Structured Predictions Cascades

David Weiss  
*University of Pennsylvania*

Ben Taskar  
*University of Pennsylvania, taskar@cis.upenn.edu*

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

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

**Recommended Citation**  

*Structured Prediction Cascades*, D. Weiss and B. Taskar. *International Conference on Artificial Intelligence and Statistics (AISTATS), Sardinia, Italy*, May 2010.  
Copyright is held by the authors.

This paper is posted at ScholarlyCommons. [http://repository.upenn.edu/cis_papers/546](http://repository.upenn.edu/cis_papers/546)  
For more information, please contact repository@pobox.upenn.edu.
Structured Predictions Cascades

Abstract
Structured prediction tasks pose a fundamental trade off between the need for model complexity to increase predictive power and the limited computational resources for inference in the exponentially-sized output spaces such models require. We formulate and develop structured prediction cascades: a sequence of increasingly complex models that progressively filter the space of possible outputs. We represent an exponentially large set of filtered outputs using max marginals and propose a novel convex loss function that balances filtering error with filtering efficiency. We provide generalization bounds for these loss functions and evaluate our approach on handwriting recognition and part-of-speech tagging. We find that the learned cascades are capable of reducing the complexity of inference by up to five orders of magnitude, enabling the use of models which incorporate higher order features and yield higher accuracy.

Disciplines
Computer Sciences

Comments

Copyright is held by the authors.
Structured Prediction Cascades

David Weiss and Ben Taskar
University of Pennsylvania

Abstract

Structured prediction tasks pose a fundamental trade-off between the need for model complexity to increase predictive power and the limited computational resources for inference in the exponentially-sized output spaces such models require. We formulate and develop structured prediction cascades: a sequence of increasingly complex models that progressively filter the space of possible outputs. We represent an exponentially large set of filtered outputs using max marginals and propose a novel convex loss function that balances filtering error with filtering efficiency. We provide generalization bounds for these loss functions and evaluate our approach on handwriting recognition and part-of-speech tagging. We find that the learned cascades are capable of reducing the complexity of inference by up to five orders of magnitude, enabling the use of models which incorporate higher order features and yield higher accuracy.

1 Introduction

The trade-off between approximation and estimation error is fundamental in learning complex prediction models. In structured prediction tasks, such as part-of-speech tagging, machine translation and gene prediction, the factor of computation time also plays an important role as models with increasing complexity of inference are considered. For example, a first order conditional random field (CRF) (Lafferty et al., 2001) is fast to evaluate but may not be an accurate model for phoneme recognition, while a fifth order model is more accurate, but prohibitively expensive for both learning and prediction. (Model complexity can also lead to overfitting problems due the sparseness of the training data. We do not specifically address this problem in our paper other than judiciously regularizing model parameters.)

In practice, model complexity is limited by computational constraints at prediction time, either explicitly by the user or implicitly because of the limits of available computation power. We therefore need to balance expected error with inference time. A common solution is to use heuristic pruning techniques or approximate search methods in order to make higher order models feasible. While it is natural and commonplace to prune graphical model state space, the problem of explicitly learning to control the error/computation tradeoff has not been addressed. In this paper, we formulate the problem of learning a cascade of models of increasing complexity that progressively filter a given structured output space, minimizing overall error and computational effort at prediction time according to a desired tradeoff. The contributions of this paper are:

- A novel convex loss function specifically geared for learning to filter accurately and effectively.
- A simple online algorithm for minimizing this loss using standard inference methods.
- Theoretical analysis of generalization of the cascade (in terms of both accuracy and efficiency).
- Evaluation on two large-scale applications: handwriting recognition and part-of-speech tagging.

2 Related Work

Heuristic methods for pruning the search space of outputs have been exploited in many natural language processing and computer vision tasks. For part-of-speech tagging, perhaps the simplest method is to limit the possible tags for each word to those only seen as its labels in the training data. For example, the MXPOST tagger (Ratnaparkhi, 1996) and many others use this technique. In our experiments, we compare to this simple trick and show that our method is much more accurate and effective in reducing the output space. In parsing, the several works (Charniak, 2000; Carreras et al., 2008; Petrov, 2009) use a “coarse-to-fine” idea.
closely related to ours: the marginals of a simple context-free grammar or dependency model are used to prune the parse chart for a more complex grammar. We also compare to this idea in our experiments. The key difference with our work is that we explicitly learn a sequence of models tuned specifically to filter the space accurately and effectively. Unlike the work of Petrov (2009), however, we do not learn the structure of the hierarchy of models but assume it is given by the designer.

It is important to distinguish the approach proposed here, in which we use exact inference in a reduced output space, with other approximate inference techniques that operate in the full output space (e.g., Druck et al. (2007), Pal et al. (2006)). Because our approach is orthogonal to such approximate inference techniques, it is likely that the structured pruning cascades we propose could be combined with existing methods to perform approximate inference in a reduced output space.

Our inspiration comes partly from the cascade classifier model of Viola and Jones (2002), widely used for real-time detection of faces in images. In their work, a window is scanned over an image in search of faces and a cascade of very simple binary classifiers is trained to weed out easy and frequent negative examples early on. In the same spirit, we propose to learn a cascade of structured models of increasing order that weed out easy incorrect assignments early on.

3 Structured Prediction Cascades

Given an input space \( \mathcal{X} \), output space \( \mathcal{Y} \), and a training set \( S = \{ (x^1, y^1), \ldots, (x^n, y^n) \} \) of \( n \) independent and identically-distributed (i.i.d.) random samples from a joint distribution \( D(X,Y) \), the standard supervised learning task is to learn a hypothesis \( h : \mathcal{X} \mapsto \mathcal{Y} \) that minimizes the expected loss \( \mathbb{E}_D [ \mathcal{L}(h(x), y) ] \) for some non-negative loss function \( \mathcal{L} : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+ \).

We consider the case of structured classification where \( Y \) is a \( \ell \)-vector of variables and \( \mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_\ell \). In many settings, the number of random variables \( Y \) differs depending on input \( X \) (for example, length of the sentence in part of speech tagging), but for simplicity of notation, we assume a fixed number \( \ell \) here. We denote the components of \( y \) as \( y = \{ y_1, \ldots, y_\ell \} \), where \( y_i \in \{ 1, \ldots, K \} \). The linear hypothesis class we consider is of the form:

\[
h_w(x) = \argmax_{y \in \mathcal{Y}} w^T f(x, y)
\]  

(1)

where \( w \in \mathbb{R}^p \) is a vector of parameters and \( f : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}^p \) is a function mapping \( (x,y) \) pairs to a set of \( p \) real-valued features. We further assume that \( f \) decomposes over a set of cliques \( \mathcal{C} \subseteq \mathbb{P}\{X,Y\} \) (where \( \mathbb{P} \) is a powerset):

\[
w^T f(x, y) = \sum_{c \in \mathcal{C}} w_c^T f_c(x, y_c).
\]  

(2)

Above, \( y_c \) is an assignment to the subset of \( Y \) variables in the clique \( c \) and we will use \( Y_c \) to refer to the set all assignments to the clique. By considering different cliques over \( X \) and \( Y \), \( f \) can represent arbitrary interactions between the components of \( x \) and \( y \). Thus, evaluating \( h_w(x) \) is not always tractable, and computational resources limit the expressiveness of the features that can be used.

For example, a first order Markov sequence model has cliques \( \{ Y_i, Y_{i+1}, X \} \) to score features depending on emissions \( \{ X,Y_i \} \) and transitions \( \{ Y_i, Y_{i+1} \} \):

\[
w^T f(x, y) = \sum_{i=1}^{\ell} w^T f_i(x, y_i, y_{i+1}).
\]  

(3)

For the first order model, evaluating (1) requires \( O(K^2 \ell) \) time using the Viterbi decoding algorithm. For an order-\( d \) Markov model, with cliques over \( \{ y_i, \ldots, y_{i+d}, x \} \), inference requires \( O(k^{d+1} \ell) \) time.

Several structured prediction methods have been proposed for learning \( w \), such as conditional random fields (Lafferty et al., 2001), structured perceptron (Collins, 2002), hidden Markov support vector machines (Altun et al., 2003) and max-margin Markov networks (Taskar et al., 2003). Our focus here is instead on learning a cascade of increasingly complex models in order to efficiently construct models that would otherwise be intractable.

3.1 Cascaded inference with max-marginals

We will discuss how to learn a cascade of models in Section 4, but first we describe the inference procedure. The basic notion behind our approach is very simple: at each level of the cascade, we receive as input a set of possible clique assignments corresponding to the cliques of the current level. Each level further filters this set of clique assignments and generates a set of possible clique assignments which are passed as input to the next level. Note that each level is able to prune further than the preceding level because it can consider higher-order interactions. At the end of the cascade, the most complex model chooses a single prediction as usual, but it needs only consider the clique assignments that have not already been pruned. Finally, pruning is done using clique max-marginals, for reasons which we describe later in this section.

An example of a cascade for sequential prediction using Markov models is shown in Figure 1. A \( d \)-order Markov model has maximal cliques \( \{ X, Y_i, Y_{i+1}, \ldots, Y_{i+d} \} \). We
can consider a cascade of sequence models of increasing order as a set of bigram models where the state space is increasing exponentially by a factor of $K$ from one model to the next. Given a list of valid assignments $\mathcal{V}^i$ in a $d$-order model, we can generate an expanded list of valid assignments $\mathcal{V}^{i+1}$ for a $(d+1)$-order model by concatenating the valid $d$-grams with all possible additional states.

More generally, for a given input $x$, for each maximal clique $c \in \mathcal{C}$ in the current level of the cascade, we have a list of valid assignments $\mathcal{V}_c \subseteq \mathcal{Y}_c$. Inference is performed only over the set of valid clique assignments $\mathcal{V} = \bigcup_{c \in \mathcal{C}} \mathcal{V}_c$. Using the filtering model $\mathbf{w}$ of the current cascade level, the clique assignments in $\mathcal{V}_c$ are scored by their max-marginals, and a threshold $t$ is chosen to prune any $y_c \in \mathcal{V}_c$ with a score less than the threshold. The sets $\mathcal{V}$ are then passed to the next level of the cascade, where higher-order cliques consistent with unpruned cliques are constructed. If we can eliminate at least a fraction of the entries in $\mathcal{V}$ on each round of the cascade, then $|\mathcal{V}|$ decreases exponentially fast and the overall efficiency of inference is controlled.

Thus, to define the cascade, we need to define: (1) the set of models to use in the cascade, and (2) a procedure to choose a threshold $t$. In the remainder of this section we discuss our approach to these decisions.

First, to define a set of models for the cascade, we require only that the sets of cliques of the models form a nesting sequence. The cliques of the models must satisfy the relation,

$$\mathcal{C}^1 \subseteq \mathcal{C}^2 \subseteq \cdots \subseteq \mathcal{C}^d,$$

where $\mathcal{C}^i$ is the set of cliques of the $i$'th model of the cascade. In other words, every clique assignment $y_c$ of the $i$'th model is contained in at least one clique assignment $y_c'$ of the $(i+1)$'th model. This property allows for the use of increasingly complex models as the depth of the cascade increases. As long as (4) holds, a simple and intuitive mapping from the set of valid cliques of the $i$'th model $\mathcal{V}^i$ to a corresponding set of valid cliques of the $(i+1)$'th model, $\mathcal{V}^{i+1}$:

$$\mathcal{V}^{i+1} = \{ y_c \in \mathcal{V}_c \mid \forall c' \in \mathcal{C}^i, c' \subseteq c, y_{c'} \in \mathcal{V}_c' \}$$

This is the set of clique assignments $y_c$ in the $(i+1)$'th model for which all consistent clique assignments $y_{c'}$ for subcliques $c' \in \mathcal{C}^i$ have not been pruned by the $i$'th model.

In order to filter clique assignments, we use their max-marginals. We introduce the shorthand $\theta_x(y) = \mathbf{w}^\top \mathbf{f}(x,y)$ for the cumulative score of an output $y$, and define the max marginal $\theta^*_x(y_c)$ as follows:

$$\theta^*_x(y_c) = \max_{y_c' \in \mathcal{V}_c} \{ \theta_x(y'_c) : y'_c = y_c \}.$$

Computing max-marginals can be achieved using the same dynamic programming inference procedures in factor graphs as would be used to evaluate (1). Most importantly, max marginals satisfy the following simple property:

**Lemma 1 (Safe Filtering).** If $\theta_x(y) > \theta^*_x(y'_c)$ for some $c$, then $y_c \neq y'_c$.

Lemma 1 states that if the score of the true label $y$ is greater than the max marginal of a clique assignment $y'_c$, then that clique assignment $y'_c$ must be inconsistent with the truth $y$. The lemma follows from the fact that the max marginal of any clique assignment $y_c$ consistent with $y$ is at least the score of $y$.

A consequence of Lemma 1 is that on training data, a sufficient condition for the target output $y$ not to be pruned is that the threshold $t$ is lower than the score $\theta_x(y)$. Note that a similar condition does not generally hold for standard sum-product marginals of a CRF (where $p(y|x) \propto e^{\theta_x(y)}$), which motivates our use of max-marginals.

The next component of the inference procedure is choosing a threshold $t$ for a given input $x$. Note that the threshold cannot be defined as a single global value but should instead depend strongly on the input $x$ and $\theta_x(\cdot)$ since scores are on different scales. We also have the constraint that computing a threshold function must be fast enough such that sequentially computing scores and thresholds for multiple models in the cascade does not adversely effect the efficiency of the whole procedure. One might choose a quantile function to consistently eliminate a desired proportion of the max marginals for each example. However, quantile functions are discontinuous in the score function, and we instead approximate a quantile threshold.
with a more efficient mean-max threshold function, defined as a convex combination of the mean of the max marginals and the maximum score $\theta^* = \max_y \theta_y(y)$,

$$ t_x(\alpha) = \alpha \theta^*_x + (1 - \alpha) \frac{1}{|\mathcal{V}|} \sum_{c \in \mathcal{C}, y_c \in \mathcal{V}_c} \theta^*_x(y_c). \quad (6) $$

Choosing a mean-max threshold is therefore choosing $\alpha \in [0, 1)$. Note that $t_x(\alpha)$ is a convex function of $\theta_x(\cdot)$ (in fact, piece-wise linear), which combined with Lemma 1 will be important for learning the filtering models and analyzing their generalization. In our experiments, we found that the distribution of max marginals was well centered around the mean, so that choosing $\alpha \approx 0$ resulted in $\approx 50\%$ of max marginals being eliminated on average. As $\alpha$ approaches 1, the number of max marginals eliminated rapidly approaches 100%. We used cross-validation to determine the optimal $\alpha$ in our experiments (section 6).

4 Learning the cascade

We now turn to the problem of finding the best parameters $\mathbf{w}$ and the corresponding best tuning of the threshold $\alpha$ for each level of the cascade. When learning a cascade, we have two competing objectives that we must trade off:

- **Accuracy**: Minimize the number of errors incurred by each level of the cascade to ensure an accurate inference process in subsequent models.
- **Efficiency**: Maximize the number of filtered max marginals at each level in the cascade to ensure an efficient inference process in subsequent models.

We quantify this trade-off by defining two loss functions. We define the **filtering loss** $\mathcal{L}_f$ to be a 0-1 loss indicating a mistakenly eliminated correct assignment. The **efficiency loss** $\mathcal{L}_e$ is the proportion of unfiltered clique assignments.

**Definition 1** (Filtering loss). Let $\theta_x$ be the scoring function in the current level of the cascade. A filtering error occurs when a max-marginal of a clique assignment of the correct output $y$ is pruned. We define filtering loss as $\mathcal{L}_f(y, \theta_x) = 1[\theta_x(y) \leq t_x(\alpha)]$.

**Definition 2** (Efficiency loss). Let $\theta_x$ be defined as above. The efficiency loss is the proportion of unpruned clique assignments $\mathcal{L}_e(y, \theta_x) = \frac{1}{|\mathcal{V}|} \sum_{c \in \mathcal{C}, y_c \in \mathcal{V}_c} 1[\theta^*_x(y_c) \geq t_x(\alpha)]$.

Note that we can trivially minimize either of these at the expense of maximizing the other. If we set $(\mathbf{w}, \alpha)$ to achieve a minimal threshold such that no assignments are ever filtered, then $\mathcal{L}_f = 0$ and $\mathcal{L}_e = 1$. Alternatively, if we choose a threshold to filter every assignment, then $\mathcal{L}_f = 1$ while $\mathcal{L}_e = 0$. To learn a cascade of practical value, we can minimize one loss while constraining the other below a threshold $\epsilon$. Since the ultimate goal of the cascade is accurate classification, we focus on the problem of minimizing efficiency loss while constraining the filtering loss to be below a desired tolerance $\epsilon$.

We can express the cascade learning objective as a joint optimization over $\mathbf{w}$ and $\alpha$:

$$ \min_{\mathbf{w}, \alpha} \mathbb{E} [\mathcal{L}_e(Y, \theta_X)] \text{ s.t. } \mathbb{E} [\mathcal{L}_f(Y, \theta_X)] \leq \epsilon, \quad (7) $$

We solve this problem with a two-step procedure. First, we define a convex upper-bound on the filter error $\mathcal{L}_f$, making the problem of minimizing $\mathcal{L}_f$ convex in $\mathbf{w}$ (given $\alpha$). We learn $\mathbf{w}$ to minimize filter error for several settings of $\alpha$ (thus controlling filtering efficiency). Second, given $\mathbf{w}$, we optimize the objective (7) over $\alpha$ directly, using estimates of $\mathcal{L}_f$ and $\mathcal{L}_e$ computed on a held-out development set. In section 5 we present a theorem bounding the deviation of our estimates of the efficiency and filtering loss from the expectation of these losses.

To learn one level of the structured cascade model $\mathbf{w}$ for a fixed $\alpha$, we pose the following convex margin optimization problem:

$$ \text{SC} : \inf_{\mathbf{w}} \frac{\lambda}{2} ||\mathbf{w}||^2 + \frac{1}{n} \sum_i H(\mathbf{w}; (x^i, y^i)), \quad (8) $$

where $H$ is a convex upper bound on the filter loss $\mathcal{L}_f$,

$$ H(\mathbf{w}; (x^i, y^i)) = \max \{0, \ell + t_x(\alpha) - \mathbf{w}^T \mathbf{f}(x^i, y^i)\}. $$

The upper-bound $H$ is a hinge loss measuring the margin between the filter threshold $t_x(\alpha)$ and the score of the truth $\mathbf{w}^T \mathbf{f}(x^i, y^i)$: the loss is zero if the truth scores above the threshold by margin $\ell$ (in practice, the length $\ell$ can vary by example). We solve (8) using stochastic sub-gradient descent. Given a sample $(x, y)$, we apply the following update if $H(\mathbf{w}; (x, y))$ (i.e., the sub-gradient) is non-zero:

$$ \mathbf{w} \leftarrow (1 - \lambda) \mathbf{w} + \eta H(x, y) - \eta \alpha \mathbf{f}(x, y^*) - \eta (1 - \alpha) \frac{1}{|\mathcal{V}|} \sum_{c \in \mathcal{C}, y_c \in \mathcal{V}_c} \mathbf{f}(x, y^*_c). \quad (9) $$

Above, $\eta$ is a learning rate parameter, $y^* = \arg\max_y \theta_x(y)$ and $y^*_c = \arg\max_{y_c} \theta_x(y_c)$. The key distinguishing feature of the this update as compared to structured perceptron is that it sub-tracks features included in all max marginal assignments $y^*_c$.

Note that because (8) is $\lambda$-strongly convex, if we chose $\eta_t = 1/(\lambda t)$ and add a projection step to keep $\mathbf{w}$ in a
closed set, the update would correspond to the Pegasos update with convergence guarantees of $O(1/\epsilon)$ iterations for $\epsilon$-accurate solutions (Shalev-Shwartz, Singer, and Srebro, 2007).

5 Generalization Analysis

In this section, we give generalization bounds on the filtering and efficiency loss functions for a single level of a cascade. These bounds depend on Lipschitz dominating cost functions $\phi_f$ and $\phi_e$ that upper bound $L_f$ and $L_e$. To formulate these functions, we define the the scoring function $\theta_x : \mathcal{X} \mapsto \mathbb{R}^m$ where $m = \sum_{c \in C} |\mathcal{Y}_c|$ to be the set of scores for all possible clique assignments, $\theta_x = \{ w^\top f_c(x, y_c) \mid c \in \mathcal{C}, y_c \in \mathcal{Y}_c \}$. Thus, given $\theta_x$, the score $\theta_x(y)$ can be computed as the inner product $\langle \theta_x, y \rangle$, where we treat $y$ as a $m$-vector of indicators where $y_i = 1$ if the $i$th clique assignment appears in $y$. Finally, we define the auxiliary function $\phi$ to be the difference between the score of output $y$ and the threshold as $\phi(y, \theta_x) = \theta_x(y) - t_x(\alpha)$. We now state the main result of this section.

**Theorem 1.** Let $\theta_x$, $L_e$, $L_f$, and $\phi$ be defined as above. Let $\Theta$ be the class of all scoring functions $\theta_x$ with $||w||_2 \leq B$, the total number of cliques $\ell$, and $||f(x, y_c)||_2 \leq 1$ for all $x$ and $y_c$. Define the dominating cost functions $\phi_f(y, \theta_X) = r_\gamma(\phi(y, \theta_X))$ and $\phi_e(y, \theta_X) = \frac{1}{m} \sum_{c \in \mathcal{C}, y_c} r_\gamma(\phi(y^c, y_c), -\theta_X)$, where $r_\gamma(\cdot)$ is the ramp function with slope $\gamma$. Then for any integer $n$ and any $0 < \delta < 1$ with probability $1 - \delta$ over samples of size $n$, every $\theta_X \in \Theta$ and $\alpha \in [0, 1]$ satisfies:

$$
\mathbb{E} [L_f(Y, \theta_X)] \leq \hat{\mathbb{E}}[\phi_f(Y, \theta_X)] + O \left( \frac{mn^{3/2}B}{\gamma \sqrt{n}} \right) + \sqrt{\frac{8 \ln(2/\delta)}{n}},
$$

where $\hat{\mathbb{E}}$ is the empirical expectation with respect to training data. Furthermore, (10) holds with $L_f$ and $\phi_f$ replaced by $L_e$ and $\phi_e$.

This theorem relies on the general bound given in Bartlett and Mendelson (2002), the properties of Rademacher and Gaussian complexities (also in Bartlett and Mendelson (2002)), and the following lemma:

**Lemma 2.** $\phi_f(y, \cdot)$ and $\phi_e(y, \cdot)$ are Lipschitz (with respect to Euclidean distance on $\mathbb{R}^m$) with constant $\sqrt{\ell}/\gamma$.

A detailed proof of Theorem 1 and Lemma 2 is given in the appendix.

Theorem 1 provides theoretical justification for the definitions of the loss functions $L_e$ and $L_f$ and the structured cascade objective; if we observe a highly accurate and efficient filtering model $(w, \alpha)$ on a finite sample of training data, it is likely that the performance of the model on unseen test data will not be too much worse as $n$ gets large. Theorem 1 is the first theoretical guarantee on the generalization of accuracy and efficiency of a structured filtering model.

6 Experiments

**Handwriting Recognition.** We first evaluated the accuracy of the cascade using the handwriting recognition dataset from Taskar et al. (2003). This dataset consists of 6877 handwritten words, with average length of $\sim$8 characters, from 150 human subjects, from the data set collected by Kassell (1995). Each word was segmented into characters, each character was rasterized into an image of 16 by 8 binary pixels. The dataset is divided into 10 folds; we used 9 folds for training and a single withheld for testing (note that Taskar et al. (2003) used 9 folds for testing and 1 for training due to computational limitations, so our results are not directly comparable). Results are averaged across all 10 folds.

Our objective was to measure the improvement in predictive accuracy as higher order models were incorporated into the cascade. The final cascade consisted of four Markov models of increasing order. Pixels are used as features for the lowest order model, and 2, 3, and 4-grams of letters are the features for the three higher order models, respectively. The maximum filter loss threshold $\epsilon$ was set to 1%, 2%, and 4% (this avoids over-penalizing higher order models), and we trained structured cascades (SC) using $\alpha$’s from the candidate set $\{0, 0.25, 0.5\}$. To simplify training, we fixed $\eta = 1$, $\lambda = 0$, and used an early-stopping procedure to choose $w$ that achieved optimal tradeoff according to (7).

Results are summarized in Table 1. We found that the use of higher order models dramatically increased accuracy of the predictions, raising accuracy at the character above 90% and more than tripling the word-level accuracy. Furthermore, the cascade reduced the search space of the 4-gram model by 5 orders of magnitude, while incurring only 3.41% filtering error.

**Part-of-Speech Tagging.** We next evaluated our approach on several part of speech (POS) tagging tasks. Our objective was to rigorously compare the efficacy of our approach to alternative methods on a problem while reproducing well-established benchmark prediction accuracy. The goal of POS tagging is to assign a label to each word in a sentence. We used three different languages in our experiments: English, Portuguese and Bulgarian. For English, we used vol-
volumes 0-17 of the Wall Street Journal portion of the
Penn TreeBank (Marcus et al., 1993) for training, vol-
umes 18-20 for development, and volumes 21-24 for
testing. We used the Bulgarian BulTReeBank (Simov
et al., 2002) and the Bosque subset of the Portuguese
Floresta Sinta(c)tica (Afonso et al., 2002) for Bulgar-
ian and Portuguese datasets, respectively.

For these experiments, we focused on comparing the
efficiency of our approach to the efficiency of several
alternative approaches. We again used a set of Markov
models of increasing order; however, to increase filter-
ing efficiency, we computed max marginals over sub-
cliques representing \((d - 1)\) order state assignments
rather than \(d\)-order cliques representing transitions.
Thus, the bigram model filters unigrams and the tri-
gram model filters bigrams, etc. Although we ran the
cascades up to 5-gram models, peak accuracy was
reached with trigrams on the POS task.

We compared our SC method to two baselines: the
standard structured perceptron (SP) and a maximum
a posteriori CRF. All methods used the same stan-
dard set of binary features: namely, a feature each
(word,tag) pair \(1[ X_t = x_t, Y_t = y_t]\) and feature for
each \(d\)-gram in the model. For the baseline methods,
we trained to minimize classification error (for SP) or
log-loss (for CRF) and then chose \(\alpha \in [0, 1]\) to achieve
minimum \(\mathcal{L}_c\) subject to \(\mathcal{L}_f \leq \epsilon\) on the development set.
For the CRF, we computed sum-product marginals
\(P(y_c|x) = \sum_{y^{c'}:y_{c'}=y_c} P(y'|x)\), and used the threshold
\(t_x(\alpha) = \alpha\) to eliminate all \(y_c\) such that
\(P(y_c|x) \leq \alpha\). For all algorithms, we used grid search over several val-
ues of \(\eta\) and \(\lambda\), in addition to early stopping, to choose
the optimal \(w\) based on the development set. For SC
training, we considered initial \(\alpha\)'s from the candidate set \(\{0, 0.2, 0.4, 0.6, 0.8\}\).

We first evaluated our approach on the WSJ dataset.
To ensure accuracy of the final predictor, we set a strict
threshold of \(\epsilon = 0.01\%\). All methods were trained us-
ing a structured perceptron for the final classifier. The
results are summarized in Table 2. SC was compared
to CRF, an unfiltered SP model (Full), and a heuris-
tic baseline in which only POS tags associated with a
given word in the training set were searched during in-
ference (Tags). SC was two orders of magnitude faster
than the full model in practice, with the search space
reduced to only \(\approx 4\) states per position in inference
(roughly the complexity of a greedy approximate beam
search with 4 beams.) SC also outperformed CRF by
a factor of 2.6 and the heuristic by a factor of 6.8.
Note that because of the trade-off between accuracy
and efficiency, CRF suffered less filter loss relative to
SC due to pruning less aggressively, although neither
method suffered enough filtering loss to affect the ac-
curacy of the final classifier. Finally, training the full

<table>
<thead>
<tr>
<th>Model Order:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy, Char. (%)</td>
<td>77.44</td>
<td>85.69</td>
<td>87.95</td>
<td>92.25</td>
</tr>
<tr>
<td>Accuracy, Word (%)</td>
<td>26.65</td>
<td>49.44</td>
<td>73.83</td>
<td>84.46</td>
</tr>
<tr>
<td>Filter Loss (%)</td>
<td>0.56</td>
<td>0.99</td>
<td>3.41</td>
<td>—</td>
</tr>
<tr>
<td>Avg. Num n-grams</td>
<td>26.0</td>
<td>123.8</td>
<td>88.6</td>
<td>5.4</td>
</tr>
</tbody>
</table>

Table 1: Summary of handwriting recognition results. For
each level of the cascade, we computed prediction accuracy
(at character and word levels) using a standard voting per-
ceptron algorithm as well as the filtering loss and average
number of unfiltered n-grams per position for the SC on
the test set.

<table>
<thead>
<tr>
<th>Model:</th>
<th>Full</th>
<th>SC</th>
<th>CRF</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>96.83</td>
<td>96.82</td>
<td>96.84</td>
<td>—</td>
</tr>
<tr>
<td>Filter loss (%)</td>
<td>0.024</td>
<td>0.012</td>
<td>0.118</td>
<td></td>
</tr>
<tr>
<td>Test Time (ms)</td>
<td>173.28</td>
<td>1.56</td>
<td>4.16</td>
<td>10.6</td>
</tr>
<tr>
<td>Avg. Num States</td>
<td>1935.7</td>
<td>3.93</td>
<td>11.845</td>
<td>95.39</td>
</tr>
</tbody>
</table>

Table 2: Summary of WSJ Results. Accuracy is the accu-
tracy of the final trigram model. Filter loss is the number
of incorrectly pruned bigrams at the end of the cascade.
The last row is the average number of states considered at
each position in the test set.

We next investigated the efficiency vs. filtering accu-
tracy trade-off of SC compared to the SP and CRF
baselines on all three languages. For each of the
three languages, we generated 10 different training
sets from 40% of the full training datasets. Random-
ization was taken with the same 10 seeds for each
dataset/algorithm pair. For all methods, we trained
bigram models under two conditions: first, as the ini-
tial step of a cascade (e.g., no prior filtering), and sec-
ownd, as the second step of a cascade after initial filter-
ing by the SC algorithm with \(\epsilon = 0.05\%), and analyzed
the resulting trade-off between efficiency and accuracy.

The results are presented in Figure 2. The figures were
generated by computing for each \(\epsilon\) along the x-axis the
corresponding test efficiency \(\mathcal{L}_c\) when the constraint on
filter loss (\(\mathcal{L}_f \leq \epsilon\)) is enforced using development set
data. Points for which the constraint could not be en-
forced (even with \(\alpha = 0\)) are not shown. SC handily
beats the competitors in both the filtered and unfil-
tered case. Note that in the unfiltered condition, the
CRF is unable to achieve significant pruning efficiency
for any \(\epsilon\), while the SP cannot achieve filtering accu-
tracy for small \(\epsilon\). However, because SC and SP become
equivalent as \(\alpha\) approaches 1, we observe that the perfor-
mance of SC and SP converge as \(\epsilon\) increases.
imate inference is an interesting open question.\footnote{Theorem 1 and 2} \[ \text{where we treat} \ y \ \text{as the inner product} \ \langle \cdot, \cdot \rangle. \]

Furthermore, there are absolute constants \(c\) and \(C\) such that for every class \(F\) and every integer \(n\),
\[
e^{cR_n(F)} \leq G_n(F) \leq C \ln n R_n(F). \tag{12} \]

Let \( A = \mathbb{R}^m \) and \( F : \mathcal{X} \rightarrow A \) be a class of functions that is the direct sum of real-valued classes \( F_1, \ldots, F_m \). Then, for every integer \( n \) and every sample \((X_1, \ldots, X_n, Y_n)\),
\[
\hat{G}_n(\hat{\phi} \circ F) \leq 2L \frac{1}{n} \sum_{i=1}^m \hat{G}_n(F_i). \tag{13} \]

Let \( F = \{ x \mapsto w^T f(x, \cdot) \mid \|w\|_2 \leq B, \|f(x, \cdot)\|_2 \leq 1 \} \).

Then,
\[
\hat{G}_n(F) \leq \frac{2B}{\sqrt{n}}. \tag{14} \]

Proof of Theorem 1. Let \( A = \mathbb{R}^m \) and \( F = \Theta_X \). By (12), we have that \( R_n(\hat{\phi}_f \circ F) = O(G_n(\hat{\phi}_f \circ F)) \). Since \( \hat{\phi}_f \) passes through the origin, then by (13), \( G_n(\hat{\phi}_f \circ F) = O(2L(\hat{\phi}_f) \sum_i G_n(f_i)) = O(mL(\hat{\phi}_f)G_n(\mathcal{H})) \). With Lemma 2 and \( L(\hat{\phi}_f) = L(\phi_f) \), we then have that \( R_n(\hat{\phi}_f \circ F) = O(mn^{-1} \sqrt{G_n(F_i)}) \), where \( F_i \) is a linear function scoring the \( i \)th clique assignment. Thus the results follows from (14) and (11). Finally, because \( L(\phi_f) = L(\phi_c) \), the same results applies to \( \mathcal{L}_c \) as well. \( \square \)

A.1 Proof of Lemma 2

To prove Lemma 2, we first bound the slope of the difference \( \phi(y, \theta_x) \). We observe that we can consider the scores of output \( y \) as the inner product \( \langle y, \theta_x \rangle \), where we treat \( y \) as a \( m \)-vector of indicators and \( y_i = 1 \) if the \( i \)th clique assignment appears in \( y \).
Lemma 3. Let \( f(\theta_z) = \langle y, \theta_z \rangle - \max_{\gamma'} \langle y', \theta_z \rangle \). Then \( f(u) - f(v) \leq \sqrt{2L||u - v||_2} \).

Proof. Let \( y_u = \arg \max_{\gamma'} \langle y', u \rangle \) and \( y_v = \arg \max_{\gamma'} \langle y', v \rangle \). Then we have,

\[
\begin{align*}
f(u) - f(v) &= \langle y, u \rangle - \langle y, v \rangle + \langle y, v \rangle - \langle y, v \rangle \\
&= \langle y_u - y, v \rangle + \langle y - u, v \rangle + \langle y, u - v \rangle - \langle y, u \rangle \\
&\leq \langle y_u - y, v - u \rangle \\
&\leq \sqrt{2L||u - v||_2}.
\end{align*}
\]

The last two steps follow from the fact that \( y_u \) maximizes \( \langle y_u, u \rangle \) (so \( \langle u, y_v - y_u \rangle \) is negative), application of Cauchy-Schwarz, and from the fact that there are at most \( \ell \) cliques appear, each of which can contribute a single non-zero entry in \( y \) or \( y_v \).

Lemma 4. Let \( f'(\theta_z) = \langle y, \theta_z \rangle - \frac{1}{m} \sum_{c \in C,y_c, \gamma_c} \max_{\gamma', y_c'} \langle y, \gamma', \theta_z \rangle \). Then \( f(u) - f(v) \leq \sqrt{2L||u - v||_2} \).

Proof. Let \( y_{ui} = \arg \max_{\gamma', y_c'} \langle y', u \rangle \) for the \( i \)'th clique assignment \( y_i \), and \( y_{vi} \) the same for \( v \). Then we have,

\[
\begin{align*}
f'(u) - f'(v) &= \frac{1}{m} \sum_{i=1}^m \langle y, u \rangle - \langle y_{ui}, u \rangle + \langle y_{vi}, v \rangle - \langle y, v \rangle \\
&\leq \frac{1}{m} \sum_{i=1}^m \langle y_{vi} - y, v - u \rangle \\
&\leq \frac{1}{m} \sum_{i=1}^m \sqrt{2L||u - v||_2} \leq \sqrt{2L||u - v||_2}.
\end{align*}
\]

Here we have condensed the same argument used to prove the previous lemma.

Lemma 5. Let \( g(\theta_z) = \langle y, \theta_z \rangle - t_\alpha(\alpha) \). Then \( g(u) - g(v) \leq \sqrt{2L||u - v||_2} \).

Proof. Plugging in the definition of \( t_\alpha(\alpha) \), we see that \( g(\theta_z) = \alpha f(\theta_z) + (1 - \alpha) f'(\theta_z) \). Therefore from the previous two lemmas we have that \( g(u) - g(v) = \alpha(f(u) - f(v)) + (1 - \alpha)(f'(u) - f'(v)) \leq \sqrt{2L||u - v||_2} \).

From Lemma 5, we see that \( \phi(y, \cdot) = g(\cdot) \) is Lipschitz with constant \( \sqrt{2L} \). We can now show that \( \phi_f \) and \( \phi_e \) are Lipschitz continuous with constant \( \sqrt{2L}/\gamma \). Let \( L(\cdot) \) denote the Lipschitz constant. Then \( L(\phi_f) = L(r_{\gamma}) \cdot L(\phi(y, \cdot)) \leq \sqrt{2L}/\gamma \).

To show \( L(\phi_e) \) requires more bookkeeping because we must bound \( \phi(y^*(y_c), \theta_z) \). We can prove equivalent lemmas to lemmas 3, 4 and 5 where we substitute \( \langle y, \theta_z \rangle \) with \( \max_{y', y_c' = y_c} \langle y', \theta_z \rangle \), and thus show that \( L(\phi^*(y_c, \cdot)) \leq \sqrt{2L} \). Therefore, \( L(\phi_e) = \frac{1}{m} \sum_{c \in C,y_c, r_{\gamma}} L(r_{\gamma}) \cdot L(\phi^*(y_c, \cdot)) \leq \sqrt{2L}/\gamma \), as desired.

Acknowledgements

This work was partially supported by NSF Grant 0803256.

References


