Structured Sparsity in Structured Prediction

Andre F.T. Martins  
*Carnegie Mellon University*

Noah A. Smith  
*Carnegie Mellon University, nasmith@cs.cmu.edu*

Pedro M.Q. Aguiar  
*Universidade de Lisboa*

Mario A. T. Figueiredo  
*Universidade de Lisboa*

Follow this and additional works at: [http://repository.cmu.edu/lti](http://repository.cmu.edu/lti)

Part of the [Computer Sciences Commons](http://repository.cmu.edu/lti)

Published In

This Conference Proceeding is brought to you for free and open access by the School of Computer Science at Research Showcase @ CMU. It has been accepted for inclusion in Language Technologies Institute by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.
Structured Sparsity in Structured Prediction

Andr´e F. T. Martins∗† Noah A. Smith∗ Pedro M. Q. Aguiar‡ M´ario A. T. Figueiredo†
∗School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
†Instituto de Sistemas e Rob´otica, Instituto Superior T´ecnico, Lisboa, Portugal
‡Instituto de Telecomunica¸c˜oes, Instituto Superior T´ecnico, Lisboa, Portugal
{afm,nasmith}@cs.cmu.edu, aguiar@isr.ist.utl.pt, mtf@lx.it.pt

Abstract

Linear models have enjoyed great success in structured prediction in NLP. While a lot of progress has been made on efficient training with several loss functions, the problem of endowing learners with a mechanism for feature selection is still unsolved. Common approaches employ ad hoc filtering or $L_1$-regularization; both ignore the structure of the feature space, preventing practitioners from encoding structural prior knowledge. We fill this gap by adopting regularizers that promote structured sparsity, along with efficient algorithms to handle them. Experiments on three tasks (chunking, entity recognition, and dependency parsing) show gains in performance, compactness, and model interpretability.

1 Introduction

Models for structured outputs are in demand across natural language processing, with applications in information extraction, parsing, and machine translation. State-of-the-art models usually involve linear combinations of features and are trained discriminatively; examples are conditional random fields (Lafferty et al., 2001), structured support vector machines (Altun et al., 2003; Taskar et al., 2003; Tsochantaridis et al., 2004), and the structured perceptron (Collins, 2002a). In all these cases, the underlying optimization problems differ only in the choice of loss function; choosing among them has usually a small impact on predictive performance.

In this paper, we are concerned with model selection: which features should be used to define the prediction score? The fact that models with few features (“sparse” models) are desirable for several reasons (compactness, interpretability, good generalization) has stimulated much research work which has produced a wide variety of methods (Della Pietra et al., 1997; Guyon and Elisseeff, 2003; McCallum, 2003). Our focus is on methods which embed this selection into the learning problem via the regularization term. We depart from previous approaches in that we seek to make decisions jointly about all candidate features, and we want to promote sparsity patterns that go beyond the mere cardinality of the set of features. For example, we want to be able to select entire feature templates (rather than features individually), or to make the inclusion of some features depend on the inclusion of other features.

We achieve the goal stated above by employing regularizers which promote structured sparsity. Such regularizers are able to encode prior knowledge and guide the selection of features by modeling the structure of the feature space. Lately, this type of regularizers has received a lot of attention in computer vision, signal processing, and computational biology (Zhao et al., 2009; Kim and Xing, 2010; Jenatton et al., 2009; Obozinski et al., 2010; Jenatton et al., 2010; Bach et al., 2011). Eisenstein et al. (2011) employed structured sparsity in computational sociolinguistics. However, none of these works have addressed structured prediction. Here, we combine these two levels of structure: structure in the output space, and structure in the feature space. The result is a framework that allows building structured predictors with high predictive power, while reducing manual feature engineering. We obtain models that are interpretable, accurate, and often much more compact than $L_2$-regularized ones. Compared with $L_1$-regularized models, ours are often more accurate and yield faster runtime.
2 Structured Prediction

We address structured prediction problems, which involve an input set \( \mathcal{X} \) (e.g., sentences) and an output set \( \mathcal{Y} \), assumed large and structured (e.g., tags or parse trees). We assume that each \( x \in \mathcal{X} \) has a set of candidate outputs \( \mathcal{Y}(x) \subseteq \mathcal{Y} \). We consider linear models, in which predictions are made according to

\[
\hat{y} = \arg \max_{y \in \mathcal{Y}(x)} \theta \cdot \phi(x, y), \quad (1)
\]

where \( \phi(x, y) \in \mathbb{R}^D \) is a vector of features, and \( \theta \in \mathbb{R}^D \) is the vector of corresponding weights. Let \( \mathcal{D} = \{(x_i, y_i)\}_{i=1}^N \) be a training sample. We assume a cost function is defined such that \( c(\hat{y}, y) \) is the cost of predicting \( \hat{y} \) when the true output is \( y \); our goal is to learn \( \theta \) with small expected cost on unseen data. To achieve this goal, linear models are usually trained by solving a problem of the form

\[
\hat{\theta} = \arg \min_\theta \Omega(\theta) + \frac{1}{N} \sum_{i=1}^N L(\theta, x_i, y_i), \quad (2)
\]

where \( \Omega \) is a regularizer and \( L \) is a loss function. Examples of losses are: the negative conditional log-likelihood used in CRFs (Lafferty et al., 2001),

\[
L_{\text{CRF}}(\theta, x, y) = -\log P(y|x), \quad (3)
\]

where \( P(y|x) \propto \exp(\theta \cdot \phi(x, y)) \) is a log-linear model; the margin rescaled loss of structured SVMs (Taskar et al., 2003; Tsochantaridis et al., 2004),

\[
L_{\text{SVM}}(\theta, x, y) = \max_{y' \in \mathcal{Y}(x)} \theta \cdot \delta\phi(y') + c(y', y), \quad (4)
\]

where \( \delta\phi(y') = \phi(x, y') - \phi(x, y) \); and the loss underlying the structured perceptron (Collins, 2002a),

\[
L_{\text{SP}}(\theta, x, y) = \max_{y' \in \mathcal{Y}(x)} \theta \cdot \delta\phi(y'). \quad (5)
\]

Empirical comparison among these loss functions can be found in the literature (see, e.g., Martins et al., 2010, who also consider interpolations of the losses above). In practice, it has been observed that the choice of loss has far less impact than the model design and choice of features. Hence, in this paper, we focus our attention on the regularization term in Eq. 2. We specifically address ways in which this term can be used to help design the model by promoting structured sparsity. While this has been a topic of intense research in signal processing and computational biology (Jenatton et al., 2009; Liu and Ye, 2010; Bach et al., 2011), it has not yet received much attention in the NLP community, where the choice of regularization for supervised learning has essentially been limited to the following:

- **L_2-regularization** (Chen and Rosenfeld, 2000):

\[
\Omega_{\lambda}^{L_2}(\theta) \triangleq \frac{\lambda}{2} \| \theta \|_2^2 = \frac{\lambda}{2} \sum_{d=1}^D \theta_d^2, \quad (6)
\]

- **L_1-regularization** (Kazama and Tsujii, 2003; Goodman, 2004):

\[
\Omega_{\tau}^{L_1}(\theta) \triangleq \tau \| \theta \|_1 = \tau \sum_{d=1}^D |\theta_d|. \quad (7)
\]

The latter is known as “Lasso,” as popularized by Tibshirani (1996) in the context of sparse regression. In the two cases above, \( \lambda \) and \( \tau \) are nonnegative coefficients controlling the intensity of the regularization. \( \Omega_{\lambda}^{L_2} \) usually leads to easier optimization and robust performance; \( \Omega_{\tau}^{L_1} \) encourages sparser models, where only a few features receive nonzero weights; see Gao et al. (2007) for an empirical comparison. More recently, Petrov and Klein (2008b) applied \( L_1 \) regularization for structure learning in phrase-based parsing; a comparison with \( L_2 \) appears in Petrov and Klein (2008a). Elastic nets interpolate between \( L_1 \) and \( L_2 \), having been proposed by Zou and Hastie (2005) and used by Lavergne et al. (2010) to regularize CRFs.

Neither of the regularizers just described “looks” at the structure of the feature space, since they all treat each dimension independently—we call them unstructured regularizers, as opposed to the structured ones that we next describe.

3 Structured Sparsity

We are interested in regularizers that share with \( \Omega_{\tau}^{L_1} \) the ability to promote sparsity, so that they can be used for selecting features. In addition, we want to endow the feature space \( \mathbb{R}^D \) with additional structure, so that features are not penalized individually (as in the \( L_1 \)-case) but collectively, encouraging entire groups of features to be discarded. The choice of groups will allow encoding prior knowledge regarding the kind of sparsity patterns that are intended in the model. This can be achieved with group-Lasso regularization, which we next describe.
3.1 The Group Lasso

To capture the structure of the feature space, we group our $D$ features into $M$ groups $G_1, \ldots, G_M$, where each $G_m \subseteq \{1, \ldots, D\}$. Ahead, we discuss meaningful ways of choosing group decompositions; for now, let us assume a sensible choice is obvious to the model designer. Denote by $\theta_m = \langle \theta_d \rangle_{d \in G_m}$ the subvector of those weights that correspond to the features in the $m$-th group, and let $d_1, \ldots, d_M$ be nonnegative scalars (one per group).

We consider the following group-Lasso regularizers:

$$\Omega_d^{GL} = \sum_{m=1}^M d_m \| \theta_m \|^2. \quad (8)$$

These regularizers were first proposed by Bakin (1999) and Yuan and Lin (2006) in the context of regression. If $d_1 = \ldots = d_M$, $\Omega_d^{GL}$ becomes the “$L_1$ norm of the $L_2$ norm.” Interestingly, this is also a norm, called the mixed $L_{2,1}$-norm. These regularizers subsume the $L_1$ and $L_2$ cases, which correspond to trivial choices of groups:

- If each group is a singleton, i.e., $M = D$ and $G_d = \{\theta_d\}$, and $d_1 = \ldots = d_M = \tau$, we recover $L_1$-regularization (cf. Eqs. 7–8).
- If there is a single group spanning all the features, i.e., $M = 1$ and $G_1 = \{1, \ldots, D\}$, then the right hand side of Eq. 8 becomes $d_1 \| \theta \|^2$. This is equivalent to $L_2$ regularization.\(^{2}\)

We next present some non-trivial examples concerning different topologies of $\mathcal{G} = \{G_1, \ldots, G_M\}$.

Non-overlapping groups. Let us first consider the case where $\mathcal{G}$ is a partition of the feature space: the groups cover all the features ($\bigcup_m G_m = \{1, \ldots, D\}$), and they do not overlap ($G_a \cap G_b = \emptyset$, $\forall a \neq b$). Then, $\Omega_d^{GL}$ is termed a non-overlapping group-Lasso regularizer. It encourages sparsity patterns in which entire groups are discarded. A judicious choice of groups can lead to very compact models and pinpoint relevant groups of features. The following examples lie in this category:

- The two cases above ($L_1$ and $L_2$ regularization).
- Label-based groups. In multi-label classification, where $\mathcal{Y} = \{1, \ldots, L\}$, features are typically designed as conjunctions of input features with label indicators, i.e., they take the form $\phi(x, y) = \psi(x) \otimes e_y$, where $\psi(x) \in \mathbb{R}^{DX}$, $e_y \in \mathbb{R}^L$ has all entries zero except the $y$-th entry, which is 1, and $\otimes$ denotes the Kronecker product. Hence $\phi(x, y)$ can be reshaped as a $DX$-by-$L$ matrix, and we can let each group correspond to a row. In this case, all groups have the same size and we typically set $d_1 = \ldots = d_M$. A similar design can be made for sequence labeling problems, by considering a similar grouping for the unigram features.\(^{3}\)
- Template-based groups. In NLP, features are commonly designed via templates. For example, a template such as $w_0 \land p_0 \land p_{-1}$ denotes the word in the current position ($w_0$) conjoined with its part-of-speech ($p_0$) and that of the previous word ($p_{-1}$). This template encloses many features corresponding to different instantiations of $w_0$, $p_0$, and $p_{-1}$. In §5, we learn feature templates from the data, by associating each group to a feature template, and letting that group contain all features that are instantiations of this template. Since groups have different sizes, it is a good idea to let $d_m$ increase with the group size, so that larger groups pay a larger penalty for being included.

Tree-structured groups. More generally, we may let the groups in $\mathcal{G}$ overlap but be nested, i.e., we may want them to form a hierarchy (two distinct groups either have empty intersection or one is contained in the other). This induces a partial order on $\mathcal{G}$ (the set inclusion relation $\supseteq$), endowing it with the structure of a partially ordered set (poset).

A convenient graphical representation of the poset $\langle \mathcal{G}, \supseteq \rangle$ is its Hasse diagram. Each group is a node in the diagram, and an arc is drawn from group $G_a$ to group $G_b$ if $G_b \subseteq G_a$ and there is no $b'$ s.t. $G_b \subset G_{b'} \subset G_a$. When the groups are nested, this diagram is a forest (a union of directed trees). The corresponding regularizer enforces sparsity patterns

\(^{1}\)In the statistics literature, such mixed-norm regularizers, which group features and then apply a separate norm for each group, are called composite absolute penalties (Zhao et al., 2009); other norms besides $L_{2,1}$ can be used, such as $L_{\infty,1}$ (Quattoni et al., 2009; Wright et al., 2009; Eisenstein et al., 2011).

\(^{2}\)Note that Eqs. 8 and 6 do not become exactly the same: in Eq. 6, the $L_2$ norm is squared. However it can be shown that both regularizers lead to identical learning problems (Eq. 2) up to a transformation of the regularization constant.

\(^{3}\)The same idea is also used in multitask learning, where labels correspond to tasks (Caruana, 1997).
where a group of features is only selected if *all its ancestors are also selected*. Hence, entire subtrees in the diagram can be pruned away. Examples are:

- **The elastic net.** The diagram of $G$ has a root node for $G_1 = \{1, \ldots, D\}$ and $D$ leaf nodes, one per each singleton group (see Fig. 1).
- **The sparse group-Lasso.** This regularizer was proposed by Friedman et al. (2010):
  \[
  \Omega_{d, \tau}^{\text{SGL}}(\theta) = \sum_{m=1}^{M'} (d_m \| \theta_m \|^2 + \tau_m \| \theta_m \|_1),
  \]
  where the total number of groups is $M = M' + D$, and the components $\theta_1, \ldots, \theta_{M'}$ are non-overlapping. This regularizer promotes sparsity at both group and feature levels (i.e., it eliminates entire groups and sparsifies within each group).

**Graph-structured groups.** In general, the groups in $G$ may overlap without being nested. In this case, the Hasse diagram of $G$ is a directed acyclic graph (DAG). As in the tree-structured case, a group of features is only selected if all its ancestors are also selected. Based on this property, Jenatton et al. (2009) suggested a way of reverse engineering the groups from the desired sparsity pattern. We next describe a strategy for *coarse-to-fine feature template selection* that directly builds on that idea.

Suppose that we are given $M$ feature templates $T = \{T_1, \ldots, T_M\}$ which are partially ordered according to some criterion, such that if $T_a \preceq T_b$ we would like to include $T_b$ in our model only if $T_a$ is also included. This criterion could be a measure of coarseness: we may want to let coarser part-of-speech features precede finer lexical features, e.g., $p_0 \wedge p_1 \preceq w_0 \wedge w_1$, or conjoined features come after their elementary parts, e.g., $p_0 \preceq p_0 \wedge p_1$. The order does not need to be total, so some templates may not be comparable (e.g., we may want $p_0 \wedge p_{-1}$ and $p_0 \wedge p_1$ not to be comparable). To achieve the sparsity pattern encoded in $(T, \preceq)$, we choose $G = \langle G_1, \ldots, G_M \rangle$ as follows: let $I(T_a)$ be the set of features that are instantiations of template $T_a$; then define $G_a = \bigcup_{b:a \preceq b} I(T_b)$, for $a = 1, \ldots, M$. It is easy to see that $(G, \supseteq)$ and $(T, \preceq)$ are isomorphic posets (their Hasse diagrams have the same shape; see Fig. 1). The result is a “coarse-to-fine” regularizer, which prefers to select feature templates that are coarser before zooming into finer features.

### 3.2 Bayesian Interpretation

The prior knowledge encoded in the group-Lasso regularizer (Eq. 8) comes with a Bayesian interpretation, as we next describe. In a probabilistic model (e.g., in the CRF case, where $L = L_{\text{CRF}}$), the optimization problem in Eq. 2 can be seen as maximum *a posteriori* estimation of $\theta$, where the regularization term $\Omega(\theta)$ corresponds to the negative log of a prior distribution (call it $p(\theta)$). It is well-known that $L_2$-regularization corresponds to choosing independent zero-mean Gaussian priors, $\theta_d \sim N(0, \lambda^{-1})$, and that $L_1$-regularization results from adopting zero-mean Laplacian priors, $p(\theta_d) \propto \exp(\tau |\theta_d|)$.

Figueiredo (2002) provided an alternative interpretation of $L_1$-regularization in terms of a two-level hierarchical Bayes model, which happens to generalize to the non-overlapping group-Lasso case, where $\Omega = \Omega_{d, \tau}^{\text{GL}}$. As in the $L_2$-case, we also assume that each parameter receives a zero-mean Gaussian prior, but now with a *group-specific variance* $\tau_m$, i.e., $\theta_m \sim N(0, \tau_m I)$ for $m = 1, \ldots, M$. This reflects the fact that some groups should have their feature weights shrunk more towards zero than others. The variances $\tau_m \geq 0$ are not pre-specified but rather generated by a one-sided exponential hyper-prior $p(\tau_m |d_m) \propto \exp(-d_m^2 \tau_m/2)$. It can be shown that after marginalizing out $\tau_m$, we obtain

\[
p(\theta_m |d_m) = \int_0^{\infty} p(\theta_m |\tau_m)p(\tau_m |d_m)d\tau_m \propto \exp(-d_m^2 \| \theta_m \|). \tag{10}
\]

Hence, the non-overlapping group-Lasso corresponds to the following two-level hierarchical Bayes model: independently for each $m = 1, \ldots, M$,

\[
\tau_m \sim \text{Exp}(d_m^2/2), \quad \theta_m \sim N(0, \tau_m I). \tag{11}
\]

### 3.3 Prox-operators

Before introducing our learning algorithm for handling group-Lasso regularization, we need to define the concept of a *$\Omega$-proximity operator*. This is the function $\text{prox}_\Omega : \mathbb{R}^D \to \mathbb{R}^D$ defined as follows:

\[
\text{prox}_\Omega(\theta) = \arg \min_\theta \frac{1}{2} \| \theta' - \theta \|^2 + \Omega(\theta'). \tag{12}
\]
Proximity operators generalize Euclidean projections and have many interesting properties; see Bach et al. (2011) for an overview. By requiring zero to be a subgradient of the objective function in Eq. 12, we obtain the following closed expression (called soft-thresholding) for the $\Omega^{L_1}$-proximity operator:

$$[\text{prox}_{\Omega^{L_1}}(\theta)]_d = \begin{cases} 
\theta_d - \tau & \text{if } \theta_d > \tau \\
0 & \text{if } |\theta_d| \leq \tau \\
\theta_d + \tau & \text{if } \theta_d < -\tau.
\end{cases}$$

For the non-overlapping group Lasso case, the proximity operator is given by

$$[\text{prox}_{\Omega^{G_1}}(\theta)]_m = \begin{cases} 
0 & \text{if } \|\theta_m\|_2 \leq d_m \\
\frac{\|\theta_m\|_2 - d_m}{\|\theta_m\|_2} \theta_m & \text{otherwise}.
\end{cases}$$

which can be seen as a generalization of Eq. 13: if the $L_2$-norm of the $m$-th group is less than $d_m$, the entire group is discarded; otherwise it is scaled so that its $L_2$-norm decreases by an amount of $d_m$.

When groups overlap, the proximity operator lacks a closed form. When $\mathcal{G}$ is tree-structured, it can still be efficiently computed by a recursive procedure (Jenatton et al., 2010). When $\mathcal{G}$ is not tree-structured, no specialized procedure is known, and a convex optimizer is necessary to solve Eq. 12.

### 4 Online Prox-Grad Algorithm

We now turn our attention to efficient ways of handling group-Lasso regularizers. Several fast and scalable algorithms having been proposed for training $L_1$-regularized CRFs, based on quasi-Newton optimization (Andrew and Gao, 2007), coordinate descent (Sokolovska et al., 2010; Lavergne et al., 2010), and stochastic gradients (Carpenter, 2008; Langford et al., 2009; Tsuruoka et al., 2009). The algorithm that we use in this paper (Alg. 1) extends the stochastic gradient methods for group-Lasso regularization; a similar algorithm was used by Martins et al. (2011) for multiple kernel learning.

Alg. 1 addresses the learning problem in Eq. 2 by alternating between online (sub-)gradient steps with respect to the loss term, and proximal steps with respect to the regularizer. Proximal-gradient methods are very popular in sparse modeling, both in batch (Liu and Ye, 2010; Bach et al., 2011) and online (Duchi and Singer, 2009; Xiao, 2009) settings. The reason we have chosen the algorithm of Martins et al. (2011) is that it effectively handles overlapping groups, without the need of evaluating $\text{prox}_\Omega$ (which, as seen in §3.3, can be costly if $\mathcal{G}$ is not tree-structured). To do so, it decomposes $\Omega$ as

$$\Omega(\theta) = \sum_{j=1}^J \sigma_j \Omega_j(\theta)$$

for some $J \geq 1$, and nonnegative $\sigma_1, \ldots, \sigma_J$; each $\Omega_j$-proximal operator is assumed easy to compute. Such a decomposition always exists: if $\mathcal{G}$ does not have overlapping groups, take $J = 1$. Otherwise, find $J \leq M$ disjoint sets $\mathcal{G}_1, \ldots, \mathcal{G}_J$ such that $\bigcup_{j=1}^J \mathcal{G}_j = \mathcal{G}$ and the groups on each $\mathcal{G}_j$ are non-overlapping. The proximal steps are then applied sequentially, one per each $\Omega_j$. Overall, Alg. 1 satisfies the following important requirements:

- **Computational efficiency.** Each gradient step at round $t$ is linear in the number of features that fire for that instance and independent of the total number of features $D$. Each proximal step is linear in the number of groups $M$, and does not need to be performed every round (as we will see later).


**Algorithm 1 Online Sparse Prox-Grad Algorithm**

1: input: $\mathcal{D}$, $\langle \Omega_j \rangle_{j=1}^J$, $T$, gravity sequence $\langle \langle \sigma_{jt} \rangle_{j=1}^J \rangle_{t=1}^T$, stepsize sequence $\langle \eta_t \rangle_{t=1}^T$
2: initialize $\theta = 0$
3: for $t = 1$ to $T$ do
4:   take training pair $(x_t, y_t) \in \mathcal{D}$
5:   $\theta \leftarrow \theta - \eta_t \nabla L(\theta; x_t, y_t)$ (gradient step)
6:   for $j = 1$ to $J$ do
7:      $\theta \leftarrow \text{prox}_{\eta_t \sigma_j \Omega_j}(\theta)$ (proximal step)
8:   end for
9: end for
10: output: $\theta$

- **Memory efficiency.** Only a small active set of features (those that have nonzero weights) need to be maintained. Entire groups of features can be deleted after each proximal step. Furthermore, only the features which correspond to nonzero entries in the gradient vector need to be inserted in the active set; for some losses ($L_{\text{SVM}}$ and $L_{\text{SP}}$) many irrelevant features are never instantiante.

- **Convergence.** With high probability, Alg. 1 produces an $\epsilon$-accurate solution after $T \leq O(1/\epsilon^2)$ rounds, for a suitable choice of stepizes and holding $\sigma_{jt}$ constant, $\sigma_{jt} = \sigma_j$ (Martins et al., 2011). This result can be generalized to any sequence $\langle \sigma_{jt} \rangle_{t=1}^T$ such that $\sigma_j = \frac{1}{T} \sum_{t=1}^T \sigma_{jt}$.

We next describe several algorithmic ingredients that make Alg. 1 effective in sparse modeling.

**Budget-Driven Shrinkage.** Alg. 1 requires the choice of a “gravity sequence.” We follow Langford et al. (2009) and set $\langle \sigma_{jt} \rangle_{j=1}^J$ to zero for all $t$ which is not a multiple of some prespecified integer $K$; this way, proximal steps need only be performed each $K$ rounds, yielding a significant speed-up when the number of groups $M$ is large. A direct adoption of the method of Langford et al. (2009) would set $\sigma_{jt} = K \sigma_j$ for those rounds; however, we have observed that such a strategy makes the number of groups vary substantially in early epochs. We use a different strategy: for each $\mathcal{S}_j$, we specify a budget of $B_j \geq 0$ groups (this may take into consideration practical limitations, such as the available memory). If $t$ is a multiple of $K$, we set $\sigma_{jt}$ as follows:

1. If $\mathcal{S}_j$ does not have more than $B_j$ nonzero groups, set $\sigma_{jt} = 0$ and do nothing.
2. Otherwise, sort the groups in $\mathcal{S}_j$ by decreasing order of their $L_2$-norms. Check the $L_2$-norms of the $B_j$-th and $B_{j+1}$-th entries in the list and set $\sigma_{jt}$ as the mean of these two divided by $\eta_t$.
3. Apply a $\eta_t \sigma_{jt} \Omega_j$-proximal step using Eq. 14. At the end of this step, no more than $B_j$ groups will remain nonzero.

If the average of the gravity steps converge, $\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^T \sigma_{jt} \rightarrow \sigma_j$, then the limit points $\sigma_j$ implicitly define the regularizer, via $\Omega = \sum_{j=1}^J \sigma_j \Omega_j$. Hence, we have shifted the control of the amount of regularization to the budget constants $B_j$, which unlike the $\sigma_j$ have a clear meaning and can be chosen under practical considerations.

**Space and Time Efficiency.** The proximal steps in Alg. 1 have a **scaling** effect on each group, which affects all features belonging to that group (see Eq. 14). We want to avoid explicitly updating each feature in the active set, which could be time consuming. We mention two strategies that can be used for the **non-overlapping** group Lasso case.

- The first strategy is suitable when $M$ is large and only a few groups ($\ll M$) have features that fire in each round; this is the case, e.g., of label-based groups (see §3.1). It consists of making **lazy updates** (Carpenter, 2008), i.e., to delay the update of all features in a group until at least one of them fires; then apply a cumulative penalty. The amount of the penalty can be computed if one assigns a timestamp to each group.

- The second strategy is suitable when $M$ is small and some groups are very populated; this is the typical case of **template-based groups** (§3.1). Two operations need to be performed: updating each feature weight (in the gradient steps), and scaling entire groups (in the proximal steps). We adapt a trick due to Shalev-Shwartz et al. (2007): represent the weight vector of the $m$-th group, $\theta_m$, by a

---

5When overlaps exist (e.g., the coarse-to-fine case), we specify a total pseudo-budget $B$ ignoring the overlaps, which induces budgets $B_1, \ldots, B_J$ which sum to $B$. The number of actually selected groups may be less than $B$, however, since in this case some groups can be shrunk more than once. Other heuristics are possible.

6The convergence assumption can be sidestepped by freezing the $\sigma_j$ after a fixed number of iterations.
triple \((\xi_m, c_m, \rho_m) \in \mathbb{R}^{|G_m|} \times \mathbb{R}_+ \times \mathbb{R}_+\), such that \(\theta_m = c_m \xi_m\) and \(\|\theta_m\|^2 = \rho_m\). This representation allows performing the two operations above in constant time, and it keeps track of the group L₂-norms, necessary in the proximal updates.

For sufficient amounts of regularization, our algorithm has a low memory footprint. Only features that, at some point, intervene in the gradient computed in line 5 need to be instantiated; and all features that receive zero weights after some proximal step can be deleted from the model (cf. Fig. 2).

**Sparseptron and Debiasing.** Although Alg. 1 allows to simultaneously select features and learn the model parameters, it has been observed in the sparse modeling literature that Lasso-like regularizers usually have a strong bias which may harm predictive performance. A post-processing stage is usually taken (called debiasing), in which the model is refitted without any regularization and using only the selected features (Wright et al., 2009). If a final debiasing stage is to be performed, Alg. 1 only needs to worry about feature selection, hence it is appealing to choose a loss function that makes this procedure as simple as possible. Examining the input of Alg. 1, we see that both a gravity and a stepsize sequence need to be specified. The former can be taken care of by using budget-driven shrinkage, as described above. The stepsize sequence can be set as \(\eta_t = \eta_0 \sqrt{t/N}\), which ensures convergence, however \(\eta_0\) requires tuning. Fortunately, for the structured perceptron loss \(L_{SP}\) (Eq. 5), Alg. 1 is independent of \(\eta_0\), up to a scaling of \(\theta\), which does not affect predictions (see Eq. 1).\(^7\) We call the instantiation of Alg. 1 with a group-Lasso regularizer and the loss \(L_{SP}\) the sparseptron. Overall, we propose the following two-stage approach:

1. Run the sparseptron for a few epochs and discard the features with zero weights.
2. Refit the model without any regularization and using the loss \(L\) which one wants to optimize.

\(^7\)To see why this is the case, note that both gradient and proximal updates come scaled by \(\eta_0\), and that the gradient of the loss is \(\nabla L_{SP}(\theta, x_t, y_t) = \phi(x_t, \tilde{y}_t) - \phi(x_t, y_t)\), where \(\tilde{y}_t\) is the prediction under the current model, which is insensitive to the scaling of \(\theta\). This independence on \(\eta_0\) does not hold when the loss is \(L_{SVM}\) or \(L_{CRF}\).

5 Experiments

We present experiments in three structured prediction tasks for several group choices.

**Text Chunking.** We use the English dataset provided in the CoNLL 2000 shared task (Sang and Buchholz, 2000), which consists of 8,936 training and 2,012 testing sentences (sections 15–18 and 20 of the WSJ). The input observations are the token words and their POS tags; we want to predict the sequences of IOB tags representing phrase chunks. We built 96 contextual feature templates as follows:

- Up to 5-grams of POS tags, in windows of 5 tokens on the left and 5 tokens on the right;
- Up to 3-grams of words, in windows of 3 tokens on the left and 3 tokens on the right;
- Up to 2-grams of word shapes, in windows of 2 tokens on the left and 2 tokens on the right. Each shape replaces characters by their types (case sensitive letters, digits, and punctuation), and deletes repeated types—e.g., Confidence and 2,664,098 are respectively mapped to Aa and 0,0+,0+ (Collins, 2002b).

We defined unigram features by conjoining these templates with each of the 22 output labels. An additional template was defined to account for label bigrams—features in this template do not look at the input string, but only at consecutive pairs of labels.\(^8\)

We evaluate the ability of group-Lasso regularization to perform feature template selection. To do that, we ran 5 epochs of the sparseptron algorithm with template-based groups and budget-driven shrinkage (budgets of 10, 20, 30, 40, and 50 templates were tried). For each group \(G_m\), we set \(d_m = \log_2 |G_m|\), which is the average number of bits necessary to encode a feature in that group, if all features were equiprobable. We set \(K = 1000\) (the number of instances between consecutive proximal steps). Then, we refit the model with 10 iterations of the max-loss 1-best MIRA algorithm (Cramer et al., 2006).\(^9\) Table 1 compares the \(F_1\) scores and

\(^8\)State-of-the-art models use larger output contexts, such as label trigrams and 4-grams. We resort to bigram labels as we are mostly interested in identifying relevant unigram templates.

\(^9\)This variant optimizes the \(L_{SVM}\) loss (Martins et al., 2010). For the refitting, we used unregularized MIRA. For the baseline
Table 1: Results for text chunking.

<table>
<thead>
<tr>
<th>Model</th>
<th>MIRA</th>
<th>Group Lasso</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>B = 10</td>
</tr>
<tr>
<td>F1 (%)</td>
<td>93.10</td>
<td>92.99</td>
</tr>
<tr>
<td>Model size (# features)</td>
<td>5,300,396</td>
<td>71,075</td>
</tr>
</tbody>
</table>

Table 2: Results for named entity recognition. Each cell shows F1 (%) and the number of features.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MIRA</th>
<th>Lasso</th>
<th>Group-Lasso</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spa. dev/test</td>
<td>70.38/74.09</td>
<td>69.19/71.9</td>
<td>70.75/72.38</td>
</tr>
<tr>
<td></td>
<td>8,598,246</td>
<td>68,565</td>
<td>1,017,769</td>
</tr>
<tr>
<td>Duct. dev/test</td>
<td>69.15/71.54</td>
<td>64.07/66.35</td>
<td>66.82/69.42</td>
</tr>
<tr>
<td></td>
<td>5,727,004</td>
<td>164,960</td>
<td>565,704</td>
</tr>
<tr>
<td>Eng. dev/test</td>
<td>83.95/79.81</td>
<td>80.92/76.95</td>
<td>82.58/78.84</td>
</tr>
<tr>
<td></td>
<td>8,376,901</td>
<td>232,865</td>
<td>870,587</td>
</tr>
</tbody>
</table>

Figure 2: Memory footprints of the MIRA and sparseptron algorithms in text chunking. The oscillation in the first 5 epochs (bottom line) comes from the proximal steps each \( K = 1000 \) rounds. The features are then frozen and 10 epochs of unregularized MIRA follow. Overall, the sparseptron requires < 7.5% of the memory as the MIRA baseline.

Named Entity Recognition. We experiment with the Spanish, Dutch, and English datasets provided in the CoNLL 2002/2003 shared tasks (Sang, 2002; Sang and De Meulder, 2003). For Spanish, we use the POS tags provided by Car-

Multilingual Dependency Parsing. We trained non-projective dependency parsers for 6 languages using the CoNLL-X shared task datasets (Buchholz and Marsi, 2006): Arabic, Danish, Dutch, Japanese, Slovene, and Spanish. We chose the languages with the smallest datasets, because regularization is more important when data is scarce. The output to be predicted from each input sentence is the set of dependency links, which jointly define a spanning tree.
We use arc-factored models, for which exact inference is tractable (McDonald et al., 2005). We defined $M = 684$ feature templates for each candidate arc by conjoining the words, shapes, lemmas, and POS of the head and the modifier, as well as the contextual POS, and the distance and direction of attachment. We followed the same two-stage approach as before, and compared with a baseline which selects feature templates by ranking them according to the information gain criterion. This baseline assigns a score to each template $T_m$ which reflects an empirical estimate of the mutual information between $T_m$ and the binary variable $A$ that indicates the presence/absence of a dependency link:

$$IG_m = \sum_{f \in T_m} \sum_{a \in \{0,1\}} P(f,a) \log_2 \frac{P(f,a)}{P(f)P(a)},$$

where $P(f,a)$ is the joint probability of feature $f$ firing and an arc being active ($a = 1$) or inactive ($a = 0$), and $P(f)$ and $P(a)$ are the corresponding marginals. All probabilities are estimated from the empirical counts of events observed in the data.

The results are plotted in Fig. 3, for budget sizes of 200, 300, and 400. We observe that for all but one language (Spanish is the exception), non-overlapping group-Lasso regularization is more effective at selecting feature templates than the information gain criterion, and slightly better than coarse-to-fine group-Lasso. For completeness, we also display the results obtained with a standard Lasso regularizer. Table 3 shows what kind of feature templates were most selected for each language. Some interesting patterns can be observed: morphologically-rich languages with small datasets (such as Turkish and Slovene) seem to avoid lexical features, arguably due to potential for overfitting; in Japanese, contextual POS appear to be specially relevant. It should be noted, however, that some of these patterns may be properties of the datasets rather than of the languages themselves.

6 Related Work

A variant of the online proximal gradient algorithm used in this paper was proposed by Martins et al.
Table 3: Variation of feature templates that were selected across languages. Each line groups together similar templates, involving lexical, contextual POS, word shape information, as well as attachment direction and length. Empty cells denote that very few or none of the templates in that category was selected; + denotes that some were selected; ++ denotes that most or all were selected.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bilexical</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Lex. → POS</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>POS → Lex.</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>POS → POS</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Middle POS</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Shape</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Direction</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Distance</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Mixed norm regularization has been used for a while in statistics as a means to promote structured sparsity. Group Lasso is due to Bakin (1999) and Yuan and Lin (2006), after which a string of variants and algorithms appeared (Bach, 2008; Zhao et al., 2009; Jenatton et al., 2009; Friedman et al., 2010; Obozinski et al., 2010). The flat (non-overlapping) case has tight links with learning formalisms such as multiple kernel learning (Lanckriet et al., 2004) and multi-task learning (Caruana, 1997). The tree-structured case has been addressed by Kim and Xing (2010), Liu and Ye (2010) and Mairal et al. (2010), along with $L_{\infty, 1}$ and $L_{2, 1}$ regularization. Graph-structured groups are discussed in Jenatton et al. (2010), along with a DAG representation. In NLP, mixed norms have been used recently by Graça et al. (2009) in posterior regularization, and by Eisenstein et al. (2011) in a multi-task regression problem.

7 Conclusions

In this paper, we have explored two levels of structure in NLP problems: structure on the outputs, and structure on the feature space. We have shown how the latter can be useful in model design, through the use of regularizers which promote structured sparsity. We propose an online algorithm with minimal memory requirements for exploring large feature spaces. Our algorithm, which specializes into the sparseptron, yields a mechanism for selecting entire groups of features. We apply sparseptron for selecting feature templates in three structured prediction tasks, with advantages over filter-based methods, $L_1$, and $L_2$ regularization in terms of performance, compactness, and model interpretability.

Acknowledgments

We would like to thank all reviewers for their comments, Eric Xing for helpful discussions, and Slav Petrov for his comments on a draft version of this paper. A. M. was supported by a FCT/ICTI grant through the CMU-Portugal Program, and also by Priberam. This work was partially supported by the FET programme (EU FP7), under the SIMBAD project (contract 213250). N. S. was supported by NSF CAREER IIS-1054319.

References


T. Lavergne, O. Cappé, and F. Yvon. 2010. Practical very large scale CRFs. In *Proc. of ACL*.


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


