{R}^m$ and each label $y_i$ is an integer from the set $\mathcal{Y} = \{1, 2, \ldots, k\}$. In the multi-class setup, $m = n \times k$, where $n$ is the number of cues (i.e. number of the Projectron++ classifiers) and $k$ is the number of classes. Therefore, we obtained the optimal linear solution by solving the following convex optimization problem:

$$\begin{aligned} \min_{\mathbf{W},\xi} \quad & \frac{1}{2}||\mathbf{W}||^2 + C \sum_{i=1,\ldots,l,\ j\in y_i^C} \xi_{i,j} \\ \textit{subject to } & \overline{W}_{y_i} \cdot \hat{\mathbf{h}}_\mathbf{i} + \xi_{i,j} > \overline{W}_j \cdot \hat{\mathbf{h}}_\mathbf{i} \\ \forall i,\ j\in y_i^C & \\ & \xi_{i,j} \geq 0 \end{aligned}$$

where $\mathbf{W}$ is the multi-class linear weighted combination matrix of size $k \times m$, and $\overline{W}_r$ is the $r$-th row of $\mathbf{W}$. We denote the complement set of $y_i$ as $y_i^C = \mathcal{Y} \backslash y_i$. This setting is a generalization of linear binary classifiers. Therefore, the value $\overline{W}_r \cdot \hat{\mathbf{h}}_\mathbf{i}$ denotes the confidence for the $r$ class and the classifier predicts the label using the function:

$$\hat{y} = \arg\max_{r=1,\ldots,k} \{\overline{W}_r \cdot \hat{\mathbf{h}}\}$$

The regularizer $||\mathbf{W}||^2$ is introduced to prevent slack variables $\xi_{i,j}$ producing solutions close to $0^+$. The cost $C$ value is decided through cross validation.

# Online-Batch Strongly Convex Multi Kernel Learning

Francesco Orabona
Idiap Research Institute
Martigny, Switzerland
forabona@idiap.ch

Luo Jie
Idiap Research Institute
Martigny, Switzerland
jluo@idiap.ch

Barbara Caputo
Idiap Research Institute
Martigny, Switzerland
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 Multi Kernel Learning algorithm that obtains state-of-the-art performance in a considerably lower training time. We generalize the standard Multi Kernel Learning 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 open problems in computer vision today. Object categories present a wide visual variability within each class, especially for man-made objects. This, coupled with scalability over hundreds of classes and 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 sophisticated machine learning techniques [7, 10, 11, 14, 15, 24, 25].

The intuitive notion that using more features leads to better performance has been recently translated into SVM-based classifiers combined with kernels over multiple cues [2, 7, 10, 11, 24, 25]. 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 [11, 24]. 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 in performance over the naive baseline of averaging all the kernels [7].

To our knowledge, none of the MKL algorithms [17, 20, 23] provides theoretical guarantees on the convergence rate. Moreover, the learning process is usually stopped early, before reaching the optimal solution, based on the common assumption that it is enough to have an approximate solution of the optimization function. Considering the fact that the current MKL algorithms are solved based on their dual representation, this might mean being stopped far from the optimal solution [8]. From a practical point of view, this means that the observed performance of such algorithms could be low, even if potentially they could perform well. Another issue is the scalability of these algorithms to large-scale problems.

The contribution of this paper is a 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 the maximum 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 features/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 [8]. Furthermore, we show that by optimizing the primal objective function directly, 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 [7, 15]. 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 [26]. They use Cross-Validation (CV) methods to create the training set for the second layer [7, 26]. 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 [7].

Another interesting strategy uses a one-layer architecture, such as the MKL [12, 17, 20, 23, 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. The SILP algorithm [20, 28] depends polynomially on the number of training examples and number of classes with an exponent of $\sim 2.4$ and $\sim 1.7$ 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 $\beta_y^j$ 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, 20, 24].

The main contribution of this paper is showing that the one-layer formulation, beside being more principled, can also achieve a better 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 [21]. 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, namely that our OBSCURE algorithm has better performance and training time than state-of-the-art one-layer and two-layers architectures. Section 2 revises the basic definitions of Multi Kernel Learning 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 $\mathbb{Y}$ (*e.g.* [22]). 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 space. With multiple cues, we will have $F$ functions $\phi^j(\cdot,\cdot), i = 1, \cdots, F$. This will also define $F$ kernels

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

$K^j((\boldsymbol{x},y),(\boldsymbol{x}',y'))$ 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 [22]

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

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* $\|\bar{\boldsymbol{w}}\|_{2,p}^2$ on $\bar{\boldsymbol{w}}$ as

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

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 $\|\cdot\|_{2,q}$. These kind of norms have been used as *block regularization* in the LASSO literature [27].

### 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

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

This same formulation is used in [1, 20], while in [17] the proposed formulation is slightly different, although it is proved to be equivalent. Note that we chose to weight the regularization term by $\lambda$ 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. However the two formulations are fully equivalent when setting $\lambda=\frac{1}{CN}$. Hence a big value of $C$ will correspond to a small value of $\lambda$.

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

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

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 [9], so its optimization algorithm is rather complex and its rate of convergence is usually slow [1, 20].

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

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

where $1<p\leq 2$. We define $f(\bar{\boldsymbol{w}})=\frac{\lambda}{2}\|\bar{\boldsymbol{w}}\|_{2,p}^2+\frac{1}{N}\sum_{i=1}^N\ell\left(\bar{\boldsymbol{w}},\boldsymbol{x}_i,y_i\right)$ 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 $\lambda/q$-strongly convex [9]. Strongly 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 [18, 9]. 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. In the following we will show how to take advantage of the strong convexity to design a fast algorithm to solve (9).

## 3. The OBSCURE Algorithm

Our basic optimization tool is the framework developed in [18, 19]. 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, based on this framework, is the current state-of-art solver for linear SVM [19, 21].

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 [19, 21].

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 $\lambda$ is big. Moreover it is common knowledge that in many real-world problems, particularly in visual learning tasks, the best setting for $\lambda$ is very small, or equivalently $C$ is very big (the order of $10^2 - 10^3$). Notice that this is a general problem. The same problem also exists in the other SVM optimization algorithms such as SMO and similar approaches [8], as their training time also depends on the value of the parameter $C$.

Do *et al.* [5] proposed a variation of the Pegasos algorithm called proximal projected sub-gradient descent. This formulation has a better convergence rate for small values of $\lambda$, while retaining the fast convergence rate for big values of $\lambda$. A drawback is that the algorithm needs to know in advance an upper bound on the norm of the optimal solution. In [5] 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 [5] 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.

We have proved the following theorem that gives a theoretical guarantee on the convergence rate of OBSCURE to the optimal 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* $\sqrt{6}F^{\frac{2}{q}}\|\bar{\boldsymbol{u}}\|_{2,p}$.

***Proof.*** *See the supplementary material.*

**Algorithm 1** OBSCURE stage 1 (online)

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: Sample at random $(\boldsymbol{x}_t, y_t)$
5: $\hat{y}_t = \operatorname{argmax}_{y \neq y_t} \bar{\boldsymbol{w}}_t \cdot \bar{\phi}(\boldsymbol{x}_t, y)$
6: $\bar{\boldsymbol{z}}_t = \bar{\phi}(\boldsymbol{x}_t, y_t) - \bar{\phi}(\boldsymbol{x}_t, \hat{y}_t)$
7: **if** $\ell(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t) > 0$ **then** $\eta_t = \min\left(1 - \frac{\bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t}{q\|\bar{\boldsymbol{z}}_t\|_{2,q}}, 1\right)$
8: **else** $\eta_t = 0$
9: $\bar{\boldsymbol{\theta}}_{t+1} = \bar{\boldsymbol{\theta}}_t + \eta_t \bar{\boldsymbol{z}}_t$
10: $\boldsymbol{w}^j_{t+1} = \frac{1}{q}\left(\frac{\|\boldsymbol{\theta}^j_{t+1}\|_2}{\|\bar{\boldsymbol{\theta}}_{t+1}\|_{2,q}}\right)^{q-2} \boldsymbol{\theta}^j_{t+1},\ \forall j = 1, \cdots, F$
11: **end for**
12: **return** $\bar{\boldsymbol{\theta}}_{T+1}, \bar{\boldsymbol{w}}_{T+1}$
13: **return** $R = \sqrt{\|\bar{\boldsymbol{w}}_{T+1}\|^2_{2,p} + \frac{2}{\lambda N}\sum_{i=1}^N \ell\left(\bar{\boldsymbol{w}}_{T+1}, \boldsymbol{x}_i, y_i\right)}$

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 $\lambda$ (*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 $q$. A $p$ close to 1 will result in a sparse solution, 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 also shows that 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 [4].

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 [5] to general norms, using the framework in [18]. We need two additional definitions. Given a convex function $f : S \to \mathbb{R}$, its Fenchel conjugate $f^* : S \to \mathbb{R}$ is defined as $f^*(\boldsymbol{u}) = \sup_{\boldsymbol{v} \in S}(\boldsymbol{v}^T\boldsymbol{u} - f(\boldsymbol{v}))$. A vector $\boldsymbol{x}$ is a sub-gradient of a function $f$ at $\boldsymbol{v}$, indicated with $\partial f(\boldsymbol{v})$, if $\forall \boldsymbol{u} \in S, f(\boldsymbol{u}) - f(\boldsymbol{v}) \geq (\boldsymbol{u} - \boldsymbol{v})^T\boldsymbol{x}$.

**Lemma 1.** *Let* $h(\cdot) = \frac{\alpha}{2}\|\cdot\|^2$ *be a 1-strongly convex func-*

**Algorithm 2** OBSCURE stage 2 (batch)

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

**Algorithm 3** Proximal projected sub-gradient descent

1: **Input:** $R, \sigma, \boldsymbol{w}_1 \in S$
2: **Initialize:** $s_0 = 0$
3: **for** $t = 1, 2, \dots, T$ **do**
4: Receive $g_t$
5: $\boldsymbol{z}_t = \partial g_t(\boldsymbol{w}_t)$
6: $s_t = s_{t-1} + \frac{\sqrt{(\sigma t + s_{t-1})^2 + \frac{L_t}{R^2}} - \sigma t - s_{t-1}}{2}$
7: $\eta_t = (\sigma t + s_t)^{-1}$
8: $\boldsymbol{w}_{t+1} = \nabla h^*(\nabla h(\boldsymbol{w}_t) - \eta_t \boldsymbol{z}_t)$
9: **end for**

*tion w.r.t. a norm $\|\cdot\|$ over $S$. Assume that for all $t$, $g_t(\cdot)$ is a $\sigma$-strongly convex function w.r.t. $h(\cdot)$, and $\|\boldsymbol{z}_t\|_* \leq L_t$. Additionally, let $\xi_{a:b}$ be defined as $\sum_{i=a}^{b} \xi_i$. Then for any $\boldsymbol{u} : \|\boldsymbol{u} - \boldsymbol{w}_t\| \leq 2R$, Algorithm 3 achieves the following bound for all $T \geq 1$,*

$$\sum_{t=1}^{T} (g_t(\boldsymbol{w}_t) - g_t(\boldsymbol{u})) \leq \min_{\xi_1, \cdots, \xi_T} \sum_{t=1}^{T} \left[ 4\xi_t R^2 + \frac{L_t^2}{\sigma t + \frac{\xi_{1:t}}{\alpha}} \right].$$

***Proof.*** *See the supplementary material.*

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. They are transformed into $\bar{\boldsymbol{w}}_t$ in line 13, through a simple scaling.

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 [19]. 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.

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 [9]. It is similar to the $2p$-norm matrix Perceptron of [3], but it overcomes the disadvantage of being used with the same kernel on each feature. As in [3], for Algorithm 1 it is possible to prove a relative mistake bound. We omit the details for lack of space, but they are available in the supplementary material.

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

$$R := \sqrt{\|\bar{\boldsymbol{w}}_{T+1}\|_{2,p}^2 + \frac{2}{\lambda N} \sum_{i=1}^{N} \ell(\bar{\boldsymbol{w}}_{T+1}, \boldsymbol{x}_i, y_i)}$$
$$\geq \sqrt{\|\bar{\boldsymbol{w}}^*\|_{2,p}^2 + \frac{2}{\lambda N} \sum_{i=1}^{N} \ell(\bar{\boldsymbol{w}}^*, \boldsymbol{x}_i, y_i)} \geq \|\bar{\boldsymbol{w}}^*\|_{2,p}. \quad (10)$$

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

If $F$ 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, of the order of $\mathcal{O}(q^2 F^{\frac{2}{q}} \|\bar{\boldsymbol{u}}\|_{2,p}^2)$. Moreover the returned value of $R$ will be less than $\sqrt{6} F^{\frac{2}{q}} \|\bar{\boldsymbol{u}}\|_{2,p}$.*

***Proof.*** *See the supplementary material.*

The dependency on the number of kernels in this theorem is strongly sublinear, moreover if the problem (9) will be linearly separable with $F$ kernels, increasing the number of kernels to $F' > F$, we have that $\|\bar{\boldsymbol{u}}\|_{2,p}^2$ will decrease. This means that we expect Algorithm 1 to converge, under the separability assumption, 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 [16], Caltech-101 [6] and MNIST [13] datasets. Although our MATLAB implementation of the algorithm[2] 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 $p$ is chosen from the set $\{1.01, 1.05, 1.10, 1.25, 1.50, 1.75, 2\}$. When $p$ tends to 1, the solution tends to be sparse. The regularization parameter $\lambda$ is set through CV, as $\frac{1}{CN}$, where $C \in \{1, 10, 100, 1000\}$.

### 4.1. Oxford flowers

The Oxford flowers dataset [16] 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[3]. These distance matrices are transformed into kernel using $\exp(-\gamma^{-1} \cdot d)$, where $\gamma$ is the mean of the pairwise distances and $d$ is the distance between two examples.

We have implemented an extended version of the original Pegasos algorithm [19, 21] for the MKL problem of (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 [20], SimpleMKL [17] and LP-$\beta$ [7]. 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. It seems that both SILP and SimpleMKL obtain an optimal solution on datasets of this size. LP-$\beta$ is order of magnitudes faster as it uses an efficient standard SVM solver [4].

### 4.2. Caltech-101 datasets

The Caltech-101 [6] 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

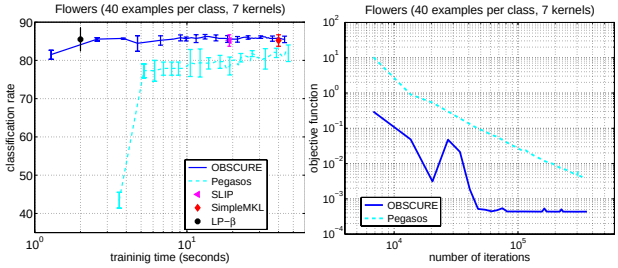


Figure 1. Comparison of performance on Oxford flowers dataset.

features and kernels of [7] which the authors made available on their website[4], with the same training and test split. This allows us to compare against them directly. Following that, we report results using all 102 classes of the Caltech-101 dataset using three 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 and refer to [7].

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 tends to be sparse. On the other hand, when $p$ equals 2, the algorithm gets a similar solution as averaging 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 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. In other words, 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-$\beta$ (about 36m). When the number of training examples increases to 30, the advantage of our algorithm

[2] Code attached with the supplementary materials.

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

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

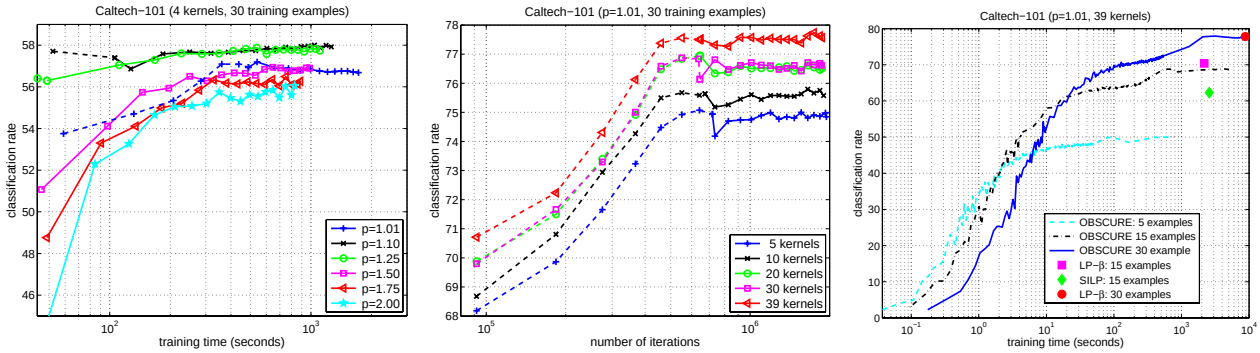


Figure 2. Behaviors of the OBSCURE algorithm on Caltech-101 dataset: (Left) the effect of different value of $p$, using four PHOG [2] kernels computed at different spatial pyramid level, as similar experiment performed in [7]; (Middle) the effect of different number of kernels randomly sampled from the 39 kernels; (Right) running time for different number of training examples using all the 39 kernels.

becomes clear, while a single run of LP-$\beta$ takes about 146m.

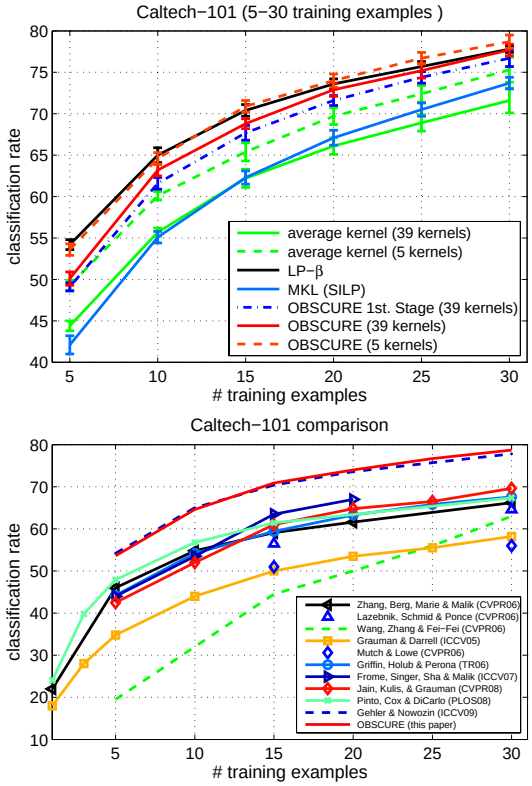


Figure 3. Performance of OBSCURE on Caltech-101 datasets: (Top) comparison with other combination methods on Caltech-101; (Down) comparison with other published methods.

In Figure 3(Top) 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-$\beta$, [7]), and outperforms the other MKL (SILP) methods. One possible explanation is that the MKL (SILP) fails to converge to an optimal solution. 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-$\beta$ [7], since several kernels were not informative. However, this may not be critical, since usually in practice all used features/kernels are informative. We did a simple test by selecting five kernels from the five different families of features [7] which achieve best performance. They were selected using a simple K-folds CV on the 30 training examples per class. 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, when provided with discriminative features, could increase performance even further. In Figure 3(Down) we compare OBSCURE with other methods in the literatures[5].

### 4.3. MNIST

For the last set of experiment we use the MNIST [13] 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 ($14 \times 14$) 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

[5]We did not include the results of [2, 24] since they are erroneous. See *e.g.* the authors' websites.

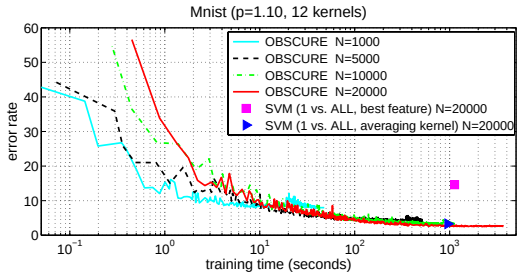


Figure 4. The generalization performance of MNIST dataset over different size of training samples.

over time, for various training size. We see that OBSCURE quickly converges to the best performance, moreover the convergence is *faster when more training samples are used*, as in [21]. 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 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 very 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 [9]. For example the framework can be very easily extended to solve other problems such as *structure output prediction*[22], 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 somehow in contrast with recent work that prefers sparse models [1, 7, 20]. As a last remark, we notice that the disadvantageous results of MKL methods, reported in [7], may be because those algorithms fail to converge to the optimal solution. By using our method, MKL can still be a popular machine learning tool for cue combination tasks.

## 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, 8

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

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

[4] 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. 4, 6

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

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

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

[8] 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

[9] 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

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

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

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

[13] 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

[14] Y.-Y. Lin, T.-L. Liu, and C.-S. Fuh. Local ensemble kernel learning for object category recognition. In *Proc. CVPR*, 2007. 1

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

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

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

[18] S. Shalev-Shwartz and Y. Singer. Logarithmic regret algorithms for strongly convex repeated games. Technical Report 2007-42, The Hebrew University, 2007. 3, 4

[19] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos:primal estimated sub-gradient solver for svm. In *Proc. ICML*, 2007. 3, 5, 6

[20] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *JMLR*, 7, 2006. 1, 2, 3, 6, 8

[21] S.Shalev-Shwartz and N.Srebro. SVM, optimization: inverse dependence on training set size. In *Proc. ICML*, 2008. 2, 3, 6, 8

[22] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. ICML'04*, 2004. 2, 3, 8

[23] M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *Proc. ICML*, 2009. 1, 2

[24] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *Proc. ICCV'07*, 2007. 1, 2, 7

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

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

[27] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Roy. Stat. Society*, 68:49–67, 2006. 3

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