Online Learning of Structured Predictors with Multiple Kernels

André F. T. Martins[*†] **Noah A. Smith**[*] **Eric P. Xing**[*] **Pedro M. Q. Aguiar**[‡] **Mário A. T. Figueiredo**[†]

[*]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

[‡]Instituto de Sistemas e Robótica, Instituto Superior Técnico, Lisboa, Portugal

[†]Instituto de Telecomunicações, Instituto Superior Técnico, Lisboa, Portugal

Abstract

Training structured predictors often requires a considerable time selecting features or tweaking the kernel. Multiple kernel learning (MKL) sidesteps this issue by embedding the kernel learning into the training procedure. Despite the recent progress towards efficiency of MKL algorithms, the structured output case remains an open research front. We propose a family of online algorithms able to tackle variants of MKL and group-LASSO, for which we show regret, convergence, and generalization bounds. Experiments on handwriting recognition and dependency parsing attest the success of the approach.

1 INTRODUCTION

Structured prediction is characterized by strong interdependence among the output variables, usually with sequential, graphical, or combinatorial structure (Bakır et al., 2007). Despite all the advances, obtaining good predictors still requires a large effort in feature/kernel design and tuning (often via cross-validation). Because discriminative training of structured predictors can be quite slow, it is appealing to learn the kernel function simultaneously.

In MKL (Lanckriet et al., 2004), the kernel is learned as a linear combination of prespecified base kernels. This framework has been made scalable with the advent of wrapper-based methods, in which a standard learning problem (*e.g.*, an SVM) is repeatedly solved in an inner loop up to a prescribed accuracy (Sonnenburg et al., 2006; Rakotomamonjy et al., 2008; Kloft et al., 2010). Unfortunately, extending such methods to structured prediction raises practical hurdles: since the output space is large, so are the kernel matrices, and the number of support vectors. Moreover, since it is typically prohibitive to tackle the inner learning problem in its batch form, one often needs to resort to online algorithms (Ratliff et al., 2006; Vishwanathan et al., 2006; Collins et al., 2008); the latter are fast learners but slow optimizers (Bottou and Bousquet, 2007), hence using them in the inner loop with early stopping can misguide the overall MKL optimization.

In this paper, we overcome the above difficulty by proposing a stand-alone online MKL algorithm which iterates between subgradient and proximal steps. The algorithm, which when applied to structured prediction is termed SPOM (*Structured Prediction by Online MKL*), has important advantages: *(i)* it is simple, flexible, and compatible with sparse and non-sparse MKL, *(ii)* it is adequate for structured prediction, *(iii)* it offers regret, convergence, and generalization guarantees. Our approach extends and kernelizes the forward-backward splitting scheme FOBOS (Duchi and Singer, 2009), whose regret bound we improve.

Our paper is organized as follows: after reviewing structured prediction and MKL (§2), we present a class of online proximal algorithms which handle composite regularizers (§3). We derive convergence rates and show how these algorithms are applicable in MKL, group-LASSO, and other structural sparsity formalisms. In §4, we apply this algorithm for structured prediction (yielding SPOM) in two experimental testbeds: sequence labeling for handwritten text recognition, and natural language dependency parsing. We show the potential of SPOM by learning combinations of kernels from tens of thousands of training instances, with encouraging results in terms of runtimes and accuracy.

2 STRUCTURED PREDICTION & MKL

2.1 Inference and Learning with Structured Outputs

We denote by $\mathcal{X}$ and $\mathcal{Y}$ the input and output sets, respectively. Given an input $x \in \mathcal{X}$, we let $\mathcal{Y}(x) \subseteq \mathcal{Y}$ be its set of admissible outputs; in structured prediction, this set is assumed to be structured and exponentially large. Two important examples of structured prediction problems are: sequence labeling, in which x is an observed sequence and

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

each $y \in \mathcal{Y}(x)$ is a corresponding sequence of labels; and natural language parsing, where x is a string and each $y \in \mathcal{Y}(x)$ is a possible parse tree that spans that string.

Let $\mathcal{U} \triangleq \{(x, y) \mid x \in \mathcal{X}, y \in \mathcal{Y}(x)\}$ denote the set of admissible input-output pairs. In supervised learning, we are given a labeled dataset $\mathcal{D} \triangleq \{(x_1, y_1), \ldots, (x_N, y_N)\} \subseteq \mathcal{U}$, and the goal is to learn a compatibility function $f_{\boldsymbol{\theta}} : \mathcal{U} \to \mathbb{R}$ that allows to make predictions on unseen data via

$$x \mapsto \hat{y}(x) \triangleq \arg \max_{y \in \mathcal{Y}(x)} f_{\boldsymbol{\theta}}(x, y). \tag{1}$$

Problem (1) is called *inference* (*decoding*) and, in strutured prediction, often involves combinatorial optimization (*e.g.*, dynamic programming). In this paper, we consider linear functions, $f_{\boldsymbol{\theta}}(x, y) \triangleq \langle \boldsymbol{\theta}, \boldsymbol{\phi}(x, y) \rangle$, where $\boldsymbol{\theta}$ is a parameter vector and $\boldsymbol{\phi}(x, y)$ a feature vector. To be general, we let these vectors live in a Hilbert space $\mathcal{H}$, with reproducing kernel $K : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$. In the sequel, features will be used implicitly or explicitly, as convenience determines.

We want $f_{\boldsymbol{\theta}}$ to generalize well, *i.e.*, given a cost function $c : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ (with $c(y, y) = 0$, $\forall y$) we want the corresponding inference criterion to have low expected risk $\mathbb{E}_{X,Y} c(Y, \hat{y}(X))$. To achieve this, one casts learning as the minimization of a regularized empirical risk functional,

$$\min_{f_{\boldsymbol{\theta}} \in \mathcal{H}} \lambda\, \Omega(f_{\boldsymbol{\theta}}) + \frac{1}{N} \sum_{i=1}^{N} L(f_{\boldsymbol{\theta}}; x_i, y_i), \tag{2}$$

where $\Omega : \mathcal{H} \to \mathcal{R}_+$ is a regularizer, $\lambda \geq 0$ is the regularization constant, and $L : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a convex loss function. Common choices are the logistic loss, in *conditional random fields* (CRF, Lafferty et al. 2001), and the structured hinge loss, in structural *support vector machines* (SVM, Taskar et al. 2003; Tsochantaridis et al. 2004):

$$L_{\mathrm{CRF}}(f_{\boldsymbol{\theta}}; x, y) \triangleq \log \textstyle\sum_{y' \in \mathcal{Y}(x)} \exp(\delta f_{\boldsymbol{\theta}}(x, y', y)), \tag{3}$$

$$L_{\mathrm{SVM}}(f_{\boldsymbol{\theta}}; x, y) \triangleq \max_{y' \in \mathcal{Y}(x)} \delta f_{\boldsymbol{\theta}}(x, y', y) + c(y', y), \tag{4}$$

where $\delta f_{\boldsymbol{\theta}}(x, y', y) \triangleq f_{\boldsymbol{\theta}}(x, y') - f_{\boldsymbol{\theta}}(x, y)$.

If the regularizer is $\Omega(f_{\boldsymbol{\theta}}) = \frac{1}{2}\|\boldsymbol{\theta}\|^2$ (ℓ_2-regularization), the solution of (2) can be expressed as a kernel expansion (structured version of the representer theorem, Hofmann et al. 2008, Corollary 13). We next discuss alternative forms of regularization that take into consideration another level of structure—now in the feature space.

2.2 Block-Norm Regularization and Kernel Learning

Selecting relevant features or picking a good kernel are ubiquitous problems in statistical learning, both of which have been addressed with *sparsity-promoting regularizers*. We first illustrate this point for the explicit feature case. Often, features exhibit a natural block structure: for example, many models in NLP consider *feature templates*—these are binary features indexed by each word w in the vocabulary, by each part-of-speech tag t, by each pair (w, t), etc. Each of these templates correspond to a *block* (also called a *group*) in the feature space $\mathcal{H}$. Thus, $\mathcal{H}$ is endowed with a block structure, where each block (indexed by $m = 1, \ldots, M$) is itself a "smaller" feature space $\mathcal{H}_m$; formally, $\mathcal{H}$ is a direct sum of Hilbert spaces: $\mathcal{H} = \bigoplus_{m=1}^{M} \mathcal{H}_m$.

Group-LASSO. Consider the goal of learning a model in the presence of many irrelevant feature templates. A well-known criterion is the group-LASSO (Bakin, 1999; Yuan and Lin, 2006), which uses the following block-norm regularizer: $\Omega_{\mathrm{GL}}(f_{\boldsymbol{\theta}}) = \sum_{m=1}^{M} \|\boldsymbol{\theta}_m\|$. This can be seen as the ℓ_1-norm of the ℓ_2-norms: it promotes sparsity w.r.t. the number of templates (groups) that are selected. When Ω_{GL} is used in (2), the following happens within the mth group: either the optimal $\boldsymbol{\theta}^*_m$ is identically zero—in which case the entire group is discarded—or it is non-sparse.

Sparse MKL. In MKL (Lanckriet et al. 2004), we learn a kernel as a linear combination $K = \sum_{m=1}^{M} \beta_m K_m$ of M prespecified base kernels $\{K_1, \ldots, K_M\}$, where the coefficients $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_M)$ are constrained to the simplex $\Delta^M \triangleq \{\boldsymbol{\beta} \geq \mathbf{0} \mid \|\boldsymbol{\beta}\|_1 = 1\}$. This is formulated as an outer minimization of (2) w.r.t. $\boldsymbol{\beta} \in \Delta^M$:

$$\min_{\boldsymbol{\beta} \in \Delta^M} \min_{f_{\boldsymbol{\theta}} \in \mathcal{H}} \frac{\lambda}{2} \sum_{m=1}^{M} \beta_m \|\boldsymbol{\theta}_m\|^2 + \frac{1}{N} \sum_{i=1}^{N} L\left(\sum_{m=1}^{M} \beta_m f_{\boldsymbol{\theta}_m}; x_i, y_i \right). \tag{5}$$

Remarkably, as shown by Bach et al. (2004) and Rakotomamonjy et al. (2008), this joint optimization over $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ can be transformed into a single optimization of the form (2) with a block-structured regularizer $\Omega_{\mathrm{MKL}}(f_{\boldsymbol{\theta}}) = \frac{1}{2}(\sum_{m=1}^{M} \|\boldsymbol{\theta}_m\|)^2$. Note that this coincides with the square of the group-LASSO regularizer; in fact, the two problems are equivalent up to a change of λ (Bach, 2008a). Hence, this MKL formulation promotes sparsity in the number of selected kernels (*i.e.*, only a few nonzero entries in $\boldsymbol{\beta}$).

Non-Sparse MKL. A more general MKL formulation (not necessarily sparse) was recently proposed by Kloft et al. (2010). Define, for $p \geq 1$, the set $\Delta_p^M \triangleq \{\boldsymbol{\beta} \geq \mathbf{0} \mid \|\boldsymbol{\beta}\|_p = 1\}$. Then, by modifying the constraint in (5) to $\boldsymbol{\beta} \in \Delta_p^M$, the resulting problem can be again transformed in one of the form (2) with the block-structured regularizer:

$$\Omega_{\mathrm{MKL},q}(f_{\boldsymbol{\theta}}) = \frac{1}{2} \left(\sum_{m=1}^{M} \|\boldsymbol{\theta}_m\|^q \right)^{2/q} \triangleq \frac{1}{2} \|\boldsymbol{\theta}\|_{2,q}^2, \tag{6}$$

where $q = 2p/(p+1)$. The function $\|.\|_{2,q}$ satisfies the axioms of a norm, and is called the $\ell_{2,q}$ *mixed norm*. Given a solution $\boldsymbol{\theta}^*$, the optimal kernel coefficients can be computed as $\beta_m^* \propto \|\boldsymbol{\theta}_m^*\|^{2-q}$. Note that the case $p = q = 1$ corresponds to sparse MKL and that $p = \infty, q = 2$ corresponds to standard ℓ_2-regularization with a sum-of-kernels.

2.3 Learning the Kernel in Structured Prediction

Up to now, all formulations of MKL apply to classification problems with small numbers of classes. The algorithmic challenges that prevent the straightforward application of these formulations to structured prediction will be discussed in §2.4; here, we formally present our MKL formulation for structured prediction.

In structured prediction, model factorization assumptions are needed to make the inference problem (1) tractable. This can be accomplished by defining a set of *parts* over which the model decomposes. Suppose, for example, that outputs correspond to vertex labelings in a Markov network (V, E). Then, we may let each part be either a vertex or an edge, and write the feature vector as $\boldsymbol{\phi}(x, y) = (\boldsymbol{\phi}_V(x, y), \boldsymbol{\phi}_E(x, y))$, with

$$\boldsymbol{\phi}_V(x, y) = \sum_{i \in V} \boldsymbol{\psi}_V(x, i) \otimes \boldsymbol{\zeta}_V(y_i) \quad (7)$$

$$\boldsymbol{\phi}_E(x, y) = \sum_{ij \in E} \boldsymbol{\psi}_E(x, ij) \otimes \boldsymbol{\zeta}_E(y_i, y_j), \quad (8)$$

where $\otimes$ denotes the Kronecker product between feature vectors, $\boldsymbol{\psi}_V$ and $\boldsymbol{\psi}_E$ are input feature vectors (which may depend globally on x), and $\boldsymbol{\zeta}_V$ and $\boldsymbol{\zeta}_E$ are *local* output feature vectors which only look, respectively, at a single vertex and a single edge. A common simplifying assumption is to let $\boldsymbol{\zeta}_V$ be the identity feature mapping, $\boldsymbol{\psi}_E \equiv 1$, and to define $\boldsymbol{\zeta}_E$ as the identity feature mapping scaled by $\beta_0 > 0$. We can then learn the kernel $K_{X,V}((x, i), (x', i')) \triangleq \langle \boldsymbol{\psi}_V(x, i), \boldsymbol{\psi}_V(x', i') \rangle$ as a combination of basis kernels $\{K_{X,V,m}\}_{m=1}^M$. This yields a kernel decomposition of the form

$$K((x, y), (x', y')) = \beta_0 \cdot |\{ij, i'j' \in E : y_i = y'_{i'}, y_j = y'_{j'}\}| + \sum_{i,i' \in V : y_i = y'_{i'}} \sum_{m=1}^{M} \beta_m K_{X,V,m}((x, i), (x', i')).$$

In our sequence labeling experiments (§4), vertices and edges correspond to label unigrams and bigrams. We explore two strategies: learning $\beta_1, \ldots, \beta_M$, with $\beta_0 = 1$ fixed, or also learning β_0.

2.4 Existing MKL Algorithms

Early approaches to MKL (Lanckriet et al., 2004; Bach et al., 2004) considered the dual of (5) in the form of a second order cone program, thus were limited to small/medium scale problems. Subsequent work focused on scalability: Sonnenburg et al. (2006) proposes a semi-infinite LP formulation and a cutting plane algorithm; Rakotomamonjy et al. (2008) proposes a gradient-based method (*SimpleMKL*) for optimizing the kernel coefficients $\boldsymbol{\beta}$; Kloft et al. (2010) performs Gauss-Seidel alternate optimization of $\boldsymbol{\beta}$ and of SVM instances; Xu et al. (2009) proposes an extended level method.

The methods mentioned in the previous paragraph are all *wrapper-based algorithms*: they repeatedly solve problems of the form (2) (or smaller chunks of it, as in Kloft et al. 2010). Although warm-starting may offer considerable speed-ups, convergence relies on the exactness (or prescribed accuracy in the dual) of these solutions, which constitutes a serious obstacle when using such algorithms for structured prediction. Large-scale solvers for structured SVMs lack strong convergence guarantees; the best methods require $O(\frac{1}{\epsilon})$ rounds to converge to ϵ-accuracy. Sophisticated second-order methods are intractable, since the kernel matrix is exponentially large and hard to invert; furthermore, there are typically many support vectors, since they are indexed by elements of $\mathcal{Y}(x_i)$.

In contrast, we tackle (5) in *primal* form. Rather than repeatedly calling off-the-shelf solvers for (2), we propose a stand-alone online algorithm with runtime comparable to that of solving a *single* instance of (2) by fast online methods. This paradigm shift paves the way for extending MKL to structured prediction, a vast unexplored territory.

3 ONLINE PROXIMAL ALGORITHMS FOR KERNEL LEARNING

3.1 An Online Proximal Gradient Scheme

The general algorithmic scheme that we propose and analyze in this paper is presented as Alg. 1. It deals (in an online fashion[1]) with problems of the form

$$\min_{\boldsymbol{\theta} \in \Theta} \lambda \Omega(\boldsymbol{\theta}) + \tfrac{1}{N} \textstyle\sum_{i=1}^N L(\boldsymbol{\theta}; x_i, y_i), \quad (9)$$

where $\Theta \subseteq \mathcal{H}$ is a convex set and the regularizer Ω has a composite form $\Omega(\boldsymbol{\theta}) = \sum_{j=1}^J \Omega_j(\boldsymbol{\theta})$. This encompasses all formulations described in §2.1–2.2: standard ℓ_2-regularized SVMs and CRFs, group LASSO, and sparse and non-sparse variants of MKL. For all these, we have $\Theta = \mathcal{H}$ and $J = 1$. Moreover, with $J > 1$ it allows new variants of block-norm regularization, as we will discuss in §3.2.

Alg. 1 is similar to stochastic gradient descent (SGD, Bottou 1991), in that it also performs "noisy" (sub-)gradient steps[2] by looking only at a single instance (line 4). Hence, as SGD, it is also suitable for problems with large N. The difference is that these subgradients are only w.r.t. the loss function L, *i.e.*, they ignore the regularizer Ω.

In turn, each round of Alg. 1 makes J proximal steps, one per each term Ω_j (line 7). Given a function $\Phi : \mathcal{H} \to \mathbb{R}$,

[1] For simplicity, we focus on the pure online setting, *i.e.*, each parameter update uses a single observation; analogous algorithms may be derived for the batch and mini-batch cases.

[2] Given a convex function $\Phi : \mathcal{H} \to \mathbb{R}$, its *subdifferential* at $\boldsymbol{\theta}$ is the set $\partial\Phi(\boldsymbol{\theta}) \triangleq \{\boldsymbol{g} \in \mathcal{H} \mid \forall \boldsymbol{\theta}' \in \mathcal{H},\ \Phi(\boldsymbol{\theta}') - \Phi(\boldsymbol{\theta}) \geq \langle \boldsymbol{g}, \boldsymbol{\theta}' - \boldsymbol{\theta} \rangle\}$, the elements of which are the *subgradients*.

the Φ-proximity operator (Moreau, 1962) is defined as:

$$\text{prox}_\Phi(\boldsymbol{\theta}) \quad = \quad \arg\min_{\boldsymbol{\xi}\in\mathcal{H}} \frac{1}{2}\|\boldsymbol{\xi}-\boldsymbol{\theta}\|^2 + \Phi(\boldsymbol{\xi}). \quad (10)$$

For sparsity-promoting regularizers, such as $\Phi(\boldsymbol{\xi}) = \|\boldsymbol{\xi}\|_1$ or $\Phi(\boldsymbol{\xi}) = \|\boldsymbol{\xi}\|_{2,1}$, prox_Φ has a shrinkage and thresholding effect on the parameter vector, pushing Alg. 1 to return sparse solutions. This does not usually happen in standard SGD: since those regularizers are non-differentiable at the origin, each subgradient step in SGD causes oscillation and the final solution is rarely sparse. Contrarily, Alg. 1 is particularly appropriate for learning sparse models.

Finally, the projection step (line 9) can be used as a trick to accelerate convergence: for example, if we take the unconstrained version of (9) and we know beforehand that the optimum lies in a convex set Θ, then the constraint $\boldsymbol{\theta} \in \Theta$ is "vacuous," but line 9 ensures that each iterate $\boldsymbol{\theta}_t$ is confined to a bounded region containing the optimum. This idea is also used in PEGASOS (Shalev-Shwartz et al., 2007).

Before analyzing Alg. 1, we remark that it includes, as particular cases, many well-known online learners:

- if $\Omega = 0$ and $\eta_t \propto \frac{1}{\sqrt{t}}$, Alg. 1 is the online projected subgradient algorithm of Zinkevich (2003);
- if $\Omega = 0$, $L = L_{\text{SVM}} + \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2$, and $\eta_t = \frac{1}{\lambda t}$, Alg. 1 becomes PEGASOS, a popular algorithm for learning SVMs that has been extended to structured prediction (Shalev-Shwartz et al., 2007, 2010);
- If $\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1$, Alg. 1 was named truncated gradient descent and studied by Langford et al. (2009);
- If $J = 1$, Alg. 1 coincides with FOBOS (Duchi and Singer, 2009), which was used for learning SVMs and also for group-LASSO (but not for structured prediction).

In §3.4, we show how to kernelize Alg. 1 and apply it to sparse MKL. The case $J > 1$ has applications in variants of MKL or group-LASSO with composite regularizers (Tomioka and Suzuki, 2010; Friedman et al., 2010; Bach, 2008b; Zhao et al., 2008). In some of those cases, the proximity operators of $\Omega_1, ..., \Omega_J$ are easier to compute than their sum Ω, making Alg. 1 more suitable than FOBOS.

3.2 Proximity Operators of Block-Norm Regularizers

For Alg. 1 to handle the MKL and group-LASSO problems (described in §2.2), it needs to compute the proximal steps for block-norm regularizers. The following proposition (proved in App. A) is crucial for this purpose.

Proposition 1 *If $\Phi(\boldsymbol{\theta}) = \varphi((\|\boldsymbol{\theta}_m\|)_{m=1}^M)$ depends on each block only through its ℓ_2-norm, where $\varphi : \mathbb{R}^M \to \mathbb{R}$, then, $[\text{prox}_\Phi(\boldsymbol{\theta})]_m = [\text{prox}_\varphi(\|\boldsymbol{\theta}_1\|, \ldots, \|\boldsymbol{\theta}_M\|)]_m(\boldsymbol{\theta}_m/\|\boldsymbol{\theta}_m\|)$.*

Algorithm 1 Online Proximal Algorithm

1: **input:** dataset $\mathcal{D}$, parameter λ, number of rounds T, learning rate sequence $(\eta_t)_{t=1,\ldots,T}$
2: initialize $\boldsymbol{\theta}_1 = \mathbf{0}$; set $N = |\mathcal{D}|$
3: **for** $t = 1$ **to** T **do**
4: take training pair (x_t, y_t) and obtain a subgradient $\boldsymbol{g} \in \partial L(\boldsymbol{\theta}_t; x_t, y_t)$
5: $\tilde{\boldsymbol{\theta}}_t = \boldsymbol{\theta}_t - \eta_t \boldsymbol{g}$ (gradient step)
6: **for** $j = 1$ **to** J **do**
7: $\tilde{\boldsymbol{\theta}}_{t+\frac{j}{J}} = \text{prox}_{\eta_t \lambda \Omega_j}(\tilde{\boldsymbol{\theta}}_{t+\frac{j-1}{J}})$ (proximal step)
8: **end for**
9: $\boldsymbol{\theta}_{t+1} = \Pi_\Theta(\tilde{\boldsymbol{\theta}}_{t+1})$ (projection step)
10: **end for**
11: **output:** the last model $\boldsymbol{\theta}_{T+1}$ or the averaged model $\bar{\boldsymbol{\theta}} = \frac{1}{T}\sum_{t=1}^T \boldsymbol{\theta}_t$

Hence, any $\ell_{2,q}^r$-proximity operator can be reduced to an ℓ_q^r one: its effect is to scale the weights of each block by an amount that depends on the latter. Examples follow.

Group-LASSO. This corresponds to $q = r = 1$, so we are left with the problem of computing the $\tau\|\cdot\|_1$-proximity operator, which has a well-known closed form solution: the soft-threshold function (Donoho and Johnstone, 1994),

$$\text{prox}_{\tau\|\cdot\|_1}(\boldsymbol{b}) = \text{soft}(\boldsymbol{b}, \tau), \quad (11)$$

where $[\text{soft}(\boldsymbol{b}, \tau)]_k \triangleq \text{sgn}(b_k) \cdot \max\{0, |b_k| - \tau\}$.

Sparse MKL. This corresponds to $q = 1$, $r = 2$, and there are two options: one is to transform the problem back into group-LASSO, by removing the square from Ω_{MKL} (as pointed out in §2, these two problems are equivalent in the sense that they have the same regularization path); the other option (that we adopt) is to tackle Ω_{MKL} directly.[3] Prop. 1 enables reducing the evaluation of a $\|\cdot\|_{2,1}^2$-proximity operator to that of a *squared* ℓ_1. However the squared ℓ_1 is not separable (unlike ℓ_1), hence the proximity operator cannot be evaluated coordinatewise as in (11). This apparent difficulty has led some authors (*e.g.*, Suzuki and Tomioka 2009) to stick with the first option. However, despite the non-separability of ℓ_1^2, this proximal step can still be efficiently computed, as shown in Alg. 2. This algorithm requires sorting the weights of each group, which has $O(M \log M)$ cost. Correctness of this algorithm is shown in App. E.[4]

Non-Sparse MKL. For the case $q \geq 1$ and $r = 2$, a direct evaluation of $\text{prox}_{\tau\|\cdot\|_{2,q}^2}$ is more involved. It seems advantageous to transform this problem into an equivalent one, which uses a separable $\ell_{2,q}^q$ regularizer instead (the two problems are also equivalent up to a change in

[3] This makes possible the comparison with other MKL algorithms, for the same values of λ, as reported in §4.

[4] A similar algorithm was proposed independently by Kowalski and Torrésani (2009) in a different context.

Algorithm 2 Proximity Operator of ℓ_1^2

1: **input:** vector $\mathbf{x} \in \mathbb{R}^M$ and parameter $\lambda > 0$
2: sort the entries of $|\mathbf{x}|$ into $\mathbf{y}$ (yielding $y_1 \geq \ldots \geq y_M$)
3: set $\rho = \max \left\{ j \in \{1, ..., M\} \,|\, y_j - \frac{\lambda}{1+j\lambda} \sum_{r=1}^{j} y_r > 0 \right\}$
4: **output:** $\mathbf{z} = \text{soft}(\mathbf{x}, \tau)$, where $\tau = \frac{\lambda}{1+\rho\lambda} \sum_{r=1}^{\rho} y_r$

the regularization constant). The resulting proximal step amounts to solving (on b) M scalar equations of the form $b - b_0 + \tau q b^{q-1} = 0$, also valid for $q \geq 2$ (unlike the method described by Kloft et al. 2010). This can be done very efficiently using Newton's method.

Other variants. Many other variants of MKL and group-LASSO can be handled by Alg. 1, with $J > 1$. For example, the elastic net MKL (Tomioka and Suzuki, 2010) uses a sum of two regularizers, $\frac{\sigma}{2}\|.\|^2 + \frac{1-\sigma}{2}\|.\|_{2,1}^2$. In hierarchical LASSO and group-LASSO with overlaps (Bach, 2008b; Zhao et al., 2008; Jenatton et al., 2009), each feature may appear in more than one group. Alg. 1 handles these problems seamlessly by enabling a proximal step for each group.[5] Sparse group-LASSO (Friedman et al., 2010) simultaneously promotes group-sparsity and sparsity *within* each group, via $\sigma\|.\|_{2,1} + (1-\sigma)\|.\|_1$ regularization; Alg. 1 can handle this regularizer by using two proximal steps, both involving simple soft-thresholding: one at the group level, and another within each group.

3.3 Regret, Convergence, and Generalization Bounds

We next show that, for a convex loss L and under standard assumptions, Alg. 1 converges up to ϵ precision, with high confidence, in $O(1/\epsilon^2)$ iterations. If L or Ω are strongly convex,[6] this bound is improved to $\tilde{O}(1/\epsilon)$, where $\tilde{O}$ hides logarithmic terms. Our proofs combine tools of online convex programming (Zinkevich, 2003; Hazan et al., 2007) and classical results about proximity operators (Moreau, 1962). The key is the following lemma (proved in App. B).

Lemma 2 *Assume that L is convex and G-Lipschitz[7] on Θ, and that $\Omega = \sum_{j=1}^{J} \Omega_j$ satisfies the following conditions:* **(i)** *each Ω_j is convex;* **(ii)** *$\forall \boldsymbol{\theta} \in \Theta, \forall j' < j$, $\Omega_{j'}(\boldsymbol{\theta}) \geq \Omega_{j'}(\text{prox}_{\lambda\Omega_j}(\boldsymbol{\theta}))$ (each proximity operator $\text{prox}_{\lambda\Omega_j}$ does not increase the previous $\Omega_{j'}$);* **(iii)** *$\Omega(\boldsymbol{\theta}) \geq \Omega(\Pi_\Theta(\boldsymbol{\theta}))$ (projecting the argument onto Θ does not increase Ω).*

Then, for any $\bar{\boldsymbol{\theta}} \in \Theta$, at each round t of Alg. 1,

$$L(\boldsymbol{\theta}_t) + \lambda\Omega(\boldsymbol{\theta}_{t+1}) \leq L(\bar{\boldsymbol{\theta}}) + \lambda\Omega(\bar{\boldsymbol{\theta}}) + \epsilon, \tag{12}$$

where $\epsilon = \frac{\eta_t}{2}G^2 + \frac{\|\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}_t\|^2 - \|\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}_{t+1}\|^2}{2\eta_t}$.

If L is σ-strongly convex this bound can be strengthened to

$$L(\boldsymbol{\theta}_t) + \lambda\Omega(\boldsymbol{\theta}_{t+1}) \leq L(\bar{\boldsymbol{\theta}}) + \lambda\Omega(\bar{\boldsymbol{\theta}}) + \epsilon', \tag{13}$$

where $\epsilon' = \epsilon - \frac{\sigma}{2}\|\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}_t\|^2$.

A related, but less tight, bound for $J = 1$ was derived by Duchi and Singer (2009); instead of our term $\frac{\eta}{2}G^2$ in ϵ, theirs is $7\frac{\eta}{2}G^2$.[8] When $\Omega = \|\cdot\|_1$, FOBOS becomes the truncated gradient algorithm of Langford et al. (2009) and our bound matches the one therein derived, closing the gap between Duchi and Singer (2009) and Langford et al. (2009). Finally, note that the conditions **(i)–(iii)** are not restrictive: they hold whenever the proximity operators are shrinkage functions—*e.g.*, if $\Omega_j(\boldsymbol{\theta}) = \|\boldsymbol{\theta}_{G_j}\|_{2,p_j}^{q_j}$, with $p_j, q_j \geq 1$ and $G_j \subseteq \{1, \ldots, M\}$, which also covers the overlapping group case where $\bigcap_{j=1}^{J} G_j \neq \varnothing$.

We next use Lemma 2 to characterize Alg. 1 in terms of its cumulative regret w.r.t. the best fixed hypothesis, *i.e.*,

$$\text{Reg}_T \triangleq \sum_{t=1}^{T} \left(\lambda\Omega(\boldsymbol{\theta}_t) + L(\boldsymbol{\theta}_t; x_t, y_t)\right) - \min_{\boldsymbol{\theta}\in\Theta} \sum_{t=1}^{T} \left(\lambda\Omega(\boldsymbol{\theta}) + L(\boldsymbol{\theta}; x_t, y_t)\right). \tag{14}$$

Proposition 3 (regret bounds) *Assume the conditions of Lemma 2, along with $\Omega \geq 0$ and $\Omega(\mathbf{0}) = 0$. Then:*

1. *Running Alg. 1 with fixed learning rate η yields*

$$\text{Reg}_T \leq \frac{\eta T}{2}G^2 + \frac{\|\boldsymbol{\theta}^*\|^2}{2\eta}, \tag{15}$$

where $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Theta} \sum_{t=1}^{T} \left(\lambda\Omega(\boldsymbol{\theta}) + L(\boldsymbol{\theta}; x_t, y_t)\right)$. Setting $\eta = \|\boldsymbol{\theta}^*\|/(G\sqrt{T})$ yields a sublinear regret of $\|\boldsymbol{\theta}^*\|G\sqrt{T}$. (Note that this requires knowing in advance $\|\boldsymbol{\theta}^*\|$ and the number of rounds T.)

2. *Assume that Θ is bounded with diameter F (i.e., $\forall \boldsymbol{\theta}, \boldsymbol{\theta}' \in \Theta$, $\|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \leq F$). Let the learning rate be $\eta_t = \eta_0/\sqrt{t}$, with arbitrary $\eta_0 > 0$. Then,*

$$\text{Reg}_T \leq \left(\frac{F^2}{2\eta_0} + G^2\eta_0\right)\sqrt{T}. \tag{16}$$

With $\eta_0 = F/(\sqrt{2}G)$, we obtain $\text{Reg}_T \leq FG\sqrt{2T}$.

3. *If L is σ-strongly convex, and $\eta_t = 1/(\sigma t)$, we obtain a logarithmic regret bound:*

$$\text{Reg}_T \leq G^2(1 + \log T)/(2\sigma). \tag{17}$$

Proof: See App. C. ∎

[5] Recently, a lot of effort has been placed on ways for computing the proximal step for regularizers with overlapping groups (Liu and Ye, 2010a,b; Mairal et al., 2010). Alg. 1 suggests an alternative approach: split the regularizer into several non-overlapping parts and apply sequential proximal steps. Although in general $\text{prox}_{\Omega_J} \circ \ldots \circ \text{prox}_{\Omega_1} \neq \text{prox}_{\Omega_J \circ \ldots \circ \Omega_1}$, Alg. 1 is still applicable, as we will see in §3.3.

[6] Φ is *σ-strongly convex* in $\mathcal{S}$ if $\forall \boldsymbol{\theta} \in \mathcal{S}, \forall \boldsymbol{g} \in \partial\Phi(\boldsymbol{\theta}), \forall \boldsymbol{\theta}' \in \mathcal{H}$, $\Phi(\boldsymbol{\theta}') \geq \Phi(\boldsymbol{\theta}) + \langle \boldsymbol{g}, \boldsymbol{\theta}' - \boldsymbol{\theta}\rangle + \frac{\sigma}{2}\|\boldsymbol{\theta}' - \boldsymbol{\theta}\|^2$.

[7] Φ is *G-Lipschitz* in $\mathcal{S}$ if $\forall \boldsymbol{\theta} \in \mathcal{S}, \forall \boldsymbol{g} \in \partial\Phi(\boldsymbol{\theta})$, $\|\boldsymbol{g}\| \leq G$.

[8] This can be seen from their Eq. 9 with $A = 0$ and $\eta_t = \eta_{t+\frac{1}{2}}$.

Similarly to other analyses of online learning algorithms, once an online-to-batch conversion is specified, regret bounds allow us to obtain PAC bounds on optimization and generalization errors. The following proposition can be proved using the techniques of Cesa-Bianchi et al. (2004) and Shalev-Shwartz et al. (2007).

Proposition 4 (optimization and estimation error) *If the assumptions of Prop. 3 hold and* $\eta_t = \eta_0/\sqrt{t}$ *as in item 2 in Prop. 3, then the version of Alg. 1 that returns the averaged model solves (9) with* ϵ*-accuracy in* $T = O((F^2G^2 + \log(1/\delta))/\epsilon^2)$ *iterations with probability at least* $1-\delta$*. If* L *is also* σ*-strongly convex and* $\eta_t = 1/(\sigma t)$ *as in item 3 of Prop. 3, then, for the version of Alg. 1 that returns* $\boldsymbol{\theta}_{T+1}$*, we get* $T = \tilde{O}(G^2/(\sigma\delta\epsilon))$*. The generalization bounds are of the same orders.*

We now pause to examine some concrete cases. The requirement that the loss is G-Lipschitz holds for the hinge and logistic losses, where $G = 2\max_{u\in\mathcal{U}} \|\boldsymbol{\phi}(u)\|$ (see App. D). These losses are not strongly convex, and therefore Alg. 1 has only $O(1/\epsilon^2)$ convergence. If the *regularizer* Ω is σ-strongly convex, a possible workaround to obtain $\tilde{O}(1/\epsilon)$ convergence is to let L "absorb" that strong convexity by redefining $\tilde{L}(\boldsymbol{\theta}; x_t, y_t) = L(\boldsymbol{\theta}; x_t, y_t) + \sigma\|\boldsymbol{\theta}\|^2/2$. Since neither the $\ell_{2,1}$-norm nor its square are strongly convex, we cannot use this trick for the MKL case, but it *does* apply for non-sparse MKL ($\ell^2_{2,q}$-norms are strongly convex for $q > 1$) and for elastic MKL. Still, the $O(1/\epsilon^2)$ rate for MKL is competitive with the best batch algorithms that tackle the dual; *e.g.*, the method of Xu et al. (2009) achieves ϵ primal-dual gap in $O(1/\epsilon^2)$ iterations.[9] Some losses of interest (*e.g.*, the squared loss, or the modified loss $\tilde{L}$ above) are G-Lipschitz in any compact subset of $\mathcal{H}$ but not in $\mathcal{H}$. However, if the optimal solution is known to lie in some compact set Θ, we can run Alg. 1 with the projection step, making the analysis still applicable.

3.4 SPOM: Structured Prediction with Online MKL

The instantiation of Alg. 1 for structured prediction and $\Omega_{\text{MKL}}(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|^2_{2,1}$ yields the SPOM algorithm (Alg. 3). We consider $L = L_{\text{SVM}}$; adapting to any generalized linear model (*e.g.*, $L = L_{\text{CRF}}$) is straightforward. As discussed in the last paragraph of §3.3, the inclusion of a vacuous projection may accelerate convergence. Hence, an optional upper bound γ on $\|\boldsymbol{\theta}\|$ is accepted as input. Suitable values of γ for the SVM and CRF case are given in App. D.

In line 5, the scores of candidate outputs are computed blockwise; as described in §2.3, a factorization over parts is assumed and the scores are for partial output assignments. Line 6 gathers all these scores and decodes (loss-augmented inference for the SVM case, or marginal inference for the CRF case). Line 10 is where the block structure is taken into account, by applying a proximity operator which corresponds to a blockwise shrinkage/thresholding.

[9] On the other hand, batch proximal gradient methods for smooth losses can be accelerated to achieve $O(1/\sqrt{\epsilon})$ convergence in the primal objective (Beck and Teboulle, 2009).

Algorithm 3 SPOM

1: **input:** $\mathcal{D}$, λ, T, radius γ, learning rates $(\eta_t)_{t=1}^T$
2: initialize $\boldsymbol{\theta}^1 \leftarrow \mathbf{0}$
3: **for** $t = 1$ **to** T **do**
4: sample an instance x_t, y_t
5: compute scores for $m = 1, \ldots, M$:
$$f_m(x_t, y'_t) = \langle \boldsymbol{\theta}^t_m, \boldsymbol{\phi}_m(x_t, y'_t)\rangle$$
6: decode: $\hat{y}_t \in \operatorname{argmax}_{y'_t \in \mathcal{Y}(x)} \sum_{m=1}^M f_m(x_t, y'_t) + c(y'_t, y_t)$
7: Gradient step for $m = 1, \ldots, M$:
$$\tilde{\boldsymbol{\theta}}^t_m = \boldsymbol{\theta}^t_m - \eta_t(\boldsymbol{\phi}_m(x_t, \hat{y}_t) - \boldsymbol{\phi}_m(x_t, y_t))$$
8: compute weights for $m = 1, \ldots, M$: $\tilde{b}^t_m = \|\tilde{\boldsymbol{\theta}}^t_m\|$
9: shrink weights $\mathbf{b}^t = \operatorname{prox}_{\eta_t\lambda\|\cdot\|^2_{2,1}}(\tilde{\mathbf{b}}^t)$ with Alg. 2
10: Proximal step for $m = 1, \ldots, M$: $\tilde{\boldsymbol{\theta}}^{t+1}_m = b^t_m/\tilde{b}^t_m \cdot \tilde{\boldsymbol{\theta}}^t_m$
11: Projection step: $\boldsymbol{\theta}^{t+1} = \tilde{\boldsymbol{\theta}}^{t+1} \cdot \min\{1, \gamma/\|\tilde{\boldsymbol{\theta}}^{t+1}\|\}$
12: **end for**
13: compute $\beta_m \propto \|\boldsymbol{\theta}^{T+1}_m\|$ for $m = 1, \ldots, M$
14: return $\boldsymbol{\beta}$, and the last model $\boldsymbol{\theta}^{T+1}$

Although Alg. 3 is described with explicit features, it can be kernelized, as shown next (one can also use explicit features in some groups, and implicit in others). Observe that the parameters of the mth block after round t can be written as $\boldsymbol{\theta}^{t+1}_m = \sum_{s=1}^t \alpha^{t+1}_{ms}(\boldsymbol{\phi}_m(x_s, y_s) - \boldsymbol{\phi}_m(x_s, \hat{y}_s))$, where

$$\begin{aligned}\alpha^{t+1}_{ms} &= \eta_s \prod_{r=s}^t \left((b^r_m/\tilde{b}^r_m)\min\{1, \gamma/\|\tilde{\boldsymbol{\theta}}^{r+1}\|\}\right) \\ &= \begin{cases} \eta_t(b^t_m/\tilde{b}^t_m)\min\{1, \gamma/\|\tilde{\boldsymbol{\theta}}^{t+1}\|\} & \text{if } s = t \\ \alpha^t_{ms}(b^t_m/\tilde{b}^t_m)\min\{1, \gamma/\|\tilde{\boldsymbol{\theta}}^{t+1}\|\} & \text{if } s < t. \end{cases}\end{aligned}$$

Therefore, the inner products in line 5 can be kernelized. The cost of this step is $O(\min\{N, t\})$, instead of the $O(d_m)$ (where d_m is the dimension of the mth block) for the explicit feature case. After the decoding step (line 6), the supporting pair $(x_t, \hat{y}_t)$ is stored. Lines 9, 11 and 13 require the *norm* of each group, which can be manipulated using kernels: indeed, after each gradient step (line 7), we have (denoting $u_t = (x_t, y_t)$ and $\hat{u}_t = (x_t, \hat{y}_t)$)

$$\begin{aligned}\|\tilde{\boldsymbol{\theta}}^t_m\|^2 &= \|\boldsymbol{\theta}^t_m\|^2 - 2\eta_t\langle\boldsymbol{\theta}^t_m, \boldsymbol{\phi}_m(u_t)\rangle + \\ &\quad \eta_t^2\|\boldsymbol{\phi}_m(\hat{u}_t) - \boldsymbol{\phi}_m(u_t)\|^2 \\ &= \|\boldsymbol{\theta}^t_m\|^2 - 2\eta_t f_m(\hat{u}_t) + \\ &\quad \eta_t^2(K_m(u_t, u_t) + K_m(\hat{u}_t, \hat{u}_t) - 2K_m(u_t, \hat{u}_t));\end{aligned}$$

and the proximal and projection steps merely scale these norms. When the algorithm terminates, it returns the kernel coefficients $\boldsymbol{\beta}$ and the sequence (α^{T+1}_{mt}).

In case of sparse explicit features, an implementation trick analogous to the one used by Shalev-Shwartz et al. (2007) (where each $\boldsymbol{\theta}_m$ is represented by its norm and an unnormalized vector) can substantially reduce the amount of computation. In the case of implicit features with a sparse kernel matrix, a sparse storage of this matrix can also significantly speed up the algorithm, eliminating its dependency on N in line 5. We do that in the experiments (§4). Note that all steps involving block-specific computation can be carried out in parallel using multi-core machines, making Alg. 3 capable of handling many kernels (large M).

4 EXPERIMENTS

We evaluate SPOM (Alg. 3) on two structured prediction tasks: a sequence labeling task (handwriting recognition) and a natural language parsing task (dependency parsing).

4.1 Handwriting Recognition

We use the OCR dataset of Taskar et al. (2003), which has a total of 6,877 words and 52,152 characters.[10] Each character (the input) is a 16×8 binary image, with one of 26 labels (`a-z`, the output to predict). As Taskar et al. (2003), we address this sequence labeling problem with a structural SVM; however, we use the SPOM algorithm to *learn* the kernel from the data. We use an indicator basis function to represent the correlation between consecutive outputs.

MKL versus average kernel. Our first experiment (upper part of Table 1; solid lines in Figure 1) compares linear, quadratic, and Gaussian kernels, either used individually, combined via a simple average, or with MKL (via SPOM). The results show that MKL outperforms the others by $\geq 2\%$, and that learning the bigram weight β_0 (§2.3) does not make any difference. Figure 1 shows that the MKL approach is able to achieve an accurate model sooner.

Feature and kernel sparsity. The second experiment aims at showing SPOM's ability to exploit both *feature* and *kernel* sparsity. We learn a combination of a linear kernel (explicit features) with a generalized B_1-spline kernel, given by $K(\mathbf{x}, \mathbf{x}') = \max\{0, 1 - \|\mathbf{x} - \mathbf{x}'\|/h\}$, with h chosen so that the kernel matrix has $\sim 95\%$ zeros. The rationale is to combine the strength of a simple feature-based kernel with that of one depending only on a few nearest neighbors. The results (bottom part of Tab. 1) show that the MKL outperforms by $\sim 10\%$ the individual kernels, and by more than 2% the averaged kernel. The accuracy is not much worse than the best one obtained in the previous experiment, while the runtime is much faster (15 versus 279 seconds). Figure 1 (dashed lines) is striking. in showing the ability of producing a reasonable model very fast.

SPOM versus wrapper-based methods. To assess the effectiveness of SPOM as a kernel learning algorithm, we compare it with two wrapper-based MKL algorithms: a Gauss-Seidel method alternating between optimizing the SVM and the kernel coefficients (see, e.g., Kloft et al. 2010), and a gradient method (*SimpleMKL*, Rakotomamonjy et al. 2008).[11] In both cases, the SVMs were tackled with structured PEGASOS. Despite the fact that each SVM is strongly convex and has $O(\frac{1}{\epsilon})$ convergence, its combination with a outer loop becomes time-consuming, even if we warm-start each SVM. This is worse when regularization is weak (small λ). In contrast, SPOM, with its overall $O(\frac{1}{\epsilon^2})$ convergence, is stable and very fast to converge to a near-optimal region, as attested in Fig. 2. This suggests its usefulness in settings where each epoch is costly.

Table 1: Results for handwriting recognition. Averages over 10 runs (same folds as Taskar et al. (2003), training on one and testing on the others). The linear and quadratic kernels are normalized to unit diagonal. In all cases, 20 epochs were used, with η_0 in (16) picked from $\{0.01, 0.1, 1, 10\}$ by selecting the one that most decreases the objective after 5 epochs. In all cases, the regularization coefficient $C = 1/(\lambda N)$ was chosen with 5-fold cross-validation from $\{0.1, 1, 10, 10^2, 10^3, 10^4\}$.

Kernel	Training Runtimes	Test Acc. (per char.)
Linear (L)	6 sec.	$71.8 \pm 3.9\%$
Quadratic (Q)	116 sec.	$85.5 \pm 0.3\%$
Gaussian (G) ($\sigma^2 = 5$)	123 sec.	$84.1 \pm 0.4\%$
Average $(L + Q + G)/3$	118 sec.	$84.3 \pm 0.3\%$
MKL $\beta_1 L + \beta_2 Q + \beta_3 G$	**279 sec.**	$\mathbf{87.5 \pm 0.3\%}$
MKL $\beta_0, \beta_1 L + \beta_2 Q + \beta_3 G$	**282 sec.**	$\mathbf{87.5 \pm 0.4\%}$
B_1-Spline (B_1)	8 sec.	$75.4 \pm 0.9\%$
Average $(L + B_1)/2$	15 sec.	$83.0 \pm 0.3\%$
MKL $\beta_1 L + \beta_2 B_1$	**15 sec.**	$\mathbf{85.2 \pm 0.3\%}$
MKL $\beta_0, \beta_1 L + \beta_2 B_1$	**16 sec.**	$\mathbf{85.2 \pm 0.3\%}$

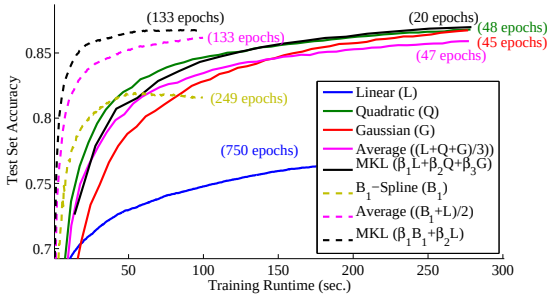

Figure 1: Test set accuracies of single kernel and multiple kernel methods as a function of the training stopping times.

4.2 Dependency Parsing

We trained non-projective dependency parsers for English, using the CoNLL-2008 shared task dataset (Surdeanu et al., 2008), in a total of 39,278 training and 2,399 test sentences. The output to be predicted from each input sentence is the

[10] Available at `www.cis.upenn.edu/~taskar/ocr`.

[11] We used the code in `http://asi.insa-rouen.fr/enseignants/~arakotom/code/mklindex.html`.

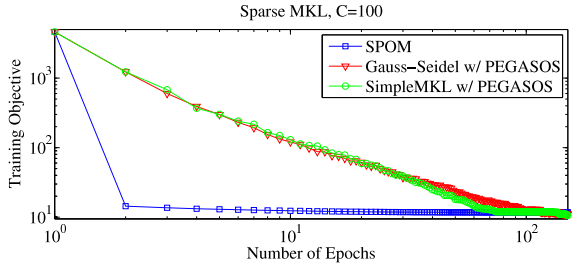

Figure 2: Comparison between SPOM (Alg. 3) and two wrapper based methods in the OCR dataset, with $C = 100$. The wrapper-based methods run 20 epochs of PEGASOS in their first SVM call; subsequent calls run 3 epochs with warm-starting. With only 20–30 passes over the data, SPOM approaches a region very close to the optimum; the wrapper-based methods need about 100 epochs.

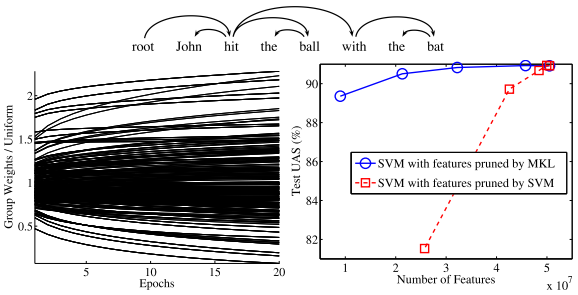

Figure 3: Top: a dependency tree (adapted from McDonald et al. 2005). Bottom left: group weights along the epochs of Alg. 3. Bottom right: results of standard SVMs with feature templates of sizes $\{107, 207, 307, 407, 507\}$, either selected via a standard SVM or by MKL, via SPOM (the UAS—*unlabeled attachment score*—is the fraction of non-punctuation words whose head was correctly assigned.)

set of dependency arcs, linking *heads* to *modifiers*, that must define a spanning tree (see example in Fig. 3). We use arc-factored models, for which exact inference is tractable via minimum spanning tree algorithms (McDonald et al., 2005). We defined $M = 507$ feature templates for each candidate arc by conjoining the words, lemmas, and parts-of-speech of the head and the modifier, as well as the parts-of-speech of the surrounding words, and the distance and direction of attachment. This instantiates > 50 million features. The feature vectors associated with each candidate arc, however, are very sparse and this is exploited in the implementation. We ran SPOM with explicit features, with each group standing for a feature template. MKL (via SPOM) did not outperform a standard SVM in this experiment (90.67% against 90.92%); however, it showed good performance at pruning irrelevant feature templates (see Fig. 3, bottom right). Besides *interpretability*, which may be useful for the understanding of the syntax of natural languages, and memory efficiency (creating a smaller model), this pruning is also appealing in a two-stage architecture, where a learner at a second stage will only need to handle a small fraction of the templates initially hypothesized.

5 RELATED WORK

Discriminative learning of structured predictors has been an active area of research since the seminal works of Lafferty et al. (2001); Collins (2002); Altun et al. (2003); Taskar et al. (2003); Tsochantaridis et al. (2004).

Following the introduction of MKL by Lanckriet et al. (2004), a string of increasingly efficient algorithms were proposed (Sonnenburg et al., 2006; Zien and Ong, 2007; Rakotomamonjy et al., 2008; Chapelle and Rakotomamonjy, 2008; Xu et al., 2009; Suzuki and Tomioka, 2009; Kloft et al., 2010), although none was applied to structured prediction. Group LASSO is due to Bakin (1999); Yuan and Lin (2006), after which many variants and algorithms appeared, all working in batch form: Bach (2008b); Zhao et al. (2008); Jenatton et al. (2009); Friedman et al. (2010).

Independently from us, Jie et al. (2010) proposed an online algorithm for multi-class MKL (called OM-2), which perform coordinate descent in the dual. Our algorithm is more flexible: while OM-2 is limited to $\ell^2_{2,q}$-regularization, with $q > 1$, and becomes slow when $q \to 1$, we efficiently handle the $\ell^2_{2,1}$ case as well as arbitrary composite regularizers.

Proximity operators are well known in convex analysis and optimization (Moreau, 1962; Lions and Mercier, 1979) and have recently seen wide use in signal processing; see Combettes and Wajs (2006), Wright et al. (2009), and references therein. Specifically, the theory of proximity operators (see App. A) underlies the proofs of our regret bounds (Prop. 3).

6 CONCLUSIONS

We proposed a new method for kernel learning and feature template selection of structured predictors. To accomplish this, we introduced a class of online proximal algorithms applicable to many variants of MKL and group-LASSO. We studied its convergence rate and used the algorithm for learning the kernel in structured prediction tasks.

Our work may impact other problems. In structured prediction, the ability to promote structural sparsity can be useful for learning simultaneously the structure and parameters of graphical models. We will explore this in future work.

Acknowledgements: A. M. was supported by a grant from FCT/ICTI through the CMU-Portugal Program, and also by Priberam Informática. N. S. was supported by NSF CAREER IIS-1054319. E. X. was supported by AFOSR FA9550010247, ONR N000140910758, NSF CAREER DBI-0546594, NSF IIS-0713379, and an Alfred P. Sloan Fellowship. This work was partially supported by the FET programme (EU FP7), under the SIMBAD project (contract 213250), and by a FCT grant PTDC/EEA-TEL/72572/2006.

References

Altun, Y., Tsochantaridis, I., and Hofmann, T. (2003). Hidden Markov support vector machines. In *Proc. of ICML*.

Bach, F. (2008a). Consistency of the group Lasso and multiple kernel learning. *JMLR*, 9:1179–1225.

Bach, F. (2008b). Exploring large feature spaces with hierarchical multiple kernel learning. *NIPS*, 21.

Bach, F., Lanckriet, G., and Jordan, M. (2004). Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*.

Bakin, S. (1999). *Adaptive regression and model selection in data mining problems*. Australian National University.

Bakır, G., Hofmann, T., Schölkopf, B., Smola, A., Taskar, B., and Vishwanathan, S. (2007). *Predicting Structured Data*. The MIT Press.

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202.

Bottou, L. (1991). Stochastic gradient learning in neural networks. In *Proc. of Neuro-Nîmes*.

Bottou, L. and Bousquet, O. (2007). The tradeoffs of large scale learning. *NIPS*, 20.

Cesa-Bianchi, N., Conconi, A., and Gentile, C. (2004). On the generalization ability of on-line learning algorithms. *IEEE Trans. on Inf. Theory*, 50(9):2050–2057.

Chapelle, O. and Rakotomamonjy, A. (2008). Second order optimization of kernel parameters. In *Proc. of the NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*.

Collins, M. (2002). Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proc. of EMNLP*.

Collins, M., Globerson, A., Koo, T., Carreras, X., and Bartlett, P. (2008). Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks. *JMLR*.

Combettes, P. and Wajs, V. (2006). Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4(4):1168–1200.

Donoho, D. and Johnstone, J. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the L1-ball for learning in high dimensions. In *ICML*.

Duchi, J. and Singer, Y. (2009). Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2873–2908.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). A note on the group lasso and a sparse group lasso. Technical report, arXiv:1001.0736v1.

Hazan, E., Agarwal, A., and Kale, S. (2007). Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192.

Hofmann, T., Scholkopf, B., and Smola, A. J. (2008). Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171.

Jenatton, R., Audibert, J.-Y., and Bach, F. (2009). Structured variable selection with sparsity-inducing norms. Technical report, arXiv:0904.3523.

Jie, L., Orabona, F., Fornoni, M., Caputo, B., and Cesa-Bianchi, N. (2010). OM-2: An online multi-class Multi-Kernel Learning algorithm. In *Proc. of the 4th IEEE Online Learning for Computer Vision Workshop*, pages 43–50. IEEE.

Kloft, M., Brefeld, U., Sonnenburg, S., and Zien, A. (2010). Non-Sparse Regularization and Efficient Training with Multiple Kernels. Technical report, arXiv:1003.0079.

Kowalski, M. and Torrésani, B. (2009). Structured sparsity: From mixed norms to structured shrinkage. In *Workshop on Signal Processing with Adaptive Sparse Structured Representations*.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72.

Langford, J., Li, L., and Zhang, T. (2009). Sparse online learning via truncated gradient. *JMLR*, 10:777–801.

Lions, P. and Mercier, B. (1979). Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979.

Liu, J. and Ye, J. (2010a). Fast Overlapping Group Lasso. Technical report, arXiv:1009.0306.

Liu, J. and Ye, J. (2010b). Moreau-yosida regularization for grouped tree structure learning. In Lafferty, J., Williams, C. K. I., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *NIPS 23*, pages 1459–1467.

Mairal, J., Jenatton, R., Obozinski, G., and Bach, F. (2010). Network flow algorithms for structured sparsity. In Lafferty, J., Williams, C. K. I., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *NIPS 23*, pages 1558–1566.

McDonald, R. T., Pereira, F., Ribarov, K., and Hajic, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

Moreau, J. (1962). Fonctions convexes duales et points proximaux dans un espace hilbertien. *CR Acad. Sci. Paris Sér. A Math*, 255:2897–2899.

Rakotomamonjy, A., Bach, F., Canu, S., and Grandvalet, Y. (2008). SimpleMKL. *JMLR*, 9:2491–2521.

Ratliff, N., Bagnell, J., and Zinkevich, M. (2006). Subgradient methods for maximum margin structured learning. In *ICML Workshop on Learning in Structured Outputs Spaces*.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*.

Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. (2010). Pegasos: primal estimated sub-gradient solver for SVM. *Mathematical Programming*, 125.

Sonnenburg, S., Rätsch, G., Schäfer, C., and Schölkopf, B. (2006). Large scale MKL. *JMLR*, 7:1565.

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. (2008). The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. *Proc. of CoNLL*.

Suzuki, T. and Tomioka, R. (2009). SpicyMKL. Technical report, arXiv:0909.5026.

Taskar, B., Guestrin, C., and Koller, D. (2003). Max-margin Markov networks. In *NIPS*.

Tomioka, R. and Suzuki, T. (2010). Sparsity-accuracy trade-off in MKL. Technical report, arXiv:1001.2615v1.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *ICML*.

Vishwanathan, S. V. N., Schraudolph, N. N., Schmidt, M. W., and Murphy, K. P. (2006). Accelerated training of conditional random fields with stochastic gradient methods. In *Proc. of ICML*.

Wright, S., Nowak, R., and Figueiredo, M. (2009). Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493.

Xu, Z., Jin, R., King, I., and Lyu, M. (2009). An extended level method for efficient multiple kernel learning. *NIPS*, 21:1825–1832.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, 68(1):49.

Zhao, P., Rocha, G., and Yu, B. (2008). Grouped and hierarchical model selection through composite absolute penalties. *Annals of Statistics*.

Zien, A. and Ong, C. (2007). Multiclass multiple kernel learning. In *ICML*.

Zinkevich, M. (2003). Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *ICML*.