Online MKL for Structured Prediction

André F. T. Martins[*†] **Noah A. Smith**[*] **Eric P. Xing**[*]
[*]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Pedro M. Q. Aguiar[‡]
[‡]Instituto de Sistemas e Robótica,
Instituto Superior Técnico, Lisboa, Portugal

Mário A. T. Figueiredo[†]
[†]Instituto de Telecomunicações,
Instituto Superior Técnico, Lisboa, Portugal

1 Introduction

Structured prediction (SP) problems are characterized by strong interdependence among the output variables, usually with sequential, graphical, or combinatorial structure [17, 7]. Obtaining good predictors often requires a large effort in feature/kernel design and tuning (usually done via cross-validation). Because discriminative training of structured predictors can be quite slow, specially in large scale settings, it is appealing to learn the kernel function simultaneously.

In multiple kernel learning (MKL, [8]), 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 [14, 11, 6]. Unfortunately, extending such methods to large-scale SP raises practical hurdles: since the output space is large, so are the kernel matrices, and the number of support vectors; moreover, it becomes prohibitive to tackle the inner problem in its batch form, and one usually resorts to online algorithms [12, 3, 13]. The latter are fast learners but slow optimizers [2]; using them in the inner loop with early stopping may misguide the overall MKL optimization.

We propose instead a stand-alone online MKL algorithm which exploits the large-scale tradeoffs directly. The algorithm iterates between subgradient and proximal steps, and has important advantages: *(i)* it is simple, flexible, and compatible with sparse and non-sparse variants of MKL, *(ii)* it is adequate for SP, *(iii)* it offers regret, convergence, and generalization guarantees. Experimentally, the proposed algorithm yields near-optimal solutions faster than wrapper-based approaches (adapted to SP). We use the two following SP experimental testbeds: sequence labeling for handwritten text recognition, and natural language dependency parsing. The potential of our approach is shown in learning combinations of kernels from tens of thousands of training instances, with encouraging results in terms of speed, accuracy, and interpretability.

2 Structured Prediction (SP) and Multiple Kernel Learning (MKL)

In SP, to each input $x \in \mathcal{X}$ corresponds a set of possible outputs, $\mathcal{Y}(x) \subseteq \mathcal{Y}$; *e.g.*, in sequence labeling, $x \in \mathcal{X}$ is an observed sequence and $y \in \mathcal{Y}(x)$ is a sequence of labels. Let $\mathcal{U} \triangleq \{(x,y) \mid x \in \mathcal{X}, y \in \mathcal{Y}(x)\}$. Given a dataset $\mathcal{D} \triangleq \{(x_1,y_1),\ldots,(x_N,y_N)\} \subseteq \mathcal{U}$, the goal is to learn a *compatibility function* $f_{\boldsymbol{\theta}} : \mathcal{U} \rightarrow \mathbb{R}$ whose maximization yields accurate predictions on new inputs, $\hat{y}(x) = \arg\max_{y \in \mathcal{Y}(x)} f_{\boldsymbol{\theta}}(x,y)$, according to some cost function $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$. In this paper, we bring in the MKL framework [1, 8] and define

$$f_{\boldsymbol{\theta}}(x,y) \triangleq \langle \boldsymbol{\theta}, \boldsymbol{\phi}(x,y) \rangle_{\mathcal{H}} = \textstyle\sum_{m=1}^{M} \langle \boldsymbol{\theta}_m, \boldsymbol{\phi}_m(x,y) \rangle_{\mathcal{H}_m}, \tag{1}$$

where $\mathcal{H} = \bigoplus_{m=1}^{M} \mathcal{H}_m$ is a direct sum of RKHS, and $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are block-structured (parameter and feature) vectors. Learning is carried out by minimizing the regularized empirical risk,

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

where L is a convex loss (*e.g.*, the structured SVM hinge loss, $L(f_{\boldsymbol{\theta}}; x, y) \triangleq \max_{y' \in \mathcal{Y}(x)} f_{\boldsymbol{\theta}}(x,y') - f_{\boldsymbol{\theta}}(x,y) + \ell(y',y))$, $\Omega : \mathcal{H} \rightarrow \mathbb{R}_+$ is a regularizer, and $\lambda \geq 0$. Some examples are:

Algorithm 1 Online Proximal MKL

```
1: input: $\mathcal{D}$, kernels $\{K_m\}_{m=1}^M$, regularizers $\{\lambda_j \Omega_j\}_{j=1}^J$, number of rounds $T$, sequence $(\eta_t)_{t=1}^T$
2: Initialize $\boldsymbol{\theta}_1 = \mathbf{0}$; set $N = |\mathcal{D}|$
3: for $t = 1$ to $T$ do
4:     Take a training pair $(x_t, y_t)$ and obtain a subgradient $\boldsymbol{g}_{\boldsymbol{\theta}_t} = \partial L(f_{\boldsymbol{\theta}_t}; x_t, y_t)$
5:     Subgradient step: $\tilde{\boldsymbol{\theta}}_t = \boldsymbol{\theta}_t - \eta_t \boldsymbol{g}_{\boldsymbol{\theta}_t}$
6:     for $j = 1$ to $J$ do
7:         Proximal step: $\tilde{\boldsymbol{\theta}}_{t+j/J} = \mathrm{prox}_{\eta_t \lambda \Omega_j}(\tilde{\boldsymbol{\theta}}_{t+(j-1)/J})$
8:     end for
9:     Projection step (optional): $\boldsymbol{\theta}_{t+1} = \tilde{\boldsymbol{\theta}}_{t+1} \cdot \min\{1, \gamma / \|\tilde{\boldsymbol{\theta}}_{t+1}\|\}$
10: end for
11: output: the last model $f_{\boldsymbol{\theta}_{T+1}}$ or the averaged model $f_{\sum_{t=1}^T \boldsymbol{\theta}_t / T}$
```

- ℓ_2-regularization, $\Omega[f_{\boldsymbol{\theta}}] = \|\boldsymbol{\theta}\|^2/2$, recovers the standard SVM with kernel $\sum_{m=1}^M K_m$;
- block-norm $\ell_{2,1}$ regularization, $\Omega[f_{\boldsymbol{\theta}}] = \frac{1}{2}\|\boldsymbol{\theta}\|_{2,1}^2 = \frac{1}{2}(\sum_{m=1}^M \|\boldsymbol{\theta}_m\|_{\mathcal{H}_m})^2$, yields sparse MKL, which is equivalent to learning a kernel of the form $\sum_{m=1}^M \beta_m K_m$ with $\boldsymbol{\beta} \geq \mathbf{0}$ and $\|\boldsymbol{\beta}\|_1 \leq 1$;
- block-norm $\ell_{2,q}$ regularization, $\Omega[f_{\boldsymbol{\theta}}] = \frac{1}{2}\|\boldsymbol{\theta}\|_{2,q}^2 = \frac{1}{2}(\sum_{m=1}^M \|\boldsymbol{\theta}_m\|_{\mathcal{H}_m}^q)^{2/q}$, generalizes the two previous cases, and for $1 < q \leq 2$ yields non-sparse MKL [6], where the kernel coefficients are constrained as $\boldsymbol{\beta} \geq \mathbf{0}$ and $\|\boldsymbol{\beta}\|_p \leq 1$ with $p = q/(2-q)$. The algorithm we propose also tackles the case $q > 2$, which cannot be handled with the procedure described in [6];
- sums of block-norm regularizers, $\Omega[f_{\boldsymbol{\theta}}] = \frac{1}{2}\sum_{j=1}^J \sigma_j \|\boldsymbol{\theta}\|_{2,q_j}^2$, with $\sigma_j \geq 0$, are a further generalization that subsume the elastic MKL of [18]. Our algorithm can also handle these.

Decomposition over parts. As usually in SP [17], features decompose over *parts*; *e.g.*, if the structure is expressed with a Markov network (V, E), then each part is associated with either a vertex or an edge, and the feature vector is written as $\boldsymbol{\phi}(x,y) = (\boldsymbol{\phi}_V(x,y), \boldsymbol{\phi}_E(x,y))$, with:

$$\boldsymbol{\phi}_V(x,y) = \sum_{i \in V} \boldsymbol{\psi}_{X,V}(x,i) \otimes \boldsymbol{\zeta}_{Y,V}(y_i) \quad \text{and} \quad \boldsymbol{\phi}_E(x,y) = \sum_{ij \in E} \boldsymbol{\psi}_{X,E}(x,ij) \otimes \boldsymbol{\zeta}_{Y,E}(y_i, y_j). \tag{3}$$

We make the common simplifying assumption that $\boldsymbol{\zeta}_{Y,V}$ is the identity feature mapping, that $\boldsymbol{\psi}_{X,E} \equiv 1$, and that $\boldsymbol{\zeta}_{Y,E}$ is the identity feature mapping scaled by a positive β_0. We then learn the kernel $K_{X,V}((x,i),(x',i')) = \langle \boldsymbol{\phi}_{X,V}(x,i), \boldsymbol{\phi}_{X,V}(x',i') \rangle$ as a combination of basis kernels $\{K_{X,V,m}\}_{m=1}^M$. This yields a kernel decomposition of the form

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

In our sequence labeling experiments, 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.

3 An Online Algorithm for MKL

To handle (2), we propose Alg. 1, which can be seen as extending FOBOS [5] by allowing multiple proximal steps to deal with composite regularizers, $\Omega[f_{\boldsymbol{\theta}}] = \sum_{j=1}^J \Omega_j[f_{\boldsymbol{\theta}}]$. Like SGD, Alg. 1 is suitable for problems with large N; the proximal steps make it effective for (non-differentiable) sparsity-inducing regularizers. As in PEGASOS [13], if a bound is known on the optimum, $\|\boldsymbol{\theta}^*\| \leq \gamma$, a projection step (line 9) is used. Naturally, Alg. 1 is entirely (or partially) kernelizable.

Subgradient step (line 4). First, local scores are computed for each part using the current parameters. Then, the highest score output, $\hat{y}_t \in \mathrm{argmax}_{y'_t \in \mathcal{Y}(x)} f_{\boldsymbol{\theta}}(x_t, y'_t) + \ell(y'_t, y_t)$, is found (*e.g.*, by dynamic programming). Finally, a subgradient is given as $\boldsymbol{g}_{\boldsymbol{\theta}} = f_{\boldsymbol{\theta}}(x_t, \hat{y}_t) - f_{\boldsymbol{\theta}}(x_t, y_t)$.

Proximal step (line 7). Given $\Omega : \mathcal{H} \to \mathbb{R}$, the Ω-*proximity operator* [4] is defined as $\mathrm{prox}_{\Omega}(\tilde{\boldsymbol{\theta}}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}\|^2 + \Omega(\boldsymbol{\theta})$. The block structure of the parameter vector makes this step particularly efficient, thanks to the following result:

Proposition 1 *Let $\Omega(\boldsymbol{\theta}) \equiv \omega((\|\boldsymbol{\theta}_m\|)_{m=1}^M)$ depend on each block only through its ℓ_2-norm. Then,* $[\mathrm{prox}_{\Omega}(\boldsymbol{\theta})]_m = [\mathrm{prox}_{\omega}(\|\boldsymbol{\theta}_1\|, \ldots, \|\boldsymbol{\theta}_M\|)]_m (\boldsymbol{\theta}_m / \|\boldsymbol{\theta}_m\|)$.

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.**	$\mathbf{85.2 \pm 0.3\%}$

Table 1: Results for handwriting recognition. Averages over 10 runs on the same folds as in [17], 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. Results are for the best regularization coefficient $C = 1/(\lambda m)$ (chosen from $\{0.1, 1, 10, 10^2, 10^3, 10^4\}$).

Hence, any $\ell_{2,q}^r$-proximity operator can be reduced to an ℓ_q^r one. For sparse MKL ($q = 1, r = 2$) this computation can be carried out in time $\tilde{O}(M)$ [9] ($\tilde{O}$ hides logarithmic terms). For non-sparse MKL, we replace $\ell_{2,q}^2$ by $\ell_{2,q}^q$ (equivalent up to a change in the regularization constant). The $\ell_{2,q}^q$-proximity operator is *separable*, hence can be computed coordinatewise.

Kernelization. The computation of local scores, which precedes the gradient step, only involves inner products. Hence, the whole algorithm can be kernelized. We only need to keep track of the ℓ_2-norm of each block, which can be updated after each gradient step via

$$\begin{aligned} \|\tilde{\boldsymbol{\theta}}_{t,m}\|^2 &= \|\boldsymbol{\theta}_{t,m}\|^2 - 2\eta_t \langle \boldsymbol{\theta}_{t,m}, \boldsymbol{\phi}_m(x_t, y_t)\rangle + \eta_t^2 \|\boldsymbol{\phi}_m(x_t, \hat{y}_t) - \boldsymbol{\phi}_m(x_t, y_t)\|^2 \\ &= \|\boldsymbol{\theta}_{t,m}\|^2 - 2\eta_t f_m(\hat{u}_t) + \eta_t^2 (K_m(u_t, u_t) + K_m(\hat{u}_t, \hat{u}_t) - 2K_m(u_t, \hat{u}_t)) \end{aligned} \quad (4)$$

and rescaled after each proximal and projection steps.

Regret, Convergence and Generalization Bounds. Let the loss L be convex and G-Lipschitz on the γ-radius ℓ_2-ball (*e.g.*, hinge or logistic), the regularizers Ω_j be convex and satisfy additional mild conditions (satisfied, *e.g.*, by $\Omega_j = \lambda_j \|\boldsymbol{\theta}\|_{p_j}^{q_j}$, for $p_j, q_j \geq 1$). We show [9] that Alg. 1 converges to ϵ precision, with high confidence, in $O(1/\epsilon^2)$ iterations (as the best batch MKL algorithms). If L or Ω are strongly convex (*e.g.*, if at least one regularizer Ω_j is a squared $\ell_{2,q}$-norm, for $q > 1$), the bound improves to $\tilde{O}(1/\epsilon)$. The generalization bounds are of the same order. This follows from the fact, shown in [9], that using rate $\eta_t = \eta_0/\sqrt{t}$ yields cumulative regret (w.r.t. the best fixed hypothesis) $\mathrm{Reg}_T \leq \left((2\gamma^2)/\eta_0 + G^2\eta_0\right)\sqrt{T}$; in the presence of σ-strong convexity, a rate $\eta_t = 1/(\sigma t)$ yields $\mathrm{Reg}_T \leq G^2(1 + \log T)/(2\sigma)$.

4 Experiments

Handwriting recognition. We use an OCR dataset [17] with 6,877 words written by 150 people (52,152 characters). Each character is a 16 × 8 binary image with one of 26 labels (a–z). Our first experiment (upper part of Tab. 1) compares linear, quadratic, and Gaussian kernels, used individually, combined via a simple average, or with MKL via Alg. 1. The results show that MKL outperforms the others, and that learning the bigram weight β_0 did not make any difference.

The second experiment shows that Alg. 1 handles both *feature* and *kernel* sparsity by learning a combination of a *linear* kernel (explicit features) with a *generalized* B_1*-spline* kernel ($K(\mathbf{x}, \mathbf{x}') = \max\{0, 1 - \|\mathbf{x} - \mathbf{x}'\|/h\}$, with h such that the kernel matrix has $\sim$ 95% zeros). The idea is to combine a simple feature-based kernel with one depending only on a few nearest neighbors. The results (Tab. 1, bottom part) show that MKL outperforms the individual kernels and the averaged kernel. The runtime is much shorter than in the previous experiment, with a mild accuracy decrease.

The third experiment compares Alg. 1 with two wrapper-based schemes: a Gauss-Seidel method alternating between optimizing the SVM and the kernel weights, and a gradient method (SimpleMKL [11]). In both cases, the SVMs were tackled with structured PEGASOS. As shown in Fig. 1, Alg. 1 converges in a small number of epochs to a near-optimal region, suggesting its usefulness in settings where each epoch is costly.

Dependency parsing. We train non-projective dependency parsers on the dataset of the CoNLL-2008 shared task [15] (39,278 training, 2,399 test sentences). The output to be predicted from each input sentence is the set of dependency arcs, which must define a directed spanning tree (example in Fig. 2). We use arc-factored models, and define a total of 507 feature groups (our linear kernels to be combined) conjoining words, lemmas, parts-of-speech, distance and direction of attachment. This

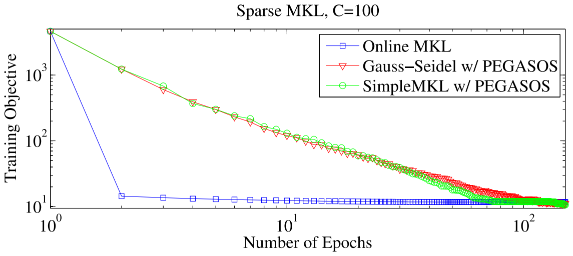

Figure 1: Comparison between Alg. 1 and two wrapper based methods in the OCR dataset, with $C = 100$. With only 20–30 passes over the data, Alg. 1 approaches a region very close to the optimum; in contrast, the wrapper-based methods need about 100 epochs.

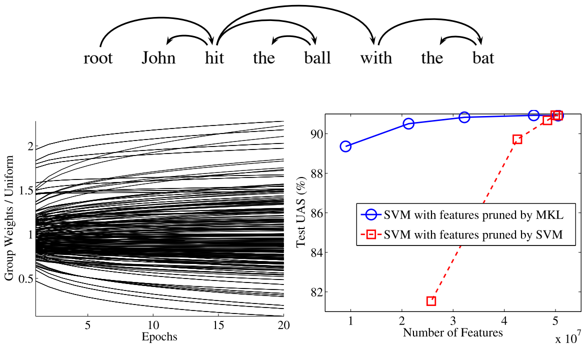

Figure 2: Top: a dependency parse tree (see [10]). Bottom left: group weights along the epochs of Alg. 1. Bottom right: results of standard SVMs trained on sets of feature templates of sizes $\{107, 207, 307, 407, 507\}$, either selected via a standard SVM or by MKL (the UAS—*unlabeled attachment score*—is the fraction of non-punctuation words whose head was correctly assigned.)

yields a problem with > 50 million instantiated features. The feature vectors associated with each candidate arc, however, are very sparse and this is exploited in the implementation. We run Alg. 1 with explicit features. Although MKL did not outperform an SVM, it showed a good performance at pruning irrelevant feature templates (Fig. 2, bottom right). Besides *interpretability*, this pruning is also appealing in a two-stage architecture, where a standard learner at a second stage will only need to handle a small fraction of the groups initially hypothesized, perhaps giving speedups in inference.

Acknowledgments

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

References

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

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

[3] 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*.

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

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

[6] Kloft, M., Brefeld, U., Sonnenburg, S., and Zien, A. (2010). Non-Sparse Regularization and Efficient Training with Multiple Kernels. *Arxiv preprint arXiv:1003.0079*.

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

[8] 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.

[9] Martins, A. F. T., Figueiredo, M. A. T., Aguiar, P. M. Q., Smith, N. A., and Xing, E. P. (2010). Online MKL for Structured Prediction. *Arxiv preprint arXiv:1010.2770*.

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

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

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

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

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

[15] 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*.

[16] Suzuki, T. and Tomioka, R. (2009). SpicyMKL. *Arxiv preprint arXiv:0909.5026*.

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

[18] Tomioka, R. and Suzuki, T. (2010). Sparsity-accuracy trade-off in MKL. *Arxiv*.