ENABLING MORE ACCURATE AND EFFICIENT STRUCTURED PREDICTION

David Weiss

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2013

Supervisor of Dissertation

_____

Ben Taskar

Associate Professor, Computer and Information Science

Graduate Group Chairperson

_____

Val Tannen, Professor, Computer and Information Science

Dissertation Committee:

Kostas Daniilidis, Professor, Computer and Information Science

Daniel Lee, Professor, Computer and Information Science

Camillo J. Taylor, Associate Professor, Computer and Information Science

Pedro Felzenszwalb, Associate Professor, Computer Science, Brown University

ENABLING MORE ACCURATE AND EFFICIENT STRUCTURED PREDICTION

For Ben: My advisor, mentor, and friend; I miss you.

Acknowledgements

I would like to first thank and acknowledge my advisor, Ben Taskar. This thesis would simply not exist without his unwavering expert guidance, encouragement, and support. His boundless energy and enthusiasm for our work pushed me to achieve what I thought was impossible. I could not have asked for a better advisor.

I am grateful to Kostas Daniliidis, the chair of my committee, for his consistent advice and confidence in me. I would also like to thank the rest of my committee–Dan Lee, CJ Taylor, and Pedro Felzenszwalb–for helping me forge my years of effort into a single coherent document. I am also indebted to my original co-advisor Michael Kearns, who shaped the start of my time in graduate school.

During my time at Penn, I was fortunate to be surrounded by an amazing community of colleagues and friends. I count myself lucky to have had the opportunity to work so closely with Benjamin Sapp. Without Andrew King, I would have missed out on an amazing paper title and acronym. Many thanks to Alex, Alex, Cody, Chris, Christine, Emily, Joao, Jenny, Katie, Katerina, Kuzman, Matthieu, Ryan, Steve, and Umar. I would not have made it through this process without you.

Finally, I am only the person I am today because of my family: my parents, Robert and Margaret, and my brothers, Michael and Jonny. Their love kept me going through the difficult times where I needed it most.

ABSTRACT

ENABLING MORE ACCURATE AND EFFICIENT STRUCTURED PREDICTION

David Weiss

Ben Taskar

Machine learning practitioners often face a fundamental trade-off between expressiveness and computation time: on average, more accurate, expressive models tend to be more computationally intensive both at training and test time. While this trade-off is always applicable, it is acutely present in the setting of *structured prediction*, where the joint prediction of multiple output variables often creates two primary, inter-related bottlenecks: inference and feature computation time.

In this thesis, we address this trade-off at test-time by presenting frameworks that enable more accurate and efficient structured prediction by addressing each of the bottlenecks specifically. First, we develop a framework based on a *cascade* of models, where the goal is to control test-time complexity even as features are added that increase inference time (even exponentially). We call this framework Structured Prediction Cascades (SPC); we develop SPC in the context of exact inference and then extend the framework to handle the approximate case. Next, we develop a framework for the setting where the feature computation is explicitly the bottleneck, in which we learn to selectively evaluate features within an instance of the mode. This second framework is referred to as Dynamic Structured Model Selection (DMS), and is once again developed for a simpler, restricted model before being extended to handle a much more complex setting. For both cases, we evaluate our methods on several benchmark datasets, and we find that it is possible to dramatically improve the efficiency and accuracy of structured prediction.

Contents

List of Tables

List of Illustrations

Chapter 1

Introduction

This thesis is motivated by the inherent trade-off in applied machine learning of *expressiveness* vs. *computation*. To make this concrete, we consider supervised discrete prediction problems where the goals is to predict a discrete output variable Y given some input X. Specifically, we assume we have a linear hypothesis of the form:

$$h(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}) \tag{1.1}$$

where $\mathcal{Y}$ is the domain of Y (the output space), $\mathbf{w}$ is a parameter weight vector and $\mathbf{f}$ is a feature generating function. This linear prediction framework encompasses most standard supervised learning settings and algorithms, where the goal is to learn a parameter vector $\mathbf{w}$ in order to minimize some expected error for a prediction task in a given domain.

In this thesis, we are instead primarily concerned with the question: what are the computational requirements of evaluating $h(\mathbf{x})$? As an example, consider an instance of the standard multi-class classification problem from computer vision is image classification, where $\mathbf{x}$ is an input image, $\mathcal{Y} = \{1, \ldots, K\}$ (where K is the number of classes of images), and $\mathbf{f}$ computes a vector of image features such as SIFT. The computation required for $h(\mathbf{x})$ is two steps: first, we compute the image features (SIFT), then, we score the feature vector for each class in $\mathcal{Y}$ and find the argmax. In general, we see that there are two potential bottlenecks for $h(\mathbf{x})$: (1) computing the features $\mathbf{f}(\mathbf{x}, \mathbf{y})$, and (2) computing the argmax in (1.1) given the scored feature vectors. If computing features is the computational bottle-

neck, then the expressiveness vs. computation trade-off is intuitively very simple: adding more features to the model increases the expressiveness of the model at additional computational cost.

Given a set of models of varying expressiveness and computational cost, the standard static *model selection* problem is to choose a model that which is closest to a given target in the space of possible trade-offs. For example, we might desire the most accurate face detector that runs in less than a fraction of a second on a typical server, or we might desire the most efficient face detector that still achieves 99.9% recall on a standard benchmark. To make the example still more concrete, suppose we have a hierarchy of feature generating functions $\mathbf{f}_1 \subseteq \mathbf{f}_2 \ldots \mathbf{f}_D$, each requiring more time to compute than the previous. One simple model selection technique would be to learn parameter vectors $\mathbf{w}_1, \ldots, \mathbf{w}_D$ separately for each model using standard machine learning techniques, evaluate each one on a development data set, and choose the one that best satisfies our problem constraints using estimates of runtime and accuracy on the development set.

In contrast, in this thesis we study test-time model selection methods that, given a model hierarchy, can achieve a much more accurate hybrid model with the same or better *expected* runtime as the static model selection scheme just described. For example, the highly influential classifier cascade of Viola and Jones (2002) provided a significant leap in state-of-the-art performance for face detection. At a high level, the classifier cascade approach to model selection is use a hybrid model that utilizes the cheapest model $\mathbf{f}_j$ in $\mathbf{f}_1, \ldots, \mathbf{f}_D$ that has margin $\mathbf{w}_j^\top \mathbf{f}_j(\mathbf{x}, \mathbf{y})$ above some threshold; in this way, "easy" examples will be evaluated quickly while "hard" examples will be assigned more computation time; on average, the accuracy can reach that of the most expensive model, while the computation time remains much lower. The contributions of this thesis can best be understood within the space of multi-stage, hybrid model selection techniques.

In particular, we present methods for improving the expressiveness-computation trade-off over static training-time model selection for the setting of *structured prediction*, where *which* features are computed is often more important to computation time than *how many*

features are computed. Put simply, structured prediction is the joint prediction of multiple discretely-valued output variables, and this problem arises in many application areas of machine learning. For example, one problem studied in this thesis is handwriting recognition, or the joint prediction of the English letters associated with a sequence of handwritten characters. Note that this problem still falls in the general framework for $h(\mathbf{x})$ from (1.1); for a sequence of length ℓ and an alphabet of size K, we simply have that $\mathcal{Y}$ is the product of ℓ discrete spaces $\{1, \ldots, K\}$. The key empirical finding is that structured prediction–making predictions for each character simultaneously–provides far more accurate results on this task than the alternative of making separate predictions for each character independently. The reason that the data is *not* generated independently for each character: because the writer is writing English words, the assignment of each letter in the word carries information on nearby letters.

Structured prediction is a natural framework both for learning high-order statistics of the data, and for incorporating hard constraints on the output space. By explicitly representing such interactions as features, structured prediction allows one to learn the higher-order statistics of the training data, and thereby ensure that predictions properly take into account interactions between output variables at test time. Furthermore, in certain settings, there exist hard constraints on the interactions between output variables: for instance, in natural language processing, producing a dependency parse of a sentence requires predicting a parent-child relationship for each word such that the resulting directed graph is a tree.

As might be expected, the additional expressiveness of structured prediction over independent predictions comes at an increased computational cost: because $\mathcal{Y}$ is exponentially large ($|\mathcal{Y}| = K^\ell$, as defined above), naively computing the argmax becomes practically infeasible for any reasonably sized problem, and therefore is often the primary bottleneck, rather than feature computation. Fortunately, better inference algorithms do exist. However, the run-time complexity of these algorithms is determined by the particular structure of the features; for instance, if we model pairwise interactions between neighboring characters, complexity of inference in the handwriting recognition problem is $O(\ell K^2)$, while

if we add triplets of neighboring characters, the complexity is $O(\ell K^3)$. Thus, for structured prediction, the expressiveness-computation trade-off can be summarized as follows: introducing additional features increases expressiveness at the cost of increased feature computation time, but may additionally increase inference time, possibly to the point of making the problem infeasible.

1.1 Thesis Overview

The primary contribution of this thesis is to provide two different and complementary methods for obtaining better expressiveness-computation trade-offs in the setting of structured prediction. The key to each of our approaches is to *learn* to control the complexity of inference or feature computation at test time, on a per-example basis. As we will see, the first methods we discuss are meant to address the inference bottleneck; the next methods are meant to address the feature computation time bottleneck. Specifically, we outline the organization of this thesis as follows:

- **Chapter 2 - Related work.** In this chapter, we place the thesis in context by reviewing the relevant literature that either precedes or competes with our approaches; much of the ideas in this work stem from related work in multi-class classification, while we compete with alternative methods in the structured prediction community.

- **Chapter 3 - Structured Prediction.** We next provide a brief overview of the key technical details of structured prediction, to provide a common foundation with which we can present and analyze the frameworks described in this thesis. Note that a complete tutorial is outside the scope of this thesis; see e.g. Nowozin and Lampert (2011) for a more complete tutorial. The purpose of this chapter is to provide enough background to understand the bottlenecks of inference and feature computation time that we address in this thesis.

- **Chapter 4 - Structured Prediction Cascades (SPC).** In the next chapter, we develop the first of the frameworks considered in this thesis: Structured Prediction

Cascades (SPC). The goal of SPC is to progressively filter or *prune* the output space such that sparse, exact inference remains feasible even when very complex features are added to the model. This chapter is based on the preliminary work (Weiss and Taskar, 2010).

- **Chapter 5 - Ensemble-SPC.** The major limitation of SPC is that it requires exact inference to be feasible in every model in the cascade; thus, in the next chapter, we extend SPC to the setting where exact inference is intractable by using *ensembles* of models. This chapter is based on preliminary work (Weiss et al., 2010).

- **Chapter 6 - Dynamic structured model selection (DMS).** We next turn to the problem of feature computation time. We propose a framework for selectively allocating resources across a batch of test examples by running a cascade of arbitrarily constructed structured models. This is based on preliminary work (Weiss and Taskar, 2013).

- **Chapter 7 - DMS-π.** In this chapter, we revisit the feature extraction problem for structured models and provide a more fine-grained formulation using principles from reinforcement learning. This allows us to selectively compute features within specific examples and learn a non-myopic feature extraction policy, which leads to large gains in efficiency over the original DMS approach. This is based on preliminary work under review at the NIPS2013 conference.

- **Chapter 8 - Future work.** Finally, we propose unifying the SPC and DMS-π frameworks into a single framework capable of trading off both the inference and feature computation bottlenecks.

Chapter 2

Related Work

The basic expressiveness-computation trade-off has been studied in the machine learning literature in various settings and for a variety of reasons. In this section, we review the relevant literature and place this thesis in a broader context. As discussed previously, there is a significant amount of related work studying the expressiveness-computation tradeoff in multi-class classification. While many of the principles or high level ideas of the same–e.g., applying a cascade of models or using reinforcement learning to model feature extraction–applying these ideas to the structured case is not trivial, and many new insights are to be gained in the process. Thus, we first review the multi-class literature and then delve into related or competing approaches for structured prediction.

However, there are several themes that consistently arise in both the multi-class and structured setting. Generally speaking, we can categorize the related literature along several axes:

- **Training time vs. test time.** Much prior work has focused on model selection during training–efficiently finding the most accurate fixed model with bounded complexity. In contrast, the contributions of this thesis allow for adaptive complexity chosen dynamically at test-time.

- **Single- vs. multi-stage.** One point of emphasis in prior work is limiting the computational complexity of a single model (1.1). This can be achieved either statically at

training or dynamically at test-time, for example by selectively computing a subset of the possible features to limit the cost. An alternative is to propose a multi-stage approach that makes a series of increasingly expensive decisions. In this case the depth of processing of a certain example determines the cost of evaluation. The work of this thesis spans both of these cases: in DMS-π, we selectively compute features in a single structured model, while in SPC and DMS, we use a cascade of increasingly expensive models.

- **Myopic vs. non-myopic.** Both the work in this thesis and related approaches allocate resources at test-time in an piecemeal fashion, repeatedly deciding whether or not more computation is necessary for a given example. One important property of such approaches is whether or not computation is allocated *myopically*, i.e. without a belief of how early steps in the computation will affect later outcomes. A typical approach to learning non-myopic strategies is to borrow techniques from the field of reinforcement learning, and that is the approach taken in this thesis for DMS-π in Chapter 7.

- **Implicit vs. explicit meta-level reasoning.** One distinction of the work in this thesis that has received far less prior attention is the idea of explicit *meta-level* reasoning: i.e. learning a distinct model with seperate meta-level features that can examine the current state of the predictive system and reason about how to proceed. For example, while our SPC framework is based around eliminating low-scoring outputs based on adaptive thresholds, the DMS/DMS-π framework explicitly learns a separate model that evaluates the output of the structured predictor and computes values that guide further computation. Most prior work, on the other hand, falls into the implicit category.

Finally, for the purely empirical contributions of this thesis–state-of-the-art results in several structured predicton tasks–we will review the related work in the presentation of the experiments.

2.1 Controlling computation of multi-class classification

The general ideas inspiring the particular methods proposed in this thesis are not new; in many cases, our intuition comes from prior work studying the computational trade-offs in the setting of multi-class classification. In many ways, one of the goals of this thesis is to translate ideas and advances from the multi-class case into the structured prediction setting. In what follows, we provide an overview of the themes from the flat classification literature that are related to or (in some cases) directly inspired the work of this thesis.

2.1.1 Computational trade-offs during model selection

Model selection has long been studied within the framework of empirical risk minimization (Bartlett et al., 2002; Vapnik and Chervonenkis, 1974), in which the primary criteria is the trade-off between *approximation error* (the bias of the model) and *estimation error* (overfitting, or the variance of the parameter estimates). For this case, model selection again consists of finding the most accurate model that falls within a complexity constraint. However, unlike the model selection considered in this thesis, the complexity measure is not defined with regards to computation. Rather, it is defined with respect to the expressive power of the model. The general intuition in this case is that as more data is obtained, the parameter estimates for more more expressive models become useful.

Several recent works have considered incorporating the *optimization error* into the model selection criteria, which is the additional error due to optimizing an objective within a fixed precision during the estimation procedure. These works attempt to solve a related but different problem to that which we focus on in this thesis: namely, how do we best allocate resources *during training* to achieve better model selection? Notably, Bottou and Bousquet (2008) show analytically that for large-scale classification, stochastic/online methods yield smaller asymptoptic error rates than batch-methods when the model and training time are fixed. Furthermore, Shalev-Shwartz and Srebro (2008) show empirically that training time to reach a given overall error rate for stochastic sub-gradient methods decreases as more data is added. These results suggest that when plenty of training data

is available and computation time at training is limited, stochastic optimization methods should be used. Since our setting assumes both properties, we employ such such stochastic learning techniques in this thesis.

Both of these works assume a single fixed class of models, while in more recent theoretical work, Agarwal et al. (2011) propose methods for model selection under training time computational constraints when given a hierarchy of increasingly complex models. This problem is more similar to that addressed in this thesis, but, as above, Agarwal et al. (2011) study the problem of allocating computation time during training, with no constraints on fixed test-time computational costs.

2.1.2 Predicting with fixed test-time budgets

More directly related to the work in this thesis is prior work on explicitly controlling the fixed test-time cost of the model through regularization during training. These works attempt to answer the question: how do we efficiently learn an accurate model that has hard constraints on a fixed test-time cost? A popular approach is to express test-time complexity as convex regularization term that penalizes expensive models during training, for example learning *sparse* (Langford et al., 2009) or *low rank* (Harchaoui et al., 2012) parameter vectors. Sparse models use only subset of the features, and therefore certain features may not need to be computed; low rank models use an intermediate representation that is shared among different classes, speeding up the argmax problem for multi-class classification when K is very large. However, while these works also seek to control test-time costs, these approaches yield a single model of fixed computational cost for every test example. In contrast, the methods presented in this thesis learn models for expected run-time that compute different features for each different test-example.

Furthermore, although we only consider linear methods in this thesis, there has also been significant research into constructing non-linear classifiers (e.g. boosted decision trees) with controlled test-time evaluation cost. For example, Sheng and Ling (2006) learn decision trees for the medical domain to optimize a trade-off between accuracy, feature

evaluation cost, and a delay cost corresponding to the time taken to perform medical tests. More recently, Grubb and Bagnell (2012) propose a boosting algorithm designed to maximize accuracy for any fixed test-time cost, while Xu et al. (2012) propose a boosting-type algorithm for learning ensembles of decision trees that explicitly trade-off computing new features vs. re-using already computed features in later stages of boosting.

Finally, another related area for controlling test-time complexity at training is in the domain of kernelized methods: kernelized models' computational cost at test-time is determined primarily by the number of *support vectors* retained during training to implicitly parameterize the feature vector of the model. Learning kernelized models on a budget has been well studied in the on-line setting (Crammer et al., 2003; Dekel et al., 2008), where an active set of support vectors is maintained and updated as new examples are observed. However, we do not study kernelized models in this thesis, and therefore the notion of budget as number of support vectors is qualitatively different from the budget we define in terms of the cost of computing features.

2.1.3 Resource allocation for batch testing

In practice, we often care about the cost of evaluating a predictor on more than a single example in isolation. Given a batch of test examples, we can define the problem of allocating limited computational resources in order to maximize accuracy on the batch as a whole. For example, Kanani and Melville (2008) study the problem of choosing to acquire features at test-time for different instances based on estimating the reduction in uncertainty over class labels of adding features to each instance, while Grubb and Bagnell (2012) use the margin of a ensemble classifier to determine which examples should have more weak classifiers incorporated into the ensemble. We also study the problem of batch-time resource allocation when we present our DMS and DMS-π frameworks.

2.1.4 Classifier cascades

A significant source of related work to the approaches proposed here is the study of *classifier cascades*, in which a pre-determined series of classification models are sequentially applied to a test example. For binary classification problems with a highly skewed class distribution with very few true positives, these works typically obtain efficiency via an "early-exit" strategy: at test-time, simpler models may reject an example as negative without needing to evaluate the more complex models farther along the cascade. This approach has long been highly successful for object detection, using boosting methods to train the cascade of classifiers (Viola and Jones, 2002). Significant effort has been put into frameworks for jointly learning the classifiers of the cascade (Lefakis and Fleuret, 2010) or constructing cascades automatically (Saberian and Vasconcelos, 2010). Feature computation cost was not incorporated specifically into the learning procedure until more recently. For example, Raykar et al. (2010) and Chen et al. (2012) propose training procedures for jointly training a cascade that explicitly represent a trade-off between accuracy and a measure of runtime complexity. These works inspired our SPC framework, which translates the idea of early-exit cascades into the structured setting.

2.1.5 More general multi-stage decision systems

There have been a number of more general formulations of a multi-stage classifier that expand on the idea of classifier cascades. For multi-class problems, the "early-exit" strategy for classifying negative examples does not apply directly, since there is no negative class. Instead, a more natural formulation is introduce an additional "reject" class to form a $(K+1)$-way classification problem; if a classifier predicts "reject," the example is passed to the next classifier (Trapeznikov and Saligrama, 2013). In this way, predicting anything besides "reject" can be seen as an early exit strategy.

In order to learn such multi-stage classifiers, Trapeznikov et al. (2013) propose a multi-stage empirical risk minimization framework, where the risk trades-off expected accuracy and cost of running the system. In practice, their algorithm consists of a series of K-class

classifiers and two additional components: one to estimate the confidence of the classifier and another to non-myopically determine a rejection threshold based on the confidence. If the classifier is not confident enough, then the "reject" action is taken for a given stage. The whole system is learned in an iterative stage-wise fashion. A similar approach is taken by Busa-Fekete et al. (2012), who assume that the series of classifiers is fixed a priori–learned via AdaBoost–and instead introduce an additional "skip" action that allows the system to short-circuit the decision process. Unlike Trapeznikov et al. (2013), Busa-Fekete et al. (2012) use standard reinforcement learning techniques to learn a policy to determine the actions taken at prediction time. While the models used in these frameworks differs greatly from the structured setting we study, we use similar principles from reinforcement learning to non-myopically learn a policy for feature extraction in our DMS-π framework.

Another multi-stage framework addressing a somewhat different problem than those just mentioned is closely related to SPC: This is the *label tree* framework for large multi-class tasks, first proposed by Bengio et al. (2010). A label tree is a decision tree where each decision node eliminates a fraction of the label space, such that at the leave nodes only comparison between a small number of classes is needed. At a high level, this idea is similar to that of SPC proposed in this thesis, except that instead of scoring, filtering, and maintaining a list of multiple overlapping partitions of the output space, the tree maintains and repeatedly filters only a single partition. However, Deng et al. (2011) show how the label tree can be explicitly optimized for accuracy and efficiency in a manner similar to the objectives posed for SPC. More recently, Liu et al. (2013) show how to learn probabilistic classifiers at each decision node to dramatically improve the performance of the tree; however, this is significantly different from the non-probabilistic approach taken throughout this thesis.

2.1.6 Feature extraction policies

The idea of learning a feature extraction policy, as in our DMS-π framework, is not inherently new. Much prior work takes a generative approach that models the distribution

of feature values, and the policy chooses individual features to evaluate. This generative setting differs greatly from the discriminative setting studied in this thesis; in the generative approach, one models a distribution $P(\mathbf{f}(X), Y)$ and uses Bayes rule to make predictions. One approach to feature extraction given a generative model is to maximize the *value of information*, which is a measure of the information about $P(Y \mid \mathbf{f}(X))$ gained by observing new features, subject to cost constraints. Though generally infeasible, approximations exist given graphical models of the data or for special cases (Bilgic and Getoor, 2007; Krause and Guestrin, 2005, 2009). The policies can also be learned using reinforcement learning techniques; e.g. Ji and Carin (2007) formulate the feature extraction problem as a Partially Observable MDP (POMDP) and use Gaussian Mixture Models (GMM) to model the features, while Bayer-Zubek (2004) use discrete distributions in a standard MDP framework. It's noteworthy that in both cases, the best empirical policies are defined myopically, greedily optimizing a heuristic, despite formulating the problem using the standard reinforcement learning frameworks.

There has been less work on discriminative approaches to feature extraction policies. Recently, several works have proposed modeling the value of a classifier in terms of the features used. For instance, Gao and Koller (2011) explicitly model the information gain of introducing additional features for a particular example, based on a pre-trained pool of classifiers that use different subsets of features; in this case, a GMM is used to model a joint distribution over possible classifier outputs and class labels, rather than modeling the features directly. Additionally, He et al. (2012)–who use an imitation learning approach for dynamic feature selection–avoid modeling a value function altogther and instead learn a classifier using meta-features of previous outputs in order to classify which feature subset to compute next. Note that although both of these approaches are similar in spirit to the reinforcement learning framework used for DMS-π, neither of these approaches apply to the structured setting.

2.2 Related approaches for structured prediction

In this next section, we review prior work that also attempts to solve the expressiveness-computation trade-off in the structured setting. These methods are more directly comparable to our work than the literature discussed in the previous section, so we focus on highlighting the distinction between previous, related work and the work presented in this thesis.

2.2.1 Coarse-to-fine reasoning

The SPC framework proposed in this work can be considered as an instance of coarse-to-fine reasoning: as implemented in practice, early stages of the cascade eliminate outputs using a much coarser partitioning of the output space than the later stages. This basic idea has been used before in the structured setting, and we review some key examples here.

The simplest approach to coarse-to-fine reasoning is to use a two-step inference procedure: before running inference in a complex model, a simpler model computes marginals over the state space, and any states with low probability are eliminated. In natural language parsing, many works (e.g. Carreras et al. (2008); Charniak (2000)) utilize this idea: the marginals of a simple context free grammar or dependency model are used to prune the parse chart for a more complex grammar. The key difference with our work is that we explicitly learn a sequence of models tuned specifically to filter the space accurately and effectively; furthermore, because we use discriminative max-marginals instead of probabilistic marginals, we can provide theoretical bounds on the generalization error of our filtering steps. Somewhat closer to our work is that of Petrov (2009), where an entire hiearchy of coarse-to-fine grammars is learned; however, we do not learn the structure of the hierarchy of models but assume it is given by the designer, and again we use a different objective for learning the coarse-to-fine hierarchy and a different filtering scheme at test-time.

In the computer vision community, coarse-to-fine reasoning also has been well established. E.g. Fleuret and Geman (2001) propose a coarse-to-fine sequence of binary tests to detect the presence and pose of objects in an image. The learned sequence of tests

is trained to minimize expected computational cost. More recently, Pedersoli et al. (2011) use coarse-to-fine processing in the structured setting by utilizing the scores of parts in low-resolution object models to prune the search space over part locations in high-resolution models. Once again, however, our work proposes a different filtering scheme and a different learning objective–and most importantly, our filtering criterion integrates information from an entire structured model rather than using local scores to prune hypotheses.

Raphael (2001) provide a provably correct coarse-to-fine method for solving arbitrary dynamic programs. This approach relies on partitioning the full state space into "superstates," and then efficiently computing upper and lower bounds on all possible transitions between the constituents within superstates; superstates can be eliminated from the search procedure once their bounds indicate the solution does not lie within. In a certain sense, the SPC approach proposed in this thesis substitutes efficiently computing bounds with efficiently computing max marginals, which is more useful in practice, but at the cost of potential sub-optimality if the learned filters are incorrect on a given example.

2.2.2 Early stopping cascade

Felzenszwalb et al. (2010) proposed a cascade for a structured parts-based object detection model. Their cascade works by early stopping while evaluating individual parts, if the combined part scores are less than fixed thresholds. While the form of this cascade can be posed in the general SPC framework (a cascade of models with an increasing number of parts), we differ from Felzenszwalb et al. (2010) in that our pruning is based on thresholds that adapt based on inference in each test example, and we explicitly learn parameters in order to prune safely and efficiently. However, the SPC idea and that of Felzenszwalb et al. (2010) are complementary, in the sense that Felzenszwalb et al. (2010) provide an early-exit strategy to speed up MAP inference (for a complex model), while SPC uses MAP inference (in simpler models) as a sub-routine; one could apply the same early-exit strategy at any point during SPC inference to quickly reject low-scoring outputs during a detection problem.

2.2.3 Prioritized inference

While modeling the feature extraction process as an MDP has been studied in the multi-class setting, using reinforcement learning to speed-up evaluation of structured models has been less well studied. One significant prior work is Jiang et al. (2012), who apply imitation learning to speed up inference in a syntatic parsing model; however, the state of their MDP is defined solely in terms of the inference procedure and does not explore feature sets in any way. In contrast, the DMS-π framework we propose focuses on feature extraction as a bottleneck, not inference as a bottleneck.

2.2.4 Meta-learners/predicting model accuracy

Another core component of the DMS/DMS-π approach is the idea of using meta-features to evaluate the progress of a structured model; this basic idea of predicting the accuracy of a model has been studied before, albeit in different contexts. E.g. Bedagkar-Gala and Shah (2010) attempt to predict various video analysis algorithm's performance (similar in spirit to the selector we propose), but based on measures of image quality rather than properties of model output. Jammalamadaka et al. (2012) propose an evaluator for human pose estimators, but only for single-frame images, and propose only learning "correct or not" coarse-level distinctions, whereas we attempt to predict a measure of the error of each model directly. In the speech community, Lanchantin and Rodet (2010) propose a parallel method of "dynamic model selection" in which several models are continually re-evaluated in an online fashion using a generative model, which is a very different setting than that we analyze here.

2.2.5 Applications of preliminary works

The SPC framework proposed in this thesis appeared in preliminary form in Weiss and Taskar (2010); Weiss et al. (2010), and since then several authors applied the ideas to specific applications outside the scope of this work. Specifically, Sapp et al. (2010) ap-

plied the SPC framework to human pose estimation in static images, using a coarse-to-fine multi-resolution framework similar to that used in section 5.5.2 of this thesis. Subsequent to Weiss et al. (2010), Sapp et al. (2011) applied the summation of max marginals from Ensemble-SPC in a different model of human pose in order to further advance state-of-the-art in pose tracking. However, in this work, we show how a sequence model built on the output of Sapp and Taskar (2013) yields faster, more accurate predictions, and we propose DMS to further speedup the evaluation of this model.

In natural language processing, Rush and Petrov (2012) apply the ideas developed in our preliminary work to the problem of dependency parsing in natural language processing. They learn a cascade of simplified parsing models using the objective presented in section 4.3 to achieve state-of-the-art performance in dependency parsing across several languages at about two orders of magnitude less time.

2.3 Summary

We have reviewed a broad cross-section of related work to the ideas proposed in this thesis. A significant source of related work is in the multi-class classification literature, where high-level ideas similar to those proposed here are applied to speed up computation of features or allow for scaling to very large problems. In this sense, our work represents a translation of these ideas into the structured setting in a novel (when compared to other work in the structured prediction domain), empirically effective, and principled fashion.

Chapter 3

Structured Prediction: An Overview

In this chapter, we provide a short tutorial and overview of the structured prediction methods that form a basis for the work in this thesis. We also introduce notation and language that will be used throughout.

3.1 Supervised learning

Given an input space $\mathcal{X}$, output space $\mathcal{Y}$, and a training set $\{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^n, y^n)\}$ of n 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 $\mathbb{E}_D \left[\mathcal{L}\left(h(X), Y\right)\right]$ for some non-negative loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$. This minimization is performed with respect to some hypothesis space $\mathcal{H}$. For example, in the linear binary classification setting, we typically have $\mathcal{Y} = \{-1, +1\}$ being the binary label, $\mathcal{L}(Y, Y') = \mathbf{1}\left[Y \neq Y'\right]$ being the 0-1 loss indicating whether or not a prediction is correct, and learn linear hypotheses of the form $h(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{f}(\mathbf{x}))$, where $\mathbf{w} \in \mathbb{R}^m$ parameterizes the model as a linear function of a computed feature vector $\mathbf{f}$.

In practice, minimizing the expected error in this setting can be difficult for three reasons:

- The 0-1 loss is not a convex function of the parameters $\mathbf{w}$.

- We can only estimate the loss via the n samples of the training set, and minimize with respect to this estimate.

- We may run out time trying to estimate the parameters for h or even compute $h(\mathbf{x})$.

The standard solution to the first problem is to replace $\mathcal{L}$ with a convex upper bound; the choice of bound leads to different flavors of optimization problems. However, the second problem leads to a trade-off: as we increase the expressiveness of the model (as measured by the number of parameters m), we decrease the reliability of our estimate of the loss of the minimizer $h^\star$ for any fixed number of training samples n. In other words, we trade off *approximation error* with *estimation error*; a simpler model may make mistakes due to approximating $D(X, Y)$ poorly, but a more complex model may make mistakes due to poorly estimating $\mathbf{w}$. This classical trade-off between approximation and estimation error (bias/variance) is fundamental in machine learning and has been well studied. Standard statistical model selection techniques (Akaike, 1974; Barron et al., 1999; Bartlett et al., 2002; Devroye et al., 1996; Vapnik and Chervonenkis, 1974) control this trade-off to minimize expected error by exploring a hierarchy of models of increasing complexity.

In this thesis, we are primarily concerned with the third final problem, which also yields a trade-off: as discussed in Chapter 1, increasing the expressiveness of the model also typically leads to increased computation time. In the next section, we make this problem concrete for the structured setting, and make explicit the two primary potential bottlenecks (inference and feature extraction). Note that for the structured setting, as in the general supervised case, the first two problems still apply.

3.2 Structured prediction

In the setting of *structured prediction*, Y is a ℓ-vector of discrete multi-class variables. Specifically, we say that $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_\ell$, where $\mathcal{Y}_i = \{1, \ldots, K\}$. For a complete output $\mathbf{y}$, we denote the i'th component as y_i. In many settings, the number of random variables, ℓ, differs depending on input X, in which case we denote the $\mathbf{x}$-specific output

space as $\mathcal{Y}(\mathbf{x})$, but for simplicity of notation, we assume a fixed ℓ here. Thus, making a prediction for a given input $\mathbf{x}$ requires predicting ℓ K-valued outputs jointly. We consider linear hypotheses of the following form (1.1) (reproduced here):

$$h(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}),$$

where the prediction $h(\mathbf{x})$ is the highest scoring output $\mathbf{y}$ according to the inner product of the parameter vector $\mathbf{w}$ and a feature function $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}^m$ mapping $(\mathbf{x}, \mathbf{y})$ pairs to a vector of features. The problem of computing the argmax is called the *inference* problem (it is equivalent to the problem of MAP inference for certain types of probabilistic models.)

Note that as written, (1.1) is an unconstrained discrete optimization problem over an exponentially large search space; furthermore, there is no way to exactly compute (1.1) for any $\mathbf{f}$ without searching the entire space. Therefore, we inject structure into the problem through two complementary approaches:

- **Factorizing the feature generating function $\mathbf{f}(\mathbf{x}, \mathbf{y})$ over subsets of $\mathcal{Y}$.** For example, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we can define unary features $\mathbf{f}_i(\mathbf{x}, y_i)$ for each $i \in \mathcal{V}$ and pairwise features $\mathbf{f}_{ij}(\mathbf{x}, y_i, y_j)$ for each $(i, j) \in \mathcal{E}$. One possible feature generating function is then just the sum of all factored features:

  $$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i \in \mathcal{V}} \mathbf{f}_i(\mathbf{x}, y_i) + \sum_{(i,j) \in \mathcal{E}} \mathbf{f}_{ij}(\mathbf{x}, y_i, y_j) \tag{3.1}$$

  Expressing feature functions as the sum of factors is an efficient and useful way of encoding the structure of the relationships between different output variables in the model. Most importantly, it provides a template to which we can apply a generic dynamic programming inference algorithm, which yields exact solutions in some cases and approximate solutions in the others. We will describe inference in more detail using factorized representations in Section 3.2.2. Furthermore, since we consider linear hypotheses, we discuss factorizing the scoring function $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$ and factorizing the feature function interchangably.

- **Introducing constraints on the output space $\mathcal{Y}$.** For example, in the problem of dependency parsing from natural language processing, each y_i represents the a directed

edge from the y_i'th word to the i'th word. For $\mathbf{y}$ to be a valid parse, these edges must form a directed tree. Thus, we constrain $\mathcal{Y}$ to only contain $\mathbf{y}$ such that $\mathbf{y}$ is a valid parse tree. For certain feature functions, this problem can be solved efficiently even though the generic message passing algorithm does not apply (McDonald et al., 2005). In general, introducing constraints in this way can prevent the use of generic inference algorithms, making the inference problem more difficult to solve in practice. In this thesis, we will only consider problems with no constraints or constraints that can be easily incorporated into the generic inference algorithm.

3.2.1 Representing structure with factor graphs

A convenient form of representing the factorization of the scoring function is a *factor graph*. A factor graph is a bipartite undirected graph $\mathcal{G}$, where there are ℓ vertices for the output variables $y_1, \ldots, y_\ell$ and F vertices, each representing one factor in the scoring function. Each factor is connected via an edge to each variable in its scope (equivalently, each variable is connected via an edge to each factor it participates in.) Typically, factor graphs are represented visually using round vertices for variables and squares for factors. Factor graphs are useful both as a visual aid to help understand the assumed structure of a model and as a scaffold for efficient inference.

Several examples of common factor graph structures are presented in Figure 3.1.

3.2.2 Complexity of inference

Given a factor graph, let $\mathcal{F}_i$ be the set of factors that the i'th variable Y_i participates in, and $\mathcal{Y}_c$ be the scope of the factor c. Inference proceeds by passing messages from factors nodes to variable nodes and from variable nodes to factor nodes. Each message is a list of

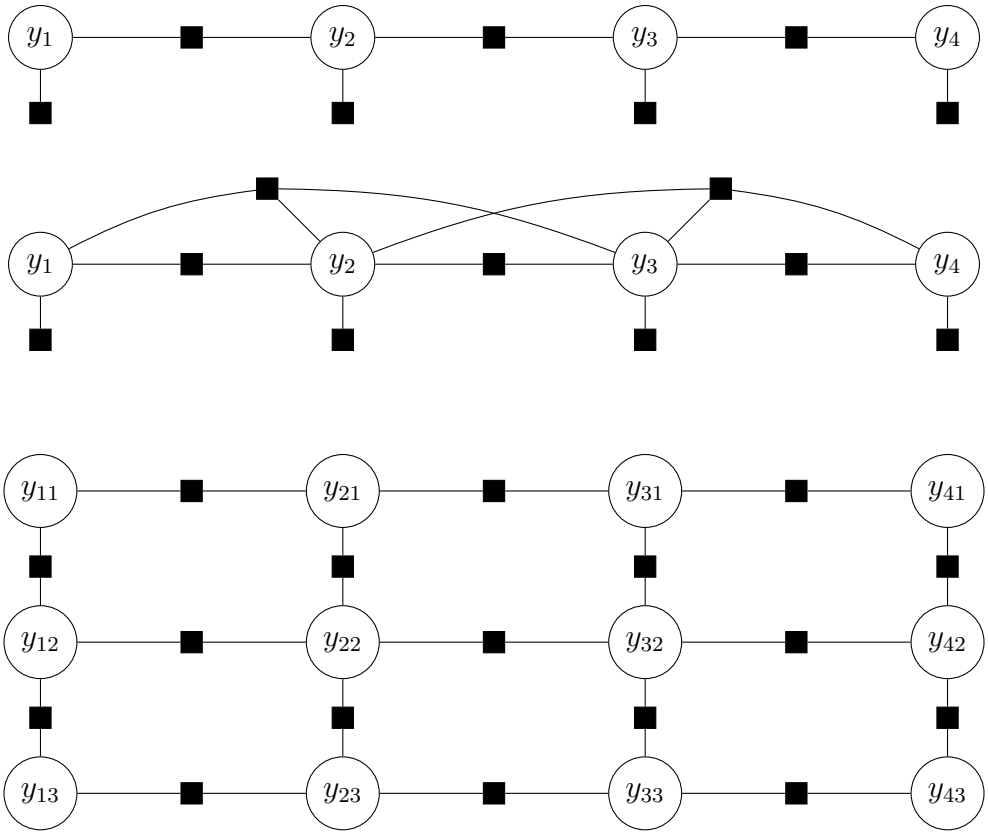


Figure 3.1: Examples of several common factor graphs. **Top:** Bigram (1-order) linear-chain model. **Middle:** Tri-gram (2-order) linear-chain model. **Bottom:** Grid model.

K values: the messages are defined as follows,

$$\omega_{c \to Y_i}(k) = \max_{y_c : y_i = k} \mathbf{w}^\top \mathbf{f}_c(\mathbf{x}, \mathbf{y}_c) + \sum_{j \in \mathcal{Y}_c \setminus i} \omega_{Y_j \to c}(y_j), \tag{3.2}$$

$$\omega_{Y_i \to c}(k) = \sum_{c' \in \mathcal{F}_i \setminus c} \omega_{c' \to Y_i}(k), \tag{3.3}$$

where $\omega_{c \to Y_i}(k)$ is the message from factor c to variable i for state k, and $\omega_{Y_i \to c}$ is the message from variable i to factor c for state k. If the factor graph is a tree, then the messages have the following natural intuition. Each variable is separated by its factors from the rest of the variables in the graph; the message $\omega_{c \to Y_i}(k)$ is the maximum score achievable by the rest of the graph (separated from Y_i by c) if Y_i takes on the value k. Similarly, the message $\omega_{Y_i \to c}(k)$ is the maximum score achievable by the component of the graph connected to the

variable Y_i if the factor c is not used and Y_i takes on value k. Note that certain messages depend upon other messages being precomputed. However, if the factor graph is a tree, then clearly any leaf of the tree can start the process by sending its message to its sole neighbor, and all messages will eventually be computed.

What is the complexity of this inference scheme? For a factor of size d, computing a single element of the message table involves taking a maximization over $d-1$ variables, each of which can take on K values. Computing the entire message table is therefore $O(K^d)$ operations; if there are F factors, then the entire inference procedure is $O(K^d F)$.

However, if the graph is not a tree (i.e. it contains a cycle), the recursive definition of messages implies that we cannot guarantee *exact* computation of messages in a finite amount of time. We can still obtain an approximate answer by running message passing scheme anyway; we simply choose an arbitrary start point and re-compute messages as the inputs change, stopping when messages converge or after a pre-determined period of time. This approximate inference scheme is a variant of *loopy belief propagation* (LBP), one of many different approximate inference algorithms for cyclic factors. There is a large body of research on approximate inference algorithms that is outside the scope of this thesis; see e.g. Boykov et al. (2001); Koller and Friedman (2009); Murphy et al. (1999); Sontag (2010) for more information.

Note that a cyclic graph does not imply that exact inference is impossible; simply that we cannot apply the message passing algorithm without modification. In fact, we can transform a loopy factor graph into an equivalent tree factor graph by creating auxiliary variable nodes that represent conjunctions of the original variables; we pay the price that these auxiliary variables may have exponentially large state spaces, so we cannot guarantee that exact inference will be efficient. We also need to introduce consistency constraints to ensure that auxiliary variables that share some underlying variable y_i agree about the maximizing assignment to y_i. We'll return to this idea for a specific set of factor graphs in section 3.3.

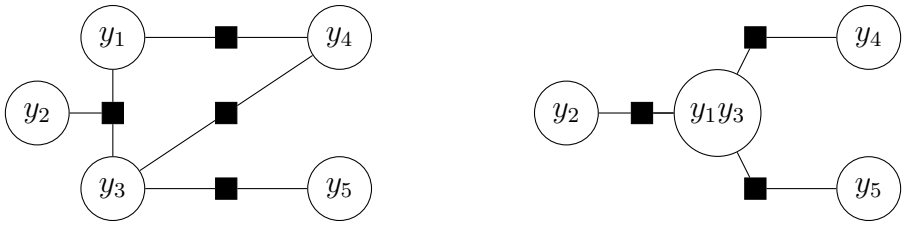


Figure 3.2: Agglomerating variable states.

3.2.3 Max-margin parameter learning

We now turn to the question of how to learn $\mathbf{w}$ to provide useful predictions from the inference procedure. We follow the structured max-margin framework (Taskar et al., 2005), which aims to solve the following:

$$\min_{\mathbf{w},\xi\geq 0} \lambda||\mathbf{w}||^2 + \frac{1}{n}\sum_i \xi_i \quad \text{s.t.} \quad \mathbf{w}^\top \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i) \geq \max_{y\in\mathcal{Y}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}^i, \mathbf{y}) + \Delta(\mathbf{y}^i, \mathbf{y}) + \xi_i, \quad \forall i. \tag{3.4}$$

Above, Δ is a loss function, such that $\Delta(\mathbf{y}^i, \mathbf{y})$ measures the difference between the target output $\mathbf{y}^i$ and any given output $\mathbf{y}$. For most of this work, we use the Hamming loss,

$$\Delta(\mathbf{y}, \mathbf{y}') = \sum_{i=0}^{\ell} \mathbf{1}\left[y_i \neq y_i'\right] \tag{3.5}$$

However, other losses may be used in practice. Intuitively, (3.4) balances a standard ℓ_2 regularization term with a penalty for violating a set of *margin* constraints. In order to satisfy the margin constraints, the true output $\mathbf{y}^i$ must have higher score than every alternative $\mathbf{y}$ by margin $\Delta(\mathbf{y}^i, \mathbf{y})$; thus, outputs that are closer to the ground truth (i.e. more correct) need less margin than those that are farther away, as measured by the loss Δ.

Although there are a variety of approaches to solving (3.4), there is a common thread to all of the solvers used in this thesis: they each require inference as a sub-problem. Intuitively, efficiently solving (3.4) requires computing which of the constraints are violated, either to compute a sub-gradient or to add constraints to a working set (Joachims et al., 2009). Thus, the complexity of learning is highly dependent on the complexity of inference.

3.3 Trading off computation and expressiveness

As we have seen, the complexity of inference is a function of the structure of the factor graph and of the feature functions $\mathbf{f}_c(\mathbf{x}, \mathbf{y})$, and inference is often a bottleneck during parameter learning. Therefore, we have a fundamental trade-off between the expressiveness of the model and the computation required to use it effectively.

Linear-chain models. As an example, consider a first-order linear-chain model (Figure 3.1). This model has two types of factors: pairwise factors between consequtive variables y_i and y_{i+1} and unary factors for each variable y_i. For ease of notation, we avoid using the subscript $\mathbf{f}_c$ and incorporate the scope of the factor directly, as follows:

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{\ell} \mathbf{f}(\mathbf{x}, y_i, y_{i+1}) + \mathbf{f}(\mathbf{x}, y_i) \tag{3.6}$$

Since we have $\ell - 1$ pairwise factors and ℓ unary factors, the complexity of inference using the message passing algorithm is $O(K^2\ell + K\ell) = O(K^2\ell)$. We will revisit such linear-chain models several times in this thesis; since the features can capture sequential dependencies between predictions, such models are well suited for tasks such as phoneme recognition, handwriting recognition, or visual tracking tasks.

Two channels for more expressiveness. Given the basic first-order linear-chain, we can introduce more expressiveness into the model through two main channels. The first is simply to incorporate more features into the unary or pairwise terms. For example, when tracking an object in video, optical flow (Horn and Schunck, 1981) – estimating the movement of each pixel in an image from one frame to the next – is an expensive but often very useful pairwise feature.

The second channel is to increase the scope of the factors in the model, or to introduce additional factors. This allows for more accurate modeling of the higher-order statistics of the data. In the case of linear-chain models, we can widen the scope of the pairwise factors to incorporate three sequential variables at a time. Note that this introduces a loop into

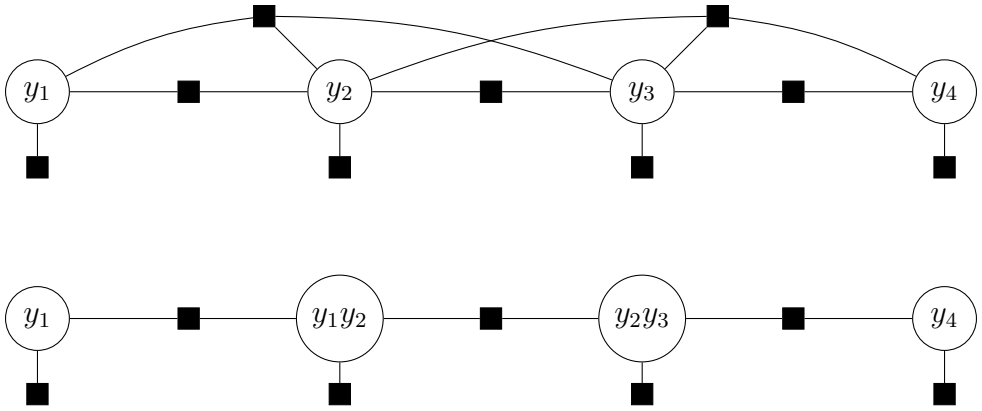


Figure 3.3: Removing loops from a trigram model. **Top:** The trigram linear-chain model contains loops due to incorporating triplets into a new 2-order factor. **Bottom:** By agglomerating neighboring states, we obtain a bigram model with no loop, but where the conjoined variables have K^2 possible assignments instead of K.

the tree structure (Figure 3.3). However, we can still perform exact inference by applying the junction method we described in the previous section; we introduce auxiliary variables representing the conjunction of sequential variables as shown in Figure 3.3. Note that the size of the state space of the auxiliary variables increases by a factor of K for each additional variable we incorporate, so exact inference in a d-gram linear-chain model takes $O(K^d \ell)$ time. Note that we also must introduce consistency requirements into the inference procedure: variables $y_1 y_2$ and $y_2 y_3$ must agree on the value of y_2 in order to map from an assignment in the new model back to a sequence of state assignments in the original model. However, these agreement constraints can be easily added by introducing a new indicator feature into the pairwise factors that measures such agreement, taking on the value $-\infty$ when agreement is not satisfied.

3.4 Summary

In this section, we have reviewed the relevant basic aspects of structured prediction: factor graphs, inference, and learning. The rest of the thesis is concerned with learning to

explicitly control the the trade-off between expressiveness and computation. As we have seen, there are two primary modes through which we can increase expressiveness: increasing factor scope, or adding features to factors. In the next chapters, we propose the SPC/Ensemble-SPC frameworks to address the former and allow for higher order factors or over finer discretizations of the state space, and the DMS/DMS-π frameworks to address the latter, and allow for more expensive features computed within a fixed set of factors.

Chapter 4

Structured Prediction Cascades (SPC)

Overview. As we have discussed, model complexity is limited by computational constraints at prediction time in practice, 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 the state space of structured models, the problem of explicitly learning to control the error/computation tradeoff has not been addressed.

In this chapter, 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 key principle of our approach is that each model in the cascade is optimized to accurately filter and refine the structured output state space of the *next* model, speeding up both learning and inference in the next layer of the cascade. Although complexity of inference increases (perhaps exponentially) from one layer to the next, the *state-space sparsity* of inference increases exponentially as well, and the entire cascade procedure remains highly efficient. We call our approach *Structured Prediction Cascades* (SPC).

Contributions. We first introduce SPC in the next section, and propose a novel method of filtering the output space of structured models based on *max marginals*. We analyze

Algorithm 1: Overview of SPC Inference.

input : Example $\mathbf{x}$, models $\mathbf{w}^0, \ldots, \mathbf{w}^T$,$\mathbf{f}^0, \mathbf{f}^T$.

output: Approximate prediction $h(\mathbf{x})$.

initialize $\mathcal{S}^0 = \mathcal{Y}$;

for $t = 0$ **to** $T - 1$ **do**

  Run sparse inference over $\mathcal{S}^t$ using model $\mathbf{w}^t, \mathbf{f}^t$;

  Eliminate a subset of low-scoring outputs;

  Output S^{t+1} for the next model;

Predict using the final level: $h(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{S}^T} \psi_{\mathbf{w}^T}(\mathbf{x}, \mathbf{y})$

this filtering method and propose two novel loss functions that bound the accuracy and efficiency of a single level of the cascade. We show how to optimize these losses and provide theoretical analysis in the form of generalization bounds–to our knowledge, these are the first generalization bounds for the accuracy/efficiency trade-off of a structured model. We propose a stage-wise learning algorithm (Algorithm 2) for SPC and apply it to two sequential prediction tasks: part-of-speech (POS) tagging and handwriting recognition. SPC provides a significant increase in efficiency on the POS tagging task and a dramatic increase in accuracy on the handwriting recognition task.

4.1 Enabling complexity via filtering $\mathcal{Y}$

In this section, we introduce the SPC framework to handle problems for which the inference problem in (1.1) is prohibitively expensive. Rather than learning a single monolithic model, we define a structured prediction cascade to be a coarse-to-fine sequence of increasingly complex models $\mathbf{w}^0, \ldots, \mathbf{w}^T$ with corresponding features $\mathbf{f}^0, \ldots, \mathbf{f}^T$. As discussed in section 3.3, there are many ways in which "increasingly complex" can be defined. However, for the purposes of introducing the method, the increasing complexity can be entirely encapsulated in the feature functions $\mathbf{f}^i$. We will later define specific instances of the cas-

cade for specific problems: for sequential prediction problems, we will define $\mathbf{f}^i$ to be a linear-chain model of order i, while for pose estimation problems we will define each $\mathbf{f}^i$ to be a pose model of increasing resolution.

Regardless of the specific layout of the cascade, the goal of each model is to filter out a large subset of the possible values for y without eliminating the correct one. The idea is that by eliminating possibilities for y, inference in the next model will require searching a much smaller space. Thus, the filtering process is feed-forward; simpler, faster models are used to quickly eliminate unlikely outputs at first while richer, more complex models can quickly search among the remaining possibilities to produce a final output.

In summary, a high-level overview of the SPC inference framework is given in Algorithm 1. Here, $\mathcal{S}^t$ denotes a sparse (filtered) version of the output space $\mathcal{Y}$. The process is also illustrated in Figure 4.1. See Figure 4.2 for a concrete example of the output of a the first two stages of a cascade for handwriting recognition and human pose estimation. We will discuss how to represent and choose $\mathcal{S}^t$ in the next section. The key challenge is that $\mathcal{S}^t$ are exponential in the number of output variables, which rules out explicit representations. The representation we propose is implicit and concise. It is also tightly integrated with parameter estimation algorithm for $\mathbf{w}^t$ that optimizes the overall accuracy and efficiency of the cascade.

4.2 Cascaded inference with max-marginals

In order to represent and filter low-scoring outputs, we use *max-marginals*, for reasons that we detail below. For any value of $\mathbf{y}_c$, we define the max-marginal $\psi^\star_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c)$ to be the maximum score of any full output $\mathbf{y}$ that is consistent with the (partial) assignment $\mathbf{y}_c$:

$$\psi^\star_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c) \triangleq \max_{\mathbf{y}' : \mathbf{y}'_c = \mathbf{y}_c} \psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}'). \tag{4.1}$$

Max-marginals can be computed exactly and efficiently for any factor c in low-treewidth graphs, although the computational cost is exponential in $|c|$ (the number of variables in the factor) when the state-space is not filtered. Note that max-marginals can be computed

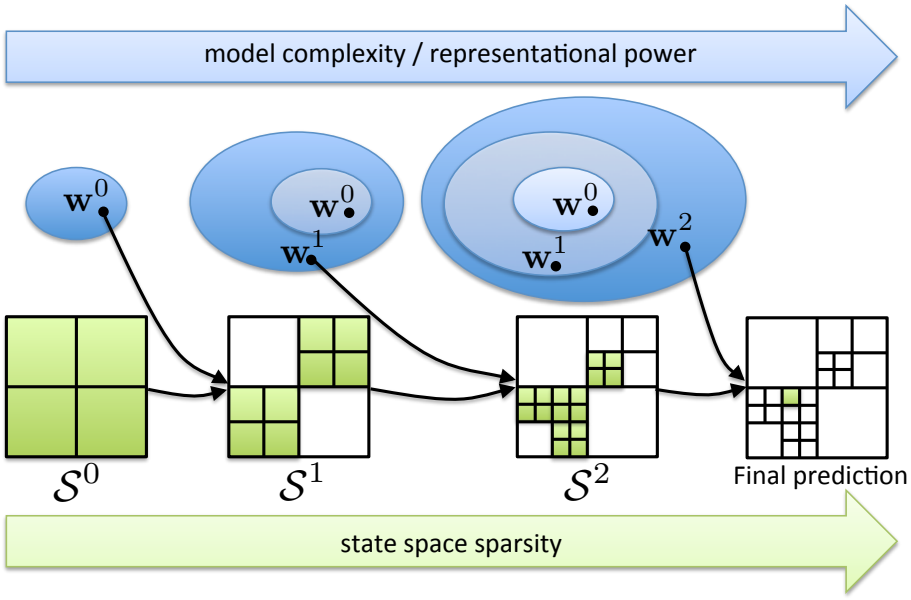


Figure 4.1: A high-level overview of the SPC inference framework. As the cascade progresses, the representational power of the models increases, yet tractability is maintained by sufficient filtering of the state space.

over *any* subset of variables c, not just the factors used in the feature function $\mathbf{f}$; for example, in Section 5.5.2, we compute max-marginals over single variables (i.e., $\mathbf{y}_c = y_j$) when performing human pose estimation, but at increasingly higher resolutions. On the other hand, in Section 4.5 we compute max-marginals over increasingly large factors for sequence models (e.g. bigram, trigrams, and quadgrams).

Exact computation of max-marginals for a factor c requires the same amount of time to run as standard exact MAP inference. This process is visualized in Figure 4.3: once forward and backward max-sum messages have been computed for MAP inference, the max-marginal for a given value $\mathbf{y}_c$ is simply the sum of the score $\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c)$ plus the incoming messages to the variables in c. Note that in practice, both stages of computation become faster as the output space becomes increasingly sparse as the input proceeds through the cascade. This algorithm can also compute the maximizing assignment for each $\mathbf{y}_c$,

$$\mathbf{y}^{\star}_{\mathbf{w}}(\mathbf{x}, y_c) \triangleq \operatorname*{argmax}_{\mathbf{y}' : \mathbf{y}'_c = \mathbf{y}_c} \psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}'). \tag{4.2}$$

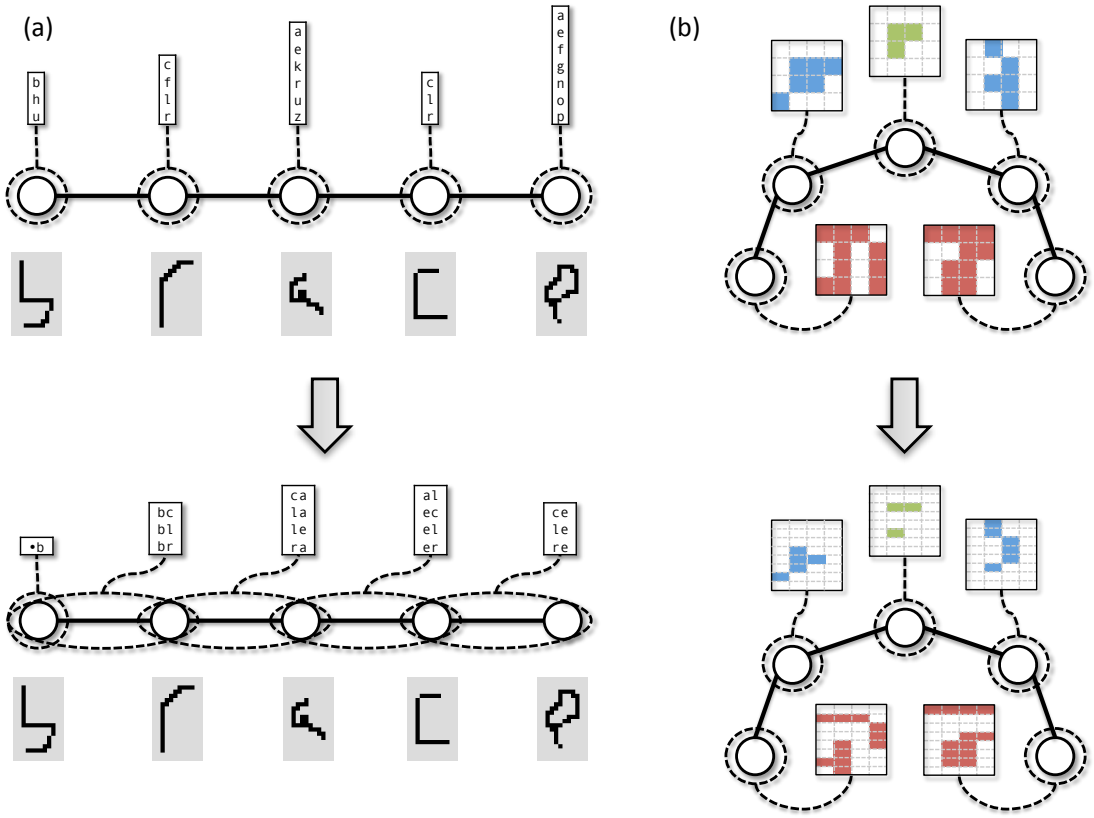


Figure 4.2: Sample output from the first two layers of a cascade. Circles represent output variables, and the dashed lines indicate factors that are being filtered at a given level of the cascade, with the attached tables representing the sparse state space. The solid lines indicate the graph used for inference and features. **(a)** Output from a handwriting recognition cascade (Section 4.5) of increasing Markov order. The first level outputs a sparse set of possible letters for each image. The second level takes as input the sparse set of letters, and further refines this to a very sparse set of *bigrams* at each position. **(b)** Output from a coarse-to-fine human pose cascade (Sapp et al., 2010). The colored areas indicate valid 2D locations for each joint. Unlike the sequence cascade (a), the factors stay the same from one layer to another. Instead, the resolution of the state space doubles with each additional layer.

Symbol	Meaning
$\mathcal{X}, X, x$	input space, variable and value
$\mathcal{Y}, \mathbf{Y}, Y_i, \mathbf{y}, y_i$	output space, variables and value
$\mathcal{F}, c, \mathbf{y}_c$	set of factors, individual factor, factor assignment
$\mathbf{f}(x, \mathbf{y})$	features of input/output pair
$\mathbf{f}_c(x, \mathbf{y}_c)$	features of a factor assignment
$\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) \triangleq \mathbf{w}^\top \mathbf{f}(x, \mathbf{y})$	score of input/output pair
$\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c) \triangleq \mathbf{w}^\top \mathbf{f}_c(x, \mathbf{y}_c)$	score of a factor assignment
$\psi^\star_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c) \triangleq \max_{\mathbf{y}' : \mathbf{y}'_c = \mathbf{y}_c} \psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}')$	max-marginal of a factor assignment y_c
$\mathbf{y}^\star_{\mathbf{w}}(\mathbf{x}, y_c) \triangleq \operatorname{argmax}_{y' : y'_c = y_c} \psi_{\mathbf{w}}(\mathbf{x}, y')$	best scoring output consistent with factor assignment y_c

Table 4.1: Summary of key notation.

We call $\mathbf{y}^\star_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c)$ the *argmax-marginal* or *witness* for $\mathbf{y}_c$ (it might not be unique, so we break ties in an arbitrary but deterministic way).

Once max-marginals have been computed, we filter the output space by discarding any factor assignments $\mathbf{y}_c$ for which $\psi^\star_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c) \leq t$ for a threshold t (Figure 4.4). This filtering rule has two desirable properties for the cascade that follow immediately from the definition of max-marginals:

Lemma 1 (Safe Filtering)**.** *If* $\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) > t$*, then* $\forall c \;\; \psi^\star_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c) > t$*.*

Lemma 2 (Safe Lattices)**.** *If* $\max_{\mathbf{y}'} \psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}') > t$*, then* $\exists \mathbf{y} \; \forall c \;\; \psi^\star_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c) > t$*.*

By Lemma 1, ensuring that the score of the true label $\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y})$ is greater than the threshold is sufficient (although not necessary) to guarantee that no marginal assignment $\mathbf{y}_c$ consistent with the true global assignment $\mathbf{y}$ will be filtered. This condition will allow us to define a max-marginal based loss function that we propose to optimize in Section 4.3 and will analyze in Section 4.4. Lemma 2 follows from Lemma 1, which states that so long as the threshold is less than the maximizing score, there always exists a global assignment $\mathbf{y}$ with no pruned factors (i.e., a valid assignment always exists after pruning). Thus, Lemma

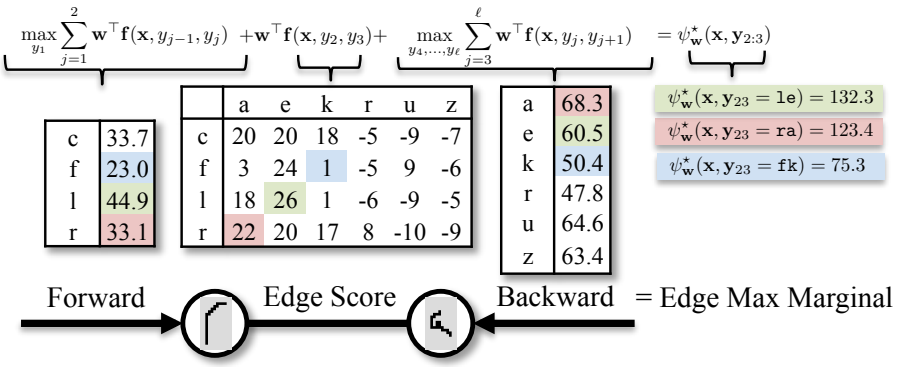


Figure 4.3: Computing max-marginals over bigrams via message passing. The input is the same as in Figure 4.2. Once forward and backward messages have been computed, the max-marginal is simply the sum of incoming messages and the score of the factor over bigrams.

2 guarantees that $|\mathcal{S}^{i+1}| \geq 1$ in the SPC algorithm introduced above, and therefore the cascade will always produce a valid output. Note that neither property generally holds for standard sum-product marginals $p(\mathbf{y}_c|\mathbf{x})$ of a log-linear CRF (where $p(\mathbf{y}|\mathbf{x}) \propto e^{\psi_{\mathbf{w}}(\mathbf{x},\mathbf{y})}$), which motivates our use of max-marginals.

The next component of the inference procedure is choosing a threshold t for a given input $\mathbf{x}$ (Figure 4.4). Note that the threshold cannot be defined as a single global value but should instead depend strongly on the input $\mathbf{x}$ and $\psi_{\mathbf{w}}(\mathbf{x}, \cdot)$ since scores are on different scales for different $\mathbf{x}$. 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 $\psi_{\mathbf{w}}(\mathbf{x}, \cdot)$ function, and we instead approximate a quantile threshold with a threshold function that is continuous and convex in $\psi_{\mathbf{w}}(\mathbf{x}, \cdot)$. We call this the *max-mean-max* threshold function (Figure 4.4), and define it as a convex combination of the maximum score and the mean of

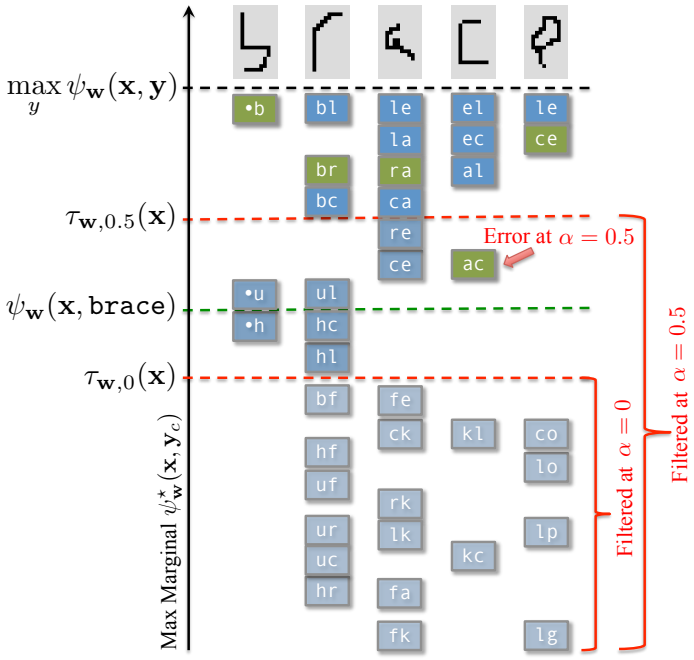


Figure 4.4: Thresholding bigrams using max-marginals. The input is the same as in Figure 4.2. The sparse set of unfiltered bigrams is shown at each position according to the max-marginal score. The bigrams corresponding to the correct label sequence, `brace`, are highlighted in green. The green dashed line indicates the score of the correct label sequence. Note that the max-marginals of the correct sequence are at least the score of the correct sequence. The black dashed line indicates the maximum score of any sequence, which is the maximum filtering threshold. The largest max-marginal values are all exactly equal to this score. The red dashed lines indicate two candidate filtering thresholds $\tau_{\mathbf{w},\alpha}(\mathbf{x}) = 0$ and $\tau_{\mathbf{w},\alpha}(\mathbf{x}) = 0.5$ and corresponding sets of filtered bigrams are highlighted. Note that a filtering error occurs at the more aggressive level of $\alpha = 0.5$.

the max-marginals:

$$\tau_{\mathbf{w},\alpha}(\mathbf{x}) = \alpha \max_{\mathbf{y}} \psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) + (1-\alpha) \frac{1}{\sum_{c \in \mathcal{F}} |\mathcal{Y}_c|} \sum_{c \in \mathcal{F}} \sum_{\mathbf{y}_c \in \mathcal{Y}_c} \psi^{\star}_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c). \tag{4.3}$$

Choosing a threshold using (4.3) is therefore equivalent to picking a $\alpha \in [0, 1)$. Note that $\tau_{\mathbf{w},\alpha}(\mathbf{x})$ is a convex function of $\mathbf{w}$ (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 α approaches 1, the number of max-marginals eliminated rapidly approaches 100%.[1]

In summary, the inner loop of the SPC algorithm can be detailed as follows. The sparse output space $\mathcal{S}^i$ is a list of valid assignments $\mathbf{y}_c$ for each factor c in the model $\mathbf{f}_i$ (e.g., Figure 4.2):

$$\mathcal{S}^i = \{\mathcal{Y}_c \mid \forall c \in \mathcal{F}\} \qquad \text{(list of valid factor values for all factors)} \tag{4.4}$$

Next, sparse max-sum message passing is used to compute max-marginals $\psi^{\star}_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c)$ (4.1) for each value $\mathbf{y}_c \in \mathcal{Y}_c$ of each factor c of interest. Finally, for a given α, a threshold is computed and low-scoring values of $\psi^{\star}_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c)$ are eliminated. Depending on the model in the next layer of the cascade, further transformation of the states may be necessary: For example, in the coarse-to-fine pose cascade (Section 6.3.2), valid 2-D locations for each limb are halved either vertically or horizontally to produce finer-resolution states for the next model (Figure 4.2b).

4.3 Learning structured prediction cascades

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.

[1] We use cross-validation to determine the optimal α in our experiments (see Section 4.5).

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

Given a training set, we can measure the accuracy and efficiency of our cascade, but what is unknown is the performance of the cascade on test data. In section 4.4, we provide a guarantee that our estimates of accuracy and efficiency will be reasonably close to the true performance measures with high probability. This suggests that optimizing parameters to achieve a desired trade-off on training data is a good idea.

We begin by quantifying accuracy and efficiency in terms of max-marginals, as used by SPC. We define the *filtering loss* $\mathcal{L}_f$ to be a 0-1 loss indicating a mistakenly eliminated correct assignment. As discussed in the previous section, Lemma 1 states that an error can only occur if $\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) \leq \tau_{\mathbf{w},\alpha}(\mathbf{x})$. We also define the *efficiency loss* $\mathcal{L}_e$ to simply be the proportion of unfiltered factor assignments.

Definition 1 (Filtering loss)**.** *A filtering error occurs when a max-marginal of a factor assignment of the correct output* $\mathbf{y}$ *is pruned. We define filtering loss as*

$$\mathcal{L}_f(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) = \mathbf{1}\left[\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) \leq \tau_{\mathbf{w},\alpha}(\mathbf{x})\right]. \tag{4.5}$$

Definition 2 (Efficiency loss)**.** *The efficiency loss is the proportion of unpruned factor assignments:*

$$\mathcal{L}_e(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) = \frac{1}{\sum_{c \in \mathcal{F}} |\mathcal{Y}_c|} \sum_{c \in \mathcal{F}, \mathbf{y}_c \in \mathcal{Y}_c} \mathbf{1}\left[\psi^{\star}_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c) > \tau_{\mathbf{w},\alpha}(\mathbf{x})\right]. \tag{4.6}$$

We now turn to the problem of learning parameters $\mathbf{w}$ and tuning of the threshold parameter α from training data. We have two competing objectives, accuracy ($\mathcal{L}_f$) and efficiency ($\mathcal{L}_e$), that we must trade off. 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 fixed level ϵ. Since the ultimate goal of the cascade is accurate classification, we focus on the problem

Algorithm 2: Forward Batch Learning of Structured Prediction Cascades.

input : Data $\{(\mathbf{x}^i, \mathbf{y}^i)\}_1^n$, features $\mathbf{f}^0, \ldots, \mathbf{f}^T$ and parameters $\alpha^0, \ldots, \alpha^{T-1}$

output: Cascade parameters $\mathbf{w}^0, \ldots, \mathbf{w}^T$

initialize $\mathcal{S}^0(\mathbf{x}^i) = \mathcal{Y}(\mathbf{x}^i)$ for each example.;

for $t = 0$ **to** $T - 1$ **do**

  Optimize (4.8) with sparse inference over the valid set $\mathcal{S}^t$ to find $\mathbf{w}^t$;

  Generate $\mathcal{S}^{t+1}(\mathbf{x}^i)$ from $\mathcal{S}^t(\mathbf{x}^i)$ by filtering low-scoring factor assignments $\mathbf{y}_c$:

$$\psi^\star_{\mathbf{w}^t}(\mathbf{x}^i, \mathbf{y}_c) \leq \tau_{\mathbf{w}^t, \alpha^t}(\mathbf{x}^i)$$

Learn $\mathbf{w}^T$ to maximize (3.4) over final sparse state spaces $\mathcal{S}^T(\mathbf{x}^i)$;

of minimizing efficiency loss while constraining the filtering loss to be below a desired tolerance.

We express the cascade learning objective for a *single level* of the cascade as a joint optimization over $\mathbf{w}$ and α:

$$\min_{\mathbf{w},\alpha} \ \mathbb{E}_{X,Y}\left[\mathcal{L}_e(X, Y; \mathbf{w}, \alpha)\right] \text{ s.t. } \mathbb{E}_{X,Y}\left[\mathcal{L}_f(X, Y; \mathbf{w}, \alpha)\right] \leq \epsilon. \tag{4.7}$$

We solve this problem for a single level of the cascade as follows. First, we define a convex upper-bound (4.8) on the filter error $\mathcal{L}_f$, making the problem of minimizing $\mathcal{L}_f$ convex in $\mathbf{w}$ (given α). We learn $\mathbf{w}$ to minimize filter error for several settings of α (thus controlling filtering efficiency). Given several possible values for $\mathbf{w}$, we optimize the objective (4.7) over α directly, using estimates of $\mathcal{L}_f$ and $\mathcal{L}_e$ computed on a held-out development set, and choose the best $\mathbf{w}$. Note that in Section 4.4, we present a theorem bounding the deviation of our estimates of the efficiency and filtering loss from the expectation of these losses.

For the first step of learning a single level of the cascade, we learn the parameters $\mathbf{w}$ for a fixed α using the following convex margin optimization problem:

$$SPC: \quad \min_{\mathbf{w}} \quad \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_i H(\mathbf{x}^i, \mathbf{y}^i; \mathbf{w}, \alpha), \tag{4.8}$$

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

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

The upper-bound H is a hinge loss measuring the margin between the filter threshold $\tau_{\mathbf{w},\alpha}(\mathbf{x}^i)$ and the score of the truth $\psi_{\mathbf{w}}(\mathbf{x}^i, \mathbf{y}^i)$; the loss is zero if the truth scores above the threshold by margin ℓ (in practice, the length ℓ can vary by example). We solve (4.8) using stochastic sub-gradient descent. Given a sample $(\mathbf{x}, \mathbf{y})$, we apply the following update if $H(\mathbf{w}, \mathbf{x}, \mathbf{y})$ (i.e., the sub-gradient) is non-zero:

$$\mathbf{w}' \leftarrow (1 - \eta\lambda)\mathbf{w} + \eta\mathbf{f}(\mathbf{x}, \mathbf{y}) - \eta\alpha\mathbf{f}(\mathbf{x}, \mathbf{y}^\star) - \eta(1 - \alpha)\frac{1}{\sum_c |\mathcal{Y}_c|} \sum_{c \in \mathcal{F}, \mathbf{y}_c \in \mathcal{Y}} \mathbf{f}(\mathbf{x}, \mathbf{y}^\star_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c)). \tag{4.9}$$

Above, η is a learning rate parameter. The key distinguishing feature of this update compared to the structured perceptron update is that it subtracts features included in all max-marginal assignments $\mathbf{y}^\star_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c)$.

Note that because (4.8) is λ-strongly convex, we can choose $\eta_t = 1/(\lambda t)$ and add a projection step to keep $\mathbf{w}$ in a fixed norm-ball. The update then corresponds to the Pegasos update with convergence guarantees of $\tilde{O}(1/\epsilon)$ iterations for ϵ-accurate solutions (Shalev-Shwartz et al., 2007).

An overview of the entire learning process for the whole cascade is given in Algorithm 2. Levels of the cascade are learned incrementally using the output of the previous level of the cascade as input. Note that Algorithm 2 trades memory efficiency for time efficiency by storing the sparse data structures $\mathcal{S}^t$ for each example. A more memory-efficient (but less time efficient) algorithm would instead run all previous layers of the cascade for each example during sub-gradient descent optimization of (4.8).

Finally, in our implementation, we can sometimes achieve better results by further tuning the threshold parameters α^t using a development set. We first learn $\mathbf{w}^t$ using some fixed α^t as before. However, we then choose an improved $\bar{\alpha}^t$ by maximizing efficiency subject to the constraint that filter loss on the development set is less than a *tolerance* ϵ_t:

$$\bar{\alpha}^t \leftarrow \operatorname*{argmin}_{0 \le \alpha' < 1} \sum_{i=1}^{n} \mathcal{L}_e(\mathbf{x}^i, \mathbf{y}^i; \mathbf{w}^t, \alpha') \quad \text{s.t.} \quad \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_f(\mathbf{x}^i, \mathbf{y}^i; \mathbf{w}^t, \alpha') \le \epsilon_t.$$

Furthermore, we can repeat this tuning process for several different starting values of α^t and pick the $(\mathbf{w}^t, \bar{\alpha}^t)$ pair with the optimal trade-off, to further improve performance. In practice, we find that this procedure can substantially improve the efficiency of the cascade while keeping accuracy within range of the given tolerance.

4.4 Generalization analysis

We now present generalization bounds on the filtering and efficiency loss functions for a single level of a cascade. To achieve bounds on the entire cascade, these can be combined provided that a fresh sample is used for each level. To prove the following bounds, we make use of Gaussian complexity results from Bartlett and Mendelson (2002), which requires vectorizing scoring and loss functions in a novel structured manner (details in Appendix A.1). The main theorem in this section depends on Lipschitz dominating cost functions $\mathcal{L}_f^\gamma$ and $\mathcal{L}_e^\gamma$ that upper bound $\mathcal{L}_f$ and $\mathcal{L}_e$. Note that as $\gamma \to 0$, we recover $\mathcal{L}_f$ and $\mathcal{L}_e$.

Definition 3 (Margin-augmented losses)**.** *We define margin-augmented filtering and efficiency losses using the usual γ-margin function:*

$$r_\gamma(z) = \begin{cases} 1 & \text{if } z < 0 \\ 1 - z/\gamma & \text{if } 0 \le z \le \gamma \\ 0 & \text{if } z > \gamma. \end{cases} \tag{4.10}$$

$$\mathcal{L}_f^\gamma(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) = r_\gamma(\psi_\mathbf{w}(\mathbf{x}, \mathbf{y}) - \tau_{\mathbf{w},\alpha}(\mathbf{x})) \tag{4.11}$$

$$\mathcal{L}_e^\gamma(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) = \frac{1}{\sum_{c \in \mathcal{F}} |\mathcal{Y}_c|} \sum_{c \in \mathcal{F}, \mathbf{y}_c \in \mathcal{Y}_c} r_\gamma(\tau_{\mathbf{w},\alpha}(\mathbf{x}) - \psi^\star_\mathbf{w}(\mathbf{x}, \mathbf{y}_c). \tag{4.12}$$

Theorem 1. *Fix $\alpha \in [0, 1]$ and let Θ be the class of all scoring functions $\psi_\mathbf{w}$ with $||\mathbf{w}||_2 \le B$, let $|\mathcal{F}|$ be the total number of factors, $m = \sum_{c \in \mathcal{F}} |\mathcal{Y}_c|$ be the total number of factor assignments, $||\mathbf{f}_c(\mathbf{x}, \mathbf{y}_c)||_2 \le 1$ for all $\mathbf{x} \in \mathcal{X}, c \in \mathcal{F}$ and $\mathbf{y}_c \in \mathcal{Y}_c$. Then there exists a constant c such that for any integer n and any $0 < \delta < 1$ with probability $1 - \delta$ over*

samples of size n, every $\psi_{\mathbf{w}} \in \Theta$ satisfies:

$$\mathbb{E}\left[\mathcal{L}_f(X,Y;\mathbf{w},\alpha)\right] \leq \hat{\mathbb{E}}\left[\mathcal{L}_f^{\gamma}(X,Y;\mathbf{w},\alpha)\right] + \frac{cmB\sqrt{|\mathcal{F}|}}{\gamma\sqrt{n}} + \sqrt{\frac{8\ln(2/\delta)}{n}}, \quad (4.13)$$

$$\mathbb{E}\left[\mathcal{L}_e(X,Y;\mathbf{w},\alpha)\right] \leq \hat{\mathbb{E}}\left[\mathcal{L}_e^{\gamma}(X,Y;\mathbf{w},\alpha)\right] + \frac{cmB\sqrt{|\mathcal{F}|}}{\gamma\sqrt{n}} + \sqrt{\frac{8\ln(2/\delta)}{n}}, \quad (4.14)$$

where $\hat{\mathbb{E}}$ is the empirical expectation with respect to the sample.

Theorem 1 provides theoretical justification for the definitions of the loss functions $\mathcal{L}_e$ and $\mathcal{L}_f$ and the structured cascade objective; if we observe a highly accurate and efficient filtering model $(\mathbf{w}, \alpha)$ 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.

4.5 Experiments

Linear-chain Markov models. In this section, we apply the structured prediction cascades framework to sequence prediction tasks with increasingly high order linear-chain models. We first introduced these models in section 3.3. To review, the state space of a linear chain model is $\forall i \ : \mathcal{Y}_i = \{1, \ldots, K\}$, where K is the number of possible states. Thus, the size of the state space is K. A d-order linear-chain model has maximal factors $\{x, y_i, y_{i-1}, \ldots, y_{i-d}\}$. Thus, for an order d factor, there are K^{d+1} possible factor assignments, although we find that in practice very few high-order factor assignments survive the first few levels of the cascade (see Table 4.3.)

For a d-order linear-chain model, the standard factorization of the score of an output y is given by a combination of unary features $\mathbf{f}_u$ that depend on the input $\mathbf{x}$ and a set of transition features $\mathbf{f}_c$ that depend only on the assignment of a given factor $\mathbf{y}_c$:

$$\psi_{\mathbf{w}}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{\ell} \mathbf{w}^{\top}\mathbf{f}_u(\mathbf{x}, y_i) + \sum_{i=1}^{\ell}\sum_{j=1}^{d} \mathbf{w}^{\top}\mathbf{f}_{c_j}(y_i, \ \ldots \ , y_{i-j}), \quad (4.15)$$

where $\mathbf{f}_{c_j}$ are the features for the factor of size $|c| = j + 1$. I.e. for a 2-order model, we have unary features $\mathbf{f}_u(\mathbf{x}, y_i)$, bigram features $\mathbf{f}_{c_1}(y_i, y_{i-1})$, and trigram features $\mathbf{f}_{c_2}$ (y_i, y_{i-1}, y_{i-2}).

In general, any d-order linear-chain model can be equivalently represented as a bigram (2-order) model with K^d states. Thus, it is simplest to implement 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 $\mathcal{S}^t$ in a d-order model, we can generate an expanded list of valid assignments $\mathcal{S}^{t+1}$ for a $(d + 1)$-order model by concatenating the valid d-grams with all possible additional states. Note that this means that filtering occurs over *edges* in each bigram model, as opposed to the traditional method of filtering *states*.

4.5.1 Speed: Part-of-speech (POS) tagging

Experimental setup. We first evaluated our approach on a standard benchmark for part-of-speech (POS) tagging in English. 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. For English, we followed the standard benchmark by using volumes 0-17 of the Wall Street Journal portion of the Penn TreeBank (Marcus et al., 1993) for training, volumes 18-20 for development, and volumes 21-24 for testing. 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 filtering efficiency, we computed max marginals over subcliques representing $(d - 1)$ order state assignments rather than d-order cliques representing transitions. Thus, the bigram model filters unigrams and the trigram model filters bigrams, etc. Although we ran the cascades up to 5-gram models, peak accuracy was reached with trigrams on the POS task.

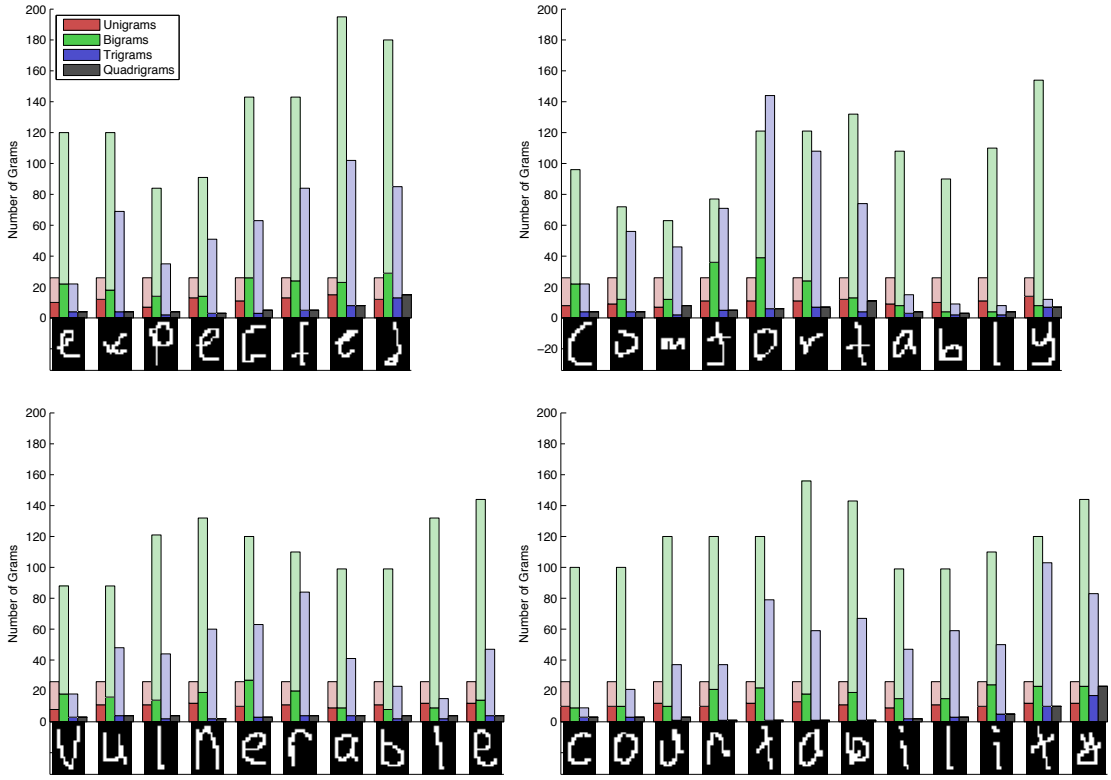


Figure 4.5: Sparsity of inference during an example sequence cascade. Each panel shows the complexity of inference on a different example from the OCR dataset at each position in the sequence. The total height of each bar represents the size of the valid assignments $\mathcal{S}^t$, while the shaded portion represents the remaining assignments after thresholding. Although complexity rises as unigrams are expanded into bigrams, filtering with bigrams and trigrams quickly reduces complexity to a few possible assignments at each position.

Baselines. We compared our SPC method to two baselines: the standard structured perceptron (SP) and a maximum a posteriori CRF. All methods used the same standard set of binary features: namely, a feature each (word,tag) pair $\mathbf{1}\left[X_t = x_t, Y_t = y_t\right]$ 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 $\mathcal{L}_e$ subject to $\mathcal{L}_f \leq \epsilon$ on the development set. For the CRF, we computed sum-product marginals $P(\mathbf{y}_c|\mathbf{x}) = \sum_{\mathbf{y}':\mathbf{y}'_c=\mathbf{y}_c} P(\mathbf{y}'|\mathbf{x})$, and used the threshold $\tau_{\mathbf{w},\alpha}(\mathbf{x}) = \alpha$ to eliminate all $\mathbf{y}_c$ such that $P(\mathbf{y}_c|\mathbf{x}) \leq \alpha$. For all algorithms, we used grid search over several values of η and λ, in addition to early stopping, to choose the optimal $\mathbf{w}$ based on the development set. For SPC training, we considered initial α's from the candidate set $\{0, 0.2, 0.4, 0.6, 0.8\}$. To ensure accuracy of the final predictor, we set a strict threshold of $\epsilon = 0.01\%$. All methods were trained using a structured perceptron for the final classifier.

SPC improves efficiency while preserving accuracy. The results are summarized in Table 4.2. SC was compared to CRF, an unfiltered SP model (Full), and a heuristic baseline in which only POS tags associated with a given word in the training set were searched during inference (Tags). SC was two orders of magnitude faster than the full model in practice, with the search space reduced to only ≈ 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 accuracy of the final classifier. Finally, training the full trigram POS tagger with exact inference is extremely slow and took several days to train; training the SC cascade took only a few hours, despite the fact that more models were trained in total.

Model:	**Full**	**SPC**	**CRF**	**Tags**
Accuracy (%)	96.83	96.82	96.84	—
Filter loss (%)	0	0.121	**0.024**	0.118
Test Time (ms)	173.28	**1.56**	4.16	10.6
Avg. Num States	1935.7	**3.93**	11.845	95.39

Table 4.2: Summary of WSJ Results. Accuracy is the accuracy 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.

4.5.2 Accuracy: Handwriting recognition

Experimental setup. We 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 Kassel (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. We trained six cascades, up to a sixth-order (sexagram) linear-chain model. This is significantly higher order than the typical third-order (trigram) models typically used in sequence classification tasks. Note that in practice, the additional accuracy gained by increasing the order of the model might be offset by the additional filtering errors incurred due to lengthening the cascade. Thus, each level of each cascade was tuned to achieve maximum efficiency subject to a maximum error *tolerance* ϵ, whereby α was set such that no more than ϵ filtering error was incurred by each level of the cascade.

SPC dramatically improves accuracy. Results are summarized in Table 4.3. We found that using higher order models led to a dramatic gain in accuracy on this dataset, increasing character accuracy from 77.35% to 98.54% and increasing word accuracy from 26.74% to 96.16%. It is interesting to note that the word level accuracy of the sixth-order model is roughly equivalent to the character-level accuracy of the trigram model. Furthermore, using a development set, we found that a stricter tolerance was required to gain accuracy from fifth- and sixth-order models, as reflected in Table 4.3. Finally, compared to previous approaches on this dataset, our accuracies are much higher; the best previously reported result on this dataset was 90.19% (Daumé et al., 2009).

In fact, the extremely high accuracies of our approach on this dataset highlight the particular features of this data. Due to the high number of subjects used, there are only 55 unique words in the handwriting recognition dataset. In fact, if just the first three letters of each word are given exactly, one can guess the identity of the word with 94.5% accuracy. Given more letters, it is possible to uniquely identify the word with 100% accuracy. However, due to inter-subject variance, previous approaches have not been able to approach this theoretical performance. By being able to utilize very high order factors, SPC overcomes this limitation.

To gain intuition about the inference process of SPC, a detailed picture of the complexity of inference for a few representative examples is presented in Figure 4.5 for the fourth-order cascade model. This figure also demonstrates the flexibility of the cascade: although a single threshold is chosen, the max marginals around unambiguous portions of the input are eliminated first.

4.6 Summary

We presented Structured Prediction Cascades, a framework for adaptively increasing the complexity of structured models on a per-example basis while maintaining efficiency of inference. This allows for the construction and training of structured models of far greater complexity than was previously possible. We proposed two novel loss functions, filtering

Model Order	**1**	**2**	**3**	**4**	**5**	**6**
Accuracy, Char. (%)	77.35	85.02	96.20	97.21	98.27	98.54
Accuracy, Word (%)	26.74	45.67	88.25	91.35	93.74	96.16
Filter Loss (%)	—	0.50	0.73	1.00	0.75	0.57
Tolerance (%)	1.00	1.00	1.00	1.00	0.50	0.25
Avg. Num n-grams	26.0	127.97	101.84	18.80	82.12	73.36

Table 4.3: Summary of handwriting recognition results. For each level of the cascade, we computed prediction accuracy (at character and word levels) using a standard voting perceptron algorithm as well as the filtering loss and average number of unfiltered n-grams per position for the SPC on the test set.

loss and efficiency loss, that measure the two objectives balanced by the cascade, and provided generalization bounds for these loss functions. We proposed a simple sub-gradient based learning algorithm to minimize these losses, and presented a stage-wise learning algorithm for the entire cascade in Algorithm 2. Finally, we demonstrated empirically that SPC is an effective method for obtaining better accuracy/computation trade-offs on two sequential prediction tasks.

Chapter 5

Ensemble-SPC: SPC for Loopy Graphs

Overview. In the previous chapter, we developed the SPC framework by thresholding max-marginals to filter the output space and speed up sparse inference. Computing max-marginals is therefore the critical component of SPC inference (Algorithm 1). However, as discussed in Chapter 3, there are loopy factor graph structures where it is not feasible to run exact inference, even with only pairwise factors. One example is tracking of human pose in video; while the model for pose in a single frame is tree-structured and tractable (Figure 4.2b), adding dependencies between each limb in neighboring frames introduces loops that make inference intractable (Figure 7.1a).

In this chapter, we propose a novel method for learning structured cascades when inference is intractable due to loops in the graphical structure. The key idea of our approach is to decompose the loopy model into a collection of equivalent tractable sub-models for which inference is tractable; we can then sum the max-marginals for each model to produce an approximate max-marginal that still allows for generalization analysis as in the previous chapter. What distinguishes our approach from other decomposition based methods e.g., Bertsekas (1999); Komodakis et al. (2007)) is that, because the cascade's objective is filtering and not decoding, our approach does not require enforcing the constraint that the sub-models agree on which output has maximum score. This makes our method far more efficient than other decomposition based approximate inference techniques. In preliminary

work (Weiss et al., 2010), this approach was called *structured ensemble cascades*, here we simply refer to it as Ensemble-SPC.

Contributions. We first introduce the Ensemble-SPC framework: decomposition of an intractable model into a set of sub-models that cover each factor in the original graph. We define a novel loss function based on the sum of max-marginals from each constituent component, and provide generalization bounds on the filtering error of an Ensemble-SPC cascade. We first evaluate our approach on a synthetic grid model to establish the utility of the decomposition approach, and then apply our approach to a coarse-to-fine cascade for the human pose tracking problem. On the tracking problem, our approach obtains a significant improvement over state-of-the-art.

5.1 Decomposition without agreement constraints

Thus far, we have assumed that (sparse) inference is feasible, so that max marginals can be computed. In this section, we describe how to apply SPC when exact max-sum message passing is computationally infeasible due to loops in the graph structure of the model. In order to simplify the presentation in this section, we will assume that the structured cascade under consideration operates in a "node-centric" coarse-to-fine manner as follows: For each variable y_j in the model, each level of the cascade filters a current set of possible states $\mathcal{Y}_j$, and any surviving states are passed forward to the next level of the cascade by substituting each state with its set of descendants in a hierarchy. For example, such hierarchies arise in pose estimation (Section 6.3.2) by discretizing the articulation of joints at multiple resolutions, or in image segmentation due to the semantic relationship between class labels (e.g., "grass" and "tree" can be grouped as "plants," "horse" and "cow" can be grouped as "animal.") Thus, in the pose estimation problem, surviving states are subdivided into multiple finer-resolution states; in the image segmentation problem, broader object classes are split into their constituent classes for the next level.

Given a loopy (intractable) graphical model, it is always possible to express the score

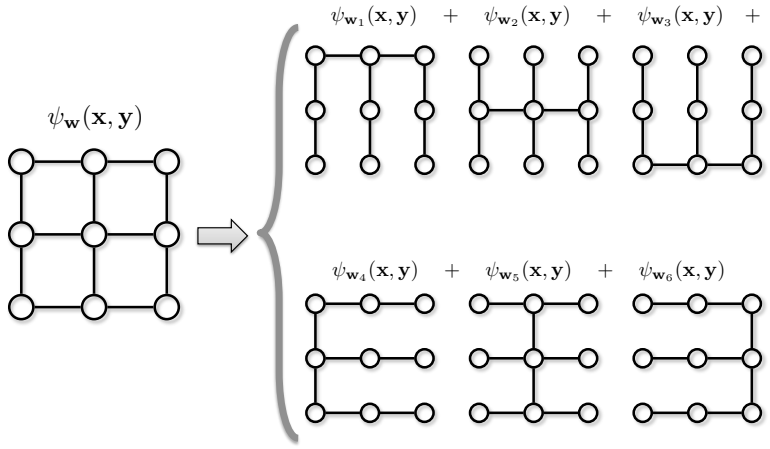


Figure 5.1: Example decomposition of a 3×3 fully connected grid into all six constituent "comb" trees. In general, a $n \times n$ grid yields $2n$ such trees.

of a given output $\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y})$ as the sum of P scores $\psi_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y})$ under sub-models that collectively cover every edge in the loopy model: $\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) = \sum_p \psi_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y})$ (Figure 5.1). However, it is *not* the case that optimizing each individual sub-model separately will yield the single globally optimum solution. Instead, care must be taken to enforce agreement between sub-models. For example, in the method of dual decomposition (Komodakis et al., 2007), it is possible to solve a relaxed MAP problem in the (intractable) full model by running inference in the (tractable) sub-models under the constraint that *all sub-models agree on the argmax solution.* Enforcing this constraint requires iteratively re-weighting unary potentials of the sub-models and repeatedly re-running inference until each sub-model convergences to the same argmax solution.

However, for the purposes of SPC, we are only interested in computing the max-marginals $\psi^{\star}_{\mathbf{w}}(\mathbf{x}, y_j)$. In other words, we are only interested in knowing whether or not a configuration $\mathbf{y}$ consistent with y_j that scores highly in each sub-model $\psi_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y})$ *exists.* We show in the remainder of this section that the requirement that a *single* $\mathbf{y}$ consistent with y_j optimizes the score of each submodel (i.e, that all sub-models *agree*) is not necessary for the purposes of filtering. Thus, because we do not have to enforce agreement between sub-models, we can apply SPC to intractable (loopy) models, but pay only a linear (factor

of P) increase in inference time over the tractable sub-models.

Formally, we define a single level of the Ensemble-SPC as a set of P models such that $\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) = \sum_p \psi_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y})$. We let $\psi^\star_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y}_c)$, $\psi^\star_{\mathbf{w}_p}(\mathbf{x})$ and $\tau_{\mathbf{w}_p,\alpha}(\mathbf{x})$ denote the max-marginals, max score, and threshold of the p'th model, respectively. Recall that the *argmax-marginal* or *witness* $\mathbf{y}^\star_{\mathbf{w}_p}(\mathbf{x}, y_j)$ is defined as the maximizing complete assignment of the corresponding max-marginal $\psi^\star_{\mathbf{w}_p}(\mathbf{x}, y_j)$. Then we have that

$$\psi^\star_{\mathbf{w}}(\mathbf{x}, y_j) = \sum_p \psi^\star_{\mathbf{w}_p}(\mathbf{x}, y_j) \qquad \text{(with agreement: } \mathbf{y} = \mathbf{y}^\star_{\mathbf{w}_p}(\mathbf{x}, y_j), \forall p) \tag{5.1}$$

$$\psi^\star_{\mathbf{w}}(\mathbf{x}, y_j) \leq \sum_p \psi^\star_{\mathbf{w}_p}(\mathbf{x}, y_j) \qquad \text{(in general)} \tag{5.2}$$

Note that if we do not require the sub-models to agree, then $\psi^\star_{\mathbf{w}}(\mathbf{x}, y_j)$ is strictly less than $\sum_p \psi^\star_{\mathbf{w}_p}(\mathbf{x}, y_j)$. Nonetheless, as we show next, the approximation $\psi^\star_{\mathbf{w}}(\mathbf{x}, y_j) \approx \sum_p \psi^\star_{\mathbf{w}_p}(\mathbf{x}, y_j)$ is still useful and sufficient for filtering in a structured cascade.

5.2 Safe filtering

We now show that if a given label y has a high score in the full model, it must also have a large ensemble max-marginal score, even if the sub-models do not agree on the argmax. This extends Lemma 1 for the ensemble case, as follows:

Lemma 3 (Joint Safe Filtering)**.** *If* $\sum_p \psi_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y}) > t$*, then* $\sum_p \psi^\star_{\mathbf{w}_p}(\mathbf{x}, y_j) > t$ *for all* j*.*

Proof. In English, this lemma states that if the global score is above a given threshold, then the sum of sub-model max-marginals is also above threshold (with no agreement constraint). The proof is straightforward. For any y_j consistent with $\mathbf{y}$, we have $\psi^\star_{\mathbf{w}_p}(\mathbf{x}, y_j) \geq \psi_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y})$. Therefore $\sum_p \psi^\star_{\mathbf{w}_p}(\mathbf{x}, y_j) \geq \sum_p \psi_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y}) > t$. □

Therefore, we see that an agreement constraint is not necessary in order to filter safely: if we ensure that the combined score $\sum_p \psi_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y})$ of the true label y is above threshold, then we can filter without making a mistake if we compute max-marginals by running inference separately for each sub-model. However, there is still potentially a price to pay

for disagreement. If the sub-models do not agree, *and* the truth is not above threshold, then the threshold may filter *all* of the states for a given variable y_j and therefore "break" the cascade. This results from the fact that without agreement, there is no single argmax output $\mathbf{y}^\star$ that is always above threshold for any α; therefore, we do not have an equivalent to Lemma 2 for the ensemble case. However, we note that in our experiments (Section 5.5.2), we never experienced such breakdown of the cascades.

5.3 Learning with ensembles

It is straightforward to adapt Algorithm 2 for the Ensemble-SPC case. As in the previous chapter, we first define the natural loss function for sums of max-marginals, as suggested by Lemma 3. We define the *joint filtering loss* as follows,

Definition 4 (Joint Filtering Loss)**.**

$$\mathcal{L}_{joint}(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) = \mathbf{1}\left[\sum_p \psi_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y}) \leq \sum_p \tau_{\mathbf{w}_p,\alpha}(\mathbf{x})\right]. \tag{5.3}$$

We now discuss how to minimize the joint filter loss (5.3) given a dataset. We rephrase the SPC optimization problem (4.8) using the ensemble max-marginals to form the ensemble cascade margin problem,

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_P,\xi\geq 0} \frac{\lambda}{2}\sum_p ||\mathbf{w}_p||^2 + \frac{1}{n}\sum_i \xi^i \quad \text{s.t.} \quad \sum_p \psi_{\mathbf{w}_p}(\mathbf{x}^i, \mathbf{y}^i) \geq \sum_p \tau_{\mathbf{w}_p,\alpha}(\mathbf{x}^i) + \ell^i - \xi^i. \tag{5.4}$$

Seeing that the constraints can be ordered to show $\xi^i \leq \sum_p \tau_{\mathbf{w}_p,\alpha}(\mathbf{x}^i) - \sum_p \psi_{\mathbf{w}_p}(\mathbf{x}^i, \mathbf{y}^i) + \ell^i$, we can form an equivalent unconstrained minimization problem,

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_P} \frac{\lambda}{2}\sum_p ||\mathbf{w}_p||^2 + \frac{1}{n}\sum_i \left[\sum_p \tau_{\mathbf{w}_p,\alpha}(\mathbf{x}^i) - \psi_{\mathbf{w}_p}(\mathbf{x}^i, \mathbf{y}^i) + \ell^i\right]_+, \tag{5.5}$$

where $[z]_+ = \max\{z, 0\}$. Finally, we take the subgradient of the objective in (5.5) with respect to each parameter $\mathbf{w}_p$. This yields the following update rule for the p'th model:

$$\mathbf{w}_p \leftarrow (1-\lambda)\mathbf{w}_p + \begin{cases} 0 & \text{if} \quad \sum_p \psi_{\mathbf{w}_p}(\mathbf{x}^i, \mathbf{y}^i) \geq \sum_p \tau_{\mathbf{w}_p,\alpha}(\mathbf{x}^i) + \ell^i, \\ \nabla\psi_{\mathbf{w}_p}(\mathbf{x}^i, \mathbf{y}^i) - \nabla\tau_{\mathbf{w}_p,\alpha}(\mathbf{x}^i) & \text{otherwise.} \end{cases} \tag{5.6}$$

This update is identical to the original SPC update with the exception that we update each model individually only when the ensemble has made a mistake *jointly*. Thus, learning to filter with the ensemble requires only P times as many resources as learning to filter with any of the models individually. We simply replace the optimization over (4.8) step in Algorithm 2 with an optimization over (5.5).

5.4 Generalization analysis

We now turn to ensemble setting and define an appropriate margin-augmented loss:

Definition 5 (Ensemble margin-augmented loss)**.**

$$\mathcal{L}^{\gamma}_{joint}(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) \quad = \quad r_{\gamma}\left(\sum_{p} \psi_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y}) - \tau_{\mathbf{w}_p, \alpha}(\mathbf{x})\right) \tag{5.7}$$

Theorem 2. *Fix $\alpha \in [0, 1]$ and let $||\mathbf{w}_p||_2 \leq B/P$ for all p, and $||\mathbf{f}_c(\mathbf{x}, \mathbf{y}_c)||_2 \leq 1$ for all $\mathbf{x}$ and $\mathbf{y}_c$. Then there exists a constant c such that for any integer n and any $0 < \delta < 1$ with probability $1 - \delta$ over samples of size n, every $\mathbf{w} = \{\mathbf{w}_1, \ldots, \mathbf{w}_P\}$ satisfies:*

$$\mathbb{E}\left[\mathcal{L}_{joint}(X, Y; \mathbf{w}, \alpha)\right] \leq \hat{\mathbb{E}}\left[\mathcal{L}^{\gamma}_{joint}(X, Y; \mathbf{w}, \alpha)\right] + \frac{cmBP\sqrt{|\mathcal{F}|}}{\gamma\sqrt{n}} + \sqrt{\frac{8\ln(2/\delta)}{n}}, \tag{5.8}$$

where $\hat{\mathbb{E}}$ is the empirical expectation with respect to the sample.

The proof of Theorem 2 is given in Appendix A.1.

5.5 Experiments

We evaluated Ensemble-SPC in two experiments. First, we analyzed the "best-case" filtering performance of the summed max-marginal approximation to the true marginals on a synthetic image segmentation task, assuming the true scoring function $\psi_{\mathbf{w}}(\mathbf{x}, y)$ is available for inference. Second, we evaluated the real-world accuracy of our approach on a difficult, real-world human pose dataset (VideoPose). In both experiments, the max-marginal ensemble outperforms state-of-the-art baselines.

5.5.1 Synthetic loopy graphs with Ensemble-SPC

Experimental setup. We first evaluated the filtering accuracy of the max-marginal ensemble on a synthetic 8-class segmentation task. For this experiment, we removed variability due to parameter estimation and focused our analysis on accuracy of inference. We compared our approach to Loopy Belief Propagation (Loopy BP) (McEliece et al., 1998; Murphy et al., 1999; Pearl, 1988), on a 11×11 two-dimensional grid MRF.[1] For the ensemble, we used 22 unique "comb" tree structures to approximate the full grid model. To generate a synthetic instance, we generated unary potentials $\omega_i(k)$ uniformly on $[0, 1]$ and pairwise potentials log-uniformly: $\omega_{ij}(k, k') = e^{-v}$, where $v \sim \mathcal{U}[-25, 25]$ was sampled independently for every edge and every pair of classes. (Note that for the ensemble, we normalized unary and edge potentials by dividing by the number of times that each potential was included in any model.) It is well known that inference for such grid MRFs is generally difficult (Koller and Friedman, 2009), and we observed that Loopy BP failed to converge for at least a few variables on most examples we generated.

Ensemble outperforms Loopy BP. We evaluated our approach on 100 synthetic grid MRF instances. For each instance, we computed the accuracy of filtering using marginals from Loopy BP, the ensemble, and each individual sub-model. We determined error rates by counting the number of times "ground truth" was incorrectly filtered if the top K states were kept for each variable, where we sampled 1000 "ground truth" examples from the true joint distribution using Gibbs sampling. To obtain a good estimate of the true marginals, we restarted the chain for each sample and allowed 1000 iterations of mixing time. The result is presented in Figure 5.3 for all possible values of K (filter aggressiveness.) We found that the ensemble outperformed Loopy BP and the individual sub-models by a significant margin for all K.

[1] We used the UGM Matlab Toolbox by Mark Schmidt for the Loopy BP and Gibbs MCMC comparisons, see: `http://people.cs.ubc.ca/~schmidtm/Software/UGM.html`.

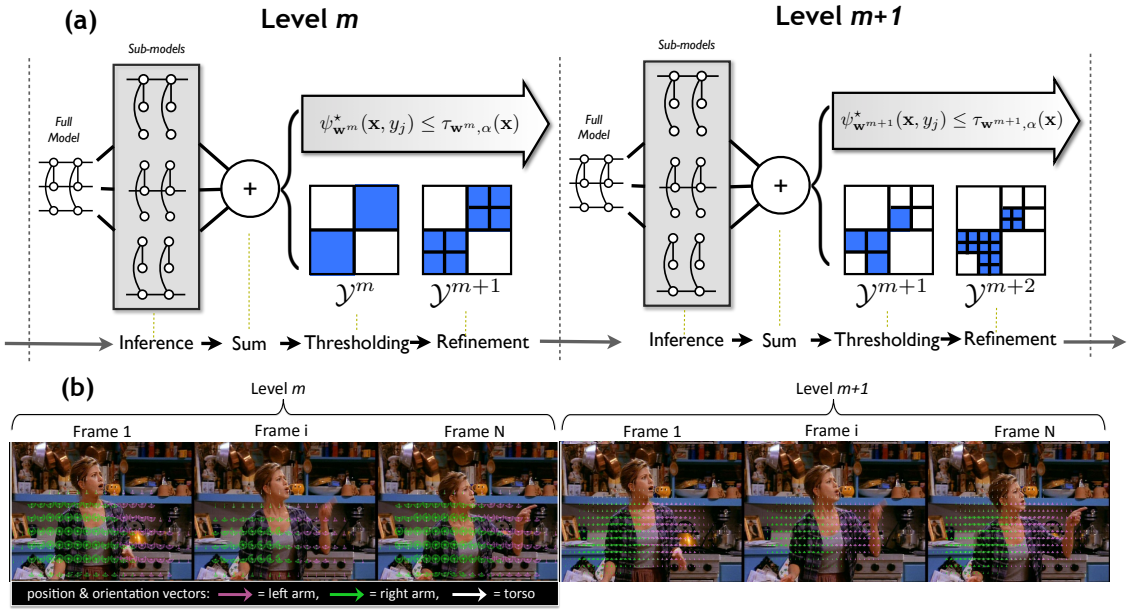


Figure 5.2: **(a)** Schematic overview Ensemble-SPC for human pose tracking. The m'th level of the cascade takes as input a sparse set of states $\mathcal{Y}^m$ for each variable y_j. The full model is decomposed into constituent sub-models (above, the three tree models used in the pose tracking experiment) and sparse inference is run. Next, the max marginals of the sub-models are summed to produce a single max marginal for each variable assignment: $\psi^\star_{\mathbf{w}}(\mathbf{x}, y_j) = \sum_p \psi^\star_{\mathbf{w}_p}(\mathbf{x}, y_j)$. Note that each level and each constituent model will have different parameters as a result of the learning process. Finally, the state spaces are thresholded based on the max-marginal scores and low-scoring states are filtered. Each state is then refined according to a state hierarchy (e.g., spatial resolution, or semantic categories) and passed to the next level of the cascade. This process can be repeated as many times as desired. In **(b)**, we illustrate two consecutive levels of the ensemble cascade on real data, showing the filtered hypotheses left for a single video example.

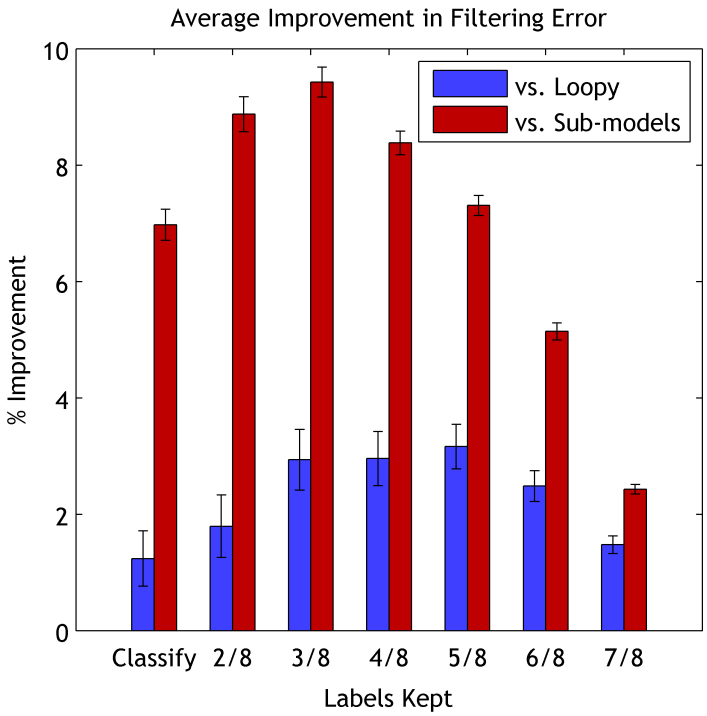


Figure 5.3: Improvement over Loopy BP and constituent tree-models on the synthetic segmentation task. Error bars show standard error.

Effect of sub-model agreement. We next investigated the question of whether or not the ensembles were most accurate on variables for which the sub-models tended to agree. For each variable y_{ij} in each instance, we computed the mean pairwise Spearman correlation between the ranking of the 8 classes induced by the max marginals of each of the 22 sub-models. We found that complete agreement between all sub-models never occurred (the median correlation was 0.38). We found that sub-model agreement was significantly correlated ($p < 10^{-15}$) with the error of the ensemble for all values of K, peaking at $\rho = -0.143$ at $K = 5$. Thus, increased agreement predicted a decrease in error of the ensemble. We then asked the question: Does the effect of model agreement explain the *improvement* of the ensemble over Loopy BP? In fact, the improvement in error compared to Loopy BP was *not* correlated with sub-model agreement for any K (maximum $\rho = 0.0185$, $p < 0.05$). Thus, sub-model agreement does *not* explain the improvement over Loopy BP, indicating that sub-model disagreement is not related to the difficulty in inference problems that causes Loopy BP to under perform relative to the ensembles (e.g., due to convergence failure.)

5.5.2 Articulated pose tracking cascade

Experimental setup. The VideoPose dataset[2] consists of 34 video clips of approximately 50 frames each. The clips were harvested from three popular TV shows: 3 from *Buffy the Vampire Slayer*, 27 from *Friends*, and 4 from *LOST*. Clips were chosen to highlight a variety of situations and and movements when the camera is largely focused on a single actor. In our experiments, we use the *Buffy* and half of the *Friends* clips as training (17 clips), and the remaining *Friends* and *LOST* clips for testing. In total we test on 901 individual frames. The *Friends* are split so no clips from the same episode are used for both training and testing. We further set aside 4 of the *Friends* test clips to use as a development set. Each frame of each clip is hand-annotated with locations of joints of a full pose model; for simplicity, we use only the torso and upper arm annotations in this work, as these have the strongest continuity across frames and strong geometric relationships.

[2]The VideoPose dataset is available online at `http://vision.grasp.upenn.edu/video/`.

Figure 5.4: Qualitative test results. Points shown are the position of left/right shoulders and torsos at the last level of the ensemble SC (blue square, green dot, white circle resp.). Also shown (green line segments) are the best-fitting hypotheses to groundtruth joints, selected from within the top 4 max-marginal values. Shown as dotted gray lines is the best guess pose returned by Ferrari et al. (2008).

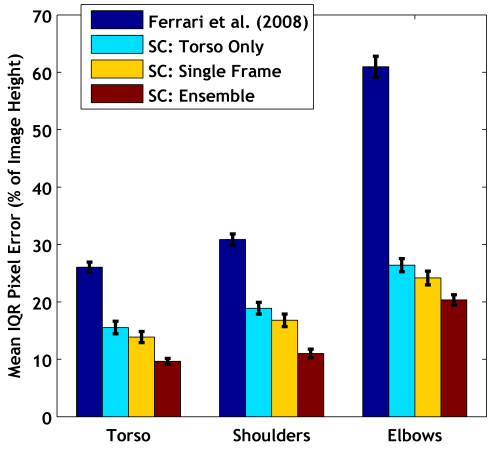
Articulated pose model. All of the models we evaluated on this dataset share the same basic structure: a variable for each limb's (x, y) location and angle rotation (torso, left arm, and right arm) with edges between torso and arms to model pose geometry. We refer to this basic model, evaluated independently on each frame, as the "Single Frame" approach. For the VideoPose dataset, we augmented this model by adding edges between limb states in adjacent frames (Figure 7.1), forming an intractable, loopy model. Our features in a single frame are the same as in the beginning levels of the pictorial structure cascade (Section 6.3.2): unary features are discretized Histogram of Gradient (HoG) part detectors scores, and pairwise terms measure relative displacement in location and angle between neighboring parts. Pairwise features connecting limbs across time also express geometric displacement, allowing our model to capture the fact that human limbs move smoothly over time.

Coarse-to-Fine Ensemble. We learned a coarse-to-fine structured cascade with six levels for tracking as follows. The six levels use increasingly finer state spaces for joint locations, discretized into bins of resolution 10×10 up to 80×80, with each stage doubling one of the state space dimensions in the refinement step. All levels use an angular discretization of 24 bins. For the ensemble cascade, we learned three sub-models simultaneously (Figure 7.1), with each sub-model accounting for temporal consistency for a different limb by adding edges connecting the same limb in consecutive frames.

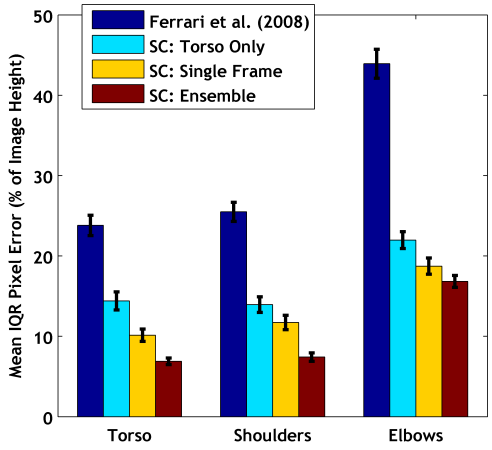
Results. A summary of results are presented in Figure 5.5. We compared the single-frame cascade and the ensemble cascade to a state-of-the-art single-frame pose detector (Ferrari et al. (Ferrari et al., 2008)) and to one of the individual sub-models, modeling torso consistency only ("Torso Only"). We evaluated the method from Ferrari et al. (2008) on only the first half of the test data due to computation time (taking approximately 7 minutes/frame). We found that the ensemble cascade was the most accurate for every joint in the model, that all cascades outperformed the state-of-the-art baseline, and, interestingly, that the single-frame cascade outperformed the torso-only cascade. We suspect that the poor performance of the torso-only model may arise because propagating only torso states through time leads to an over-reliance on the relatively weak torso signal to determine the location of all the limbs. Sample qualitative output from the ensemble is presented in Figure 5.4.

5.6 Summary

In this chapter, we presented Ensemble-SPC, an extension of the SPC framework to allow for coarse-to-fine inference in loopy factor graphs. Like other approximate inference methods, we decompose the loopy graph into an ensemble of tractable sub-graphs; however, our method only requires agreement with respect to a single variable at a time. We also extended the generalization analysis from Chapter 4 to provide a generalization bound on the filtering error of the ensemble. Finally, we showed that the method is effective on both synthetic and real world data.



(a) Decoding Error.



(b) Top $K = 4$ Error.

Level	*State Dimensions*	$PCP_{0.25}$ *in top K=4*	*Efficiency* (%)
0	$10 \times 10 \times 24$	–	–
2	$20 \times 20 \times 24$	98.8	87.5
4	$40 \times 40 \times 24$	93.8	96.9
6	$80 \times 80 \times 24$	84.6	99.2

(c) Ensemble efficiency.

Figure 5.5: **(a)**,**(b)**: Prediction error for VideoPose dataset. Reported errors are the average distance from a predicted joint location to the true joint for frames that lie in the [25,75] inter-quartile range (IQR) of errors. Error bars show standard errors computed with respect to clips. All SC models outperform Ferrari et al. (2008); the "torso only" persistence cascade introduces additional error compared to a single-frame cascade, but adding arm dependencies in the ensemble yields the best performance. **(c)**: Summary of test set filtering efficiency and accuracy for the ensemble cascade. $PCP_{0.25}$ measures Oracle % of correctly matched limb locations given unfiltered states; see Sapp et al. (2010) for more details.

Chapter 6

Dynamic Structured Model Selection (DMS)

Overview. In this chapter, we develop an alternative framework for addressing the expressiveness v. computation trade-off. Whereas the SPC framework was developed in order to allow for sparse inference in very large state spaces, this new framework is developed in order to dynamically control the cost of feature extraction at test-time. Specifically, we study the problem of managing the cost of low-level vision processing: dense evaluation of input features common in visual processing (e.g., normalized cut segmentation, optical flow, contour detection, etc.). In many object detection systems, the cost of computing these features is several times greater then the time spent on structured inference given the features. Furthermore, there is a tremendous variation of cost of low-level processing based on resolution and other accuracy parameters; thus, choosing one global setting is often not sufficient for complex images and wasteful on simple ones.

The basic principle behind this framework is very simple: at test-time, we selectively apply different models to different examples. Each model utilizes its own set of features and represents a different point on the expressiveness-computation trade-off curve. We treat each model as a black box that takes a fixed cost to run for each example. To simplify the algorithm, we assume a fixed ordering and hierarchy of models; the first (baseline)

model is the cheapest and least expressive, and subsequent models increase the predictive power at the price of paying an additional cost. Our goal is as follows: given a batch of test examples, allocate models to examples to maximize accuracy within an overall budget constraint. We call our approach *Dynamic Structured Model Selection* (DMS).

We propose a novel two-tier architecture that provides dynamic speed v. accuracy trade-offs through a simple type of *introspection*. The key idea is a division of labor between a hierarchy of models/inference algorithms (tier one) and meta-level model selector (tier two), which decides when to use expensive models adaptively, where they are most likely to improve the accuracy of predictions. The two tiers have complementary strengths: Tier one models provide increasingly accurate and more expensive inference over structured outputs. Tier two model-selectors use arbitrary sparsely-computed features and long-range dependencies, which would make inference intractable, in order to evaluate the outputs of the first tier and decide when to stop. While the first tier optimizes over a combinatorial set of possibilities using inference over densely computed features, the second tier simply evaluates proposals of the first. The advantage of this division is that both tiers are efficient and the second tier has more information than the first that allows it to reason about the success of the first.

Contributions. We introduce dynamic structured model selection (DMS), a novel two-tier framework for creating faster and more accurate structured prediction systems. In section 6.1, we propose a simple greedy algorithm for maximizing test-time accuracy given a budgetary constraint. We apply our approach to two sequential modeling tasks: handwriting recognition (section 6.3.1) and articulated pose estimation in videos (section 6.3.2). On the handwriting recognition task, we use DMS to achieve a significant increase in accuracy over baseline while at the same time being nearly $3\times$ faster. On the pose task, we propose a novel sequence model based re-ranker that utilizes the recently introduced model of Sapp and Taskar (2013) to achieve state-of-the-art accuracy on a benchmark dataset while being $23\times$ faster than the previous best method. We then apply DMS to achieve even faster times on a new, much larger benchmark dataset, reducing the re-ranking model runtime by

Algorithm 3: Dynamic structured model selection (DMS).

input : Test set $\{\mathbf{x}^j\}_1^n$, hypotheses $h_1, \ldots, h_T$, costs $c_1, \ldots, c_T$, selector ν, and budget B.

output: Predictions $\mathbf{y}^1, \ldots, \mathbf{y}^n$.

initialize $B' \leftarrow 0$, $\tau_j \leftarrow 1$, $\mathbf{y}^j \leftarrow h_1(\mathbf{x}^j)$;

initialize priority queue Q with priority-value pairs $\langle \nu(h_2, \mathbf{x}^j), j \rangle$;

while $B' < B$ *and* Q *is not empty* **do**

    Pop value j from Q with max priority;

    if $c_{\tau_j+1} \leq (B - B')$ **then**

        $\tau_j \leftarrow \tau_j + 1$;

        $B' \leftarrow B' + c_{\tau_j}$;

        $\mathbf{y}^j \leftarrow h_{\tau_j}(\mathbf{x})$;

        Insert $\left\langle \nu(h_{\tau_j+1}, \mathbf{x}^j), j \right\rangle$ into Q;

a factor of $2\times$ with no decrease in accuracy for wrist localization.

6.1 Meta-learning with a value-based selector

In this section, we introduce our approach to dynamic model selection and provide an overview of the algorithm. The core idea behind our approach is very simple: we learn to predict the *value* of choosing a more expensive model over a cheaper one using introspective *meta-features*, and we use these predicted values to allocate computational resources at test time. In this fashion, we only apply the computationally expensive models to those examples where we will receive the most benefit.

Value of a model. We work within the framework presented in Chapter 3 for structured prediction. For the dynamic model selection problem we consider here, we assume that we are given a *set* of models, $h_1, \ldots, h_T$, that require some amortized cost $c_1, \ldots, c_T$ to

evaluate on any given example $\mathbf{x}$. Given a fixed ordering of the models, we define the *value* of evaluating model h_i on example $\mathbf{x}$,

$$V(h_i, \mathbf{x}, \mathbf{y}) = \mathcal{L}(h_{i-1}(\mathbf{x}), \mathbf{y}) - \mathcal{L}(h_i(\mathbf{x}), \mathbf{y}), \tag{6.1}$$

where $\mathcal{L}(\mathbf{y}, \mathbf{y}')$ is a non-negative loss function. Note that a positive value signifies a decrease in loss, while a negative value signifies an increase is loss. (While more expensive models usually increase accuracy on *average*, in practice we find that there are many examples where the more expensive features hurt performance.) Our proposed goal for meta-learning is to learn a *selector* $\nu(h_i, \mathbf{x})$ to approximate the value function.

Batch inference. Given a set of models $h_1, \ldots, h_T$, a selector ν, and a test set $\{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$, we perform inference using Algorithm 3. The algorithm is very simple: we greedily optimize the total predicted value,

$$J(\tau_1, \ldots, \tau_n, \eta) = \sum_{j=1}^{n} \sum_{i=1}^{\tau_j} \nu(h_i, \mathbf{x}^j), \tag{6.2}$$

where τ_j is the stopping point on the j'th example. This can be implemented easily using a single priority queue. The value for selecting the next model for each example is put in the queue, and we allocate resources by repeatedly extracting the highest value element from the queue and computing the next model's predictions to update the value. Note that even if all predicted $\nu(h_i, \mathbf{x}^j)$ are negative, Algorithm 3 continues to greedily choose more expensive models as long as budget is available.

6.2 Learning the models and selector

Learning the models. Although we treat each model as a black box, in this work we assume the same linear hypothesis class used throughout this thesis (1.1). Each model h_i has an associated weight vector $\mathbf{w}_i$ and feature function $\mathbf{f}_i$. We use the structured max margin learning framework as outlined in Chapter 3 to learn each model independently.

In particular, for the handwriting recognition task, we approximately optimize (3.4) using the structured perceptron algorithm, which has been shown to work well for this task (Weiss and Taskar, 2010). For the video pose estimation task, we optimize (3.4) directly using the recent stochastic Frank-Wolfe block-coordinate descent method of Lacoste-Julien et al. (2013), which we found to be more robust. In both cases, we choose regularization parameter λ and a stochastic stopping time through cross-validation using a development set.

Learning the selector. In order for Algorithm 3 to succeed, the selector ν must provide a useful estimate of the value V. We formulate the selector as a linear function of *meta-features* computed on the output of the models. The key idea is that, while the feature generating function $\mathbf{f}$ for a structured prediction model decomposes over subsets of $\mathbf{y}$ in order to maintain feasible inference, the meta-features ϕ need only be computed efficiently for the specific outputs $h_1(\mathbf{x})$ through $h_{i-1}(\mathbf{x})$. We provide detail on the specific meta-features used in each application in section 6.3.

We learn the selector by learning a weight vector β to approximate the value function. On the training set, we first compute the value for every model on every training example. We then minimize the following ℓ_2-regularized squared loss over a training set,

$$\frac{\lambda}{2}||\beta||_2^2 + \sum_{i=1}^{m}\sum_{j=1}^{n}\left(V(h_i, \mathbf{x}^j, \mathbf{y}^j) - \beta^\top \phi(\mathbf{x}^j, h_{1:i-1})\right)^2 \tag{6.3}$$

where the function ϕ is a function generating *meta-features* that takes all predictions h_1, $\ldots, h_{i-1}$ as input and λ is a regularization parameter chosen via cross validation.

Preventing overfitting. Some care is needed when learning the selector in order to avoid re-using the same training set for learning both the models and the selector; i.e. if h_i was trained on example $(\mathbf{x}^j, \mathbf{y}^j)$, we expect $\mathcal{L}(h_i(\mathbf{x}^j), \mathbf{y}^j)$ to be unrealistically low and thus the value may be unrepresentative of the test distribution. However, a simple N-fold cross-validation scheme suffices to prevent this. We train N different models $h_i^1, \ldots, h_i^N$ in the standard way and use the model not trained on example j when evaluating (6.3).

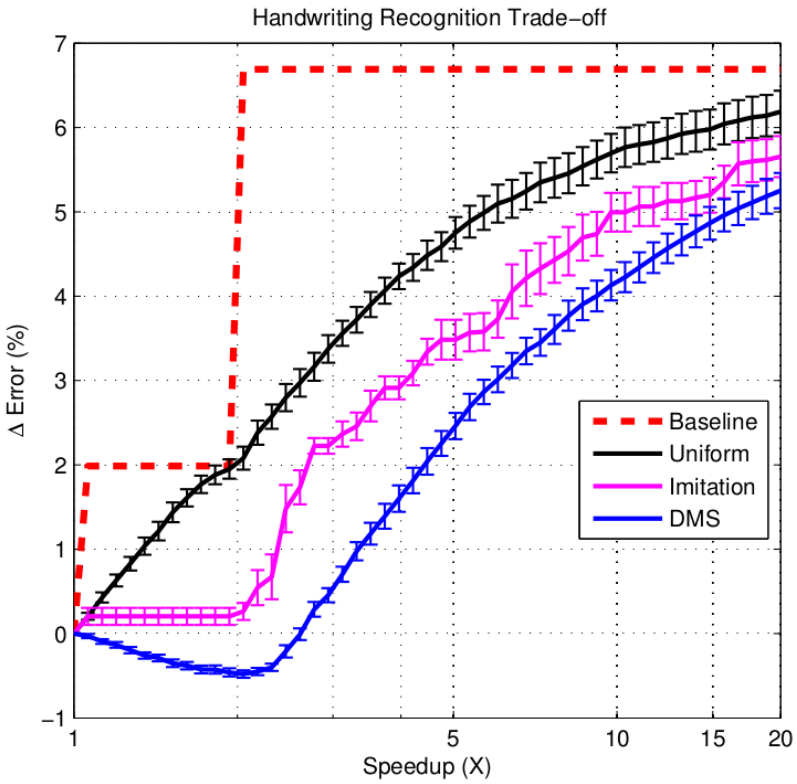


Figure 6.1: Trade-off on handwriting recognition task, displayed as a function of the efficiency speedup w.r.t the final model (Speedup) vs. the change in error rate w.r.t the final model (ΔError). To draw each curve, we sweep the budget B or tradeoff parameter η until we find a point with at least the target speedup and record the error rate. Our approach (DMS) significantly outperforms imitation learning, yielding an error rate *below* that of the final model. The Uniform method consists of picking which element to expand uniformly at random until all examples use the same model, and the Baseline method consists of picking a single entire fixed stage of models *a priori.*

6.3 Application to sequential prediction

In the next sections, we discuss two applications of our dynamic structured model selection framework to computer vision problems: handwriting recognition and human pose estimation from 2D video. In both settings, DMS provides for a far more efficient structured prediction model.

Sequence model. In both settings, we use the standard linear-chain structured prediction model first introduced in Chapter 1. For completeness, we briefly recap the details: Given an input $\mathbf{x}$ of length ℓ, we wish to predict a sequence of discrete K-valued outputs $y_1, \ldots, y_\ell$, where $y_i \in \{1, \ldots, K\}$. For the handwriting recognition problem, each y_i corresponds to one letter of the written word; for human pose estimation, each y_i corresponds to one of K possible predicted poses. For any given sequence $y_1, \ldots, y_\ell$, the combined score of the sequence is the sum of unary and pairwise potentials,

$$\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{\ell} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, y_i) + \sum_{i=2}^{\ell} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, y_{i-1}, y_i), \tag{6.4}$$

where $\mathbf{w}$ is a (learned) weight vector and $\mathbf{f}$ is a feature generating function. At test time, we can efficiently make predictions using the Viterbi algorithm to find the state sequence that maximizes (6.4).

6.3.1 Handwriting recognition

We first apply our method to the handwriting recognition dataset of Taskar et al. (2003). For our purposes, this dataset represents a "best case" type of scenario: while there are thousands of examples of handwritten words in the dataset, examples were generated by enlisting many different people to rewrite the same list of less than a hundred unique words. Therefore we expect high-order features (e.g., 5-grams) to be very informative, but these features are computationally infeasible to include in the linear-chain model directly. Instead, they are ideally suited as informative meta-features for the selector. In this way, the

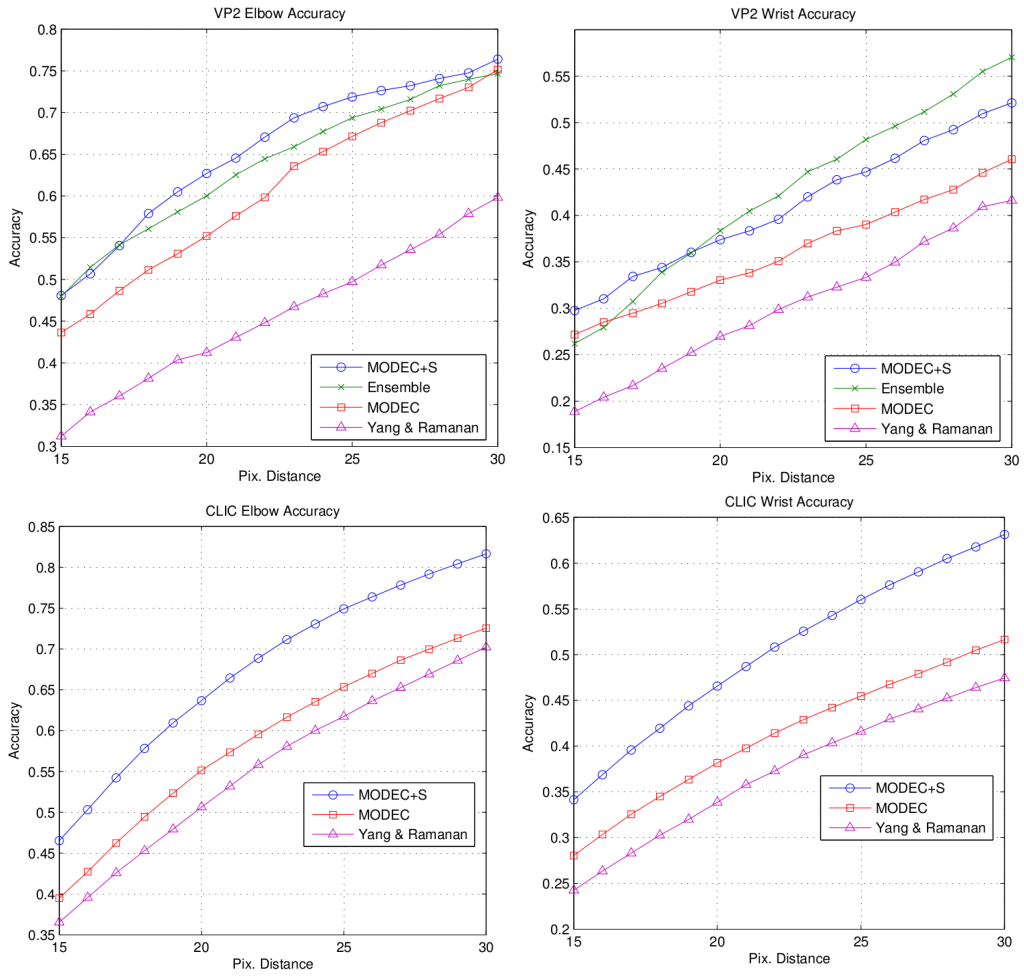


Figure 6.2: Exceeding state-of-the-art on VideoPose2 Sapp et al. (2011) and CLIC datasets. The MODEC+S model matches the much slower ensemble approach of Sapp et al. (2011) (Ensemble) in elbow accuracy and exceeds it in wrist accuracy (at high precisions), and provides a significant boost in performance over MODEC. However, MODEC is still competitive and more accurate than previous state-of-the-art Yang and Ramanan (2011) on both datasets.

selector is ideally suited to direct computation at test time, and a very fast and effective method is the result.

Models. We use three different models for the handwriting recognition problem, differing only in the unary term features of the sequence model. In each model, we have K^2 binary pairwise features $\mathbf{f}_{k,k'}(y_{i-1}, y_i) = \mathbf{1}\left[y_{i-1} = k, y_i = k'\right]$ as well as a unary feature for every binary pixel activation in the 16×8 image. The second model h_2 computes a coarse Histogram of Gradients (HoG) in 3×3 bins and the third model h_3 additionally computes HoG in smaller 2×2 bins. Since the pixels are given as in the input, and HoG takes constant time for fixed input size, we have $c_1 = 0, c_2 = 1, c_3 = 1$.

Selector Meta-features. We use two sets of meta-features for the selector. The first are computed from the output of $h_i(\mathbf{x})$, consisting of the relative difference in the scores of the top two outputs and the average of the mean, min, and max entropies of the marginal distributions predicted by h_i at each position in the sequence[1]. The second set of meta-features count the number of times an n-gram was predicted in $h_i(\mathbf{x})$ that occurred zero times in the training set, computed for $n = 3, 4, 5$. Both of these features take negligible time to compute compared to the HoG computation.

Imitation learning baseline. We compare to an alternative method for dynamic model selection inspired by imitation learning methods for feature selection (He et al., 2012). For this baseline, we first pick a trade-off parameter η, and then for each example $(\mathbf{x}^j, \mathbf{y}^j)$ in the training set independently decide the optimal stopping point,

$$\tau_j^\star = \operatorname*{argmin}_\tau \mathcal{L}(h_\tau(\mathbf{x}^j), \mathbf{y}^j) + \eta \cdot c_\tau. \tag{6.5}$$

These stopping points define an optimal policy $\pi^\star(i, \mathbf{x}^j) = \mathbf{1}\left[\tau_j^\star < i\right]$, where the policy $\pi^\star(i, \mathbf{x}^j)$ is 1 if computation should continue on example $\mathbf{x}^j$ after model i and 0 otherwise.

[1] In addition to Viterbi, we also ran sum-product inference in order to compute probabilistic marginals to obtain entropies

We then learn an approximate policy $\pi(i, \mathbf{x}^j)$ using a linear SVM classifier trained with the same meta-features as the selector uses; we generated training data points by sampling all trajectories generated by the optimal policy on the training set. Note that for each η, we obtain one error/cost trade-off point. For all experiments, we swept η across many values to generate all possible unique optimal policies for each fold of cross validation.

Results. We visualize the trade-off between error rate and computation time on the handwriting recognition task in Figure 7.3. All results are plotted in terms relative to the third, most expensive model. Our approach significantly outperforms imitation learning, and both imitation learning and our approach provide a significant increase in efficiency over choosing one of the models *a priori* or uniformly at random. Most significantly, the model/selector approach leads to a predictor that is more accurate than the final model (due to choosing the most accurate models first) while yielding a roughly $2.5\times$ speedup compared to the most expensive model. Note that we also computed a different uniform baseline where examples were advanced to the next model uniformly at random without ensuring that all examples reached the same stage; we found this performed equivalent or worse than the baseline shown.

Imitation learning vs. DMS. Besides the improvement in accuracy and speedup, there are several practical advantages of DMS over the imitation learning baseline. Unlike DMS, each choice of η yields a different policy; in order to sweep a curve, we must re-run learning for every point we wish to generate on the curve. Furthermore, imitation learning as defined using equation 6.5 does not guarantee that computation over a batch of test examples will run within a fixed budget; each choice of η yields a fixed trade-off that will approximately run at some budget that is a function of the interaction between computation and accuracy on the given training set.

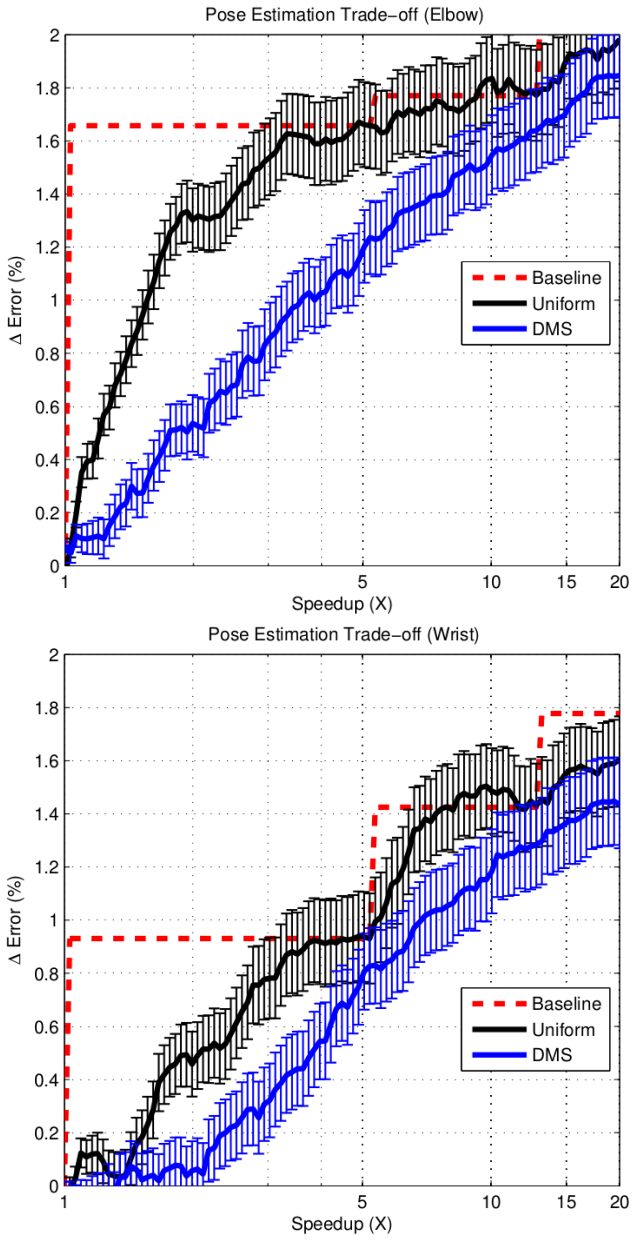


Figure 6.3: Dynamic model selection on the CLIC dataset. See caption of Figure 7.3 for explanation of axes/baselines. DMS provides a significant increase in speedup with very little accuracy cost compared to picking elements uniformly at random; e.g. for elbows, a $2\times$ speedup can be obtained for hardly any accuracy cost, while a $5\times$ speedup with DMS can be obtained for the same accuracy of a $3\times$ speedup when picking uniformly at random.

6.3.2 Human pose estimation in video

Our approach to video pose estimation can be summarized as follows. We propose a simple bigram linear-chain model, one per arm. The model consists of 32 states per frame. We adapt the efficient and current state-of-the-art MODEC pose model Sapp and Taskar (2013) to generate the states: each state corresponds to the highest scoring prediction of one of the 32 MODEC sub-models. Next, given the set of states for each clip in our training database, we learn to predict a path through the states using high level features such as color and flow consistency. The resulting model is both more accurate than previous state-of-the-art and is roughly $23\times$ faster. We then apply dynamic model selection to choose the features in the sequence models on-the-fly for even greater efficiency gains on a new, large dataset.

Related works in pose estimation. There is considerable research into human pose estimation from 2D images; far more than we can review here. However, as state-of-the-art pose estimation can take upwards of several minutes per frame (Ladický et al., 2013; Sapp et al., 2011) there is significantly less prior work on pose estimation in video clips. Buehler et al. (2011) use a single pictorial structure model for upper body in the different setting of extended signing sequences, where e.g. a static background over long periods leads to useful models of background. Park and Ramanan (2011) propose a related approach to our method by stitching together N hypothesized poses per frame into video tracks, using $N = 300$ and evaluating their approach on 4 video sequences. In contrast, we use $N = 32$ proposals from Sapp et al. (2011) (assuming the scale and location of the person is known), learn additional sequence models using features computed over proposed tracks, and evaluate on hundreds of short clips from cinema. Sapp et al. (2011) tackles video clips from TV shows and is therefore the most relevant competitor to our approach, but (as we demonstrate) our method is both more accurate and an order of magnitude faster.

MODEC Proposals. A key relevant modeling innovation in the MODEC method is the joint learning of a mixture of 32 articulated part-based models. Each mixture component, or *mode*, represents a different canonical pose. To generate our 32 states, we find the argmax

arm configuration for each of the 32 modes in MODEC. Note that MODEC models each arm as a separate pose model, but chooses a single mode for each arm based on a combined compatibility score between the two poses; for our purposes, we ignore the compatibility score and take the 32 separate predictions for each arm independently. Experimentally, we find that with 32 states per arm, at least one state is typically very close to the true arm pose for a given image (i.e. on the VideoPose2 dataset of Sapp et al. (2011), greater than 80% of elbows and wrists are within 20 pixels of the ground truth, on average.)

MODEC+S sequence model. Given a video sequence, we generate 32 states for each arm for each frame independently using the MODEC model. The problem then becomes selecting which of the 32 poses for each arm and frame to choose. We apply a standard linear-chain bigram model for this task. Let y_i be state at frame i; for each assignment to y_i we have a corresponding MODEC argmax pose on the i'th frame, which we denote $p_i(y_i)$. For each state y_i in the i'th frame we allow transitions from the 5 closest states y'_{i-1} in the previous frame, as measured by the distance in joint configuration space $||p_i(y_i) - p_{i-1}(y'_{i-1})||_2$. This yields a total of 160 possible transitions for each frame. At test time, we can efficiently make predictions using the Viterbi algorithm to find the state sequence that maximizes (6.4) with practically negligible runtime due to the tiny size of the state and transition space. We call our approach MODEC+S.

Learning. We learn the scoring functions ψ from a set of n training examples as follows. Given a video clip $\mathbf{x}^j$ and a labeled pose p_i^j for each frame i in $\mathbf{x}^j$, we define the ground truth state y_i^j to be the state with the closest pose $y_i^j = \operatorname{argmin} ||p_i(y_i) - p_i^j||_2$. This results in the training set $\{(\mathbf{x}^j, \mathbf{y}^j)\}_{j=1}^n$ which we use as the basis for the rest of our analysis.

Features. As in the handwriting recognition task, we use a fixed hierarchy of features to create a series of four increasingly complex base models. The first model uses unary features consisting of a prior term and the normalized MODEC score for each mode, and binary features consisting of a (mode,mode) transition prior and several kinematic terms

(angular joint and limb velocities and x, y joint location velocities). The second model adds an image-dependent pairwise term, the χ^2-distance between color histograms of the predicted arm locations from one frame to the next. The third model adds an image-dependent unary term; each image is quickly segmented into superpixels using Felzenszwalb and Huttenlocher (2004), and we compute the intersection-over-union (IoU) score between the predicted arm rectangles and superpixels selected by the rectangles. Finally, the fourth model computes a very fast and coarse optical flow using Liu (2009); we obtain an estimate of the *foreground* flow by subtracting the median flow outside the target bounding box. We then compute a flow-based pairwise feature as follows: for each predicted arm location in the first frame, we shift each arm pixel by its estimated flow to produce a predicted arm location in the next frame, and compute the IoU between the flow-shifted arm and each possible predicted arm location in the next frame. Although the computational cost depends on the size of the input images, for the CLIC dataset (described below), we compute the per-frame amortized costs of each model (in seconds)[2] to be $c_1 = 0, c_2 = 0.41$, $c_3 = 1$, and $c_4 = 5.2$ (optical flow is by far the most expensive feature of MODEC+S).

Meta-features. We use similar meta-features as in the handwriting recognition setting (n-gram occurrences and distribution and entropy of the sequence model marginals) with one set of additional image-dependent features. For every n-gram in the image, where $n = 2k$, we compute the mean and max χ^2 distance between a center frame predicted arm location and the k frames before and after. The feature is then the average number of times these distances exceed 0.5, indicating a significant difference between the predicted arm color of the center frame and the surround frames. For $k = 1, 2, 3$, these features yield a total of 40 features for the selector. Finally, for computing metafeatures at model level i, we also compute the features of level $i - 1$ and the change in these features from $i - 1$ to i. The per-frame amortized cost of evaluation is 0.014 seconds.

[2]All computation was carried out on a AMD Opteron 4284 CPU @ 3.00 GHz with 16 cores.

A new dataset. We introduce a new publicly available dataset, Clips Labeled in Cinema (CLIC).[3] This dataset consists of 362 annotated clips from the same movies as the Frames Labeled in Cinema (FLIC) dataset from Sapp and Taskar (2013), for a total of roughly 15,000 individual frames. The annotations were obtained from a Amazon Mechanical Turk crowd-sourcing annotation tool. Due to the difficulty of labeling an entire video sequence, we labeled a significantly larger pool of video clips, but kept only those frames where inter-annotator agreement was within the 90th percentile of the distribution of agreement across all frames in all video clips. Each clip is between 10 and 61 frames in length, with a median clip length of 46 frames. Although all of the data will be publicly available, for our experiments in the following, we selected half of the clips that contained the most arm motion to provide for a more challenging dataset.

6.3.3 Evaluation of MODEC+S

Experimental setup. Before applying DMS, we evaluated the utility of the final, most complex MODEC+S model for articulated pose estimation in video. We first compared to the approach of Sapp et al. (2011) on the VideoPose2 (VP2) dataset (Sapp et al., 2011), which represents state-of-the-art in human pose estimation in challenging videos. (We also compare to state-of-the-art single frame inference, as represented by Yang and Ramanan (2011)). For the VP2 dataset we used the same train/test partitioning of the VP2 dataset as Sapp et al. (2011). We next compared our approach on the new CLIC dataset. Due to excessive runtime, it was infeasible to apply Sapp et al. (2011) to this much larger dataset, so we compared to single-frame methods Sapp and Taskar (2013) and Yang and Ramanan (2011). To evaluate on CLIC, we divided the dataset into two halves so that different movies were contained in different halves. Since our new dataset CLIC uses the same movies as the FLIC dataset in Sapp and Taskar (2013), we re-trained MODEC on the corresponding

[3]This dataset was collected and annotated by collaborators, and not the author of this thesis. The primary contribution of the author was in manually reviewing, fixing errors, and curating a subset of the dataset for use in this work.

movies from FLIC contained in each half and tested MODEC on the other half. To train and test MODEC+S, we used these test evaluations of MODEC as input, again training on one half of the data and testing on the other. Note that for all experiments, we ignored the human detection problem and fed all algorithms pre-localized and scaled images using the annotations in the data, although only MODEC and Sapp et al. (2011) explicitly take advantage of this fact.

Results. The results are summarized in Figure 6.2, and qualitative results are presented in Figure 6.5. On VP2, MODEC+S achieves similar or better accuracies to Sapp et al. (2011), but whereas the downloadable code package for Sapp et al. (2011) took 367 seconds/frame of computation time, MODEC+S takes only 16 seconds/frame, a 22.9$\times$ speedup. In particular, the new model is significantly more accurate on the wrist than either the previous best or the single-frame MODEC model. However, even the single-frame MODEC is close to state-of-the-art (after smoothing), which is even faster than our approach, and Yang and Ramanan (2011) is faster still, though not as competitive. On CLIC, MODEC+S dominates the other methods by an even more significant margin, and all accuracies are generally higher than VP2 for all methods.

6.3.4 Evaluation of DMS for MODEC+S

Experimental setup. We evaluated our dynamic model selection framework on the CLIC dataset with 200 random partitions of the dataset. For each partition, we used 70% of the data for training, 10% as development, and 20% as test. Within the training set, we ran 3-fold cross-validation to generate model predictions for learning the selector as described in section 6.1. When learning the selector, we focused on minimizing wrist test error, counting as an error any frame that the wrist was not localized to within 20 pixels. We also smoothed the predictions of each model before passing them to the selector since we found this improved the overall accuracy of the system.

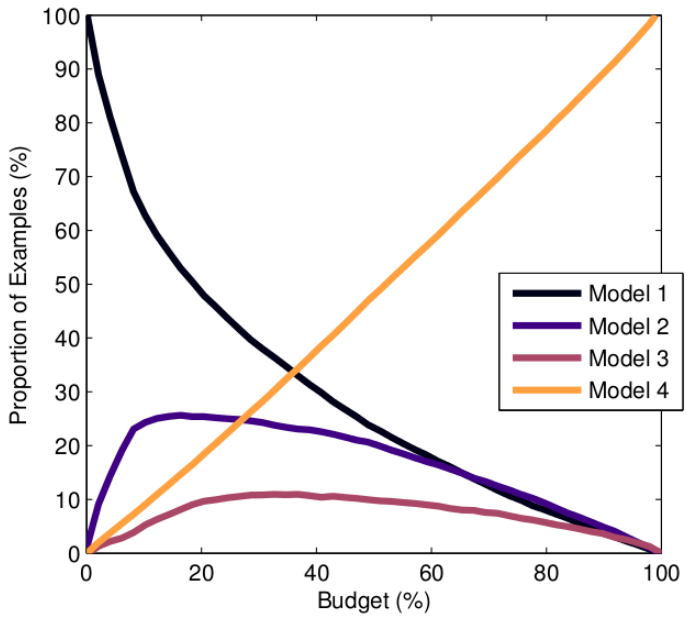


Figure 6.4: Expansion distribution of DMS on the CLIC dataset. For each % of budget used, the distribution of stopping points for the batch examples between the four possible models is shown. As can be seen, our approach quickly begins using the most expensive model in order to obtain higher accuracy for less overall computational cost.

Results. The results are shown in Figure 6.3. For wrist localization, our approach was able to obtain a $2\times$ speedup for little to no accuracy cost, and maintain a significant speedup compared to the uninformed model selection baseline. For elbow localization, our approach yields a speedup of $5\times$ at the same accuracy cost as an uninformed $2\times$ speedup, a significant improvement. Note that these improvements include the slight additional cost of evaluating the DMS meta-features.

To further shed light on the difference between DMS and the uniform expansion, we investigated whether or not Algorithm 3 was choosing models to use by simply choosing the cheapest first (Figure 6.4). We found this not to be the case; instead, the most expensive model is assigned to examples very early on in the budget allocation process. This suggests that the computational gains of DMS stem from being able to allocate resources to very difficult examples very quickly.

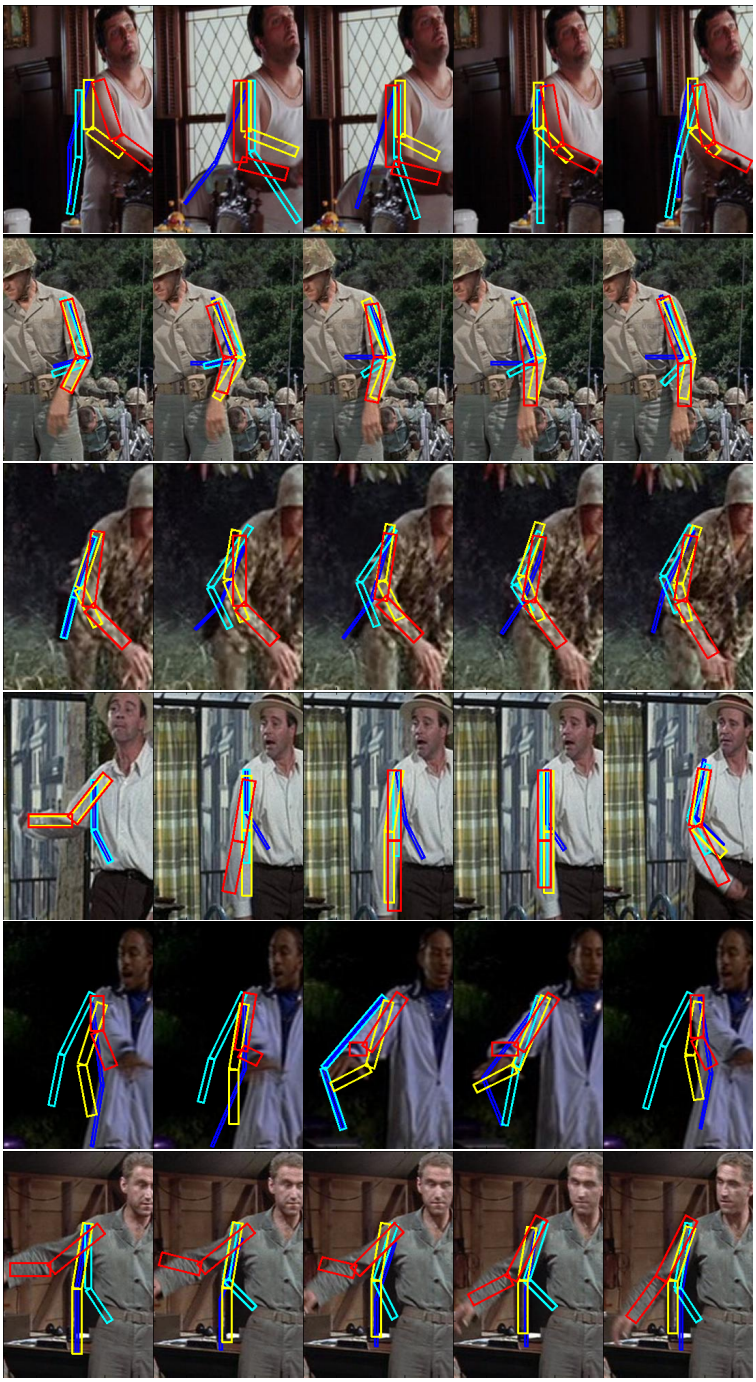


Figure 6.5: Qualitative results on the CLIC dataset. Shown are the predictions of the 4 base models (blue, cyan, yellow, red, respectively). The optical flow based features (red) are often times significantly more accurate than the other features.

6.4 Summary

We presented *dynamic structured model selection* (DMS), a simple but powerful meta-learning algorithm that leverages typically intractable features in structured learning problems in order to automatically determine which of several models should be used at test-time in order to maximize accuracy under a fixed budgetary constraint. In two domains, we found significant improvements in accuracy and efficiency compared to alternative or uninformed approaches. We also established a new state-of-the-art in human pose estimation in video with an implementation that is $23\times$ faster than the previous standard implementation.

Chapter 7

DMS-π: Policy-based Model Selection

Overview. In this chapter, we extend and reformulate the basic DMS framework presented in the previous chapter to obtain significant improvements in the expressiveness-computation trade-off. We achieve this improvement by "opening up" the black box models from the previous section: rather than reason at a meta-level about *models*, we reason about *features*. We call this new framework DMS-π because we redefine the optimization problem in terms of a *feature extraction policy* that determines specifically which features are computed within the linear structured prediction framework of (1.1). Thus, whereas the goal was previously to choose an optimal stopping point in a series of structured models, our goal is now to select an optimal set of features that maximize accuracy within a fixed computational budget.

One approach to this problem is to assume a joint probabilistic model of the input and output variables and a utility function measuring payoffs. The expected value of information measures the increase in expected utility after observing a given variable (Howard, 1966; Lindley, 1956) Unfortunately, the problem of computing optimal conditional observation plans is computationally intractable even for simple graphical models like Naive Bayes (Krause and Guestrin, 2009). Moreover, assuming and learning a joint model of input and output is typically quite inferior to discriminative models of output given input (Altun et al., 2003; Collins, 2002; Lafferty et al., 2001; Taskar et al., 2003).

The approach developed in this chapter is to adapt the two-tier structure of DMS to the feature extraction problem. Both tiers must change to adapt to the new setting. First, we can no longer assume a single sequence of models, because we allow for features to be computed on-the-fly for different factors even within a single example. We must be able to evaluate predictions for any combination of features computed within the various factors. Therefore, the first tier is a *information-adaptive* prediction model that is trained to produce hypotheses given any of a set of extracted feature combinations. These feature combinations factorize over an assumed graph structure, and we allow for sparsely computed features such that different vertices and edges may utilize different features of the input.

Next, rather than learning a selector that sequentially selects models for evaluation, we frame the meta-level control problem in terms of finding a feature extraction policy that sequentially adds features to the models until a budget limit is reached. Whereas in DMS features are re-computed for an entire example with each selection step, a single action for the DMS-π policy consists of choosing a factor within a given example and computes additional features for that factor. Once again, we take a discriminative approach and learn a parametrized value function that utilizes a set of meta-features to predict the value of each action; however, we redefine the value of information to non-myopically include future changes in accuracy. In order to estimate this new value function, we adopt a more sophisticated learning scheme based on techniques from the reinforcement learning community.

Once again, the critical advantage here is that the meta-features can incorporate valuable functions of the output space that are infeasible to include at inference time: while features for inference must be computed densely for every assignment to the clique they are computed over, the meta-features are computed sparsely on-the-fly for only the proposed outputs of the adaptive model. Thus, the meta-features can include qualitatively different long-range dependencies that convey information about the self-consistency of a proposed output. We learn to weigh the meta-features for the value function using linear

function approximation techniques from reinforcement learning.

Contributions. We propose DMS-π for adaptively controlling the costs of feature extraction in structured models at test-time. We first learn a prediction model that is trained to use subsets of features computed sparsely across the structure of the input; we then use reinforcement learning to estimate a value function that adaptively computes an approximately optimal set of features given a budget constraint. Because of the particular structure of our problem, we show how we can apply value function estimation in a batch setting using standard least-squares solvers. We revisit the applications from Chapter 6 and apply our method to two sequential prediction domains: articulated human pose estimation and handwriting recognition. In both domains, we achieve greater accuracy using a small fraction of the available features. Finally, in the handwriting domain where inference rivals feature extraction as a bottleneck, we provide a self-limiting message-passing inference algorithm that we show empirically can dramatically reduce inference time while preserving the benefits of our approach.

7.1 Q-Learning a feature extraction policy

Setup. We follow the structured prediction framework introduced in Chapter 3. However, we introduce an additional *explicit* feature extraction state vector $\mathbf{z}$:

$$h(\mathbf{x}, \mathbf{z}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{z}). \tag{7.1}$$

Above, $\mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is a sparse vector of D features that takes time $\mathbf{c}^\top \mathbf{z}$ to compute for a non-negative cost vector $\mathbf{c}$ and binary indicator vector $\mathbf{z}$ of length $|\mathbf{z}| = F$. Intuitively, $\mathbf{z}$ indicates which of F sets of features are extracted when computing $\mathbf{f}$; $\mathbf{z} = \mathbf{1}$ means every possible feature is extracted, while $\mathbf{z} = \mathbf{0}$ means that only a minimum set of features is extracted.

Note that by incorporating $\mathbf{z}$ into the feature function, the predictor h can learn to use different linear weights for the same underlying feature value by conditioning the feature

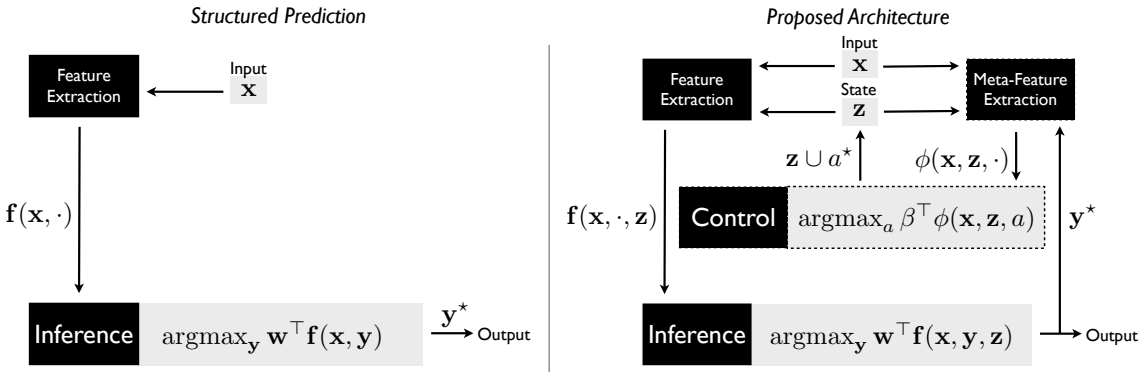


Figure 7.1: **Overview of framework architecture.** (Left) Standard structured prediction architecture; features are extracted from the input and passed to the inference procedure, which produces the output $\mathbf{y}^\star$. (Right) Proposed framework; we introduce the binary state vector $\mathbf{z}$ to represent explicitly which sets of features have been extracted. The output of the inference procedure (which performs inference given any $\mathbf{z}$) produces an output $\mathbf{y}^\star$, and this output is then used to extract *meta-features* ϕ, which are in turn used to compute an update $a^\star$ to the state $\mathbf{z}$ by a *control* model. We use stochastic subgradient to learn the inference model $\mathbf{w}$ first and reinforcement learning to learn the control model β given $\mathbf{w}$.

on the value of $\mathbf{z}$. As we discuss in Section 7.4, adapting the weights in this way is crucial to building a predictor h that works well for *any* subset of features. We will discuss how to construct such features in more detail in Section 7.2.

Suppose we have learned such a model h. At test time, our goal is to make the most accurate predictions possible for an example under a fixed budget B. Specifically, given h and a loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$, we wish to find the following:

$$H(\mathbf{x}, B) = \operatorname*{argmin}_{\mathbf{z}} \mathbb{E}_{\mathbf{y}|\mathbf{x}}[\mathcal{L}(\mathbf{y}, h(\mathbf{x}, \mathbf{z}))] \tag{7.2}$$

In practice, there are three primary difficulties in optimizing equation (7.2). First, the distribution $P(Y|X)$ is unknown. Second, there are exponentially many $\mathbf{z}$s to explore. Most important, however, is the fact that we do not have free access to the objective function. Instead, given $\mathbf{x}$, we are optimizing over $\mathbf{z}$ using a *function oracle* since we cannot compute $\mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ without paying $\mathbf{c}^\top \mathbf{z}$, and the total cost of all the calls to the oracle must not exceed B. Our approach to solving these problems is outlined in Figure 7.1; we learn a *control model* (i.e. a policy) by posing the optimization problem as an MDP and using reinforcement learning techniques.

Adaptive extraction MDP. We model the budgeted prediction optimization as the following Markov Decision Process. The state of the MDP s is the tuple $(\mathbf{x}, \mathbf{z})$ for an input $\mathbf{x}$ and feature extraction state $\mathbf{z}$ (for brevity we will simply write s). The start state is $s_0 = (\mathbf{x}, \mathbf{0})$, with $\mathbf{x} \sim P(X)$, and $\mathbf{z} = \mathbf{0}$ indicating only a minimal set of features have been extracted. The action space $\mathcal{A}(s)$ is $\{i \mid z_i = 0\} \cup \{0\}$, where z_i is the i'the element of $\mathbf{z}$; given a state-action pair (s, a), the next state is deterministically $s' = (\mathbf{x}, \mathbf{z} + \mathbf{e}_a)$, where $\mathbf{e}_a$ is the indicator vector with a 1 in the a'th component or the zero vector if $a = 0$. Thus, at each state we can choose to extract one additional set of features, or none at all (at which point the process terminates.) Finally, for fixed h, we define the shorthand $\eta(s) = \mathbb{E}_{\mathbf{y}|\mathbf{x}} \mathcal{L}(\mathbf{y}, h(\mathbf{x}, \mathbf{z}))$ to be the expected error of the predictor h given state $\mathbf{z}$ and input $\mathbf{x}$.

We now define the expected reward to be the adaptive value of information of extracting

the a'th set of features given the system state and budget B:

$$R(s, a, s') = \begin{cases} \eta(s) - \eta(s') & \text{if } \mathbf{c}^\top \mathbf{z}(s') \leq B \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

Intuitively, (7.3) says that each time we add additional features to the computation, we gain reward equal to the decrease in error achieved with the new features (or pay a penalty if the error increases.) However, if we ever exceed the budget, then any further decrease does not count; no more reward can be gained. Furthermore, assuming $\mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ can be cached appropriately, it is clear that we pay only the additional computational cost c_a for each action a, so the entire cumulative computational burden of reaching some state s is exactly $\mathbf{c}^\top \mathbf{z}$ for the corresponding $\mathbf{z}$ vector.

Given a trajectory of states $s_0, s_1, \ldots, s_T$, computed by some deterministic policy π, it is clear that the final cumulative reward $R_\pi(s_0)$ is the difference between the starting error rate and the rate of the last state satisfying the budget:

$$R_\pi(s_0) = \eta(s_0) - \eta(s_1) + \eta(s_1) - \cdots = \eta(s_0) - \eta(s_{t^\star}), \tag{7.4}$$

where $t^\star$ is the index of the final state within the budget constraint. Therefore, the optimal policy $\pi^\star$ that maximizes expected reward will compute $\mathbf{z}^\star$ minimizing (7.2) while satisfying the budget constraint.

Learning an approximate policy with long-range meta-features. In this work, we focus on a straightforward method for learning an approximate policy: a batch version of least-squares policy iteration (Lagoudakis and Parr, 2003) based on Q-learning (Watkins and Dayan, 1992). We parametrize the policy as the greedy optimization of a linear function of *meta-features* ϕ computed from the current state $s = (\mathbf{x}, \mathbf{z})$: $\pi_\beta(s) = \operatorname{argmax}_a \beta^\top \phi(\mathbf{x}, \mathbf{z}, a)$. The meta-features (which we abbreviate as simply $\phi(s, a)$ henceforth) need to be rich enough to represent the value of choosing to expand feature a for a given partially-computed example $(\mathbf{x}, \mathbf{z})$. Note that we already have computed $\mathbf{f}(\mathbf{x}, h(\mathbf{x}, \mathbf{z}), \mathbf{z})$, which may be useful in estimating the confidence of the model on a given example. However, we have

much more freedom in choosing $\phi(s, a)$ than we had in choosing $\mathbf{f}$; while $\mathbf{f}$ is restricted to ensure that inference is tractable, we have no such restriction for ϕ. We therefore compute functions of $h(\mathbf{x}, \mathbf{z})$ that take into account large sets of output variables, and since we need only compute them for the particular output $h(\mathbf{x}, \mathbf{z})$, we can do so very efficiently. We describe the specific ϕ we use in our experiments in Section 7.4, typically measuring the self-consistency of the output as a surrogate for the expected accuracy.

One-step off-policy Q-learning with least-squares. To simplify the notation, we will assume given current state s, taking action a deterministically yields state s'. Given a policy π, the value of a policy is recursively defined as the immediate expected reward plus the discounted value of the next state:

$$Q_\pi(s, a) = R(s, a, s') + \gamma Q_\pi(s', \pi(s')). \tag{7.5}$$

The goal of Q-learning is to learn the Q for the optimal policy $\pi^\star$ with maximal $Q_{\pi^\star}$; however, it is clear that we can increase Q by simply stopping early when $Q_\pi(s, a) < 0$ (the future reward in this case is simply zero.) Therefore, we define the *off-policy* optimized value $Q^\star_\pi$ as follows:

$$Q^\star_\pi(s_t, \pi(s_t)) = R(s_t, \pi(s_t), s_{t+1}) + \gamma \left[Q^\star_\pi(s_{t+1}, \pi(s_{t+1}))\right]_+ . \tag{7.6}$$

We propose the following one-step algorithm for learning Q from data. Suppose we have a finite trajectory $s_0, \ldots, s_T$. Because both π and the state transitions are deterministic, we can unroll the recursion in (7.6) and compute $Q^\star_\pi(s_t, \pi(s_t))$ for each sample using simple dynamic programming. For example, if $\gamma = 1$ (there is no discount for future reward), we obtain $Q^\star_\pi(s_i, \pi(s_i)) = \eta(s_i) - \eta(s_{t^\star})$, where $t^\star$ is the optimal stopping time that satisfies the given budget.

We therefore learn parameters $\beta^\star$ for an approximate Q as follows. Given an initial policy π, we execute π for each example $(\mathbf{x}^j, \mathbf{y}^j)$ to obtain trajectories $s^j_0, \ldots, s^j_T$. We then solve the following least-squares optimization,

$$\beta^\star = \operatorname*{argmin}_\beta \lambda ||\beta||^2 + \frac{1}{nT} \sum_{j,t} \left(\beta^\top \phi(s^j_t, \pi(s^j_t)) - Q^\star_\pi(s^j_t, \pi(s^j_t))\right)^2 , \tag{7.7}$$

using cross validation to determine the regularization parameter λ.

We perform a simple form of policy iteration as follows. We first initialize β by estimating the expected reward function (this can be estimated from pairs (s, s'), which are more efficient to compute than Q-functions on trajectories). We then compute trajectories under π_β and use these trajectories to compute $\beta^\star$ that approximates $Q^\star_\pi$. We found that additional iterations of policy iteration did not noticeably change the results.

Learning for multiple budgets. One potential drawback of our approach just described is that we must learn a different policy for every desired budget. A more attractive alternative is to learn a single policy that is tuned to a range of possible budgets. One solution is to set $\gamma = 1$ and learn with $B = \infty$, so that the value $Q^\star_\pi$ represents the best improvement possible using some optimal budget $B^\star$; however, at test time, it may be that $B^\star$ is greater than the available budget B and $Q^\star_\pi$ is an over-estimate. By choosing $\gamma < 1$, we can trade-off between valuing reward for short-term gain with smaller budgets $B < B^\star$ and longer-term gain with the unknown optimal budget $B^\star$.

In fact, we can further encourage our learned policy to be useful for smaller budgets by adjusting the reward function. Note that two trajectories that start at s_0 and end at $s_{t^\star}$ will have the same reward, yet one trajectory might maintain much lower error rate than the other during the process and therefore be more useful for smaller budgets. We therefore add a shaping component to the expected reward in order to favor the more useful trajectory as follows:

$$R_\alpha(s, a, s') = \eta(s) - \eta(s') - \alpha \left[\eta(s') - \eta(s)\right]_+ . \tag{7.8}$$

This modification introduces a term that does not cancel when transitioning from one state to the next, *if the next state has higher error than our current state.* Thus, we can only achieve optimal reward $\eta(s_0) - \eta(s_{t^\star})$ when there is a sequence of feature extractions that never increases the error rate[1]; if such a sequence does not exist, then the parameter α controls the trade-off between the importance of reaching $s_{t^\star}$ and minimizing any errors

[1]While adding features decreases training error on average, even on the training set additional features may lead to increased error for any particular example.

Features	Error (%)	Time	Overhead	Speedup
All	44.3	5.20s	—	1×
	44.0	**1.20s**	**0.10s**	**4**×
Variable	44.5	0.56s	0.08s	8×
(Q-learning)	45.6	0.24s	0.08s	16×
	46.5	0.08s	0.07s	32×
All−Flow	46.3	1.01s	0.00s	5.1×
Base only	47.7	—	—	—

Table 7.1: Trade-off between average elbow and wrist error rate and computation time achieved by our method on the pose dataset, for a variety of different budgets. The methods are stopped once the desired speedup is reached. Note that due to meta-feature precomputation in our implementation, there is 0.07s minimal overhead. Runtime is measure in seconds per frame. Our approach yields a more accurate model that is 4× faster, as well as much greater efficiency than *a priori* avoiding the expensive features for every example.

along the way. Note that we can still use the procedure described above to learn β when using R_α instead of R. We use a development set to tune α as well as γ to find the most useful policy when sweeping B across a range of budgets.

7.2 Design of the information-adaptive predictor h

Learning. We now address the problem of learning $h(\mathbf{x}, \mathbf{z})$ from n labeled data points $\{(\mathbf{x}^j, \mathbf{y}^j\}_{j=1}^n$. Since we do not necessarily know the test-time budget during training (nor would we want to repeat the training process for every possible budget), we formulate the problem of minimizing the *expected* training loss according to a uniform distribution over budgets:

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}}\, \lambda||\mathbf{w}||^2 + \frac{1}{n}\sum_{j=1}^{n} \mathbb{E}_{\mathbf{z}}[\mathcal{L}(\mathbf{y}^j, h(\mathbf{x}^j, \mathbf{z})]. \tag{7.9}$$

Note that if $\mathcal{L}$ is convex, then (7.9) is a weighted sum of convex functions and is also convex. Our choice of distribution for $\mathbf{z}$ will determine how the predictor h is calibrated.

In our experiments, we used the following generative process: (1) first sample a budget $B \in [0, |\mathbf{c}|_1]$ uniformly at random, (2) then sample $\mathbf{z}$ uniformly from $\{\mathbf{z} \mid \mathbf{c}^\top \mathbf{z} = B\}$ by greedily adding feature sets to $\mathbf{z}$ in random order. To learn $\mathbf{w}$, we use Pegasos-style (Shalev-Shwartz et al., 2007) stochastic sub-gradient descent; we approximate the expectation in (7.9) by resampling $\mathbf{z}$ every time we pick up a new example $(\mathbf{x}^j, \mathbf{y}^j)$. We set λ and a stopping-time criterion through cross-validation onto a development set.

Feature design. We now turn to the question of designing $\mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{z})$. In the standard pair-wise graphical model setting (before considering $\mathbf{z}$), we decompose a feature function $\mathbf{f}(\mathbf{x}, \mathbf{y})$ into unary and pairwise features:

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i \in \mathcal{V}} \mathbf{f}_u(\mathbf{x}, y_i) + \sum_{(i,j) \in \mathcal{E}} \mathbf{f}_e(\mathbf{x}, y_i, y_j) \tag{7.10}$$

We consider several different schemes of incorporating $\mathbf{z}$ of varying complexity. The simplest scheme is to use several different feature functions $\mathbf{f}^1, \ldots, \mathbf{f}^T$. Then $|\mathbf{z}| = F$, and $\mathbf{z}_a = 1$ indicates that $\mathbf{f}^a$ is computed. Thus, we have the following expression, where we use $z(a)$ to indicate the a'th element of z:

$$\mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sum_{a=1}^{T} z(a) \left[\sum_{i \in \mathcal{V}} \mathbf{f}_u^a(\mathbf{x}, y_i) + \sum_{(i,j) \in \mathcal{E}} \mathbf{f}_e^a(\mathbf{x}, y_i, y_j) \right] \tag{7.11}$$

Note that in practice we can choose each $\mathbf{f}^a$ to be a sparse vector such that $\mathbf{f}^a \cdot \mathbf{f}^{a'} = 0$ for all $a' \neq a$; that is, each feature function $\mathbf{f}^a$ "fills out" a complementary section of the feature vector $\mathbf{f}$. Note that this feature extraction scheme is essentially equivalent to the series of linear models of the standard DMS scheme, since adding features sequentially in this way corresponds to using entirely different feature functions.

A much more powerful approach is to create a feature vector as the composite of different extracted features for each vertex and edge in the model. In this setting, we set $\mathbf{z} = [\mathbf{z}_u \ \mathbf{z}_e]$, where $|\mathbf{z}| = (|\mathcal{V}| + |\mathcal{E}|)T$, and we have

$$\mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sum_{i \in \mathcal{V}} \sum_{a=1}^{T} z_u(a, i) \mathbf{f}_u^a(\mathbf{x}, y_i) + \sum_{(i,j) \in \mathcal{E}} \sum_{a=1}^{T} z_e(a, ij) \mathbf{f}_e^a(\mathbf{x}, y_i, y_j). \tag{7.12}$$

We refer to this latter feature extraction method a *factor-level* feature extraction, and the former as *example-level*. As we will show empirically, the factor-level extraction method allows for much finer-grain control of computation and therefore much more favorable trade-offs. However, these gains come at the cost of increased inference time; inference is re-run many more times per equivalent budget increase for the factor-level extractions as compared to the example-level extractions. This is because each action taken by the controller uses up smaller chunks of the budget and computes far fewer features when features are only added to a single factor at at time.

Finally, we further note that the restriction (7.13) also allows us to increase the complexity of the feature function $\mathbf{f}$ as follows; when using the a'th extraction, we allow the model to re-weight the features from extractions 1 through a. In other words, we condition the value of the feature on the current set of features that have been computed; since there are only T sets in the restricted setting (and not 2^F), this is a feasible option. We simply define $\hat{\mathbf{f}}^a = [0 \ \ldots \ \mathbf{f}^1 \ \ldots \ \mathbf{f}^a \ \ldots \ 0]$, where we add duplicates of features $\mathbf{f}^1$ through $\mathbf{f}^a$ for each feature block a. Thus, the model can learn different weights for the same underlying features based on the current level of feature extraction; we found that this was crucial for optimal performance.

Reducing inference overhead. Feature computation time is only one component of the computational cost in making predictions; computing the argmax (7.1) given $\mathbf{f}$ can also be expensive. Note that for reasons of simplicity, we only consider low tree-width models in this work for which (7.1) can be efficiently solved via a standard max-sum message-passing algorithm. Nonetheless, since $\phi(s, a)$ requires access to $h(\mathbf{x}, \mathbf{z})$ then we must run message-passing every time we compute a new state s in order to compute the next action. Therefore, we run message passing *once* and then perform less expensive local updates using saved messages from the previous iteration. We define a simple algorithm for such *quiescent* inference; we refer to this inference scheme as q-inference. The intuition is that we stop propagating messages once the magnitude of the update to the max-marginal decreases below a certain threshold q; we define q in terms of the margin of the current

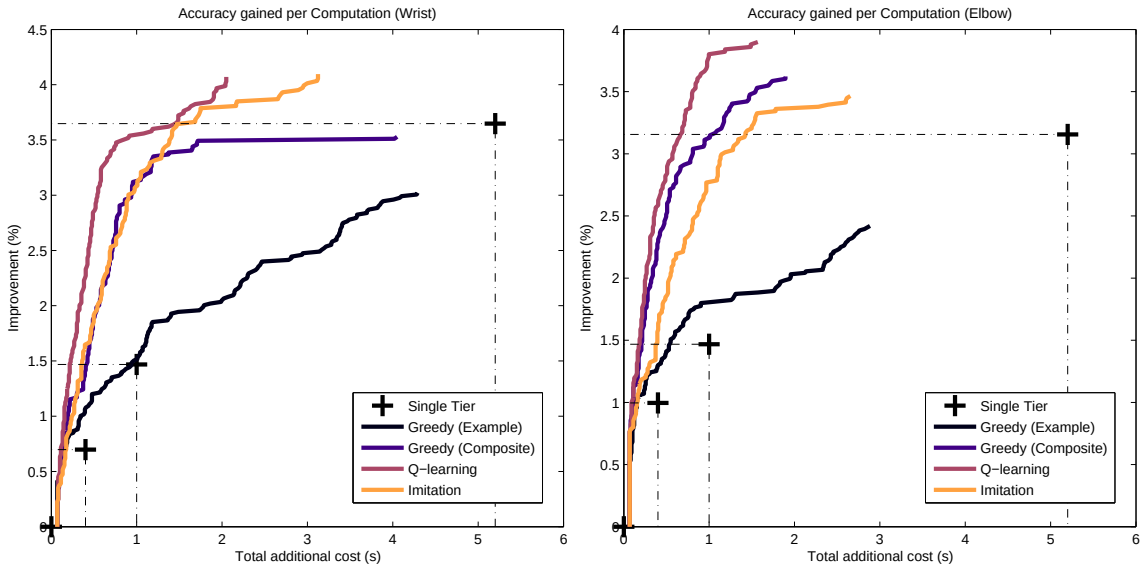


Figure 7.2: **Trade-off performance on the pose dataset for wrists (left) and elbows (right).** The curve shows the additional computation time per frame (including all overhead) required to achieve an increase in accuracy over the minimal-feature model. The "Single Tier" baseline indicates computing all features in order up to a given level for all examples. The "Greedy" baselines consist of a policy trained to maximize an estimate of reward, with no measure of future value of an action; the "example" variant computes new features at each position in the sequence (i.e. using the *example-level* action space); this is far less effective than the factor-level (composite) computation strategies. Note that Q-learning in particular achieves higher accuracy models at a fraction of the computational cost of using all features, and is more effective than imitation learning.

Algorithm 4: Quiescent Forward-Backward Algorithm.

input : Forward/backward messages α/β, target position p, scores θ, tolerance q.

output: Updated messages α',β'.

Initialize $\alpha' \leftarrow \alpha$, $\beta' \leftarrow \beta$, $i \leftarrow p, j \leftarrow p, \Delta\alpha = 1, \Delta\beta = 1$;

Let $\tau(i) = \max_k\{\alpha_{ik} + \beta_{ik}\} - \min_k\{\alpha_{ik} + \beta_{ik}\}$;

while $i \leq L$ *and* $\Delta\alpha > q$ **do**

  $\forall k: \quad \alpha'_{ik} \leftarrow \theta(i,k) + \max_{k'} \alpha'_{i-1,k'} + \theta(i,k',k)$;

  $\Delta\alpha \leftarrow \max_k |\alpha'_{ik} - \alpha_{ik}|/\tau(i)$;

while $j \geq 1$ *and* $\Delta\beta > q$ **do**

  $\forall k: \quad \beta'_{jk} \leftarrow \max_{k'} \beta'_{j+1,k'} + \theta(j+1,k,k') + \theta(j+1,k')$;

  $\Delta\beta \leftarrow \max_k |\beta'_{jk} - \beta_{jk}|/\tau(j)$;

MAP decoding at the given position, since that margin must be surpassed if the MAP decoding will change as a result of inference.

The algorithm for local updates to message passing is in Algorithm 4. This algorithm is similar to the standard forward-backward max-sum message passing algorithm for linear-chain models, but it assumes that messages have already been precomputed. Based on these precomputed messages, computation stops if the new message does not change by a specified fraction q of the amount needed to change the argmax at a given position in the sequence. It furthermore begins computation at a position p (where scores have presumably changed) and propagates the changes outward from that position.

7.3 Batch mode inference

We now have all the elements in place to define the final DMS-π algorithm. At test time, we are typically given a *test set* of n examples, rather than a single example. In this setting the budget applies to the entire inference process, and it may be useful to spend more of the budget on difficult examples rather than allocate the budget evenly across all examples.

In this case, we extend our framework to concatenate the states of all n examples $s = (\mathbf{x}^1, \ldots, \mathbf{x}^n, \mathbf{z}^1, \ldots, \mathbf{z}^n)$. The action consists of choosing an example and then choosing an action within that example's sub-state; our policy searches over the space of *all* actions for *all* examples simultaneously. Because of this, we impose additional constraints on the action space, specifically:

$$z(a, \ldots) = 1 \implies z(a', \ldots) = 1, \quad \forall a' < a. \tag{7.13}$$

Equation (7.13) states that there is an inherent *ordering* of feature extractions, such that we cannot compute the a'th feature set without first computing feature sets $1, \ldots, a-1$. This greatly simplifies the search space in the batch setting while at the same time preserving enough flexibility to yield significant improvements in efficiency.

A precise form of the algorithm is given in Algorithm 5. Note that we use the factor-level feature extraction scheme given in 7.2; an action chooses an element α from a graph structure $\mathcal{G}$ and a feature tier level t (assuming T tiers), and computes that specific feature tier of features for the specific factor corresponding to the graph element α. Thus, for a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, there are $(|\mathcal{V}| + |\mathcal{E}|)T$ possible actions per example.

Imitation learning baseline. As an alternative to the approach just described, we also consider an imitation learning scheme in which we learn a classifier to reproduce a target policy given by an oracle. We use the same trajectories used to $Q^\star_\pi$, but instead we create a classification dataset of positive and negative examples given a budget B by assigning all state/action pairs along a trajectory within the budget as positive examples and all budget violations as negative examples. We tune the budget B using a development set to optimize the overall trade-off when the policy is evaluated with multiple budgets.

7.4 Experiments

We evaluate our approach on linear-chain models by revisiting the two distinct domains of the previous chapter. The first is the tracking of human pose in video, in which feature

Algorithm 5: Factor-level DMS-π batch inference.

input : Test set $\{\mathbf{x}^j\}_1^n$, adaptive model h, controller weights β, graph $\mathcal{G}$, budget B.

output: Predictions $\mathbf{y}^1, \ldots, \mathbf{y}^n$.

initialize $B' \leftarrow 0$, $\mathbf{z}^j \leftarrow \mathbf{0}$, $\mathbf{y}^j \leftarrow h(\mathbf{x}^j, \mathbf{z}^j)$;

define an action a as a pair $\langle \alpha \in \mathcal{G}, t \in \{1, \ldots, T\} \rangle$;

initialize priority queue Q with pairs $\left\langle \beta^\top \phi(\mathbf{x}^j, \mathbf{z}^j, a), (j, a) \right\rangle, \forall a = \langle \alpha, 1 \rangle$;

while $B' < B$ *and* Q *is not empty* **do**

  Pop values (j, a) from Q with max priority;

  if $c_a \leq (B - B')$ **then**

    $\mathbf{z}^j \leftarrow \mathbf{z}^j + a$;

    $B' \leftarrow B' + c_a$;

    $\mathbf{y}^j \leftarrow h(\mathbf{x}^j, \mathbf{z}^j)$;

    $a' \leftarrow \langle \alpha, t + 1 \rangle$);

    Insert $\left\langle \beta^\top \phi(\mathbf{x}^j, \mathbf{z}^j, a'), (j, a') \right\rangle$ into Q;

computation completely dominates the running time of the algorithm, and the second is optical handwriting recognition, in which inference time is on par with feature computation. In both very different settings, our method achieves much faster models that use an order of magnitude fewer features while yielding higher accuracy than the baseline.

7.4.1 Tracking of human pose in video

Setup. For this problem, our goal is to predict the joint locations of human limbs in video clips extracted from Hollywood movies. Our testbed is the MODEC+S model proposed in the previous chapter; the MODEC+S model uses the MODEC model of Sapp and Taskar (2013) to generate 32 proposed poses per frame of a video sequence, and then combines the predictions using a linear-chain structured sequential prediction model. There are four types of features used by MODEC+S, the final and most expensive of which is a coarse-to-fine optical flow Liu (2009); we assume the cheapest are always computed and minimize additional computation time. We present cross validation results averaged over 50 80/20 train/test splits of the dataset. We measure localization performance or elbow and wrists in terms of percentage of times the predicted locations fall within 20 pixels of the ground truth.

Meta-features. We define the meta-features $\phi(s, a)$ in terms of the targeted position in the sequence i and the current predictions $\mathbf{y}^\star = h(\mathbf{x}, \mathbf{z})$. Specifically, we concatenate the already computed unary and edge features of $y_i^\star$ and its neighbors (conditioned on the value of $\mathbf{z}$ at i), the margin of the current MAP decoding at position i, and a measure of self-consistency computed on $\mathbf{y}^\star$ as follows. For all sets of m frames overlapping with frame i, we extract color histograms for the predicted arm segments and compute the maximum χ^2-distance from the first frame to any other frame; we then also add an indicator feature each of these maximum distances exceeds 0.5, and repeat for $m = 2, \ldots, 5$. We also add several bias terms for which sets of features have been extracted around position i.

Discussion. We present a short summary of our pose results in Table 7.1, and compare to various baselines in Figure 7.2. We found that our Q-learning approach is consistently more effective than the imitation learning alternative; Q-learning yields a model that has $4\times$ faster runtime (including the overhead of our approach) and that is more accurate than a baseline model trained with all features. Note that in terms of the proportion of features used, our method is extremely efficient: we achieve 56% of the possible improvement using only 4.6% of all possible features, or 96% of the improvement while using only 10.7% of the features.

7.4.2 Handwriting recognition

Setup. For this problem, we use the OCR dataset from Taskar et al. (2003), which is pre-divided into 10 folds that we use for cross validation. We use three sets of features: the original pixels (free), and two sets of Histogram-of-Gradient (HoG) features computed on the images for different bin sizes. Unlike the pose setting, the features are very fast to compute compared to inference. Thus, we evaluate the effectiveness of q-inference with various thresholds to minimize inference time. For meta-features, we use the same construction as for pose, but instead of inter-frame χ^2-distance we use a binary indicator as to whether or not the specific m-gram occurred in the training set. The results are summarized in Figure 7.3; see caption for details.

Discussion. As in the pose setting, our method is extremely efficient in terms of the features computed for h; however, in this case, the overhead of inference is on par with the feature computation. Thus, we obtain a more accurate model with $q = 0.5$ that is $1.5\times$ faster, even though it uses only $1/5$ of the features; if the implementation of inference were improved, we would expect a speedup much closer to $5\times$. Furthermore, once again the composite (factor-wise) features are far more efficient in terms of features extracted than the example-wise policy.

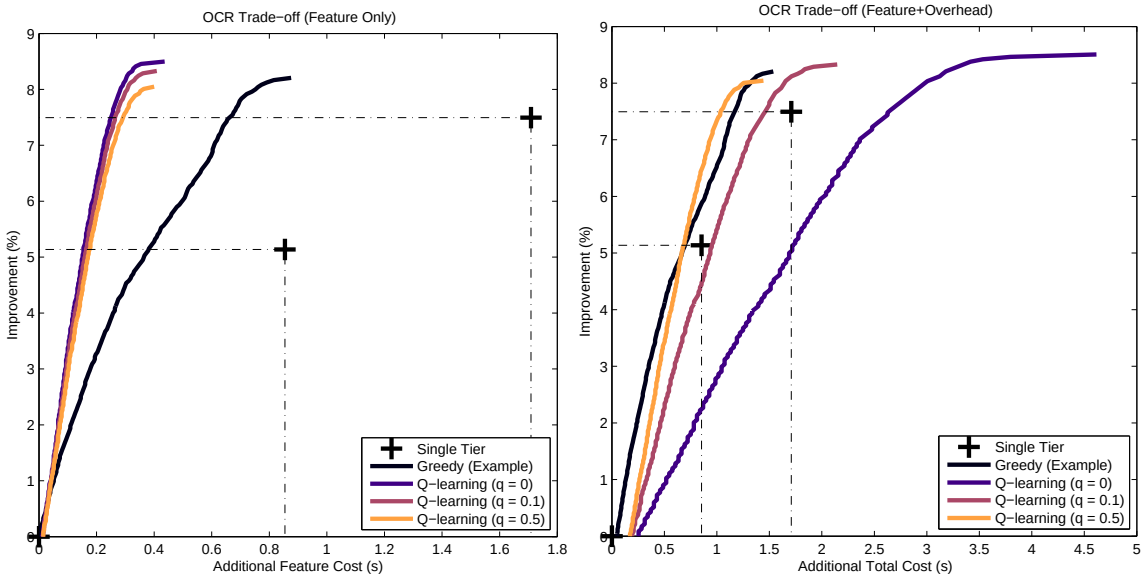


Figure 7.3: **Controlling overhead on the OCR dataset.** While our approach is is extremely efficient in terms of how many features are extracted (Left), the additional overhead of inference is prohibitively expensive for the OCR task without applying q-inference (Right) with a large threshold. Furthermore, although the example-wise strategy is less efficient in terms of features extracted, it is more efficient in terms of overhead.

7.5 Summary

We have introduced a framework for learning feature extraction policies and predictive models that adaptively select features for extraction in a factor-wise, on-line fashion. On two tasks our approach yields models that both more accurate and far more efficient; our work is a significant step towards eliminating the feature extraction bottleneck in structured prediction.

Chapter 8

Future Work

In the preceding chapters, we have presented several new frameworks for addressing the expressiveness-computation trade-off in structured prediction. However, although the bulk of the thesis work is completed, there are several further questions worth investigating. In this chapter, I propose a primary direction for future work: combining and extending the existing frameworks into a single method, SPC-π.

8.1 Combining SPC and DMS-π

Addressing complementary limitations. The DMS-π has significant advantages over the DMS framework originally proposed: DMS-π is capable of optimizing expressiveness vs. computation within a single example and at a much finer-grained level than DMS, which can only select between different models. However, as the framework currently stands, the DMS-π approach is limited to adding more features into existing factors. In contrast, consider the limitations of SPC: while SPC could be applied to the same setting, the bottleneck of feature computation may not depend on the size of the state space of the output, e.g. computing HoG on the handwriting recognition data depends only on the size of the input image. On the other hand, SPC is capable of incorporating factors with large scopes when the features computation bottleneck does depend on the size of the output

space.

In other words, we can see that the limitations of the DMS-π and SPC frameworks are complementary: of the two channels we have for increasing expressiveness–adding features or adding scope to factors–DMS-π is well suited for the former, while SPC is well suited for the latter. It is natural to therefore consider combining the approaches into a single framework that is capable of intelligently allocating computation across both channels simultaneously.

Introducing SPC-π. We call this new hybrid approach SPC-π: Policy-based Structured Prediction Cascades. The key idea is to expand the action space of DMS-π to incorporate introducing additional factors into the model. A proposed high-level algorithm for inference in this new framework is given in Algorithm 6; the key inputs are now an *action space* $\mathcal{A}$ that defines either additional features for a given factor or adding a new factor to the model. For simplicity of presentation we assume a policy $\pi(\mathbf{x}, \mathbf{z})$ (as opposed to explicitly defining a priority queue value based policy as in Algorithm 5). Once again, we require a single model that can adapt to any feature set and make any predictions. Note that if features are added to an existing factor, the procedure is almost identical to DMS-π. However, as in SPC, we maintain a sparse list of valid assignments for each factor; thus, whenever a new factor is added, it is constructed on the fly from the list of valid assignments to other factors, scored, and then filtered again based on the max marginals.

Adaptive cascade structure. We can also interpret SPC-π as an adaptive SPC cascade structure that is specific to each test example. For example, suppose the pool of available features always adds new factors to the model: each subsequent addition could be considered a different model $\mathbf{f}^0, \ldots, \mathbf{f}^T$. However, instead of using a set of pre-trained filter models that are applied in fixed order, these models are constructed incrementally by adding a single factor at a time. Thus, the hybrid approach represents an intelligent means of adapting the *structure* of the cascade in the SPC to each example. In this way, we can "close the loop" and remove the remaining human element (construction of the cascade)

Algorithm 6: High-level overview of SPC-π inference.

input : Example $\mathbf{x}$, adaptive model h, controller π, action space $\mathcal{A}$, budget B.

output: Prediction $\mathbf{y}$.

initialize $\mathbf{z} \leftarrow \mathbf{0}$, $\mathbf{y} \leftarrow h(\mathbf{x}, \mathbf{z})$;

initialize max-marginals $\psi^{\star}_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c)$ using the baseline model and initialize: $\mathcal{S}(\mathbf{x}) = \{\mathbf{y}_c : \psi^{\star}_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c) \geq \tau_{\mathbf{w},\alpha}(\mathbf{x})\}$;

while *budget remains* **do**

- Get an action a from the policy $\pi(\mathbf{x}, \mathbf{z})$ with cost c_a;
- Sparsely compute $\mathbf{f}_a$ according to $\mathcal{S}(\mathbf{x})$;
- Update max-marginals $\psi^{\star}_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c)$ and output $h(\mathbf{x}, \mathbf{z})$;
- If not yet filtered, filter the scope of $\mathbf{f}_a$ according to $\tau_{\mathbf{w},\alpha}(\mathbf{x})$;

from the SPC framework.

However, one significant difference between this hybrid approach and the standard SPC approach is that the max-marginals can be filtered at any time. In other words, we no longer can train a pre-set pool of filtering models; in fact, in Algorithm 6, there is only a single adaptive model that is responsible for both filtering and making predictions. An interesting research question is whether or not we can learn such a model that is useful for both; alternatively, we might learn and maintain two separate models $\mathbf{w}_1$ and $\mathbf{w}_2$ for prediction and filtering respectively. Regardless, we now need to learn a filtering model that utilizes arbitrary subsets of features, much as the adaptive prediction model is learned in DMS-π. Finally, another significant difference is that the SPC-π approach as defined is not suitable for coarse-to-fine SPC problems where the state space (and not the factors) is changed from one model to the next, as predictions are not useful in this setting until the final (fine) stage of the cascade.

New challenges. This new model raises some additional difficulties that must be solved. In order to implement Algorithm 6, we need to first define an action space $\mathcal{A}$ that includes incorporating factors of new scope into the model, yet is still efficient to search over and that

lends itself to computing meta-features. We then need an inference algorithm for the max-marginals that can be computed using an arbitrary set of factors: if the additional factors would introduce loops, exact inference may not be possible. Thus, the original formulation for features used in DMS-π, which depended on unary and pairwise terms only, is not suitable, but neither is the tree-decomposition method of Ensemble-SPC, which assumes a fixed structure. Therefore, a fruitful direction for future research will be exploring (1) an approximate method for computing max-marginals (e.g. loopy message passing), (2) a dynamically expanding tree-decomposition, and (3) alternative LP relaxations that allow for message passing algorithms (e.g. Sontag (2010)).

8.1.1 SPC-π on Linear-chains

Tractable subcase. Nonetheless, for the specific case of sequential prediction, there is a relatively straightforward action space that should admit exact inference. As we have seen, it is possible to retain exact inference while incorporating higher order Markov terms. This suggests a simple action space for the SPC-π framework: for each position in the sequence, we have one action, corresponding to increasing the order of the n-gram at that position. We might also allow for adding features to the current largest factor at that position. Whenever a higher order factor is added, we simply concatenate the state spaces (as before, in Chapter 4) of the two lower-order factors involved to form a linear-chain model with expanded states.

8.1.2 Extending to loopy graphs

Potential Issues. As discussed, it is less straightforward to compute max-marginals in a loopy factor graph that is being constructed on-the-fly in a piecewise fashion. The first problem is defining a tractable action space: this would almost certainly require domain knowledge. E.g. in a grid model, one might add factors over increasingly large subsets of neighboring variables in the grid, defining one action per possible subset. The next problem is how to approximately compute max-marginals, given the factor graph. The simplest

method would be to simply apply loopy messaging passing, but this notoriously can lead to instabilities Koller and Friedman (2009), and does not come with any generalization bounds.

There are two alternatives that would be worth exploring. The first would be to adapt the Ensemble-SPC approach and construct an ensemble of trees on the fly. The simplest method would be to add an additional tree for each high-order factor being added to the model: since each high order factor only participates in a single tree, we would not need to re-run inference in the previous trees in each iteration. The disadvantage of this approach is that if more features are added to an existing factor that is shared across multiple trees, recomputing max-marginals could become very expensive. Furthermore, we are making the assumption that the action space always admits efficiently constructing such a tree for each factor.

The second approach would be to adopt the dual decomposition approach from LP relaxation methods (e.g. Sontag (2010)); in this case, we would use ensembles of sub-graphs consisting of single factors that communicate through message passing between dual variables. As in the loopy message passing case, we would run multiple iterations of message passing to compute approximate global max-marginals rather than exactly computing max marginals within a small set of sub-trees, but because we are solving a relaxed dual problem, it may possible to define a filtering loss function using these approximate max marginals.

8.1.3 Summary

In this chapter, we have proposed a framework to tie together the themes of this thesis: SPC-π. The basis of this framework is to use the reinforcement-learning approach of DMS-π to dynamically create a factor graph that is specifically tailored for each test instance. It is our goal for the future to use this framework to expand upon the ideas presented in this thesis, and open up new applications beyond the sequential prediction settings discussed here.

Chapter 9

Conclusion

In this thesis, we have presented several frameworks for enabling more accurate and efficient structured prediction. The SPC framework is designed for coarse-to-fine inference, utilizing simpler models to filter the exponentially large state space of more complex models. By using decomposition of a loopy graph into trees, Ensemble-SPC allows us to apply the gains of SPC to a much more general case. Next, we developed the DMS framework, in which we use meta-features to model the error rate of a sequence of increasingly feature-expensive structured prediction models. We extended this framework to DMS-π, allowing for selective feature extraction within a single example and enabling significantly higher gains in efficiency compared to DMS. Finally, we propose for future work unifying the SPC and DMS-π frameworks into a single coherent method that is capable of addressing all aspects of the bottlenecks inherent in structured prediction.

Appendix A

Theorem Proofs

A.1 Proofs of Theorems 1 and 2

We first summarize the Rademacher and Gaussian complexity definitions and results from Bartlett et al. (2002) required to prove the theorems.

Definition 6 (Rademacher and Gaussian complexities)**.** *Let $H : \mathcal{X} \mapsto \mathbb{R}$ be a function class and $\mathbf{x}^1, \ldots, \mathbf{x}^n$ be n independent samples from a fixed distribution. Define the random variables:*

$$\hat{R}(H) = \mathbb{E}_\sigma \left[\sup_{h \in H} \left| \frac{2}{n} \sum_{i=1}^{n} \sigma_i h(\mathbf{x}^i) \right| \middle| \mathbf{x}^1, \ldots, \mathbf{x}^n \right], \tag{A.1}$$

$$\hat{G}(H) = \mathbb{E}_g \left[\sup_{h \in H} \left| \frac{2}{n} \sum_{i=1}^{n} g_i h(\mathbf{x}^i) \right| \middle| \mathbf{x}^1, \ldots, \mathbf{x}^n \right], \tag{A.2}$$

where $\sigma_i \in \pm 1$ are independent uniform and $g_i \in \mathbb{R}$ are independent standard Gaussian. Then $R(H) = \mathbb{E}[\hat{R}(H)]$ and $G(H) = \mathbb{E}[\hat{G}(H)]$ are the Rademacher and Gaussian complexities of H.

Consider a general loss function $\Phi(\mathbf{y}, \mathbf{h}(\mathbf{x}))$ where $\mathbf{h}(\mathbf{x}) \in \mathbb{R}^m$ represents the prediction function. In our case, $\mathbf{h}(\mathbf{x})$ is vector of factor assignment scores $\mathbf{w}^\top \mathbf{f}_c(\mathbf{x}, \mathbf{y}_c)$ of dimension $\sum_{c \in \mathcal{F}} |\mathcal{Y}_c|$, indexed by $\mathbf{y}_c$ (a factor and its assignment). This vector $\mathbf{h}(\mathbf{x})$ contains all the

information needed to compute the max-marginals and threshold for a given example $\mathbf{x}$. Both $\mathcal{L}_e$ and $\mathcal{L}_f$ can be written in this general form, as we detail below.

Definition 7 (Lipschitz continuity with respect to Euclidean norm)**.** *Let $\phi : \mathbb{R}^m \mapsto \mathbb{R}$, then ϕ is Lipschitz continuous with constant $L(\phi)$ with respect to Euclidean norm if for any $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^m$:*

$$|\phi(\mathbf{z}_1) - \phi(\mathbf{z}_2)| \leq L(\phi)||\mathbf{z}_1 - \mathbf{z}_2||_2. \tag{A.3}$$

We recall the relevant results in the following theorem:

Theorem 3 (Bartlett and Mendelson, 2002)**.** *Consider a loss function $\Phi : \mathcal{Y} \times \mathbb{R}^m \mapsto \mathbb{R}$ and a dominating cost function $\phi : \mathcal{Y} \times \mathbb{R}^m \mapsto \mathbb{R}$ such that $\Phi(\mathbf{y}, \mathbf{z}) \leq \phi(\mathbf{y}, \mathbf{z})$. Let $H : \mathcal{X} \mapsto \mathbb{R}^m$ be a vector-valued class of functions. Then for any integer n and any $0 < \delta < 1$, with probability $1 - \delta$ over samples of length n, every $\mathbf{h}$ in H satisfies*

$$\mathbb{E}[\Phi(Y, \mathbf{h}(X))] \leq \hat{\mathbb{E}}[\phi(Y, \mathbf{h}(X))] + R_n(\tilde{\phi} \circ H) + \sqrt{\frac{8 \ln(2/\delta)}{n}}, \tag{A.4}$$

where $\tilde{\phi} \circ H$ is a class of functions defined by centered composition of ϕ with $\mathbf{h} \in H$, $\tilde{\phi} \circ \mathbf{h} = \phi(\mathbf{y}, \mathbf{h}(x)) - \phi(\mathbf{y}, 0)$.

Furthermore, Rademacher complexity can be bounded using Gaussian complexity: there are absolute constants c and C such that for every class H and every integer n,

$$cR_n(H) \leq G_n(H) \leq (C \ln n) R_n(H). \tag{A.5}$$

Let $H : \mathcal{X} \to \mathbb{R}^m$ be a class of functions that is the direct sum of real-valued classes $H_1, \ldots, H_m$. Then, for every integer n and every sample $(\mathbf{x}^1, \ldots, \mathbf{x}^n)$,

$$\hat{G}_n(\phi \circ H) \leq 2L(\phi) \sum_{i=1}^{m} \hat{G}_n(H_i), \tag{A.6}$$

where $L(\phi)$ is the Lipschitz constant of ϕ with respect to Euclidean distance. Finally, for the 2-norm-bounded linear class of functions, $H = \{\mathbf{x} \mapsto \mathbf{w}^\top \mathbf{f}(\mathbf{x}) \mid ||\mathbf{w}||_2 \leq B, ||\mathbf{f}(\mathbf{x})||_2 \leq 1\}$,

$$\hat{G}_n(H) \leq \frac{2B}{\sqrt{n}}. \tag{A.7}$$

A.1.1 Proof of Theorem 1

We will express our loss functions $\mathcal{L}_e$ and $\mathcal{L}_f$ and dominating loss functions $\mathcal{L}_e^\gamma$ and $\mathcal{L}_f^\gamma$, in the in terms of the framework above. We reproduce the definitions side-by-side in a slightly modified form below, where $m = \sum_{c \in \mathcal{F}} |\mathcal{Y}_c|$ and the γ-margin step-function dominates the step-function $r_\gamma(z) \geq \mathbf{1}\left[z \leq 0\right]$ by construction:

$$\mathcal{L}_f(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) = \mathbf{1}\left[\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) - \tau_{\mathbf{w},\alpha}(\mathbf{x}) \leq 0\right], \tag{A.8}$$

$$\mathcal{L}_f^\gamma(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) = r_\gamma(\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) - \tau_{\mathbf{w},\alpha}(\mathbf{x})), \tag{A.9}$$

$$\mathcal{L}_e(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) = \frac{1}{m} \sum_{c \in \mathcal{F}, \mathbf{y}_c \in \mathcal{Y}_c} \mathbf{1}\left[\tau_{\mathbf{w},\alpha}(\mathbf{x}) - \psi^\star_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c) \leq 0\right], \tag{A.10}$$

$$\mathcal{L}_e^\gamma(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) = \frac{1}{m} \sum_{c \in \mathcal{F}, \mathbf{y}_c \in \mathcal{Y}_c} r_\gamma(\tau_{\mathbf{w},\alpha}(\mathbf{x}) - \psi^\star_{\mathbf{w}}(\mathbf{x}, \mathbf{y}_c)). \tag{A.11}$$

We "vectorize" our scoring function $\mathbf{w}$ and assignments $\mathbf{y}$ by defining vector-valued functions, where the vectors are indexed by factor assignments, $\mathbf{y}_c$, with total dimension m.

Definition 8 (Vectorization)**.**

$$\mathbf{h}_{\mathbf{y}_c}(\mathbf{x}) \triangleq \mathbf{w}^\top \mathbf{f}_c(\mathbf{x}, \mathbf{y}_c) \tag{A.12}$$

$$\mathbf{v}_{\mathbf{y}_c}(\mathbf{y}') \triangleq \mathbf{1}\left[\mathbf{y}'_c = \mathbf{y}_c\right] \tag{A.13}$$

$$\psi_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) = \mathbf{h}(\mathbf{x})^\top \mathbf{v}(\mathbf{y}) \tag{A.14}$$

Clearly, the m-dimensional vector $\mathbf{h}(\mathbf{x})$ contains all the information needed to compute the max-marginals and threshold for a given example $\mathbf{x}$ (we assume α is fixed). Hence we can define the losses in the form of Theorem 3:

$$\Phi_f(\mathbf{y}, \mathbf{h}(\mathbf{x})) = \mathcal{L}_f(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) \tag{A.15}$$

$$\phi_f(\mathbf{y}, \mathbf{h}(\mathbf{x})) = \mathcal{L}_f^\gamma(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) \tag{A.16}$$

$$\Phi_e(\mathbf{y}, \mathbf{h}(\mathbf{x})) = \mathcal{L}_e(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) \tag{A.17}$$

$$\phi_e(\mathbf{y}, \mathbf{h}(\mathbf{x})) = \mathcal{L}_e^\gamma(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) \tag{A.18}$$

What remains is to calculate the Lipschitz constants of ϕ_f and ϕ_e.

Theorem 4. *$\phi_f(\mathbf{y}, \cdot)$ and $\phi_e(\mathbf{y}, \cdot)$ are Lipschitz (with respect to Euclidean distance on $\mathbb{R}^m$) with constant $\sqrt{2|\mathcal{F}|}/\gamma$ for all $\mathbf{y} \in \mathcal{Y}$.*

To prove Theorem 4, we bound Lipschitz constants of constituent functions of ϕ_f and ϕ_e.

Lemma 4. *Fix any $\mathbf{y} \in \mathcal{Y}$ and let $\phi_1 : \mathbb{R}^m \mapsto \mathbb{R}$ be defined as*

$$\phi_1(\mathbf{z}) = \mathbf{z}^\top \mathbf{v}(\mathbf{y}) - \max_{\mathbf{y}' \in \mathcal{Y}} \mathbf{z}^\top \mathbf{v}(\mathbf{y}').$$

Then $\phi_1(\mathbf{z}_1) - \phi_1(\mathbf{z}_2) \leq \sqrt{2|\mathcal{F}|}||\mathbf{z}_1 - \mathbf{z}_2||_2$ for any $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^m$.

Proof. For brevity of notation in the proof below, we define $\mathbf{v} = \mathbf{v}(\mathbf{y})$, $\mathbf{v}_1 = \mathbf{v}(\text{argmax}_{\mathbf{y}'} \mathbf{z}_1^\top \mathbf{v}(\mathbf{y}'))$ and $\mathbf{v}_2 = \mathbf{v}(\text{argmax}_{y'} \mathbf{z}_2^\top \mathbf{v}(y'))$, with ties broken arbitrarily but deterministically. Then,

$$\begin{aligned}
\phi_1(\mathbf{z}_1) - \phi_1(\mathbf{z}_2) &= \mathbf{z}_1^\top \mathbf{v} - \mathbf{z}_1^\top \mathbf{v}_1 - \mathbf{z}_2^\top \mathbf{v} + \mathbf{z}_2^\top \mathbf{v}_2 \\
&= (\mathbf{z}_2 - \mathbf{z}_1)^\top (\mathbf{v}_2 - \mathbf{v}) + \mathbf{z}_1^\top (\mathbf{v}_2 - \mathbf{v}_1) \\
&\leq (\mathbf{z}_2 - \mathbf{z}_1)^\top (\mathbf{v}_2 - \mathbf{v}) \\
&\leq ||\mathbf{z}_2 - \mathbf{z}_1||_2 ||\mathbf{v}_2 - \mathbf{v}||_2 \\
&\leq \sqrt{2|\mathcal{F}|}||\mathbf{z}_1 - \mathbf{z}_2||_2.
\end{aligned}$$

The last three steps follow (1) from the fact that $\mathbf{v}_1$ maximizes $\mathbf{z}_1^\top \mathbf{v}(\mathbf{y}')$ (so that $\mathbf{z}_1^\top (\mathbf{v}_2 - \mathbf{v}_1)$ is negative), (2) from the Cauchy-Schwarz inequality, and (3) from the fact that there are $|\mathcal{F}|$ factors, each of which can contribute at most a single non-zero entry in $\mathbf{v}$ or $\mathbf{v}_2$. □

Lemma 5. *Fix any $\mathbf{y} \in \mathcal{Y}$ and let $\phi_2 : \mathbb{R}^m \mapsto \mathbb{R}$ be defined as*

$$\phi_2(\mathbf{z}) = \mathbf{z}^\top \mathbf{v}(\mathbf{y}) - \frac{1}{m} \sum_{c \in \mathcal{F}, \mathbf{y}'_c \in \mathcal{Y}_c} \max_{\mathbf{y}'' : \mathbf{y}''_c = \mathbf{y}'_c} \mathbf{z}^\top \mathbf{v}(\mathbf{y}'').$$

Then $\phi_2(\mathbf{z}_1) - \phi_2(\mathbf{z}_2) \leq \sqrt{2|\mathcal{F}|}||\mathbf{z}_1 - \mathbf{z}_2||_2$ for any $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^m$.

Proof. Let $\mathbf{v} = \mathbf{v}(\mathbf{y}), \mathbf{v}_{1\mathbf{y}'_c} = \mathbf{v}(\text{argmax}_{\mathbf{y}'':\mathbf{y}''=\mathbf{y}'_c} \mathbf{z}_1^\top \mathbf{v}(\mathbf{y}''))$ and $\mathbf{v}_{2\mathbf{y}'_c} = (\text{argmax}_{\mathbf{y}'':\mathbf{y}''=\mathbf{y}'_\mathbf{c}} \mathbf{z}_\mathbf{2}^\top \mathbf{v}(\mathbf{y}''))$.

$$\begin{aligned}
\phi_2(\mathbf{z}_1) - \phi_2(\mathbf{z}_2) &= \frac{1}{m} \sum_{c \in \mathcal{F}, \mathbf{y}'_c \in \mathcal{Y}_c} \mathbf{z}_1^\top \mathbf{v} - \mathbf{z}_1^\top \mathbf{v}_{1\mathbf{y}'_c} - \mathbf{z}_2^\top \mathbf{v} + \mathbf{z}_2^\top \mathbf{v}_{2\mathbf{y}'_c} \\
&= \frac{1}{m} \sum_{c \in \mathcal{F}, \mathbf{y}'_c \in \mathcal{Y}_c} (\mathbf{z}_2 - \mathbf{z}_1)^\top (\mathbf{v}_{2\mathbf{y}'_c} - \mathbf{v}) + \mathbf{z}_1^\top (\mathbf{v}_{2\mathbf{y}'_c} - \mathbf{v}_{1\mathbf{y}'_c}) \\
&\leq \frac{1}{m} \sum_{c \in \mathcal{F}, \mathbf{y}'_c \in \mathcal{Y}_c} (\mathbf{z}_2 - \mathbf{z}_1)^\top (\mathbf{v}_{2\mathbf{y}'_c} - \mathbf{v}) \\
&\leq \frac{1}{m} \sum_{c \in \mathcal{F}, \mathbf{y}'_c \in \mathcal{Y}_c} \sqrt{2|\mathcal{F}|} ||\mathbf{z}_1 - \mathbf{z}_2||_2 = \sqrt{2|\mathcal{F}|} ||\mathbf{z}_1 - \mathbf{z}_2||_2.
\end{aligned}$$

The inequalities follow using a similar argument to previous lemma, but made separately for each $\mathbf{y}'_c$. □

Lemma 6. *Fix any* $\mathbf{y} \in \mathcal{Y}$ *and let*

$$\phi_3(\mathbf{z}) = \alpha\phi_1(\mathbf{z}) + (1-\alpha)\phi_2(\mathbf{z}) = \overbrace{\underbrace{\mathbf{z}^\top \mathbf{v}(\mathbf{y})}^{\psi_\mathbf{w}(\mathbf{x},\mathbf{y})} - \left(\alpha \max_{\mathbf{y}'} \mathbf{z}^\top \mathbf{v}(\mathbf{y}') + \frac{1-\alpha}{m} \sum_{\mathbf{y}'_c} \max_{\mathbf{y}'':\mathbf{y}''_c=\mathbf{y}'_c} \mathbf{z}^\top \mathbf{v}(\mathbf{y}'') \right)}^{\tau_{\mathbf{w},\alpha}(\mathbf{x})},$$

where the over-braces show the relationship to the score of the correct label sequence and the threshold, assuming $\mathbf{z} = \mathbf{h}(\mathbf{x})$. *Then* $\phi_3(\mathbf{z}_1) - \phi_3(\mathbf{z}_2) \leq \sqrt{2|\mathcal{F}|} ||\mathbf{z}_1 - \mathbf{z}_2||_2$ *for any* $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^m$ *and the Lipschitz constant of* $\phi_f = r_\gamma \circ \phi_3$ *is bounded by* $\sqrt{2|\mathcal{F}|}/\gamma$.

Proof. Combining two previous lemmas we have that

$$\phi_3(\mathbf{z}_1) - \phi_3(\mathbf{z}_2) = \alpha(\phi_1(\mathbf{z}_1) - \phi_1(\mathbf{z}_2)) + (1-\alpha)(\phi_2(\mathbf{z}_1) - \phi_2(\mathbf{z}_2)) \leq \sqrt{2|\mathcal{F}|} ||\mathbf{z}_1 - \mathbf{z}_2||_2.$$

To show that ϕ_f is Lipschitz continuous with constant $\sqrt{2|\mathcal{F}|}/\gamma$, we note that $\phi_f = r_\gamma \circ \phi_3$ so $L(\phi_f) = L(r_\gamma) \cdot L(\phi_3) \leq \sqrt{2|\mathcal{F}|}/\gamma$. □

Next, we show that ϕ_e is Lipschitz continuous with the same constant.

Lemma 7. *Fix $c \in \mathcal{F}$ and $\mathbf{y}_c \in \mathcal{Y}$ and let $\phi_{[\mathbf{y}_c]} : \mathbb{R}^m \mapsto \mathbb{R}$ be defined as*

$$\phi_{[\mathbf{y}_c]}(\mathbf{z}) = \overbrace{\left(\max_{\mathbf{y}':\mathbf{y}'_c=\mathbf{y}_c} \mathbf{z}^\top \mathbf{v}(\mathbf{y}')\right)}^{\psi^\star_{\mathbf{w}}(\mathbf{x},\mathbf{y}_c)} - \overbrace{\left(\alpha \max_{\mathbf{y}'} \mathbf{z}^\top \mathbf{v}(\mathbf{y}') + \frac{1-\alpha}{m} \sum_{\mathbf{y}'_{c'}} \max_{\mathbf{y}'':\mathbf{y}''_{c'}=\mathbf{y}'_{c'}} \mathbf{z}^\top \mathbf{v}(\mathbf{y}'')\right)}^{\tau_{\mathbf{w},\alpha}(\mathbf{x})},$$

where the over-braces show the relationship to max-marginal of $\mathbf{y}_c$ and the threshold and, assuming $\mathbf{z} = \mathbf{h}(\mathbf{x})$. Then $\phi_{[\mathbf{y}_c]}(\mathbf{z}_1) - \phi_{[\mathbf{y}_c]}(\mathbf{z}_2) \leq \sqrt{2|\mathcal{F}|}||\mathbf{z}_1 - \mathbf{z}_2||_2$ for any $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^m$.

Proof. We once again apply the trick from the proof of Lemma 4. Let

$$\begin{aligned}
\mathbf{v}_1 &= \mathbf{v}(\operatorname*{argmax}_{\mathbf{y}'} \mathbf{z}_1^\top \mathbf{v}(\mathbf{y}')), \qquad \mathbf{v}_2 = \mathbf{v}(\operatorname*{argmax}_{\mathbf{y}'} \mathbf{z}_2^\top \mathbf{v}(\mathbf{y}')), \\
\mathbf{v}_{1\mathbf{y}'_{c'}} &= \mathbf{v}(\operatorname*{argmax}_{\mathbf{y}'':\mathbf{y}''=\mathbf{y}'_{c'}} \mathbf{z}_1^\top \mathbf{v}(\mathbf{y}'')), \quad \mathbf{v}_{2\mathbf{y}'_{c'}} = \mathbf{v}(\operatorname*{argmax}_{\mathbf{y}'':\mathbf{y}''=\mathbf{y}'_{c'}} \mathbf{z}_2^\top \mathbf{v}(\mathbf{y}'')).
\end{aligned}$$

Then, we have:

$$\begin{aligned}
\phi_{[\mathbf{y}_c]}(\mathbf{z}_1) - \phi_{[\mathbf{y}_c]}(\mathbf{z}_2) &= (\mathbf{z}_1^\top \mathbf{v}_{1\mathbf{y}_c} - \mathbf{z}_2^\top \mathbf{v}_{2\mathbf{y}_c}) + \alpha(\mathbf{z}_2^\top \mathbf{v}_2 - \mathbf{z}_1^\top \mathbf{v}_1) + \\
&\quad \frac{1-\alpha}{m} \sum_{\mathbf{y}'_{c'}} (\mathbf{z}_2^\top \mathbf{v}_{2\mathbf{y}'_{c'}} - \mathbf{z}_1^\top \mathbf{v}_{1\mathbf{y}'_{c'}}) \\
&\leq (\mathbf{z}_1 - \mathbf{z}_2)^\top \mathbf{v}_{1\mathbf{y}_c} + \alpha(\mathbf{z}_2 - \mathbf{z}_1)^\top \mathbf{v}_2 + \frac{1-\alpha}{m} \sum_{\mathbf{y}'_{c'}} (\mathbf{z}_2 - \mathbf{z}_1)^\top \mathbf{v}_{2\mathbf{y}'_{c'}} \\
&= \frac{1}{m} \sum_{\mathbf{y}'_{c'}} (\mathbf{z}_1 - \mathbf{z}_2)^\top \left(\mathbf{v}_{1\mathbf{y}_c} - \alpha \mathbf{v}_2 - (1-\alpha)\mathbf{v}_{2\mathbf{y}'_{c'}}\right) \\
&\leq \frac{1}{m} \sum_j \sqrt{2|\mathcal{F}|}||\mathbf{z}_1 - \mathbf{z}_2||_2 = \sqrt{2|\mathcal{F}|}||\mathbf{z}_1 - \mathbf{z}_2||_2.
\end{aligned}$$

Here once again we have condensed the argument similar to Lemma 4.

□

Finally, we note that $\phi_e(\mathbf{z}) = 1/m \sum_i r_\gamma(\phi_{[\mathbf{y}_c]}(\mathbf{z}))$. Therefore $L(\phi_e) = 1/m \sum_i \sqrt{2|\mathcal{F}|}/\gamma = \sqrt{2|\mathcal{F}|}/\gamma$, thus completing the proof of Theorem 4. Now turning back to Theorem 1, we note that the class of functions H we are working with is the direct sum of m linear classes each bounded by norm B. Hence we complete the proof of Theorem 1, by using Theorem 3, with $R_n(\tilde{\phi}_f \circ H) = R_n(\tilde{\phi}_e \circ H) \leq \frac{cmB\sqrt{|\mathcal{F}|}}{\gamma\sqrt{n}}$ for some constant c.

A.1.2 Proof of Theorem 2

We define

$$\Phi_{joint}(\mathbf{y}, \mathbf{h}(\mathbf{x})) \triangleq \mathcal{L}_{joint}(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) = \mathbf{1}\left[\left(\sum_p \psi_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y}) - \tau_{\mathbf{w}_p,\alpha}(\mathbf{x})\right) \leq 0\right],$$

$$\phi_{joint}(\mathbf{y}, \mathbf{h}(\mathbf{x})) \triangleq \mathcal{L}^{\gamma}_{joint}(\mathbf{x}, \mathbf{y}; \mathbf{w}, \alpha) = r_\gamma\left(\sum_p \psi_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y}) - \tau_{\mathbf{w}_p,\alpha}(\mathbf{x})\right)$$

Once again, we fix any $\mathbf{y} \in \mathcal{Y}$ and for each p, let (similar to Lemma 6)

$$\phi_3(\mathbf{z}_p) = \overbrace{\mathbf{z}_p^\top \mathbf{v}(\mathbf{y})}^{\psi_{\mathbf{w}_p}(\mathbf{x},\mathbf{y})} - \overbrace{\left(\alpha \max_{\mathbf{y}'} \mathbf{z}_p^\top \mathbf{v}(\mathbf{y}') + \frac{1-\alpha}{m} \sum_{\mathbf{y}'_c} \max_{\mathbf{y}'':\mathbf{y}''_c=\mathbf{y}'_c} \mathbf{z}_p^\top \mathbf{v}(\mathbf{y}'')\right)}^{\tau_{\mathbf{w}_p,\alpha}(\mathbf{x})},$$

where the over-braces show the relationship to the score of the correct label sequence under model p and the threshold for model p, assuming $\mathbf{z}_p = \mathbf{h}_p(x)$ of model p.

Then $\phi_{joint}(y, \mathbf{h}(\mathbf{x})) = r_\gamma(\sum_p \phi_3(\sum_p \mathbf{z}_p))$ has Lipschitz constant $\sqrt{2|\mathcal{F}|}P/\gamma$, since we can apply Lemma 6 for each p, and $\phi_{joint}(\mathbf{y}, \mathbf{h}(\mathbf{x})) = r_\gamma\left(\sum_p \psi_{\mathbf{w}_p}(\mathbf{x}, \mathbf{y}) - \tau_{\mathbf{w}_p,\alpha}(\mathbf{x})\right)$ has Lipschitz constant at most $\sqrt{2|\mathcal{F}|}P/\gamma$ because if composition with r_γ and the sum of P identical terms. In Theorem 2, our function class H is the direct sum of $m * P$ linear classes each bounded by norm B/P, hence $R_n(\tilde{\phi}_{joint} \circ H) \leq \frac{cmPB\sqrt{|\mathcal{F}|}}{\gamma\sqrt{n}}$ for some constant c.

Bibliography

A. Agarwal, J. Duchi, P. Bartlett, and C. Levrard. Oracle ineqaulities for computationally budgeted model selection. In *Proc. COLT*, 2011.

H. Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716 – 723, dec 1974.

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

A. Barron, L. Birgé, and P. Massart. Risk bounds for model selection via penalization. *Probability theory and related fields*, 113(3):301–413, 1999.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *JMLR*, 2002.

P. L. Bartlett, S. Boucheron, and G. Lugosi. Model selection and error estimation. *Machine Learning*, 48:85–113, 2002.

V Bayer-Zubek. Learning diagnostic policies from examples by systematic search. In *UAI*, 2004.

A. Bedagkar-Gala and S.K. Shah. Joint modeling of algorithm behavior and image quality for algorithm performance prediction. In *BMVC*, 2010.

S Bengio, J Weston, and D Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2010.

D. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

M Bilgic and L Getoor. Voila: Efficient feature-value acquisition for classification. In *AAAI*, 2007.

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *NIPS*, 2008.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 2001.

P. Buehler, M. Everingham, D.P. Huttenlocher, and A. Zisserman. Upper body detection and tracking in extended signing sequences. *IJCV*, 95:180–197, 2011.

R Busa-Fekete, D Benbouzid, and B Kegl. Fast classification using sparse decision dags. In *ICML*, 2012.

X. Carreras, M. Collins, and T. Koo. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proc. CoNLL*, 2008.

E. Charniak. A maximum-entropy-inspired parser. In *Proc. NAACL*, 2000.

M. Chen, Z. Xu, K.Q. Weinberg, O. Chapelle, and D. Kedem. Classifier cascade for minimizing feature evaluation cost. In *AISATATS*, 2012.

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

K. Crammer, J. Kandola, and Y. Singer. Online classification on a budget. In *NIPS*. MIT Press, 2003.

H. Daumé, J. Langford, and D. Marcu. Search-based structured prediction. *Machine Learning*, 75(3):297–325, 2009. URL `http://dx.doi.org/10.1007/s10994-009-5106-x`.

O. Dekel, S. Shalev-Shwartz, and Y. Singer. The Forgetron: A kernel-based Perceptron on a budget. *SIAM Journal on Computing*, 37(5):1342–1372, 2008.

J Deng, S Satheesh, A Berg, and L Fei-Fei. Fast and balanced: Efficient label tree learning for large scale object recognition. In *NIPS*, 2011.

L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Verlag, 1996.

P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Proc. CVPR*, 2010.

P.F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2), 2004.

V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *Proc. CVPR*, 2008.

F. Fleuret and D. Geman. Coarse-to-fine face detection. *IJCV*, 41(1/2), 2001.

T. Gao and D. Koller. Active classification based on value of classifier. In *NIPS*, 2011.

A. Grubb and D. Bagnell. Speedboost: Anytime prediction with uniform near-optimality. In *AISTATS*, 2012.

Z Harchaoui, M Douze, M Paulin, M Dudik, and J Malick. Large-scale image classification with trace-norm regularization. In *CVPR*, 2012.

H. He, H. Daumé III, and J. Eisner. Imitation learning by coaching. In *NIPS*, 2012.

B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 1981.

R. A Howard. Information value theory. *Systems Science and Cybernetics, IEEE Transactions on*, 2(1):22–26, 1966.

N. Jammalamadaka, A. Zisserman, M. Eichner, V. Ferrari, and C.V.Jawahar. Has my algorithm succeeded? An evaluator for human pose estimators. In *ECCV*, 2012.

S Ji and L Carin. Cost-sensitive feature acquisition and classification. *Pattern Recognition*, 2007.

J. Jiang, A. Teichart, H. Daumé III, and J. Eisner. Learned prioritization for trading off accuracy and speed. In *NIPS*, 2012.

T. Joachims, T. Finley, and C. N. J. Yu. Cutting-plane training of structural svms. *Machine Learning*, 2009.

P Kanani and P Melville. Prediction-time active feature-value acquisition for cost-effective customer targeting. In *NIPS*, 2008.

R. Kassel. *A Comparison of Approaches to On-line Handwritten Character Recognition*. PhD thesis, Massachusetts Institute of Technology, 1995.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *Proc. ICCV*, 2007.

A Krause and C Guestrin. Near-optimal value of information in graphical models. In *UAI*, 2005.

Andreas Krause and Carlos Guestrin. Optimal value of information in graphical models. *Journal of Artificial Intelligence Research (JAIR)*, 35:557–591, 2009.

S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *ICML*, 2013.

L. Ladický, P.H.S. Torr, and A. Zisserman. Human pose estimation using a joint pixel-wise and part-wise formulation. In *CVPR*, 2013.

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

M. Lagoudakis and R. Parr. Least-squares policy iteration. *JMLR*, 2003.

P. Lanchantin and X. Rodet. Dynamic model selection for spectral voice conversion. In *Interspeech*, 2010.

J Langford, L Li, and T Zhang. Sparse online learning via truncated gradient. *JMLR*, 2009.

L. Lefakis and F. Fleuret. Joint cascade optimization using a product of boosted classifiers. *NIPS*, 2010.

Dennis V Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, pages 986–1005, 1956.

B Liu, F Sadeghi, M Tappen, O Shamir, and C Liu. Probabilistic label trees for efficient large scale image classification. In *CVPR*, 2013.

C. Liu. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. PhD thesis, MIT, 2009.

M. Marcus, S. Santorini, and M. Marcinkiewicz. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

R. McDonald, F. PerF. Pereira. Ribarov, and J. Hajic. Non-projective dependency parsing using spanning tree algorithms. In *HLT/EMNLP*, 2005.

R.J. McEliece, D.J.C. MacKay, and J.F. Cheng. Turbo decoding as an instance of Pearl's belief propagation algorithm. *J. on Selected Areas in Communications*, 16(2):140–152, 1998.

K.P. Murphy, Y. Weiss, and M.I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proc. UAI*, pages 467–475, 1999.

S. Nowozin and C. H. Lampert. Structured prediction and learning in computer vision. In *Foundations and Trends in Computer Graphics and Vision*, volume 6. 2011.

D. Park and D. Ramanan. N-best maximal decoders for part models. In *ICCV*, 2011.

J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

M. Pedersoli, A. Vedaldi, and J. Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *CVPR*, 2011.

S. Petrov. *Coarse-to-Fine Natural Language Processing*. PhD thesis, University of California at Bekeley, 2009.

C. Raphael. Coarse-to-fine dynamic programming. *PAMI*, 2001.

V.C. Raykar, B. Krishnapuram, and S. Yu. Designing efficient cascaded classifiers: tradeoff between accuracy and cost. In *SIGKDD*, 2010.

A. Rush and S. Petrov. Vine pruning for efficient multi-pass dependency parsing. In *Proc. NAACL*, 2012.

M Saberian and N Vasconcelos. Boosting classifier cascades. In *NIPS*, 2010.

B. Sapp and B. Taskar. MODEC: Multimodal decomposable models for human pose estimation. In *CVPR*, 2013.

B. Sapp, A. Toshev, and B. Taskar. Cascaded models for articulated pose estimation. In *ECCV*, 2010.

B. Sapp, D. Weiss, and B. Taskar. Parsing human motion with stretchable models. In *CVPR*, 2011.

S. Shalev-Shwartz and N. Srebro. SVM optimization: inverse dependence on training set size. In *International Conference on Machine learning*, pages 928–935, 2008.

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

V Sheng and C Ling. Feature value acquisition in testing: A sequential batch test algorithm. In *ICML*, 2006.

D. Sontag. *Approximate Inference in Graphical Models using LP Relaxations*. PhD thesis, Massachusetts Institute of Technology, 2010.

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

B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *ICML*, 2005.

K. Trapeznikov and V. Saligrama. Supervised sequential classification under budget constraints. In *AISTATS*, 2013.

K Trapeznikov, V Saligrama, and D Castanon. Multi-stage classifier design. *Machine Learning*, 2013.

V. Vapnik and A. Chervonenkis. *Theory of pattern recognition*. Nauka, 1974.

P. Viola and M. Jones. Robust real-time object detection. *IJCV*, 57(2):137–154, 2002.

C. Watkins and P. Dayan. Q-learning. *Machine learning*, 1992.

D. Weiss and B. Taskar. Structured prediction cascades. In *AISTATS*, 2010.

D. Weiss and B. Taskar. Dynamic structured model selection. In *ICCV*, 2013.

D. Weiss, B. Sapp, and B. Taskar. Sidestepping intractable inference with structured ensemble cascades. In *Proc. NIPS*, 2010.

Z Xu, K Weinberger, and O Chapelle. The greedy miser: Learning under test-time budgets. In *ICML*, 2012.

Y. Yang and D. Ramanan. Articulated pose estimation using flexible mixtures of parts. In *Proc. CVPR*, 2011.