Spectral Regularization for Max-Margin Sequence Tagging

Ariadna Quattoni[1] AQUATTONI@LSI.UPC.EDU
Universitat Politècnica de Catalunya, Barcelona, Catalunya

Borja Balle[1] BBALLE@CS.MCGILL.CA
McGill University, Montreal, QC, Canada

Xavier Carreras CARRERAS@LSI.UPC.EDU
Universitat Politècnica de Catalunya, Barcelona, Catalunya

Amir Globerson GAMIR@CS.HUJI.AC.IL
The Hebrew University of Jerusalem, Jerusalem, Israel

Abstract

We frame max-margin learning of latent variable structured prediction models as a convex optimization problem, making use of scoring functions computed by input-output observable operator models. This learning problem can be expressed as an optimization problem involving a low-rank Hankel matrix that represents the input-output operator model. The direct outcome of our work is a new spectral regularization method for max-margin structured prediction. Our experiments confirm that our proposed regularization framework leads to an effective way of controlling the capacity of structured prediction models.

1. Introduction

Many important problems in machine learning can be framed as structured prediction tasks where the goal is to learn functions that map structured inputs to structured outputs such as sequences or trees. This work focuses on sequence tagging problems, where both inputs and outputs are sequences of equal length. This is an important task with applications in many domains where sequential data appears naturally, including speech and natural language processing. We note that although our contributions are described in a sequence tagging framework, ideas in this work can be generalized to other structured prediction settings.

A standard approach to structured prediction is based on discriminative factorized linear models (Lafferty et al., 2001; Taskar et al., 2004), a direct generalization of linear multiclass prediction to the setting of structured input-output pairs. The key idea is to break structures into parts and describe the relation between inputs and outputs using a feature representation of each factor. The final scoring function for an input-output pair measures the compatibility of an output y with an input x. In factorized linear models the scoring function is assumed to be linear in the features that describe part-factored (x, y) pairs.

For sequence prediction, factors are usually associated with pairs of input-output sub-sequences, and the feature vector of a complete sequence is obtained by adding the features of individual factors. With this approach, sequence prediction reduces to a linear multiclass problem with an exponential number of outputs. Prediction in this case requires solving an inference problem over the space of possible outputs. Unlike with more complex models, in factorized linear models this inference problem can be efficiently solved. In addition, for appropriate loss functions, training these models can be formalized as a convex optimization problem (e.g., see Taskar et al., 2004). In practice, the caveat with these simple linear models is that in order to achieve good generalization performance, feature functions providing a good representation of the relevant input-output patterns for each problem domain need to be manually specified.

To address the limitations of factorized linear models, researchers have proposed to introduce latent variables and make the scoring function depend on those (Quattoni et al., 2004; Zhu et al., 2009; Wang & Mori, 2009; Yu & Joachims, 2009; Girshick et al., 2011; Schwing et al., 2012). As their name indicates, these variables are not

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

[1]The first two authors contributed equally.

observed at train or test time and their values need to be induced by the learning algorithm. The main idea is that models with latent variables have more freedom in explaining the relation between inputs and outputs, and can often identify the relevant patterns in a given domain. However, this increased expressivity comes at a price: the learning problem becomes non-convex, and exact prediction in some of these models becomes intractable. In addition, as the number of parameters increases, choosing appropriate regularization strategies becomes essential to avoid overfitting.

Two popular approaches to structured prediction with latent variables are based on linear and log-linear models. The latent SVM algorithm (Yu & Joachims, 2009) trains a linear model with latent variables by minimizing a regularized structured hinge loss under the assumption that predictions are scored by maximizing over all possible assignments to the latent variables. Log-linear factorized models can be trained using a conditional logistic loss (Quattoni et al., 2004), or max-margin approaches (Wang & Mori, 2009). Scoring in these models can be done by maximizing over all possible latent assignments, like with linear models. However, when the loss function has a probabilistic interpretation (e.g. the conditional logistic loss), log-linear models can be normalized by a partition function to induce a conditional distribution over output structures given an input structure and an assignment to the latent variables. A natural score in those cases is to consider the probability obtained by marginalizing over all possible latent assignments. Unfortunately, finding the output structure that maximizes these marginalized scores is usually an intractable inference problem. Alternative approaches consider piecewise inference, where each factor of the output structure is predicted independently by marginalizing over all other factors (Quattoni et al., 2004). Overall, more powerful methods require solving more complex learning and prediction problems, and in most cases there is no clear match between the loss used at training and the inference rule used at prediction time.

In this paper we propose a model for sequence tagging with latent variables based on *observable operator models* (OOM) (Jaeger, 2000). These models, which include HMMs as a special case, provide a powerful modeling mechanism for scoring input-output sequences. Scoring functions computed by OOM naturally marginalize over the set of latent variables. This implies that finding maximum score taggings involves an intractable inference problem. To make the inference problem tractable we consider a piecewise loss function similar to segmented minimum Bayes-risk decoders used in speech recognition (Goel et al., 2004). We use this inference to define a loss function that explicitly considers the marginalized scores that will be used at prediction time, thus coupling the scores used for prediction and learning. Then, by leveraging recent techniques in spectral learning of HMM and related models (Hsu et al., 2009; Bailly et al., 2009; Balle et al., 2011; Boots et al., 2011) and Hankel matrix completion (Balle & Mohri, 2012; Bailly et al., 2013b;a), we give an efficient algorithm for learning OOM for sequence tagging with this loss function. Our algorithm combines a nuclear norm regularized optimization with a spectral technique for recovering OOM from Hankel matrices. This provides a learning algorithm without local minima, which, in contrast with other spectral algorithms, tries to agnostically fit a model to the data without making any explicit assumption about the distribution that generated it.

We report experiments where the proposed method is used to solve a word-to-phoneme transcription task. We show that spectral regularization is an effective way to control the capacity of the model, and that it outperforms standard ℓ_2 regularization. Furthermore, our method achieves accuracies similar to a feature-based Conditional Random Field for smaller training sets, and improves for larger training sets, without the need to design any features for the factors.

1.1. Notation

Bold letters are used to denote vectors $\mathbf{v} \in \mathbb{R}^d$ and matrices $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$. Given a matrix $\mathbf{M}$ we write $\|\mathbf{M}\|_*$ for its trace/nuclear norm, which is the sum of its singular values. We use $\mathbf{M}^+$ to denote the Moore–Penrose pseudo-inverse of $\mathbf{M}$. Columns and rows of a matrix will sometimes be indexed by ordered sets $\mathcal{I}$ and $\mathcal{J}$. In this case we write $\mathbf{M} \in \mathbb{R}^{\mathcal{I} \times \mathcal{J}}$ to denote a matrix of size $|\mathcal{I}| \times |\mathcal{J}|$ with rows indexed by $\mathcal{I}$ and columns indexed by $\mathcal{J}$.

Let $\mathcal{X}$ be a finite set. We use the standard notation $\mathcal{X}^\star$ to denote the set of all finite sequences with elements in $\mathcal{X}$. The empty sequence is denoted by ϵ. Given a sequence $x = x_1 \cdots x_T \in \mathcal{X}^T$ of length T and indices $1 \leq s \leq t \leq T$ we use $x_{s:t}$ to denote the subsequence $x_s \cdots x_t$. Given two sequences $u, v \in \mathcal{X}^\star$ we write $w = u \cdot v = uv$ for their concatenation; u and v are said to be a prefixes and suffixes of w respectively. Given two sets of sequences $\mathcal{P}, \mathcal{S} \subseteq \mathcal{X}^\star$ we write $\mathcal{P} \cdot \mathcal{S}$ for the set obtained by taking every sequence of the form uv with $u \in \mathcal{P}$ and $v \in \mathcal{S}$.

2. Sequence Tagging with IO-OOM

2.1. Learning Setting

Let $\mathcal{X}$ be a set of input symbols and $\mathcal{Y}$ a set of output symbols. Our goal is to learn a model for *sequence tagging* that given an input sequence of T symbols $x \in \mathcal{X}^T$ produces an output sequence $y \in \mathcal{Y}^T$. In this setting, a model is given by a *scoring function* $F : (\mathcal{X} \times \mathcal{Y})^\star \to \mathbb{R}$ assigning a score $F(x, y)$ to each pair of input and output strings. Given an input sequence $x \in \mathcal{X}^T$ the model predicts an output se-

quence $y \in \mathcal{Y}^T$ by maximizing the score function:

$$\hat{y}(x) = \operatorname*{argmax}_{y \in \mathcal{Y}^T} F(x, y) \ . \tag{1}$$

The learning problem is specified via a class of possible scoring functions $\mathfrak{F}$ and a set of labeled training examples $S = \{(x^i, y^i)\}_{i=1}^m$.

A widely used scoring function is the class of factorized linear models (e.g., see Lafferty et al., 2001; Collins, 2002; Taskar et al., 2004). Assuming that some factor size k and a feature function $\boldsymbol{\phi}$ that maps factors to a vector representation are provided, the scoring function for sequence tagging is defined as:

$$F(x, y) = \sum_{t=k+1}^{T} \mathbf{w} \cdot \boldsymbol{\phi}(x, y_{t-k:t}) \tag{2}$$

To measure the accuracy of tagging we are given a *task loss* function $\ell : (\mathcal{Y} \times \mathcal{Y})^\star \to \mathbb{R}$ that measures the difference $\ell(y, y')$ between two output sequences $y, y' \in \mathcal{Y}^T$. For sequence tagging problems, a common choice for ℓ is the Hamming distance that counts the number of symbols where y and y' differ. This can be regarded as the analog of the zero-one loss for the sequence prediction task.

Minimizing the Hamming loss on the training data is computationally intractable for most $\mathfrak{F}$ of interest. The standard approach in these cases resorts to minimizing a surrogate loss L upper bounding the task loss. The learning problem is then:

$$\operatorname*{argmin}_{F \in \mathfrak{F}} \sum_{i=1}^{m} L(x^i, y^i; F) + \tau R(F) \tag{3}$$

where $R(F)$ is a regularization term that controls the capacity of the hypothesis F, and $\tau > 0$ is a regularization parameter. A common choice for L is the *structured hinge loss* (Taskar et al., 2004):

$$L_{\text{hinge}}(x, y; F) = \max_z \left[F(x, z) - F(x, y) + \ell(y, z) \right] \ .$$

2.2. Taggers with Latent Variables

The linearity of F may be too restrictive in some settings. One mechanism for going beyond linearity is to introduce latent variables. We briefly review two such approaches below.

The main assumption behind latent variable predictors is the existence of a set of latent variables $\mathcal{H}$, each providing a possible explanation for the relation between an input x and an output y. Given one possible explanation $h \in \mathcal{H}$, these models compute a vector of features: $\boldsymbol{\phi}(x, y, h) \in \mathbb{R}^D$ describing the interactions between x, y, and h. As in the linear methods, a weight vector $\mathbf{w}$ is used to define a score on x, y, h:

$$S(x, y, h) = \sum_{t=k+1}^{T} \mathbf{w} \cdot \boldsymbol{\phi}(x, y_{t-k:k}, h_{t-k:k}) \ .$$

Given a score function $S(x, y, h)$ there are two natural ways of obtaining a prediction.

Maximizing over h: Yu & Joachims (2009) suggested to obtain a prediction by maximizing over h and then over y. Namely, first define: $F(x, y) = \max_h S(x, y, h)$, and then predict via: $\hat{y}(x) = \operatorname{argmax}_y F(x, y)$, The loss of this predictor can be approximately minimized by using an appropriate variant of the structured hinge loss. This problem is known as *latent SVM* and has shown impressive results in several applications, most notably machine vision (Girshick et al., 2011). One major caveat is that the loss in this case is no longer convex, and a concave-convex alternating minimization procedure is used to find local minima.

Summing Over h: An alternative inference procedure for y is to consider the conditional distribution on y given by:

$$p(y, h|x) \propto \exp\left(S(x, y, h)\right) \ .$$

Marginalizing this over h yields a score function $F(x, y)$ (ignoring normalizing factors):

$$F(x, y) = \sum_h \exp\left(S(x, y, h)\right) \ .$$

At this point there are two ways of predicting y. Either to further marginalize $F(x, y)$ to obtain $F(x, y_i)$ and take $\operatorname{argmax}_{y_i} F(x, y_i)$. This is what is done in latent CRFs (Quattoni et al., 2004). An alternative, more consistent with the structured prediction methodology is to maximize $F(x, y)$ over y. However, as we argue later, this is a computationally hard task even for simple HMM like models.

We note that it is also possible to consider intermediate models that interpolate smoothly between the $\max$ and sum inference criteria with soft-max type functions (Schwing et al., 2012).

Training of latent variable models always turns out to be non-convex because of the non-linear structure of the prediction. The goal of this work is to construct structured predictors that are as powerful as latent variable ones, but have convex training procedures, and tractable prediction functions. To achieve this, we turn to the powerful family of predictors corresponding to input-output observable operator models.

2.3. Input-Output Observable Operator Models

We will now define *input-output observable operator models*, which are a family of scoring functions $F(x, y)$ which

generalize the latent variable models above. An IO-OOM with input $\mathcal{X}$ and output $\mathcal{Y}$ is a tuple $A = \langle \boldsymbol{\alpha}, \boldsymbol{\beta}, \{\mathbf{A}_{a,b}\}\rangle$, where $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^n$ and $\mathbf{A}_{a,b} \in \mathbb{R}^{n\times n}$ for each $a \in \mathcal{X}$ and $b \in \mathcal{Y}$. The dimension n is called the number of states of the model. Vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are respectively the initial and final weights of the model. The matrices $\mathbf{A}_{a,b}$ are known as the observable operators of the model.

The IO-OOM can be used to define a score for each x, y, h combination as follows:

$$S(x, y, h) = \boldsymbol{\alpha}(h_0) \left(\prod_{t=1}^{T} \mathbf{A}_{x_t,y_t}(h_{t-1}, h_t) \right) \boldsymbol{\beta}(h_T) \ .$$

A score $F(x, y)$ is then naturally defined by summation over h, and is given by the compact algebraic expression:

$$F(x, y) = \sum_h S(x, y, h) = \boldsymbol{\alpha}^\top \mathbf{A}_{x_1,y_1} \cdots \mathbf{A}_{x_T,y_T} \boldsymbol{\beta} \ .$$

Prediction of y can now be done as before by maximizing $F(x, y)$ over y.

We shall write F_A to denote the scoring function computed by A. IO-OOM are a powerful formalism stemming from stochastic process models in control theory (Jaeger, 1998; 2000), with tight relations to weighted automata and transducers (Droste et al., 2009), and predictive state representations (Littman et al., 2001).

To link the $F(x, y)$ of IO-OOM to that of latent variable models in Section 2.2, note that if the $\boldsymbol{\phi}(x, y, h)$ used in the latter uses the scope x_i, y_i, h_t, h_{t-1} then it is exactly captured by an IO-OOM. This is just a different way of saying that IO-OOM can compute the same functions as HMM. In fact it is well known that the class of functions computed by IO-OOM is a superset of those computed by HMMs (Jaeger, 2000).

From now on we will take IO-OOM with arbitrary coefficients as our class of hypothesis scoring functions. In the next section we show how to predict using these models, and in the sequel we focus on learning them from data.

2.4. Piecewise Prediction for IO-OOM

The first challenge to using IO-OOM is that of performing prediction, i.e. maximizing $F(x, y)$ in (2.3). It turns out that this is an NP hard task (Lyngsø & Pedersen, 2002) so that approximations are required.

A common approach to approximate the most likely output of IO-OOM is to use a piecewise version of F for scoring k-grams of y independently and then obtain an output that maximizes the sum of those scores using a dynamic programming algorithm (Goel et al., 2004). More formally, suppose we are given a scoring function F, an order $k \geq 1$, and sequences x, y of length $T \geq k$. Then one can consider the following approximate inference rule

$$\begin{aligned} \hat{y}_k(x) &= \operatorname*{argmax}_y \sum_{t=k+1}^{T} F(x_{t-k:t}, y_{t-k:t}) \\ &= \operatorname*{argmax}_y F_k(x, y) \ . \end{aligned} \tag{4}$$

The above only considers dependencies between input and output k-gram, and their additive effect. A more general version of this can be obtained by looking at l-grams for all $l \leq k$:

$$\begin{aligned} \hat{y}_{[k]}(x) &= \operatorname*{argmax}_y \sum_{l=1}^{k} \sum_{t=l+1}^{T} F(x_{t-l:t}, y_{t-l:t}) \\ &= \operatorname*{argmax}_y F_{[k]}(x, y) \ . \end{aligned}$$

Using these piecewise approximations of F we can define tractable relaxations of the structured hinge loss for IO-OOM. In particular, we define the *Viterbi-hinge losses* given by $L_k(x, y; F) = L_{\text{hinge}}(x, y; F_k)$ and $L_{[k]}(x, y; F) = L_{\text{hinge}}(x, y; F_{[k]})$. For sequences of length T these loss functions can be computed in time $O(T|\mathcal{Y}|^k)$ using Viterbi's algorithm.[2]

Now that we have a loss that we can evaluate for the class of IO-OOM, we can ask what is a natural regularizer for this class. In this case, an obvious choice is the number of states of an IO-OOM, which we denote by $|A|$. Using $\mathfrak{F}$ to denote the class of all scoring functions computed by IO-OOM, we would now like to learn a sequence tagger of the form F_A by solving the optimization problem

$$\operatorname*{argmin}_{A\in\mathfrak{F}} \sum_{i=1}^{m} L(x^i, y^i; F_A) + \tau|A| \ , \tag{5}$$

with $L = L_k$ or $L = L_{[k]}$ for some given order k.

In general this turns out to be a hard problem because even if L can be efficiently evaluated for small values of k, when $k \geq 2$ the dependence of $L(x, y; F_A)$ on the parameters of A is non-convex. In particular, assuming a fixed number of states for A, we see that the expression for $L(x, y; F_A)$ involves terms of the form $\boldsymbol{\alpha}^\top \mathbf{A}_{x_t,y_t} \cdots \mathbf{A}_{x_{t+l},y_{t+l}} \boldsymbol{\beta}$, which are polynomials of degree $l + 3$ in the parameters of $A = \langle \boldsymbol{\alpha}, \boldsymbol{\beta}, \{\mathbf{A}_{a,b}\}\rangle$, and are non-convex in A. Our agenda for the following sections is to obtain an approximate solution for this problem by relaxing the objective function and splitting the learning procedure into two tractable subproblems.

[2] We assume here that the cost of evaluating F is constant.

3. A Spectral Algorithm for Learning Max-Margin IO-OOM

To address the non-convexity of (5) we first observe that the loss is convex in the actual *values* of F_A.[3] This means that if we represent the optimization variable as a list of values on $(\mathcal{X} \times \mathcal{Y})^{\leq k}$ given by some function F instead of an IO-OOM A, the optimization problem becomes convex. However, this poses three key challenges:

- The values of F should correspond to valid IO-OOM.
- The regularizer $|A|$ should correspond to the rank of the IO-OOM described by F.
- Given an F that corresponds to an IO-OOM, how can one recover the matrices A that define it?

In what follows, we show that all these difficulties can be addressed, and that optimizing with respect to the values of F_A instead of the parameters of A yields an optimization problem over a set of Hankel matrices. Then, by leveraging recent techniques on spectral learning of OOM via Hankel matrix completion, we can recover an IO-OOM from the solution of this optimization problem with a simple SVD computation.

3.1. IO-OOM and Hankel Matrices

To explain our approach we first introduce the concept of Hankel matrices of F. These matrices provide an algebraic formalism to study the problem of recovering an IO-OOM from evaluations of its scoring function $F : (\mathcal{X} \times \mathcal{Y})^\star \to \mathbb{R}$.

Let $\mathcal{P}, \mathcal{S} \subseteq (\mathcal{X} \times \mathcal{Y})^\star$ be sets of input-output sequences. We call the pairs $(u, w) \in \mathcal{P}$ prefixes and the pairs $(v, z) \in \mathcal{S}$ suffixes. A Hankel matrix $\mathbf{H} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ for F over the basis $(\mathcal{P}, \mathcal{S})$ is obtained by taking the entries of $\mathbf{H}$ to be

$$\mathbf{H}((u, w), (v, z)) = F(uv, wz) \ . \tag{6}$$

A well-known theorem (Schützenberger, 1961; Carlyle & Paz, 1971; Fliess, 1974) states that a function $F : (\mathcal{X} \times \mathcal{Y})^\star \to \mathbb{R}$ can be realized by an IO-OOM with n states if and only if, for very possible basis the corresponding Hankel matrix $\mathbf{H}$ of F has rank at most n. A constructive version of this theorem lies at the very heart of recent spectral algorithms for learning HMM (Hsu et al., 2009), weighted automata (Bailly et al., 2009; Balle & Mohri, 2012), weighted transducers (Balle et al., 2011), and other families of recursively defined functions over sequences with discrete observations (Siddiqi et al., 2010; Boots et al., 2011).

When applied to IO-OOM, the spectral algorithm works as follows. Let us assume that F can be realized by a minimal IO-OOM with n states and that we are given a basis $(\mathcal{P}, \mathcal{S})$ such that the corresponding Hankel matrix of F, which we denote by $\mathbf{H}_{\epsilon,\epsilon}$, has rank n. Suppose we are also given, for each $(a, b) \in \mathcal{X} \times \mathcal{Y}$, the Hankel matrix $\mathbf{H}_{a,b} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ with entries $\mathbf{H}_{a,b}((u, w), (v, z)) = F(uav, wbz)$, and vectors $\mathbf{h}_\mathcal{P} \in \mathbb{R}^\mathcal{P}$ and $\mathbf{h}_\mathcal{S} \in \mathbb{R}^\mathcal{S}$ with entries given by $\mathbf{h}_\mathcal{P}(u, w) = F(u, w)$ and $\mathbf{h}_\mathcal{S}(v, z) = F(v, z)$. Then the following procedure recovers an IO-OOM $A = \langle \boldsymbol{\alpha}, \boldsymbol{\beta}, \{\mathbf{A}_{a,b}\} \rangle$ such that $F_A = F$: first, take the reduced SVD of $\mathbf{H}_{\epsilon,\epsilon} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$; and then, build A by taking $\mathbf{A}_{a,b} = (\mathbf{H}_{\epsilon,\epsilon}\mathbf{V})^+ \mathbf{H}_{a,b}\mathbf{V}$, $\boldsymbol{\alpha}^\top = \mathbf{h}_\mathcal{P}^\top \mathbf{V}$, and $\boldsymbol{\beta} = (\mathbf{H}_{\epsilon,\epsilon}\mathbf{V})^+ \mathbf{h}_\mathcal{S}$.

A fundamental property of this spectral algorithm is its robustness to noise. In particular, when the entries in these Hankel matrices are noisy estimates of values computed by an IO-OOM, the algorithm will produce a model which is close to the true model in terms of ℓ_1 distance. The paradigmatic example of this approach is the algorithm for learning HMM of (Hsu et al., 2009), where the noisy Hankel matrices come from empirical estimates of observation probabilities. A different approach, which is the one we pursue here, is to obtain an approximate Hankel matrix by solving a matrix completion problem with a loss function defined in terms of the task loss (Balle & Mohri, 2012; Bailly et al., 2013b;a). The next section shows how to apply this method to max-margin learning of IO-OOM.

3.2. Max-Margin Completion of Hankel Matrices

The discussion in the previous section implies that the set of F values for an IO-OOM of rank n is equivalent to the set of Hankel matrices of rank n. Thus, the optimization of (5) can be equivalently performed on Hankel matrices with rank regularization. We next elaborate on this problem, and address the difficulty of optimizing the rank.

To obtain a Hankel matrix completion problem from (5) we proceed as follows. First we parameterize an IO-OOM via its corresponding Hankel matrix $\mathbb{H}(\mathcal{P}, \mathcal{S})$ (over some fixed basis $(\mathcal{P}, \mathcal{S})$) instead of the original A matrices representation. Next we change the regularizer: instead of the number of states of an IO-OOM, the Hankel/IO-OOM equivalence theorem says that we can take the rank of the Hankel matrix as a regularizer that directly controls the number of states of the learned IO-OOM. This yields the following optimization problem

$$\operatorname*{argmin}_{\mathbf{H} \in \mathbb{H}(\mathcal{P},\mathcal{S})} \sum_{i=1}^{m} L(x^i, y^i; \mathbf{H}) + \tau \operatorname{rank}(\mathbf{H}) \ , \tag{7}$$

which by the relation between IO-OOM and Hankel matrices is almost equivalent to (5), with the only difference that now the search is conducted over the class of all IO-OOM that can be recovered from Hankel matrices in $\mathbb{H}(\mathcal{P}, \mathcal{S})$.

The choice of a right basis $(\mathcal{P}, \mathcal{S})$ is essential if we want to be able to use information from all training examples,

[3] Specifically, it is piecewise linear in those.

and also guarantee that we can recover an IO-OOM from the learned Hankel matrix. When using loss $L_{[k]}$, the first point requires that we have $\cup_{l=1}^{k} \text{grams}_l(S) \subseteq \mathcal{P} \cdot \mathcal{S}$, where $\text{grams}_l(S)$ is the set of all input-output l-grams observed in the training sample S. Similarly, for $L = L_k$ we need a basis such that $\text{grams}_k(S) \subseteq \mathcal{P} \cdot \mathcal{S}$. To recover operators for F using the spectral method, we will retrieve the $\mathbf{H}_{\epsilon,\epsilon}$, $\mathbf{H}_{a,b}$, $\mathbf{h}_{\mathcal{P}'}$ and $\mathbf{h}_{\mathcal{S}'}$ defined in Section 3.1 as sub-blocks of the learned Hankel matrix; note that these sub-blocks will correspond to a basis $(\mathcal{P}', \mathcal{S}')$ smaller than the $(\mathcal{P}, \mathcal{S})$ used in (7) (see (Balle & Mohri, 2012) for further details). One way to guarantee that $\mathbf{H}$ contains the right sub-blocks is by choosing an initial basis $(\mathcal{P}', \mathcal{S}')$ with $(\epsilon, \epsilon) \in \mathcal{P}' \cap \mathcal{S}'$, and then take $\mathcal{P} = \mathcal{P}' \cdot (\mathcal{X}' \times \mathcal{Y}')$ and $\mathcal{S} = \mathcal{S}'$, where $\mathcal{X}' = \mathcal{X} \cup \{\epsilon\}$ and $\mathcal{Y}' = \mathcal{Y} \cup \{\epsilon\}$. Note that (6) implies that entries in Hankel matrices corresponding to pairs of prefixes and suffixes that yield the same input-output sequence must have the same value. Taking a larger basis implies that more constraints will need to be satisfied in the matrix search space considered by the completion algorithm. Therefore, a good strategy to keep the optimization as simple as possible is to choose the minimal basis satisfying the constraints outlined above.

The last step needed to obtain a convex optimization is to relax the regularization term in (7), replacing it with the nuclear norm of the Hankel matrix. This last step is a usual approach in matrix completion problems, and can be justified by observing that because the nuclear norm of a matrix is precisely the ℓ_1 norm of its singular values, minimizing the nuclear norm yields (approximately) low rank matrices in the same way that ℓ_1 regularization yields sparse vectors (Candès & Recht, 2009). Therefore, assuming the basis $(\mathcal{P}, \mathcal{S})$ is given, we can put together the ingredients described so far to get the following optimization problem over a space of Hankel matrices:

$$\hat{\mathbf{H}}_S \in \operatorname*{argmin}_{\mathbf{H} \in \mathbb{H}(\mathcal{P},\mathcal{S})} \sum_{i=1}^{m} L(x^i, y^i; \mathbf{H}) + \tau \|\mathbf{H}\|_* \ . \quad (8)$$

One last observation is that in the case $L = L_k$, it is possible to choose the basis in a way that the space of Hankel matrices in (8) contains no equality constraints. This can be interesting if we are willing to give up on the ability to recover an IO-OOM from the learned matrix, and use the values in $\hat{\mathbf{H}}_S$ for predicting using the inference rule (4). From an algorithmic point of view, this can be interpreted as taking a trade-off between memory and time for predicting with the learned model. Obtaining the IO-OOM operators from $\hat{\mathbf{H}}_S$ yields a high rate of compression, at the price of needing several matrix multiplications every time a score needs to be computed. On the other hand, one can store the whole matrix $\hat{\mathbf{H}}_S$ and treat it as cached scores for each possible input-output k-gram.

4. Optimization Details

In this section we describe the details of an optimization algorithm for solving problem 8. For simplicity we focus on the setting $L = L_k$ with a basis $(\mathcal{P}, \mathcal{S})$ where the Hankel matrices in $\mathbb{H}(\mathcal{P}, \mathcal{S})$ contain no equality constraints. It is easy to extend the algorithm for basis with equality constraints by adding an extra projection step.

Recall that our goal is to minimize the following function:

$$g(\mathbf{H}) = \sum_{i=1}^{m} L_k(x^i, y^i; \mathbf{H}) + \tau \|\mathbf{H}\|_* \ ,$$

To simplify notation we denote the overall loss by $L_S(\mathbf{H})$ so that $g(\mathbf{H}) = L_S(\mathbf{H}) + \tau \|\mathbf{H}\|_*$.

Since both the loss and the trace norm are convex it turns out that $g(\mathbf{H})$ is convex, albeit non differentiable. In recent years, many algorithms have been proposed for optimizing trace norm regularized problems (e.g., see Jaggi & Sulovsk, 2010; Shalev-Shwartz et al., 2011; Ji & Ye, 2009). Some of these methods only apply to smooth losses and are thus not applicable here.[4] Furthermore, some of these require solving optimization problems that are costly in our setting (e.g., backward fitting as in Shalev-Shwartz et al., 2011).

Here we use a simple optimization scheme known as Forward Backward Splitting, or FOBOS (Duchi & Singer, 2009). FOBOS is similar to proximal gradient, with the exception that it linearizes the loss part of $g(\mathbf{H})$ (and not the regularizer). It corresponds to the following repeated updates on $\mathbf{H}_t$. First, take a step in the direction of the subgradient of the loss (ignoring the trace norm):

$$\mathbf{H}_{t+0.5} = \mathbf{H}_t - \eta_t \frac{\partial L_S(\mathbf{H}_t)}{\partial \mathbf{H}}$$

where $\eta_t = \frac{c}{\sqrt{t}}$ is a step size and $\frac{\partial L_S(\mathbf{H}_t)}{\partial \mathbf{H}}$ is a sub gradient of the loss at $\mathbf{H}_t$. This is easily evaluated for the hinge loss we consider, and involves finding the argmax of the hinge loss. In the second step, find an $\mathbf{H}$ that is close to $\mathbf{H}_{t+0.5}$ but with a trace norm penalty:

$$\mathbf{H}_{t+1} = \operatorname*{argmin}_{\mathbf{H}} \|\mathbf{H}_{t+0.5} - \mathbf{H}\|_2^2 + \eta_t \tau \|\mathbf{H}\|_* \ .$$

This step can be solved via SVD thresholding (see Cai et al., 2010) as follows. Use SVD to write $\mathbf{H}_{t+1}$ as $\mathbf{H}_{t+1} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ with $\mathbf{\Sigma}$ a diagonal matrix and $\mathbf{U}, \mathbf{V}$ orthogonal matrices. Denote by σ_i the i^{th} element on the diagonal of $\mathbf{\Sigma}$. Define a new matrix $\bar{\mathbf{\Sigma}}$ with diagonal elements $\bar{\sigma}_i = \max[\sigma_i - \eta_t \tau, 0]$. The update is then: $\mathbf{H}_{t+1} = \mathbf{U}\bar{\mathbf{\Sigma}}\mathbf{V}^\top$. Each update requires calculating an

[4] Note however that it is possible to smooth the loss using standard methods (Nesterov, 2005) and then use algorithms for smooth objectives.

Algorithm 1 FOBOS minimization of $L_S(\mathbf{H}) + \tau\|\mathbf{H}\|_*$.

Initialize $\mathbf{H}_0 = 0$
while $t <$ MaxIter **do**
  Set $\mathbf{G}_t$ to a subgradient of $L_S(\mathbf{H})$ at $\mathbf{H}_t$.
  Set $\eta_t = \frac{c}{\sqrt{t}}$.
  Set $\mathbf{H}_{t+0.5} = \mathbf{H}_t - \eta_t \mathbf{G}_t$.
  Calculate the SVD of $\mathbf{H}_{t+0.5}$ as $\mathbf{H}_{t+0.5} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$.
  Define a diagonal matrix $\bar{\mathbf{\Sigma}}$ such that $\bar{\sigma}_i = \max[\sigma_i - \eta_t\tau, 0]$.
  Set $\mathbf{H}_{t+1} = \mathbf{U}\bar{\mathbf{\Sigma}}\mathbf{V}^\top$.
end while

SVD of $\mathbf{H}$. For the problem sizes we considered this was fairly fast to compute. It can be further sped up by using the fact that only leading singular vectors are required and using methods that only calculate these. Furthermore $\mathbf{H}_{t+0.5}$ is typically a low rank plus sparse matrix, which can be used for further speedups.

It can be shown (Duchi & Singer, 2009) that FOBOS converges to the global optimum of $g(\mathbf{H})$ at a rate of $O(\varepsilon^{-2})$. Although rates of $O(\varepsilon^{-1})$ are possible via accelerated gradient (Ji & Ye, 2009), we find it is sufficient for the applications we consider.

5. Experimental Results

In this section we present experimental results on a text-to-phonemes task. We compare the max-margin sequence tagging method with spectral regularization to several alternative methods.

We used the "Nettalk" dataset (*a.k.a.* the Connectionist Bench), available from the UCI repository (Sejnowski & Rosenberg, 1987). The data consists of 20,008 English words (formed with 26 letters) paired with a phonetic representation (using 51 phoneme symbols). For example, `apple` is paired with `@p-L-` and `hippopotamus` with `hIp-xpatxmxs`. Input-output sequence pairs in the data have one-to-one monotonic alignments (using a special symbol "-" to represent many-to-one letter to phoneme associations, as shown in the examples), which results in paired sequences of equal length. We consider the task of predicting the phoneme sequence given an input word. We use Hamming accuracy to compare the performance of different methods. We randomly divided the dataset into 15,000 training sequences, 1,034 development sequences and 3,974 test sequences. We created 6 training sets of increasing size to obtain a learning curve.

We trained several types of sequence tagging models that exploit trigram factorizations, and for all of them we used the Viterbi algorithm as inference routine[5]. We compare the following models:

- **IO-OOM with Spectral Max-Margin:** To set the basis of the Hankel matrix, we use all observed bi-symbols for prefixes and for suffixes. In this configuration the matrix $\hat{\mathbf{H}}_S$ we obtain has values for all input-output trigrams, and we can directly use it to tag sequences. The parameters of the method are the regularization constant τ, an initial learning rate c and the number of iterations.

- **IO-OOM with Unregularized Max-Margin:** This method drops the spectral regularizer. It is equivalent to setting $\tau = 0$.

- **IO-OOM with L2 Max-Margin:** This method replaces the spectral regularizer with a standard ℓ_2 penalty on the coefficients of the Hankel matrix; that is, the trace norm regularizer $\|\mathbf{H}\|_*$ is replaced with a Frobenius norm regularizer $\|\mathbf{H}\|_F$.

- **Spectral IO-HMM:** The standard spectral method for HMM applied to joint sequences (Hsu et al., 2009), which estimates Hankel matrices directly from empirical counts on the training sample. We also tried the spectral method for conditional IO models by (Balle et al., 2011) but we obtained development accuracies significantly lower than other methods, around 70%.

- **Latent SVM:** The latent SVM by (Yu & Joachims, 2009), which follows a Max-Max approach and attempts to solve a non-convex problem. We used their implementation adapted to sequence tagging, using the same features as an IO-OOM seen as a log-linear model. We could only find configurations that obtained very moderate training accuracies, at the level of 60%.

- **CRF and Averaged Perceptron:** A standard feature-based trigram Conditional Random Field tagger (CRF), that represents input-output trigrams using sub-pattern features. We used the code by (Collins et al., 2008), which implements structured prediction learning algorithms and tagging models that obtain state-of-the-art accuracies in part-of-speech tagging. As features, it exploits each bi-symbol with various combinations of the input context, and with various combinations of the the output context in the trigram.

[5]The number of output trigrams is a constant of all sequence taggers, and considering all possible output trigrams for this data (i.e. 51^3) is impractical. Instead of pairing each input symbol with each output symbol, for each training set we restricted to the set of bi-symbols observed in the training data. For the largest training set, input symbols are paired with 2 to 18 output symbols. This dramatically reduces the number of possible trigrams, and has very low impact in prediction accuracy. This strategy is common in tagging tasks such as part-of-speech tagging.

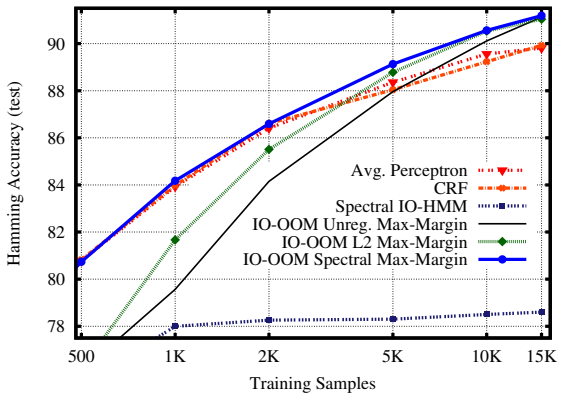

Figure 1. Learning curve for various methods. We plot test accuracy with respect to size of the training set.

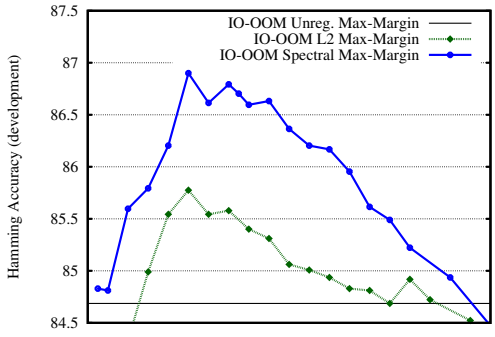

Figure 2. Regularization path for Spectral and L2 regularization, using a training set of 2,000 samples. We plot development accuracy with respect to values of the regularization constant τ. The range of relevant τ values of each model has been normalized.

> For parameter estimation we tried the log-linear loss (i.e. CRF), structured max-margin loss, and averaged Perceptron. All methods performed similarly.

We trained all models on training sets of different sizes, trying a wide range of regularization constants when applicable. We used accuracy on development sequences to pick the best configuration. Figure 1 plots test accuracies for each method. We can see that spectral regularization largely improves over the unregularized model: in the first points of the curve the spectral approach requires half of the training samples to get to the same accuracy as the unregularized one. Comparing regularizers, the spectral one shows a better curve than the ℓ_2, especially in the first part of the curve. We attribute this to the ability of the regularizer to factor input-output trigrams using a hidden representation. As the number of samples increases, the non-factored representation of trigrams is effective and the two regularizers perform similarly. Figure 2 presents the regularization path for the spectral and ℓ_2 regularizers for 2,000 training samples. At their best configurations in development, the Spectral improves 2.2 points over the unregularized model, while the ℓ_2 improves 1.1 points.

We can observe that in the first part of the curve the spectral regularization performs very similarly to feature-based models trained with Perceptron or CRF. This suggests that the factorization obtains a representation that is as effective as manually-specified sub-pattern features. As the number of examples increases, the full trigram parameters become useful and the spectral regularizer leverages them, while the feature-based models stop improving. In all, the spectral regularization technique seems to perform the best in any of the empirical scenarios.

6. Conclusion

The central contribution of this paper is to derive a convex formulation for max-margin structured prediction with latent variables and max-sum prediction rules. The main outcome of our work is a new regularization approach, inspired by spectral techniques, which is specifically designed for structured prediction tasks. This means that, instead of designing the features of a factorized linear model, we can consider rich function spaces and implicitly induce features via proper regularization. Our experiments confirm that the proposed regularization framework leads to an effective way of controlling the capacity of structured prediction models.

Spectral methods for latent variable models require a single SVD calculation and are non-iterative in nature. While this is an attractive feature, it is justified only in the case where data is indeed generated from the assumed latent variable model (e.g., in Hsu et al., 2009, it is assumed that data is generated from an HMM). In our formalism no such assumptions are made. The latent variable construct is only a mechanism for generating a prediction function. Thus the Hankel matrix we consider does not correspond to observed statistics of some assumed model but rather corresponds to unknown parameters of a prediction function. Our work suggests that core ideas behind spectral learning can have wide applicability to structured prediction. Our results can be generalized to other loss functions as long as they have surrogate piecewise approximations.

Acknowledgments

We thank the reviewers for their helpful comments. This work was supported by projects XLike (FP7-288342), ERA-Net CHISTERA VISEN, TACARDI (TIN2012-38523-C02-00) and by the ISF Centers of Excellence (grant 1789/11). Xavier Carreras was supported by the Ramón y Cajal program of the Spanish Government (RYC-2008-02223). Borja Balle received support from NSERC and the James McGill Research Fund.

References

Bailly, R., Denis, F., and Ralaivola, L. Grammatical inference as a principal component analysis problem. *ICML*, 2009.

Bailly, R., Carreras, X., Luque, F., and Quattoni, A. Unsupervised spectral learning of WCFG as low-rank matrix completion. *EMNLP*, 2013a.

Bailly, R., Carreras, X., and Quattoni, A. Unsupervised spectral learning of finite state transducers. In *NIPS*, 2013b.

Balle, B. and Mohri, M. Spectral learning of general weighted automata via constrained matrix completion. *NIPS*, 2012.

Balle, B., Quattoni, A., and Carreras, X. A spectral learning algorithm for finite state transducers. *ECML*, 2011.

Boots, B., Siddiqi, S., and Gordon, G. Closing the learning planning loop with predictive state representations. *International Journal of Robotics Research*, 2011.

Cai, J-F., Candès, E.J., and Shen, Z. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 2010.

Candès, E.J. and Recht, B. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 2009.

Carlyle, J. W. and Paz, A. Realizations by stochastic finite automata. *Journal of Computer Systems Science*, 1971.

Collins, M. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the conference on Empirical methods in natural language processing*, pp. 1–8, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

Collins, M., Globerson, A., Koo, T., Carreras, X., and Bartlett, P. Exponentiated Gradient Algorithms for Conditional Random Fields and Max-Margin Markov Networks. *Journal of Machine Learning Research*, 9(2):1775–1822, 2008.

Droste, M., Kuich, W., and Vogler, H. *Handbook of weighted automata*. Springer, 2009.

Duchi, J. and Singer, Y. Efficient online and batch learning using forward backward splitting. *The Journal of Machine Learning Research*, 2009.

Fliess, M. Matrices de Hankel. *Journal de Mathématiques Pures et Appliquées*, 1974.

Girshick, Ross B., Felzenszwalb, Pedro F., and Mcallester, David. Object detection with grammar models. In *In NIPS*, 2011.

Goel, V., Kumar, S., and Byrne, W. Segmental minimum bayes-risk decoding for automatic speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 2004.

Hsu, D., Kakade, S. M., and Zhang, T. A spectral algorithm for learning hidden Markov models. *COLT*, 2009.

Jaeger, H. Discrete-time, discrete-valued observable operator models: A tutorial. Technical report, GMD Report 42, German National Research Center for Information Technology, 1998.

Jaeger, H. Observable operator models for discrete stochastic time series. *Neural Computation*, 2000.

Jaggi, M. and Sulovsk, M. A simple algorithm for nuclear norm regularized problems. In *ICML*, 2010.

Ji, S. and Ye, J. An accelerated gradient method for trace norm minimization. In *ICML*, 2009.

Lafferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pp. 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

Littman, M. L., Sutton, R. S., and Singh, S. Predictive representations of state. In *NIPS*, 2001.

Lyngsø, R. B. and Pedersen, C. N. S. The consensus string problem and the complexity of comparing hidden markov models. *J. Comput. Syst. Sci.*, 2002.

Nesterov, Yu. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

Quattoni, A., Collins, M., and Darrell, T. Conditional random fields for object recognition. In *In NIPS*, pp. 1097–1104. MIT Press, 2004.

Schützenberger, M. P. On the definition of a family of automata. *Information and Control*, 1961.

Schwing, A.G., Hazan, T., Pollefeys, M., and Urtasun, R. Efficient structured prediction with latent variables for general graphical models. In *ICML*, 2012.

Sejnowski, T.J. and Rosenberg, C.R. Parallel networks that learn to pronounce english text. *Complex Systems*, 1:145–168, 1987.

Shalev-Shwartz, S., Gonen, A., and Shamir, O. Large-scale convex minimization with a low-rank constraint. In *ICML*, 2011.

Siddiqi, S.M., Boots, B., and Gordon, G.J. Reduced-rank hidden Markov models. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Taskar, B., Guestrin, C., and Koller, D. Max margin Markov networks. In Thrun, S., Saul, L., and Schölkopf, B. (eds.), *Advances in Neural Information Processing Systems 16*, pp. 25–32. MIT Press, Cambridge, MA, 2004.

Wang, Y. and Mori, G. Max-margin hidden conditional random fields for human action recognition. In *CVPR*, 2009.

Yu, C-N. and Joachims, T. Learning structural svms with latent variables. In *ICML*, 2009.

Zhu, Jun, Xing, Eric P., and Zhang, Bo. Partially observed maximum entropy discrimination markov networks. In *Proc. NIPS*, 2009.