Learning Structured Prediction Models: A Large Margin Approach

Ben Taskar TASKAR@CS.BERKELEY.EDU
Electrical Engineering and Computer Science, UC Berkeley, Berkeley, CA 94720

Vassil Chatalbashev VASCO@CS.STANFORD.EDU
Daphne Koller KOLLER@CS.STANFORD.EDU
Computer Science, Stanford University, Stanford, CA 94305

Carlos Guestrin GUESTRIN@CS.CMU.EDU
Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213

Abstract

We consider large margin estimation in a broad range of prediction models where inference involves solving combinatorial optimization problems, for example, weighted graph-cuts or matchings. Our goal is to learn parameters such that inference using the model reproduces correct answers on the training data. Our method relies on the expressive power of convex optimization problems to compactly capture inference or solution optimality in structured prediction models. Directly embedding this structure within the learning formulation produces concise convex problems for efficient estimation of very complex and diverse models. We describe experimental results on a matching task, disulfide connectivity prediction, showing significant improvements over state-of-the-art methods.

1. Introduction

Structured prediction problems arise in many tasks where multiple interrelated decisions must be weighed against each other to arrive at a globally satisfactory and consistent solution. In natural language processing, we often need to construct a global, coherent analysis of a sentence, such as its corresponding part-of-speech sequence, parse tree, or translation into another language. In computational biology, we analyze genetic sequences to predict 3D structure of proteins, find global alignment of related DNA strings, and recognize functional portions of a genome. In computer vision, we segment complex objects in cluttered scenes, reconstruct 3D shapes from stereo and video, and track motion of articulated bodies.

Many prediction tasks are modeled by combinatorial optimization problems; for example, alignment of 2D shapes using weighted bipartite point matching (Belongie et al., 2002), disulfide connectivity prediction using weighted non-bipartite matchings (Baldi et al., 2004), clustering using spanning trees and graph cuts (Duda et al., 2000), and other combinatorial and graph structures. We define a *structured model* very broadly, as a scoring scheme over a set of combinatorial structures and a method of finding the highest scoring structure. The score of a model is a function of the weights of vertices, edges, or other parts of a structure; these weights are often represented as parametric functions of a set of input features. The focus of this paper is the task of learning this parametric scoring function. Our training data consists of instances labeled by a desired combinatorial structure (matching, cut, tree, etc.) and a set of input features to be used to parameterize the scoring function. Informally, the goal is to find parameters of the scoring function such that the highest scoring structures are as close as possible to the desired structures on the training instances.

Following the lines of the recent work on maximum margin estimation for probabilistic models (Collins, 2002; Altun et al., 2003; Taskar et al., 2003), we present a discriminative estimation framework for structured models based on the large margin principle underlying support vector machines. The large-margin criterion provides an alternative to probabilistic, likelihood-based estimation methods by concentrating directly on the robustness of the decision boundary of a model. Our framework defines a suite of efficient learning algorithms that rely on the expressive power of convex optimization problems to compactly capture inference or solution optimality in structured

Appearing in *Proceedings of the 22st International Conference on Machine Learning*, Bonn, Germany, 2005.

models. We present extensive experiments on disulfide connectivity in protein structure prediction showing superior performance to state-of-the-art methods.

2. Structured models

As a particularly simple and relevant example, consider modeling the task of assigning reviewers to papers as a maximum weight bipartite matching problem, where the weights represent the "expertise" of each reviewer for each paper. More specifically, suppose we would like to have R reviewers per paper, and that each reviewer be assigned at most P papers. For each paper and reviewer, we have a score s_{jk} indicating the qualification level of reviewer j for evaluating paper k. Our objective is to find an assignment for reviewers to papers that maximizes the total weight. We represent a matching using a set of binary variables y_{jk} that are set to 1 if reviewer j is assigned to paper k, and 0 otherwise. The score of an assignment is the sum of edge scores: $s(\mathbf{y}) = \sum_{jk} s_{jk} y_{jk}$. We define $\mathcal{Y}$ to be the set of bipartite matchings for a given number of papers, reviewers, R and P. The maximum weight bipartite matching problem, $\arg\max_{\mathbf{y}\in\mathcal{Y}} s(\mathbf{y})$, can be solved using a combinatorial algorithm or the following linear program:

$$\begin{aligned} \max \quad & \sum_{jk} s_{jk} y_{jk} & (1) \\ \text{s.t.} \quad & \sum_j y_{jk} = R, \quad \sum_k y_{jk} \le P, \quad 0 \le y_{jk} \le 1. \end{aligned}$$

This LP is guaranteed to have integral (0/1) solutions (as long as P and R are integers) for any scoring function $s(\mathbf{y})$ (Schrijver, 2003).

The quality of the assignment found depends critically on the choice of weights that define the objective. A simple scheme could measure the "expertise" as the percent of word overlap in the reviewer's home page and the paper's abstract. However, we would want to weight certain words more heavily (words that are relevant to the subject and infrequent). Constructing and tuning the weights for a problem is a difficult and time-consuming process to perform by hand.

Let *webpage*(j) denote the bag of words occurring in the home page of reviewer j and *abstract*(k) denote the bag of words occurring in the abstract of paper k. Then let $\mathbf{x}_{jk}$ denote the intersection of the bag of words occurring in *webpage*$(j) \cap$*abstract*(k). We can let the score s_{jk} be simply $s_{jk} = \sum_d w_d \mathbb{I}(word_d \in \mathbf{x}_{jk})$, a weighted combination of overlapping words (where $\mathbb{I}(\cdot)$ is the indicator function). Define $f_d(\mathbf{x}_{jk}) = \mathbb{I}(word_d \in \mathbf{x}_{jk})$ and $f_d(\mathbf{x}, \mathbf{y}) = \sum_{jk} y_{jk} \mathbb{I}(word_d \in \mathbf{x}_{jk})$, the number of times word d was in both the web page of a reviewer and the abstract of the paper that were matched in $\mathbf{y}$. We can represent the objective $s(\mathbf{y})$ as a weighted combination of a set of features $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$, where $\mathbf{w}$ is the set of parameters, $\mathbf{f}(\mathbf{x}, \mathbf{y})$ is the set of features. We develop a large margin framework for learning the parameters of such a model from training data, in our example, paper-reviewer assignments from previous years.

In general, we consider prediction problems in which the input $\mathbf{x} \in \mathcal{X}$ is an arbitrary structured object and the output is a vector of values $\mathbf{y} = (y_1, \ldots, y_{L_\mathbf{x}})$, for example, a matching or a cut in the graph. We assume that the length $L_\mathbf{x}$ and the structure of $\mathbf{y}$ depend deterministically on the input $\mathbf{x}$. In our bipartite matching example, the output space is defined by the number of papers and reviewers as well as R and P. Denote the output space for a given input $\mathbf{x}$ as $\mathcal{Y}(\mathbf{x})$ and the entire output space is $\mathcal{Y} = \bigcup_{\mathbf{x}\in\mathcal{X}} \mathcal{Y}(\mathbf{x})$. We assume that we can define the output space for a structured example $\mathbf{x}$ using a set of constraint functions: $g_d(\mathbf{x}, \mathbf{y}) : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ such that $\mathcal{Y}(\mathbf{x}) = \{\mathbf{y} : \mathbf{g}(\mathbf{x}, \mathbf{y}) \le 0\}$.

The class of structured prediction models $\mathcal{H}$ we consider is the linear family:

$$h_\mathbf{w}(\mathbf{x}) = \underset{\mathbf{y}\,:\,\mathbf{g}(\mathbf{x},\mathbf{y})\le 0}{\arg\max}\; \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}), \tag{2}$$

where $\mathbf{f}(\mathbf{x}, \mathbf{y})$ is a vector of functions $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}^n$. This formulation is very general; clearly, for many $\mathbf{f}, \mathbf{g}$ pairs, finding the optimal $\mathbf{y}$ is intractable. We focus our attention on models where this optimization problem can be solved in polynomial time. Such problems include: probabilistic models such as certain types of Markov networks and context-free grammars; combinatorial optimization problems such as min-cut and matching; and convex optimization problems such as linear, quadratic and semi-definite programming. In intractable cases, such as certain types of matching and graph cuts, we can use an *approximate* polynomial time optimization procedure that provides upper/lower bounds on the solution.

3. Max-margin estimation

Our input consists of a set of training instances $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^m$, where each instance consists of a structured object $\mathbf{x}^{(i)}$ (such as a graph) and a target solution $\mathbf{y}^{(i)}$ (such as a matching). We develop methods for finding parameters $\mathbf{w}$ such that:

$$\underset{\mathbf{y}\in\mathcal{Y}^{(i)}}{\arg\max}\, \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y}) \;\approx\; \mathbf{y}^{(i)}, \quad \forall i,$$

where $\mathcal{Y}^{(i)} = \{\mathbf{y} : \mathbf{g}(\mathbf{x}^{(i)}, \mathbf{y}) \le 0\}$. Note that the solution space $\mathcal{Y}^{(i)}$ depends on the structured object $\mathbf{x}^{(i)}$; for example, the space of possible matchings depends on the precise set of nodes and edges in the graph, as well as on the parameters R and P.

We describe two general approaches to solving this problem, and apply them to bipartite and non-bipartite matchings. Both of these approaches define a convex optimization problem for finding such parameters $\mathbf{w}$. This formulation provides an effective and exact polynomial-time algorithm for many variants of this problem. Moreover, the reduction of this task to a standard convex optimization problem allows the use of highly optimized off-the-shelf software.

Our framework extends the max-margin formulation for Markov networks (Taskar et al., 2003; Taskar et al., 2004a) and context free grammars (Taskar et al., 2004b), and is similar to other formulations (Altun et al., 2003; Tsochantaridis et al., 2004).

As in the univariate prediction, we measure the error of prediction using a loss function $\ell(\mathbf{y}^{(i)}, \mathbf{y})$. In structured problems, where we are jointly predicting multiple variables, the loss is often not just the simple 0-1 loss. For structured prediction, a natural loss function is a kind of Hamming distance between $\mathbf{y}^{(i)}$ and $h(\mathbf{x}^{(i)})$: the number of variables predicted incorrectly.

We can express the requirement that the true structure $\mathbf{y}^{(i)}$ is the optimal solution with respect to $\mathbf{w}$ for each instance i as:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 & (3) \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}), \quad \forall i, \ \ \forall \mathbf{y} \in \mathcal{Y}^{(i)}, \end{aligned}$$

where $\ell_i(\mathbf{y}) = \ell(\mathbf{y}^{(i)}, \mathbf{y})$, and $\mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y})$.

We can interpret $\frac{1}{||\mathbf{w}||}\mathbf{w}^\top[\mathbf{f}_i(\mathbf{y}^{(i)}) - \mathbf{f}_i(\mathbf{y})]$ as the *margin* of $\mathbf{y}^{(i)}$ over another $\mathbf{y} \in \mathcal{Y}^{(i)}$. The constraints enforce $\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \geq \ell_i(\mathbf{y})$, so minimizing $||\mathbf{w}||$ maximizes the smallest such margin, scaled by the loss $\ell_i(\mathbf{y})$. We assume for simplicity that the features are rich enough to satisfy the constraints, which is analogous to the separable case formulation in SVMs. We can add slack variables to deal with the non-separable case; we omit details for lack of space.

The problem with Eq. (3) is that it has $\sum_i |\mathcal{Y}^{(i)}|$ linear constraints, which is generally exponential in L_i, the number of variables in $\mathbf{y}_i$. We present two equivalent formulations that avoid this exponential blow-up for several important structured models.

4. Min-max formulation

We can rewrite Eq. (3) by substituting a single max constraint for each i instead of $|\mathcal{Y}^{(i)}|$ linear ones:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 & (4) \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) \geq \max_{\mathbf{y} \in \mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})], \quad \forall i. \end{aligned}$$

The above formulation is a convex quadratic program in $\mathbf{w}$, since $\max_{\mathbf{y} \in \mathcal{Y}^{(i)}}[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$ is convex in $\mathbf{w}$ (maximum of affine functions is a convex function).

The key to solving Eq. (4) efficiently is the *loss-augmented inference* $\max_{\mathbf{y} \in \mathcal{Y}^{(i)}}[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$. This optimization problem has precisely the same form as the prediction problem whose parameters we are trying to learn — $\max_{\mathbf{y} \in \mathcal{Y}^{(i)}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$ — but with an additional term corresponding to the loss function. Even if $\max_{\mathbf{y} \in \mathcal{Y}^{(i)}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$ can be solved in polynomial time using convex optimization, tractability of the loss-augmented inference also depends on the form of the loss term $\ell_i(\mathbf{y})$. In this paper, we assume that the loss function decomposes over the variables in $\mathbf{y}^{(i)}$. A natural example of such a loss function is the Hamming distance, which simply counts the number of variables in which a candidate solution $\mathbf{y}$ differs from the target output $\mathbf{y}^{(i)}$.

Assume that we can reformulate loss-augmented inference as a convex optimization problem in terms of a set of variables μ_i, with an objective $\widetilde{f}_i(\mathbf{w}, \mu_i)$ concave in μ_i and subject to convex constraints $\widetilde{\mathbf{g}}_i(\mu_i)$:

$$\max_{\mathbf{y} \in \mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})] = \max_{\mu_i : \widetilde{\mathbf{g}}_i(\mu_i) \leq 0} \widetilde{f}_i(\mathbf{w}, \mu_i). \quad (5)$$

We call such formulation *concise* if the number of variables μ_i and constraints $\widetilde{\mathbf{g}}_i(\mu_i)$ is polynomial in L_i, the number of variables in $\mathbf{y}^{(i)}$. Note that $\max_{\mu_i : \widetilde{\mathbf{g}}_i(\mu_i) \leq 0} \widetilde{f}_i(\mathbf{w}, \mu_i)$ must be convex in $\mathbf{w}$, as Eq. (4) is. Likewise, we can assume that it is feasible and bounded if Eq. (4) is.

For example, the Hamming loss for bipartite matchings counts the number of different edges in the matchings $\mathbf{y}$ and $\mathbf{y}^{(i)}$:

$$\ell_i^H(\mathbf{y}) = \sum_{jk} \mathbb{I}(y_{jk} \neq y_{jk}^{(i)}) = RN_p^{(i)} - \sum_{jk} y_{jk} y_{jk}^{(i)},$$

where the last equality follows from the fact that any valid matching for training example i has R reviewers for $N_p^{(i)}$ papers. Thus, the loss-augmented matching problem can be then written as an LP in μ_i similar to Eq. (1) (without the constant term $RN_p^{(i)}$):

$$\begin{aligned} \max \quad & \sum_{jk} \mu_{i,jk}[\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}^{(i)}) - y_{jk}^{(i)}] \\ \text{s.t.} \quad & \sum_j \mu_{i,jk} = R, \quad \sum_k \mu_{i,jk} \leq P, \quad 0 \leq \mu_{i,jk} \leq 1. \end{aligned}$$

In terms of Eq. (5), $\widetilde{f}_i$ and $\widetilde{\mathbf{g}}_i$ are affine in μ_i: $\widetilde{f}_i(\mathbf{w}, \mu_i) = RN_p^{(i)} + \sum_{jk} \mu_{i,jk}[\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}^{(i)}) - y_{jk}^{(i)}]$ and $\widetilde{\mathbf{g}}_i(\mu_i) \leq 0 \Leftrightarrow \sum_j \mu_{i,jk} = R, \ \sum_k \mu_{i,jk} \leq P, \ 0 \leq \mu_{i,jk} \leq 1$.

Generally, when we can express $\max_{\mathbf{y}\in\mathcal{Y}^{(i)}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y})$ as an LP, we have a similar LP for the loss-augmented inference:

$$d_i + \max\ (\mathbf{F}_i \mathbf{w} + \mathbf{c}_i)^\top \mu_i \ \text{ s.t. } \ \mathbf{A}_i \mu_i \le \mathbf{b}_i,\ \mu_i \ge 0, \quad (6)$$

for appropriately defined $d_i, \mathbf{F}_i, \mathbf{c}_i, \mathbf{A}_i, \mathbf{b}_i$, which depend only on $\mathbf{x}^{(i)}$, $\mathbf{y}^{(i)}$, $\mathbf{f}(\mathbf{x}, \mathbf{y})$ and $\mathbf{g}(\mathbf{x}, \mathbf{y})$. Note that $\mathbf{w}$ appears only in the objective.

In cases (such as these) where we can formulate the loss-augmented inference as a convex optimization problem as in Eq. (4), and this formulation is concise, we can use Lagrangian duality (see (Boyd & Vandenberghe, 2004) for an excellent review) to define a joint, concise convex problem for estimating the parameters $\mathbf{w}$. The Lagrangian associated with Eq. (4) is

$$L_{i,\mathbf{w}}(\mu_i, \lambda_i) = \widetilde{f}_i(\mathbf{w}, \mu_i) - \lambda_i^\top \widetilde{\mathbf{g}}_i(\mu_i), \quad (7)$$

where $\lambda_i \ge 0$ is a vector of Lagrange multipliers, one for each constraint function in $\widetilde{\mathbf{g}}_i(\mu_i)$. Since we assume that $\widetilde{f}_i(\mathbf{w}, \mu_i)$ is concave in μ_i and bounded on the non-empty set $\{\mu_i : \widetilde{\mathbf{g}}_i(\mu_i) \le 0\}$, we have *strong duality*:

$$\max_{\mu_i : \widetilde{\mathbf{g}}_i(\mu_i) \le 0} \widetilde{f}_i(\mathbf{w}, \mu_i) = \min_{\lambda_i \ge 0} \max_{\mu_i} L_{i,\mathbf{w}}(\mu_i, \lambda_i).$$

For many forms of $\widetilde{f}$ and $\widetilde{\mathbf{g}}$, we can write the Lagrangian dual $\min_{\lambda_i \ge 0} \max_{\mu_i} L_{i,\mathbf{w}}(\mu_i, \lambda_i)$ explicitly as:

$$\min\ \ h_i(\mathbf{w}, \lambda_i) \qquad \text{s.t.} \quad \mathbf{q}_i(\mathbf{w}, \lambda_i) \le 0, \quad (8)$$

where $h_i(\mathbf{w}, \lambda_i)$ and $\mathbf{q}_i(\mathbf{w}, \lambda_i)$ are convex in both $\mathbf{w}$ and λ_i. (We folded $\lambda_i \ge 0$ into $\mathbf{q}_i(\mathbf{w}, \lambda_i)$ for brevity.) Since the original problem had polynomial size, the dual is polynomial size as well. For example, the dual of the LP in Eq. (6) is

$$d_i + \min \mathbf{b}_i^\top \lambda_i \quad \text{s.t.} \quad \mathbf{A}_i^\top \lambda_i \ge \mathbf{F}_i \mathbf{w} + \mathbf{c}_i,\ \lambda_i \ge 0, \quad (9)$$

where $h_i(\mathbf{w}, \lambda_i) = d_i + \mathbf{b}_i^\top \lambda_i$ and $\mathbf{q}_i(\mathbf{w}, \lambda_i) \le 0$ is $\{\mathbf{F}_i \mathbf{w} + \mathbf{c}_i - \mathbf{A}_i^\top \lambda_i \le 0, -\lambda_i \le 0\}$.

Plugging Eq. (8) into Eq. (4), we get

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 & (10)\\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) \ge \min_{\mathbf{q}_i(\mathbf{w},\lambda_i) \le 0} h_i(\mathbf{w}, \lambda_i), \quad \forall i. \end{aligned}$$

We can now combine the minimization over λ with minimization over $\mathbf{w}$. The reason for this is that if the right hand side is not at the minimum, the constraint is tighter than necessary, leading to a suboptimal solution $\mathbf{w}$. Optimizing jointly over λ as well will produce a solution to $\mathbf{w}$ that is optimal.

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 & (11)\\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) \ge h_i(\mathbf{w}, \lambda_i), \quad \forall i; \\ & \mathbf{q}_i(\mathbf{w}, \lambda_i) \le 0, \quad \forall i. \end{aligned}$$

Hence we have a joint and concise convex optimization program for estimating $\mathbf{w}$. The exact form of this program depends strongly on $\widetilde{f}$ and $\widetilde{\mathbf{g}}$. For our LP-based example, we have a QP with linear constraints:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 & (12)\\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) \ge d_i + \mathbf{b}_i^\top \lambda_i, \quad \forall i; \\ & \mathbf{A}_i^\top \lambda_i \ge \mathbf{F}_i \mathbf{w} + \mathbf{c}_i, \quad \forall i; \quad \lambda_i \ge 0, \quad \forall i. \end{aligned}$$

This formulation generalizes the idea used by Taskar et al. (2004a) to provide a polynomial time estimation procedure for a certain family of Markov networks; it can also be used to derive the max-margin formulations of Taskar et al. (2003); Taskar et al. (2004b).

5. Certificate formulation

In the previous section, we assumed a *concise* convex formulation of the loss-augmented max in Eq. (4). There are several important combinatorial problems which allow polynomial time solution yet do not have a concise convex optimization formulation. For example, a maximum weight spanning tree and perfect *non-bipartite* matching problems can be expressed as linear programs with *exponentially* many constraints, but no polynomial formulation as a convex optimization problem is known (Schrijver, 2003). Both of these problems, however, can be solved in polynomial time using combinatorial algorithms. In some cases, though, we can find a concise *certificate of optimality* that guarantees that $\mathbf{y}^{(i)} = \arg\max_{\mathbf{y}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$ without expressing loss-augmented inference as a concise convex program. Intuitively, verifying that a given assignment is optimal can be easier than actually finding one.

As a simple example, consider the maximum weight spanning tree problem. A basic property of a spanning tree is that cutting any edge (j, k) in the tree creates two disconnected sets of nodes $(\mathcal{V}_j[jk], \mathcal{V}_k[jk])$, where $j \in \mathcal{V}_j[jk]$ and $k \in \mathcal{V}_k[jk]$. A spanning tree is optimal with respect to a set of edge weights if and only if for every edge (j, k) in the tree connecting $\mathcal{V}_j[jk]$ and $\mathcal{V}_k[jk]$, the weight of (j, k) is larger than (or equal to) the weight of any other edge (j', k') in the graph with $j' \in \mathcal{V}_j[jk], k' \in \mathcal{V}_k[jk]$ (Schrijver, 2003). These conditions can be expressed using linear inequality constraints on the weights.

More generally, suppose that we can find a *concise* convex formulation of these conditions via a polynomial (in L_i) set of functions $\mathbf{q}_i(\mathbf{w}, \nu_i)$, jointly convex in $\mathbf{w}$ and auxiliary variables ν_i such that for fixed $\mathbf{w}$, $\exists \nu_i : \mathbf{q}_i(\mathbf{w}, \nu_i) \le 0 \Leftrightarrow \forall \mathbf{y} \in \mathcal{Y}^{(i)} : \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) \ge \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$. Then $\min \frac{1}{2}||\mathbf{w}||^2$ such that $\mathbf{q}_i(\mathbf{w}, \nu_i) \le 0$ for all i is a joint convex program in $\mathbf{w}, \nu$ that computes the max-margin parameters.

Expressing the spanning tree optimality does not require additional variables ν_i, but in other problems, such as in perfect matching optimality (see below), such auxiliary variables are needed.

Consider the problem of finding a matching in a non-bipartite graph; we begin by considering perfect matchings, where each node has *exactly* one neighbor, and then provide a reduction for the non-perfect case (each node has *at most* one neighbor). Let M be a perfect matching for a complete undirected graph $G = (\mathcal{V}, \mathcal{E})$. In an *alternating cycle/path* in G with respect to M, the edges alternate between those that belong to M and those that do not. An alternating cycle is *augmenting* with respect to M if the score of the edges in the matching M is smaller that the score of the edges not in the matching M.

Theorem 5.1 (Edmonds, 1965) *A perfect matching M is a maximum weight perfect matching if and only if there are no augmenting alternating cycles.*

The number of alternating cycles is exponential in the number of vertices, so simply enumerating all of them is infeasible. Instead, we can rule out such cycles by considering shortest paths.

We begin by negating the score of those edges not in M. In the discussion below we assume that each edge score s_{jk} has been modified this way. We also refer to the score s_{jk} as the length of the edge jk. An alternating cycle is augmenting if and only if its length is negative. A condition ruling out negative length alternating cycles can be stated succinctly using a type of distance function. Pick an arbitrary root node r. Let d_j^e, with $j \in \mathcal{V}$, $e \in \{0, 1\}$, denote the length of the shortest distance alternating path from r to j, where $e = 1$ if the last edge of the path is in M, 0 otherwise. These shortest distances are well-defined if and only if there are no negative alternating cycles. The following constraints capture this distance function.

$$
\begin{aligned}
s_{jk} &\geq d_k^0 - d_j^1, \qquad s_{jk} \geq d_j^0 - d_k^1, \qquad \forall\ jk \notin M; \\
s_{jk} &\geq d_k^1 - d_j^0, \qquad s_{jk} \geq d_j^1 - d_k^0, \qquad \forall\ jk \in M.
\end{aligned}
\tag{13}
$$

Theorem 5.2 *There exists a distance function $\{d_j^e\}$ satisfying the constraints in Eq. (14) if and only if no augmenting alternating cycles exist.*

The proof is presented in Taskar (2004), Chap. 10.

In our learning formulation, assuming Hamming loss, we have the loss-augmented edge weights $s_{jk}^{(i)} = (2y_{jk}^{(i)} - 1)(\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}) + 1 - 2y_{jk}^{(i)})$, where the $(2y_{jk}^{(i)} - 1)$ term in front negates the score of edges not in the matching. Let $\mathbf{d}_i$ be a vector of distance variables d_j^e, $\mathbf{F}_i$ and $\mathbf{G}_i$ be matrices of coefficients and $\mathbf{q}_i$ be a vector such that $\mathbf{F}_i\mathbf{w} + \mathbf{G}_i\mathbf{d}_i \geq \mathbf{q}_i$ represents the constraints in Eq. (14) for example i. Then the following joint convex program in $\mathbf{w}$ and $\mathbf{d}$ (with $\mathbf{d}_i$ playing the role of ν_i) computes the max-margin parameters:

$$
\min \ \frac{1}{2}||\mathbf{w}||^2 \qquad \text{s.t.} \quad \mathbf{F}_i\mathbf{w} + \mathbf{G}_i\mathbf{d}_i \geq \mathbf{q}_i, \quad \forall i. \tag{14}
$$

If our features are not rich enough to predict the training data perfectly, we can introduce a slack variable vector to allow violations of the constraints.

The case of non-perfect matchings can be handled by a reduction to perfect matchings as follows (Schrijver, 2003). We create a new graph by making a copy of the nodes and the edges and adding edges between each node and the corresponding node in the copy. We extend the matching by replicating its edges in the copy and for each unmatched node, introduce an edge to its copy. We define $\mathbf{f}(\mathbf{x}_{jk}) \equiv 0$ for edges between the original and the copy. Perfect matchings in this graph projected onto the original graph correspond to non-perfect matchings in the original graph.

6. Duals and kernels

Both the min-max formulation and the certificate formulation produce primal convex optimization problems. Rather than solving them directly, we consider their dual versions. In particular, as in SVMs, we can use the kernel trick in the dual to efficiently learn in high-dimensional feature spaces.

Consider, for example, the primal problem in Eq. (15). In the dual, each training example i has $L_i(L_i - 1)$ variables, two for each edge ($\alpha_{jk}^{(i)}$ and $\alpha_{kj}^{(i)}$), since we have two constraints per edge in the primal. Let $\alpha^{(i)}$ be the vector of dual variables for example i. The dual quadratic problem is:

$$
\begin{aligned}
\min \ & \sum_i \mathbf{q}_i^\top \alpha^{(i)} + \frac{1}{2}\left|\left|\sum_i \mathbf{F}_i^\top \alpha^{(i)}\right|\right|^2 \\
\text{s.t.} \quad & \mathbf{G}_i\alpha^{(i)} = 0, \quad \alpha^{(i)} \geq 0, \quad \forall i.
\end{aligned}
$$

The only occurrence of feature vectors is in the expansion of the squared-norm term in the objective:

$$
\mathbf{F}_i^\top \alpha^{(i)} = \sum_{j<k} (\alpha_{jk}^{(i)} + \alpha_{kj}^{(i)})(2y_{jk}^{(i)} - 1)\mathbf{f}(\mathbf{x}_{jk}^{(i)}) \tag{15}
$$

Therefore, we can apply the kernel trick and let $\mathbf{f}(\mathbf{x}_{jk}^{(i)})^\top \mathbf{f}(\mathbf{x}_{mn}^{(t)}) = K(\mathbf{x}_{jk}^{(i)}, \mathbf{x}_{mn}^{(t)})$. At prediction time, we can also use the kernel trick to compute:

$$
\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{mn}) = \sum_i \sum_{j<k} (\alpha_{jk}^{(i)} + \alpha_{kj}^{(i)})(2y_{jk}^{(i)} - 1) K(\mathbf{x}_{jk}^{(i)}, \mathbf{x}_{mn}).
$$

The dual of min-max Eq. (12) is also kernelizable.

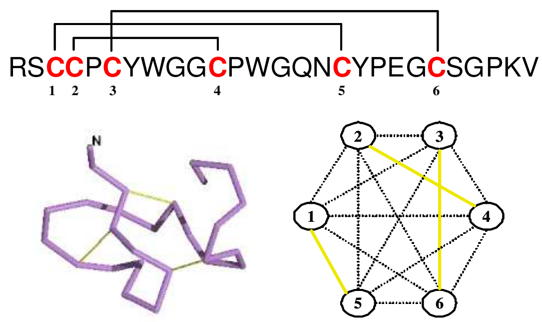

Figure 1. PDB protein 1ANS: amino acid sequence, 3D structure, and graph of potential disulfide bonds. Actual disulfide connectivity is shown in yellow in the 3D model and the graph of potential bonds.

7. Experiments

We apply our framework to the task of disulfide connectivity prediction. Proteins containing cysteine residues form intra-chain covalent bonds known as *disulfide bridges*. Such bonds are a very important feature of protein structure, as they enhance conformational stability by reducing the number of configurational states and decreasing the entropic cost of folding a protein into its native state (Baldi et al., 2004). Knowledge of the exact disulfide bonding pattern in a protein provides information about protein structure and possibly its function and evolution. Furthermore, as the disulfide connectivity pattern imposes structural constraints, it can be used to reduce the search space in both protein folding prediction as well as protein 3D structure prediction. Thus, the development of efficient, scalable and accurate methods for the prediction of disulfide bonds has important practical applications.

Recently, there has been increasing interest in applying computational techniques to the task of predicting disulfide connectivity (Fariselli & Casadio, 2001; Baldi et al., 2004). Following the lines of Fariselli and Casadio (2001), we predict the connectivity pattern by finding the maximum weighted matching in a graph in which each vertex represents a cysteine residue, and each edge represents the "attraction strength" between the cysteines it connects. For example, 1ANS protein in Fig. 1 has six cysteines, and three disulfide bonds.

We parameterize the attraction scoring function via a linear combination of features, which include the protein sequence around the two residues, evolutionary information in the form of multiple alignment profiles, secondary structure or solvent accessibility information, etc. We then learn the weights of the different features using a set of solved protein structures, in which the disulfide connectivity patterns are known.

Datasets. We used two datasets containing sequences with experimentally verified bonding patterns: SP39 (used in Baldi et al. (2004); Vullo and Frasconi (2004); Fariselli and Casadio (2001)) and DIPRO2.[1]

We report results for two experimental settings: when the bonding state is known (that is, for each cysteine, we know whether it bonds with some other cysteine) and when it is unknown. The known state setting corresponds to prefect non-bipartite matchings, while the unknown to imperfect matchings. For the case when bonding state is known, in order to compare to previous work, we focus on proteins containing between 2 and 5 bonds, which covers the entire SP39 dataset and over 90% of the DIPRO2 dataset.

In order to avoid biases during testing, we adopt the same dataset splitting procedure as the one used in previous work (Fariselli & Casadio, 2001; Vullo & Frasconi, 2004; Baldi et al., 2004).[2]

Models. The experimental results we report use the dual formulation of Eq. (16) and an RBF kernel $K(\mathbf{x}_{jk}, \mathbf{x}_{st}) = \exp\left(\frac{\|\mathbf{x}_{jk} - \mathbf{x}_{st}\|^2}{2\sigma^2}\right)$, with $\sigma \in [1, 2]$. We used commercial QP software (CPLEX) to train our models. Training time took around 70 minutes for 450 examples.

The set of features $\mathbf{x}_{jk}$ for a candidate cysteine pair jk is based on local windows of size n centered around each cysteine (we use $n = 9$). The simplest approach is to encode for each window, the actual sequence as a $20 \times n$ binary vector, in which the entries denote whether or not a particular amino acid occurs at the particular position. However, a common strategy is to compensate for the sparseness of these features by using multiple sequence alignment profile information. Multiple alignments were computed by running PSI-BLAST using default settings to align the sequence with all sequences in the NR database (Altschul et al., 1997). Thus, the input features for the first model, *PROFILE*, consist of the proportions of occurrence of each of the 20 amino-acids in the alignments at each position in the local windows.

The second model, *PROFILE-SS*, augments the *PROFILE* model with secondary structure and solvent-accessibility information. The DSSP program

[1] The DIPRO2 dataset was made publicly available by Baldi et al. (2004). It consists of 1018 non-redundant proteins from PDB (Berman et al., 2000) as of May 2004 which contain intra-chain disulfide bonds. The sequences are annotated with secondary structure and solvent accessibility information from DSSP (Kabsch & Sander, 1983).

[2] We split SP39 into 4 different subsets, with the constraint that proteins with sequence similarity of more than 30% belong to different subsets. We ran all-against-all rigorous Smith-Waterman local pairwise alignment (using BLOSUM65, with gap penalty/extension 12/4) and considered pairs with less than 30 aligned residues as dissimilar.

K	*SVM*	*PROFILE*	*DAG-RNN*
2	0.63/0.63	**0.77/0.77**	0.74/0.74
3	0.51/0.38	**0.62/0.52**	0.61/0.51
4	0.34/0.12	**0.51/0.36**	0.44/0.27
5	0.31/0.07	**0.43/0.13**	0.41/0.11

(a)

K	*PROFILE*	*PROFILE-SS*
2	0.76/0.76	**0.78/0.78**
3	0.67/0.59	**0.74/0.65**
4	0.59/0.42	**0.64/0.49**
5	0.39/0.13	**0.46/0.22**

(b)

K	*PROFILE*	*DAG-RNN*
2	**0.57**/0.59/**0.44**	0.49/0.59/0.40
3	**0.48**/**0.52**/0.28	0.45/0.50/**0.32**
4	**0.39**/**0.40**/0.14	0.37/0.36/**0.15**
5	0.31/**0.33**/**0.07**	0.31/0.28/0.03

(c)

Table 1. Numbers indicate *Precision/Accuracy* for known bonded state setting in (a) and (b) and *Precision/Recall/Accuracy* for unknown bonded state in (c). *K* denotes the true number of bonds. Best results in each row are in **bold**. (a) *PROFILE* vs. *SVM* and *DAG-RNN* model (Baldi et al., 2004) on SP39 . (b) *PROFILE* vs. *PROFILE-SS* models on the DIPRO2 dataset. (c) *PROFILE* vs. *DAG-RNN* model on SP39 (unknown state).

produces 8 types of secondary structure, so we augment each local window of size n with an additional length $8 \times n$ binary vector, as well as a length n binary vector representing the solvent accessibility at each position. Because DSSP utilizes *true* 3D structure information in assigning secondary structure, the model cannot be used in for unknown proteins. Rather, its performance is useful as an upper bound of the potential prediction improvement using features derived via accurate secondary structure prediction algorithms.

Known Bonded State. When bonding state is known, we evaluate our algorithm using two metrics: accuracy measures the fraction of entire matchings predicted correctly, and precision measures fraction of correctly predicted individual bonds.

We apply the model to the SP39 dataset by using 4-fold cross-validation, which replicates the experimental setup of Baldi et al. (2004); Vullo and Frasconi (2004). First, we evaluate the advantage that our learning formulation provides over a baseline approach, where we use the features of the *PROFILE* model, but ignore the constraints in the learning phase and simply learn $\mathbf{w}$ using an SVM by labeling bonded pairs as positive examples and non-bonded pairs as negative examples. We then use these SVM-learned weights to score the edges. The results are summarized in Table 1(a) in the column *SVM*. The *PROFILE* model uses our certificate-based formulation, directly incorporating the matching constraint in the learning phase. It achieves significant gains over *SVM* for all bond numbers, illustrating the importance of explicitly modelling structure during parameter estimation. We also compare our model to the *DAG-RNN* model of Baldi et al. (2004), the current top-performing system, which uses recursive neural networks also with the same set of input features. Our performance is better for all bond numbers.

As a final experiment, we examine the role that secondary structure and solvent-accessibility information plays in the model *PROFILE-SS*. Table 1(b) shows that the gains are significant, especially for sequences with 3 and 4 bonds. This highlights the importance of developing even richer features, perhaps through more complex kernels.

Unknown Bonded State. We also evaluated our learning formulation for the case when bonding state is unknown using the graph-copy reduction described in Sec. 5. Here, we use an additional evaluation metric: *recall*, which measures correctly predicted bonds as a fraction of the total number of bonds. We apply the model to the SP39 dataset for proteins of 2-5 bonds, without assuming knowledge of bonded state. Since some sequences contain over 50 cysteines, for computational efficiency during training, we only take up to 14 bonded cysteines by including the ones that participate in bonds, and randomly selecting additional cysteines from the protein (if available). During testing, we used all the cysteines.

The results are summarized in Table 1(c), with a comparison to the *DAG-RNN* model, which is currently the only other model to tackle this more challenging setting. Our results compare favorably having better precision and recall for all bond numbers, but our connectivity pattern accuracy is slightly lower for 3 and 4 bonds.

8. Discussion and Conclusion

We present a framework for learning a wide range of structured prediction models where the set of outcomes is a class of combinatorial structures such as matchings and graph-cuts. We present two formulations of structured max-margin estimation that define a concise convex optimization problem. The first formulation, *min-max*, relies on the ability to express inference in a model as a concise convex optimization problem. The second one, *certificate*, only requires expressing optimality of a desired solution according to a model. We illustrate how to apply these formulations to the problem of learning to match, with bipartite and non-bipartite matchings. These formulations can be also applied to min-cuts, max-flows, trees, colorings and many other combinatorial problems.

Beyond the closely related large-margin methods for probabilistic models, our max-margin formulation has ties to a body of work called inverse combinatorial and convex optimization (for a recent survey, see Heuberger (2004)). An *inverse optimization problem* is defined by an instance of an optimization problem $\max_{\mathbf{y}\in\mathcal{Y}} \mathbf{w}^\top \mathbf{f}(\mathbf{y})$, a set of nominal weights $\mathbf{w}^0$, and a target solution $\mathbf{y}^t$. The goal is to find the weights $\mathbf{w}$ closest to the nominal $\mathbf{w}^0$ in some p-norm, which make the target solution optimal. $\min\ ||\mathbf{w} - \mathbf{w}^0||_p \ \text{ s.t. } \ \mathbf{w}^\top \mathbf{f}(\mathbf{y}^t) \geq \mathbf{w}^\top \mathbf{f}(\mathbf{y}), \ \ \forall \mathbf{y} \in \mathcal{Y}$. The solution $\mathbf{w}$ depends critically on the choice of nominal weights; for example, if $\mathbf{w}^0 = 0$, then $\mathbf{w} = 0$ is trivially the optimal solution. In our approach, we have a very different goal, of learning a parameterized objective function that depends on the input $\mathbf{x}$ and will generalize well in prediction on new instances.

Our approach attempts to find weights that obtain a particular target solution for each training instance. While a unique solution is a reasonable assumption in some domains (e.g., disulfide bonding), in others there may be several "equally good" target solutions. It would be interesting to extend our approach to accommodate multiple target solutions.

The estimation problem is tractable and exact whenever the prediction problem can be formulated as a concise convex optimization problem or a polynomial time combinatorial algorithm with concise convex optimality conditions. For intractable models (e.g., tripartite matching, quadratic assignment, max-cut, etc.), we can use our framework to learn approximate parameters by exploiting approximations that only provide upper/lower bounds on the optimal structure score or provide a certificate of optimality in a large neighborhood around the desired structure.

Using off-the-shelf convex optimization code for our learning formulation is convenient and flexible, but it is likely that problem-specific methods that use combinatorial subroutines will outperform generic solvers. Design of such algorithms is an open problem.

We have addressed estimation of models with discrete output spaces. Similarly, we can consider a whole range of problems where the prediction variables are continuous. Such problems are a natural generalizations of regression, involving correlated, interconstrained real-valued outputs. Examples include learning models of metabolic flux in cells, which obeys stoichiometric constraints, and learning game payoff matrices from observed equilibria strategies.

Our learning framework addresses a large class of prediction models with rich and interesting structure. We hope that continued research in this vein will help tackle evermore sophisticated prediction problems.

Acknowledgements. This work was supported by NSF grant DBI-0345474. and by DARPA's EPCA program under subcontract to SRI.

References

Altschul, S., Madden, T., Schaffer, A., Zhang, A., Miller, W., & Lipman (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleid Acids Res.*, *25*, 3389–3402.

Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003). Hidden markov support vector machines. *Proc. ICML*.

Baldi, P., Cheng, J., & Vullo, A. (2004). Large-scale prediction of disulphide bond connectivity. *Proc. NIPS*.

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, *24*.

Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I., & Bourne, P. (2000). The protein data bank.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

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

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification*. New York: Wiley Interscience. 2nd edition.

Edmonds, J. (1965). Maximum matching and a polyhedron with 0-1 vertices. *Journal of Research at the National Bureau of Standards*, *69B*, 125–130.

Fariselli, P., & Casadio, R. (2001). Prediction of disulfide connectivity in proteins. *Bioinformatics*, *17*, 957–964.

Heuberger, C. (2004). Inverse combinatorial optimization: A survey on problems, methods, and results. *Journal of Combinatorial Optimization*, *8*.

Kabsch, W., & Sander, C. (1983). Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features.

Schrijver, A. (2003). *Combinatorial optimization: Polyhedra and efficiency*. Springer.

Taskar, B. (2004). *Learning structured prediction models: A large margin approach*. Doctoral dissertation, Stanford University.

Taskar, B., Chatalbashev, V., & Koller, D. (2004a). Learning associative Markov networks. *Proc. ICML*.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max margin Markov networks. *Proc. NIPS*.

Taskar, B., Klein, D., Collins, M., Koller, D., & Manning, C. (2004b). Max margin parsing. *Proc. EMNLP*.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. *Proc. ICML*.

Vullo, A., & Frasconi, P. (2004). Disulfide connectivity prediction using recursive neural networks and evolutionary information. *Bioinformatics*, *20*, 653–659.