Journal of Machine Learning Research 1 (2008) 1-26 Submitted 3/08; Published ??/08

Structured Prediction of Generalized Matching Graphs

Stuart J. Andrews ANDREWS@CS.COLUMBIA.EDU
Tony Jebara JEBARA@CS.COLUMBIA.EDU
Department of Computer Science
Columbia University
New York, NY 10027, USA

Editor: Leslie Pack Kaelbling

Abstract

A structured prediction approach is proposed for completing missing edges in a graph using partially observed connectivity between n nodes. Unlike previous approaches, edge predictions depend on the node attributes (features) as well as graph topology. To overcome unrealistic *i.i.d.* edge prediction assumptions, the structured prediction framework is extended to an output space of directed subgraphs that satisfy in-degree and out-degree constraints. An efficient cutting plane algorithm is provided which interleaves the estimation of an edge score function with exact inference of the maximum weight degree-constrained subgraph. Experiments with social networks, protein-protein interaction graphs and citation networks are shown.

Keywords: Graph Inference, Structured Prediction, Generalized Matching, Transduction, Cutting-Planes

1. Introduction

Graphs are abstract models used to represent pairwise relationships between a set of objects. Interesting examples can be found in natural language processing, social networks, as well as in structural and systems biology. Nodes represent individual objects, while edges encode relationships between pairs of objects. In practice, the nodes and edges have attributes associated with them too. For example, a collection of documents forms a graph called a citation network wherein nodes represent papers, and directed connections between nodes represent citations between papers. What is common across many applied domains is the understanding that the likelihood or plausibility of a graph depends on both the attributes of the nodes and edges as well as the structure or topology of the graph connectivity implied by the edges.

The goal of this article is graph inference and reconstruction. In other words, filling in missing relationships in a graph, assuming that some are known and others are hidden. Given a partially observed connectivity between n objects, the goal is to complete the connectivity to produce a realistic graph. This discovery of new or missing interactions is important to many applied disciplines. For instance, there are practical implications for the recovery of new interactions in biological networks as well as the prediction of human preferences or relationships from partially completed social networks. Examples of graph reconstruction efforts in biology are numerous ranging from metabolic networks (Herrgaard et al., 2004, Vitkup et al., 2006, Chechik et al., 2007), molecular networks (Baldi et al., 1999, Punta and Rost, 2005, Gassend et al., 2007), regulatory networks (Middendorf et al., 2004), and cell signaling networks (Sachs et al., 2005). Edge prediction is also relevant to a wider range of structured prediction problems that can also be cast as graph problems including sentence parsing and image processing.

In previous approaches, edge predictions or graph reconstruction typically depends either on intrinsic properties of the graph (topology) or on extrinsic attribute values (features), but rarely on both simultaneously. For example, metric learning can be viewed as a graph reconstruction method and has recently had considerable progress in the machine learning community (Xing et al., 2003, Kondor and Jebara, 2006, Shalev-Shwartz et al., 2004, Globerson and Roweis, 2006). Typically, an affinity or metric is learned between pairs of examples such that pairs of objects in an equivalence class have high similarity while those in different classes produce low similarity. To perform graph inference, pairwise similarity above a threshold indicates the presence of an edge between a pair of objects. However, using a learned affinity metric in a strictly pairwise manner makes all connectivity decisions independently in the graph. Clearly, in realistic scenarios, the presence or absence of one edge in a graph influences and depends on the presence or absence of other edges in the graph. Thus, an affinity metric alone may produce graphs whose structure is unlike that of realistic graphs.

Alternatively, in parallel efforts such as (Liben-Nowell and Kleinberg, 2003), topology is used to predict new edges in time-evolving citation networks. Exploiting topology is plausible since many real-world graphs exhibit a high degree of structural regularity. For instance, social network graphs have consistently small diameters and/or characteristic degree distributions. Complex biological systems graphs are composed of frequently occurring motifs (Kashtan et al., 2004). Indeed, many generative models have been proposed that can replicate similar networks and their evolution (Even-Dar and Kearns, 2006).

Therefore, it is worthwhile to augment both schools of thought by combining independent node/edge attribute learning methods with structural algorithms to preserve the regularities seen in real-world graphs. In this paper, we present an algorithm for the prediction of naturally structured graphs using both topology and attribute data, which is based on a novel coupling of the structured-outputs framework (Altun et al., 2003, Taskar et al., 2003, Altun et al., 2004, Tsochantaridis et al., 2004, Altun et al., 2005, Taskar et al., 2006) and the class of degree-constrained subgraphs. Our model learns to predict connectivity of ordered pairs of objects while enforcing in-degree and out-degree constraints. We show that by analyzing edge attributes while concurrently enforcing topological structure constraints, our model yields a better predictor for graph connectivity. We believe that our model is the first to combine topology and attribute-value learning to *significant advantage* for graph completion.

Our approach also presents a novel transductive variant of the structured-outputs framework to address graph completion. Transduction has previously been studied (Altun et al., 2005, Zien et al., 2007) to help design better structured predictors; however, these methods use labeled and unlabeled structured objects that are independent. The assumption is, for example, that the parse tree of one sentence does not depend on another and can therefore be predicted independently. In contrast, our model considers transduction within a single structured object, a graph over n objects. Edges in the graph are either labeled or unlabeled however, transduction leverages the assumption that *all* labels are constrained by a single structure and *all* edge predictions are inter-dependent. This transductive inference of degree constrained graphs or *generalized matchings* is ultimately interleaved in a cutting-plane structured prediction framework to complete real graphs.

This paper is organized as follows. Section 2 introduces graph completion in a formal setting. Next, in Sections 3 and 4, we discuss inference and learning within a maximum margin structured outputs framework. Section 5 discusses a number of implementation details of interest to practitioners. In Section 6 we demonstrate the approach using synthetic and real-world graphs. Finally, to wrap up, Section 7 discusses related work, before some concluding remarks are presented in Section 8.

2. Graph Completion

In general, we are given a fixed set of nodes $\mathcal{V} = \{1, \ldots, n\}$ which may be connected by a maximal set $\mathcal{E} = \{(j,k)\,|1 \leq j,k \leq n\}$ of n^2 directed edges, if we allow self-loops. Binary edge labels $y_{j,k} \in \{0,1\}$ indicate the *presence* or *absence* of edge (j,k), while edge attributes are represented by a feature vector $\mathbf{x}_{j,k} \in \mathbb{R}^d$. In the sequel, we occasionally refer to edges as being *positive* or *negative* even though we are using 0-1 binary labels. We denote by $\mathbf{X}$ the set of all edge feature vectors and by $\mathbf{Y}$ the set of all labels.

Graph inference concerns the prediction of missing or unobserved edge labels from $\mathbf{Y}$, assuming we have observed only some of the entries together with the information encoded in the features $\mathbf{X}$. Let $\mathcal{O}$ be the set of observed edges, $\mathcal{U}$ be the set of unobserved edges, where typically the partition of edges satisfies $\mathcal{O} \cup \mathcal{U} = \mathcal{E}$. Because some of the labels are observed, and the goal is to predict the remaining labels, this process is also called *completion*. For example, suppose we have $\mathcal{V} = \{1,2,3\}$ and we observe $\mathbf{Y}_{\mathcal{O}} = \{y_{1,1}, y_{1,2}, y_{2,1}, y_{2,2}\}$, then we would like to complete the network by predicting values for $\mathbf{Y}_{\mathcal{U}} = \{y_{1,3}, y_{2,3}, y_{3,1}, y_{3,2}, y_{3,3}\}$.

In the literature, several variants of the graph completion problem that depend on the choice of $\mathcal{O}$ and $\mathcal{U}$ have been proposed. We describe four tasks, which are based on *bipartite* partitions, and one that is non-bipartite. Each bipartite task starts with a train-test partition of the nodes, which induces a block decomposition of the $n \times n$ adjacency matrix representation of the graph edges (see Figure 1). We refer to the four blocks in the decomposition as: train-train, train-test, test-test and test-train.

In (Yamanishi et al., 2004), the set $\mathcal{O}$ is comprised of train-train edges, while the set $\mathcal{U}$ is comprised of the remaining blocks: train-test, test-test and test-train. In (Airoldi et al., 2007), the set $\mathcal{O}$ is comprised of train-train edges, while the set $\mathcal{U}$ is comprised of the test-test edges. Finally, in (Bleakley et al., 2007), the set $\mathcal{O}$ is comprised of train-train edges, while the set $\mathcal{U}$ is comprised of all the remaining blocks except the test-test edges. These graph completion tasks, which serve distinct but related purposes, are depicted in Figure 2, columns (a-c). Two additional completion tasks are displayed in columns (d) and (e), the last being non-bipartite in the sense described above. As far as prediction is concerned, the number and variety of completion tasks present unique challenges to be solved. Figure 2 provides an alternative depiction of the graph completion task from column (a) for a graph with a clear geometric structure. In this case, one half of the nodes (1-12) are observed yielding a partition of positive and negative training edges (middle row) and positive and negative testing edges (bottom row).

2.1 A Probabilistic Model

We first start with a probabilistic treatment of the problem. The most naive assumption is that the distribution $p(\mathbf{Y}|\mathbf{X})$ is parameterized by some unknown $\mathbf{w}$ and is independent across all $\mathbf{Y}$ entries and factorizes as follows

$$p_{\mathbf{w}}(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z_{\mathbf{w}}(\mathbf{X})} \prod_{j,k} \phi_{\mathbf{w}}(y_{j,k}|\mathbf{x}_{j,k}) \ , \tag{1}$$

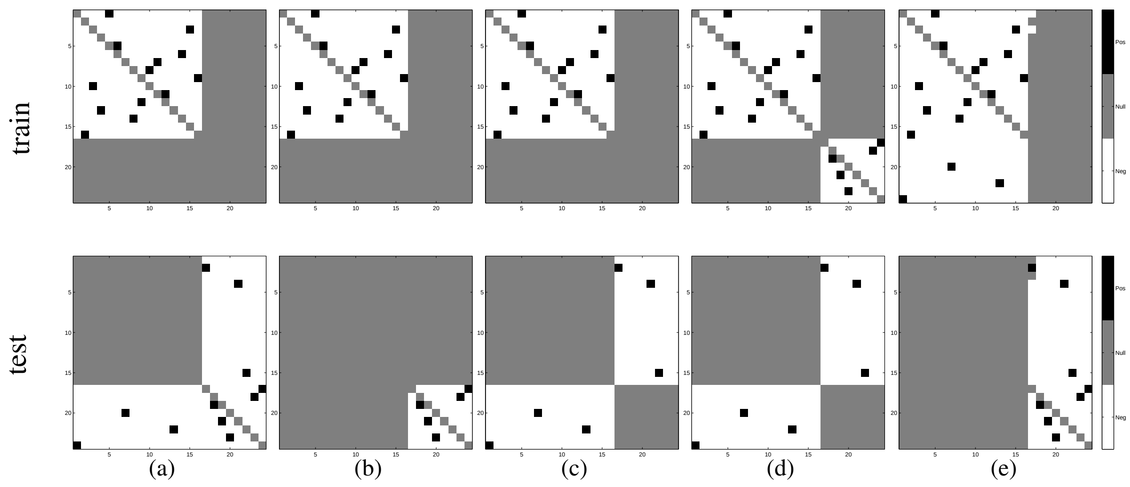


Figure 1: Supervised graph completion predicts missing edges in a graph using partially observed connectivity. Each column depicts an instance of the problem, where the top row shows the edges that are observed and the bottom row shows the edges that are missing. Black and white matrix entries indicate positive (on) and negative (off) edges, while grey entries are excluded from consideration. The semantics are (a) extending or growing a network, (b) predicting a similar but independent network, (c) linking new nodes to an existing network, and (d) linking two networks. The last column shows (e) an example of a non-bipartite partition (see text).

and $Z_{\mathbf{w}}(\mathbf{X})$ is the normalization constant. For instance, we may assume that the presence or absence of a edge depends on the feature vector $\mathbf{x}_{j,k}$ associated with the nodes it connects, and a parameter $\mathbf{w}$ by way of a log-linear potential function

$$\phi_{\mathbf{w}}(y_{j,k}|\mathbf{x}_{j,k}) = \exp\left(\left(\mathbf{w}^T\mathbf{x}_{j,k}\right) y_{j,k}\right) . \tag{2}$$

However, since some connectivity structures are more likely than others, the independence assumption is too simple. More realistically, there are dependencies between the labels. For instance, consider a set of cliques $c \in C$ over the edge labels in $\mathbf{Y}$. This gives the probability distribution

$$p_{\mathbf{w}}(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z}\prod_{c\in C}\phi_c(y_c)\prod_{j,k}\phi_{\mathbf{w}}(y_{j,k}|\mathbf{x}_{j,k}) , \tag{3}$$

where each ϕ_c is a multivariate clique potential function over a set of edges. By choosing different forms for the ϕ_c, we can tailor our beliefs about which network topologies are more plausible than others.

Given a setting of $\mathbf{w}$, the problem of finding the most likely completion of a partially observed network amounts to solving

$$\underset{\mathbf{Y}_{\mathcal{U}}}{\operatorname{argmax}}\, p_{\mathbf{w}}(\mathbf{Y}_{\mathcal{U}}|\mathbf{Y}_{\mathcal{O}},\mathbf{X}) = \underset{\mathbf{Y}_{\mathcal{U}}}{\operatorname{argmax}}\, p_{\mathbf{w}}(\mathbf{Y}_{\mathcal{U}},\mathbf{Y}_{\mathcal{O}}|\mathbf{X}) . \tag{4}$$

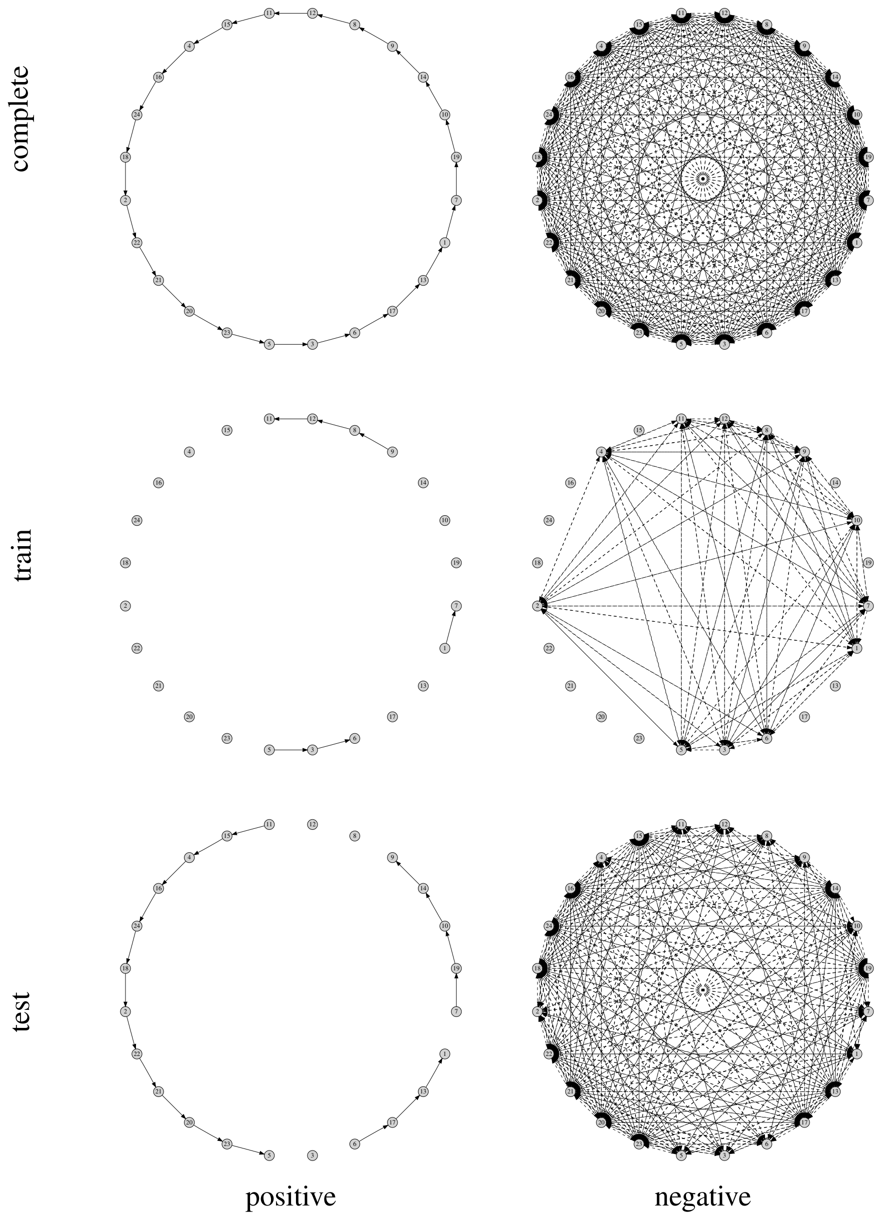


Figure 2: Graph inference on a structured graph. (Top-left) 24 equally spaced nodes on a circle are connected by positive edges running in a counter clockwise direction. As depicted in (top-right) using dashed lines, there are many more negative edges exhibiting a complementary structure. In a typical graph inference setting, a subset of the nodes *and their connectivity* are observed. In this case, one half of the graph nodes numbered 1-12 are observed resulting in (middle-left) positive and (middle-right) negative edges available for training. The goal is to correctly classify the remaining edges into positive and negative categories, as shown in (bottom-left) and (bottom-right) respectively.

Solving Equation (4) is typically called inference. A more challenging problem is that of learning the parameters $\mathbf{w}$ from data. For learning, we use the observed edge features $\mathbf{X}$ and the partially observed connectivity in $\mathbf{Y}_{\mathcal{O}}$ as training data. The first objective in learning $\mathbf{w}$ is to ensure that the connectivity predicted by the model matches that of $\mathbf{Y}_{\mathcal{O}}$. The second objective is to ensure that the model is able to generalize by predicting the correct labels for unobserved edges. A principled approach to achieve this objective is maximum likelihood. The maximum likelihood estimate for the parameter $\mathbf{w}$ is given by the following optimization problem which integrates over the unknown labels

$$\underset{\mathbf{w}}{\operatorname{argmax}} \sum_{\mathbf{Y}_{\mathcal{U}}} p_{\mathbf{w}}\left(\mathbf{Y}_{\mathcal{U}}, \mathbf{Y}_{\mathcal{O}} | \mathbf{X}\right) . \tag{5}$$

However, the quantity being maximized involves summing $2^{|\mathbf{Y}_{\mathcal{U}}|}$ terms where $|\mathbf{Y}_{\mathcal{U}}|$ is the cardinality or the number of unknown labels and moreover the normalization constant $Z_{\mathbf{w}}$ involves summing $2^{|\mathbf{Y}|}$ terms where $|\mathbf{Y}|$ is n^2. Even though many of the terms in the summations are zero, the normalization over these binary variables is computationally too difficult. An alternative approach to achieve the same goals is based on empirical risk minimization.

3. Generalized Matchings

In many applications of graph completions, the degree $\boldsymbol{\delta}_i$ of each node is known or an accurate upper bound can be established. For example, when a structural biologist searches for the 3D structure of a macromolecule, constraints are placed on the number and types of bonds that each molecular unit can make. In the study of protein interactions (Kim et al., 2006), 3D shape analysis has been used to bound the number of mutually accessible binding sites of a protein. In many social networks, the number of connections a user has made is advertised on their profile, even though the set of users to whom they connect is kept confidential (e.g. LinkedIn[1]).

Since in practice the degrees of nodes in a graph are often readily available, and because this information is not currently used by existing methods for graph completion, we decided to study if and how this topological information could be used to improve prediction. Thus, throughout the remainder of the paper, we assume that degree information is available for the graphs that are to be completed. To this end, we consider a class of structured graphs called generalized matchings. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph consisting of all candidate edges, for example, the complete graph over n nodes.

Definition 1 *A generalized matching in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a subgraph defined by edge labels $\mathbf{Y}$ that satisfy in-degree and out-degree constraints. The set of generalized matching is defined precisely as follows*

$$\begin{aligned}\mathcal{B} \triangleq \Big\{ & \mathbf{Y} \in \{0,1\}^{n^2} \mid y_{j,k} = 0 \textit{ if } (j,k) \notin \mathcal{E}, \\ & \sum_j y_{j,k} \leq \boldsymbol{\delta}_k^{in}, \quad \sum_k y_{j,k} \leq \boldsymbol{\delta}_j^{out}, \quad j,k = 1,\ldots,n \Big\} .\end{aligned} \tag{6}$$

for given constants $\boldsymbol{\delta}_k^{in}, \boldsymbol{\delta}_j^{out} \in \{0,\ldots,n\}$.

1. www.linkedin.com

We can use generalized matchings to further specify our probabilistic model of graphs. In this case, there will be a clique in the graphical model for each row and column of random variables from the adjacency matrix $\mathbf{Y}$, and potential functions $\phi_k^{in}(\mathbf{Y}) = \mathrm{H}\left(\boldsymbol{\delta}_k^{in} - \sum_j y_{jk}\right)$ and $\phi_j^{out}(\mathbf{Y}) = \mathrm{H}\left(\boldsymbol{\delta}_j^{out} - \sum_k y_{jk}\right)$ where $\mathrm{H}(\cdot)$ is the Heaviside step function.[2] Substituting these cliques into Equation (3), the complete factorization of the model has the following form

$$p_{\mathbf{w}}(\mathbf{Y}|\mathbf{X}) \propto \prod_j \phi_j^{in}(\mathbf{Y}) \prod_k \phi_k^{out}(\mathbf{Y}) \prod_{j,k} \phi_{\mathbf{w}}(y_{j,k}|\mathbf{x}_{j,k}) \,. \tag{7}$$

Notice that the probability will be zero whenever the degree constraints are not met.

For inference by Equation (4), the generalized matching inspired cliques effectively restrict our search to $\mathbf{Y} = \mathbf{Y}_{\mathcal{U}} \cup \mathbf{Y}_{\mathcal{O}} \in \mathcal{B}$, since all other configurations have zero probability. Setting the edge scores $s_{j,k} = \log(\phi_{\mathbf{w}}(y_{j,k}|\mathbf{x}_{j,k}))$, predictions can then be made by solving

$$\mathrm{GM}: \quad \underset{\mathbf{Y}_{\mathcal{U}}}{\operatorname{argmax}} \sum_{j,k} s_{j,k} y_{j,k} \quad \text{s.t.} \quad \mathbf{Y} \in \mathcal{B}\,, \tag{8}$$

which is a 0-1 programming problem. The quantity $\sum_{j,k} s_{j,k} y_{j,k} = \sum_{j,k}\left(\mathbf{w}^T \mathbf{x}_{j,k}\right) y_{j,k}$ is called the score of the generalized matching specified by $\mathbf{Y}$.

The GM problem has been studied extensively in operations research, and is closely related to methods recently applied in machine learning. If the degrees are $\boldsymbol{\delta}_k^{in} = \boldsymbol{\delta}_j^{out} = 1$, in addition to the graph being bipartite and symmetric, then the set of degree-constrained subgraphs are exactly the 1-matchings of $\mathcal{G}$, which are used to learn structural alignments in (Chatalbashev et al., 2005, Taskar et al., 2005). In the 1-matching case, the iterative Hungarian or Kuhn-Munkres algorithms each provide efficient solutions with worst case $\mathcal{O}\left(n^3\right)$ time complexity. Moreover, it has been shown that the optimal 0-1 solution is found by the linear programming relaxation of GM, which, given the quality of industrial LP solvers, is likely to be faster still.

While in general, the GM problem with arbitrary degrees $\boldsymbol{\delta}_k^{in}$, $\boldsymbol{\delta}_j^{out}$ is significantly more complex, remarkably, it has be reduced to a balanced network flow problem, resulting in $\mathcal{O}\left(n^3\right)$ complexity. A primal-dual solver for this class of problems is available in the Goblin package (Fremuth-Paeger and Jungnickel, 1998). This solver accepts both upper and lower bounds on the degrees, has options for directed or undirected graphs, and also operates on large sparse graphs. Using the sparse solver allows us to predict missing edges without incurring the $\mathcal{O}\left(n^3\right)$ cost of solving for the complete graph connectivity.

4. Learning

In typical methods for graph completion, the edge scores $s_{j,k}$ depend in some way on the node similarity and the edge attributes. Previous researchers have proposed and debated the merits of various node similarity or correlation scores. Instead of relying on our ability to design such scoring functions by hand, we propose to select the scoring function from a large family of candidates, in a data-driven manner. In particular, we assume a scoring function $s_{j,k} = \left(\mathbf{w}^T \mathbf{x}_{j,k}\right)$ that depends linearly on the features via hyperplane parameter $\mathbf{w}$ and adopt a discriminative structured-outputs

2. The Heaviside step function is defined by $\mathrm{H}(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases}$.

framework (Taskar et al., 2003, Tsochantaridis et al., 2004) in order to learn the weight vector $\mathbf{w}$ from data. Structured output models developed within the max-margin framework have been successfully applied to predicting part-of-speech tags, word-alignments, parse trees and image segmentations (Tsochantaridis et al., 2004, Taskar et al., 2004a, 2005). By focusing effort on the margin, these formulations provide a tractable alternative to conditional likelihood modeling which would otherwise require the computation of a partition function (Lafferty et al., 2001).

We begin by learning a weight vector that can predict the edges in $\mathcal{O}$. Consider a fixed but unknown distribution $P(\mathbf{X}, \mathbf{Y})$ over generalized matchings. We would like to select a score function so that the true connectivity $\mathbf{Y}$ achieves a larger score than any alternative set of labels $\mathbf{Z} \in \mathcal{B}$ with $\mathbf{Z}_{\mathcal{O}} \neq \mathbf{Y}_{\mathcal{O}}$

$$\sum_{j,k \in \mathcal{O}} \left(\mathbf{w}^T \mathbf{x}_{j,k}\right) y_{j,k} \geq \sum_{j,k \in \mathcal{O}} \left(\mathbf{w}^T \mathbf{x}_{j,k}\right) z_{j,k} \,, \quad \forall \mathbf{Z} \in \mathcal{B},\, \mathbf{Z}_{\mathcal{O}} \neq \mathbf{Y}_{\mathcal{O}} \tag{9}$$

for any sample $(\mathbf{X}, \mathbf{Y})$ drawn from $P(\mathbf{X}, \mathbf{Y})$. If the weight vector $\mathbf{w}$ is chosen to satisfy Equation (9), then the inference procedure will predict $\hat{\mathbf{Y}} = \mathrm{argmax}_{\mathbf{Z} \in \mathcal{B}} \sum_{j,k \in \mathcal{O}} \mathbf{w}^T \mathbf{x}_{j,k} z_{j,k}$ equal to the correct graph $\mathbf{Y}$, otherwise an error is incurred.

The structured-outputs framework attempts to minimize the probability of error by maximizing a margin quantity $\gamma_{\mathbf{w}} = \gamma_{\mathbf{w}}(\mathbf{X}, \mathbf{Y})$ measured on the training data. Generalizing the multi-class case of (Crammer and Singer, 2001), the margin quantity is defined as the minimum difference between the score of the true connectivity and an alternative connectivity

$$\gamma_{\mathbf{w}}(\mathbf{X}, \mathbf{Y}) \leq \sum_{j,k \in \mathcal{O}} \mathbf{w}^T \mathbf{x}_{j,k} y_{j,k} - \sum_{j,k \in \mathcal{O}} \mathbf{w}^T \mathbf{x}_{j,k} z_{j,k} \,, \quad \forall \mathbf{Z} \in \mathcal{B},\, \mathbf{Z}_{\mathcal{O}} \neq \mathbf{Y}_{\mathcal{O}} \,. \tag{10}$$

The problem of maximizing the margin $\gamma_{\mathbf{w}}$ over bounded weight vectors $\mathbf{w}$ is equivalent to the following optimization problem

$$\begin{aligned} &\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 \\ &\text{s.t.} \sum_{j,k \in \mathcal{O}} \mathbf{w}^T \mathbf{x}_{j,k} y_{j,k} \geq \sum_{j,k \in \mathcal{O}} \mathbf{w}^T \mathbf{x}_{j,k} z_{j,k} + \Delta^{0/1}(\mathbf{Y}, \mathbf{Z}) \,, \quad \forall \mathbf{Z} \in \mathcal{B} \,, \end{aligned} \tag{11}$$

where we have introduced the 0/1 loss term $\Delta^{0/1}(\mathbf{Y}, \mathbf{Z})$ and eliminated the restriction on $\mathbf{Z}_{\mathcal{O}}$. As is common for structured-outputs learning, two further modifications are made to this maximum margin formulation. First, the Hamming loss $\Delta^H(\mathbf{Y}, \mathbf{Z})$ is substituted for the 0/1 loss $\Delta^{0/1}(\mathbf{Y}, \mathbf{Z})$ with the goal of improving generalization (Taskar et al., 2006). The Hamming loss $\Delta^H(\mathbf{Y}, \mathbf{Z})$, counts the number of edge prediction errors $\sum_{j,k} z_{j,k}(1 - y_{j,k}) + y_{j,k}(1 - z_{j,k})$ and can be expressed as $\sum_{j,k} \Delta_{j,k} z_{j,k} + \Delta_0$ for given numbers Δ_0, $\Delta_{j,k}$ depending on $y_{j,k}$. Finally, a slack variable ξ is introduced to allow for potential violations of the margin constraint, with a scaled penalty in the objective. The resulting learning problem is

$$\text{CUTSVM}: \begin{aligned} &\min_{\mathbf{w}, \boldsymbol{\xi}} \frac{1}{2} \|\mathbf{w}\|^2 + C\xi \\ &\text{s.t.} \sum_{j,k \in \mathcal{O}} \mathbf{w}^T \mathbf{x}_{j,k} y_{j,k} \geq \sum_{j,k \in \mathcal{O}} \left(\mathbf{w}^T \mathbf{x}_{j,k} + \Delta_{j,k}\right) z_{j,k} + \Delta_0 - \xi \,, \quad \forall \mathbf{Z} \in \mathcal{B} \,. \end{aligned} \tag{12}$$

Notice that the slack variables are constrained to be positive $\xi \geq 0$ because we include the possibility of $\mathbf{Z} = \mathbf{Y}$ in these constraints. More importantly, notice that the terms in the constraints involve summations over the training points only.

While we have derived our formulation using the structured prediction framework, it is interesting to note that the final optimization problem bears striking resemblance to the recently proposed 1-slack formulation of the support vector machine (Joachims, 2006), which is implemented in SVM-perf. Using the edge data $(\mathbf{x}_{j,k}, y_{j,k})$ as independent examples, the 1-slack formulation considers 2^{n^2} possible linear constraints that are indexed by n^2 binary variables $c_{j,k}\, 1 \le j,k \le n$. In our work, constraints are also indexed in the same way through $z_{j,k} \triangleq y_{j,k} - c_{j,k}$, but the total number of possible constraints is less than 2^{n^2} because the $\mathbf{Z}$ variables are constrained to be generalized matchings. We compare the performance of CUT-SVM against SVM-perf, not only because SVM-perf is the exact *i.i.d.* counterpart of CUT-SVM, but also because SVM-perf is capable of solving large SVM problems such as those encountered in pairwise classification tasks dealing with n^2 pairs.

4.1 The Cutting-Plane Algorithm

At first glance, solving CUTSVM is a formidable task due to the super-exponential number of linear constraints that define the space of feasible solutions $\mathbf{w}$. Using the technique of cutting-planes, however, we can find an approximate solution that is within a pre-specified tolerance of the optimum, in a finite number of steps (Tsochantaridis et al., 2004, Joachims, 2006).

The key idea employed by cutting-plane algorithms is that we are not so much interested in representing the complete feasible region, but instead finding a good approximation to it near the optimal solution (Kelley, 1960). The algorithm works in rounds, generating a nested sequence of approximations $\mathbf{F}_1 \supseteq \ldots \supseteq \mathbf{F}_t$ to the true feasible region, and returning the optimal solution $(\mathbf{w}_t, \xi_t)$ for the current approximation. The regions $\mathbf{F}_t$ are defined by a growing subset of the original constraints, called the cut set $\mathcal{C}$. Each round selects the most violated of the original constraints indexed by $\mathbf{Z} \in \mathcal{B}$, and adds this constraint to the cut set $\mathcal{C}$, thereby shrinking the feasible region $\mathbf{F}_{t+1}$. The algorithm terminates when there are no remaining constraints that are violated by more than the allowed tolerance $\epsilon \ge 0$.

Cut generation, the task of finding the most violated constraint, reduces to a GM subproblem using the loss-augmented edge scores $\left(\mathbf{w}^T\mathbf{x}_{j,k} + \Delta_{j,k}\right)$

$$\mathbf{Z}_t = \operatorname*{argmax}_{\mathbf{Z}\in\mathcal{B}} \sum_{j,k\in\mathcal{O}} \left(\mathbf{w}^T\mathbf{x}_{j,k} + \Delta_{j,k}\right) z_{j,k} \tag{13}$$

Notice again that the GM optimization is performed over the observed subset of edges alone. We write CUTSVM $(\mathcal{C}, C)$ to denote the solution of quadratic programming problem from Equation (12) with slack penalty C but limited to the subset of constraints $\mathbf{Z} \in \mathcal{C}$. For typical problems and with a large enough tolerance ϵ, the number of constraints in $\mathcal{C}$ is manageable and CUTSVM $(\mathcal{C}, C)$ can be solved efficiently. CUTSVM $(\mathcal{C}, C)$ may also be initialized with the solution from the previous iteration. The complete algorithm is shown in the box.

4.2 Transductive Cut Generation

In a general graph completion task, we observe $(\mathbf{X}, \mathbf{Y}_\mathcal{O})$ and the goal is to complete the connectivity $\mathbf{Y}$ by predicting the missing elements $\mathbf{Y}_\mathcal{U}$. Since the features for all edges, training and testing, are available during training, we can use the principle of transductive inference to help guide the selection of the score function, and to improve our ability to complete $\mathbf{Y}$.

Algorithm 1 The cutting-plane algorithm for learning to predict generalized matchings.

```
1:  Input: Graph data $(\mathbf{X}, \mathbf{Y}_\mathcal{O})$, slack penalty $C$, and cut tolerance $\epsilon \geq 0$.
2:  $(\mathbf{w}_1, \xi_t) \leftarrow (0,0),\ \boldsymbol{\mathcal{C}} \leftarrow \emptyset,\ t \leftarrow 0$
3:  repeat
4:    $\mathbf{Z}_t \leftarrow \mathrm{argmax}_{\mathbf{Z}\in\mathcal{B}} \sum_{j,k\in\mathcal{O}} \left(\mathbf{w}^T\mathbf{x}_{j,k} + \Delta_{j,k}\right) z_{j,k}$
5:    if $\sum_{j,k\in\mathcal{O}} \mathbf{w}^T\mathbf{x}_{j,k}y_{j,k} \leq \sum_{j,k\in\mathcal{O}} \left(\mathbf{w}^T\mathbf{x}_{j,k} + \Delta_{j,k}\right) z_{j,k} + \Delta_0 - \xi - \epsilon$ then
6:      $\boldsymbol{\mathcal{C}} \leftarrow \boldsymbol{\mathcal{C}} \cup \{\mathbf{Z}_t\}$
7:      $(\mathbf{w}_{t+1}, \xi_{t+1}) \leftarrow$ CUTSVM $(\boldsymbol{\mathcal{C}}, C)$
8:    end if
9:    $t \leftarrow t+1$
10: until no change in $\boldsymbol{\mathcal{C}}$ (i.e. all cuts satisfied within allowed tolerance $\epsilon$)
```

In the maximum-margin framework, transductive inference is usually accomplished by maximizing a margin quantity over *both labeled and unlabeled* examples (Gammerman et al., 1998, Joachims, 1999, Bennett and Demiriz, 1998, Collobert et al., 2005). Following this general approach, we have modified the method by which cutting-planes are generated, allowing the test edges to influence $\mathbf{w}$ in an indirect fashion. Ideally, we would include the test points when solving the GM with loss-augmented scores, simply by summing over $\mathcal{U}$

$$\mathbf{Z}_t = \underset{\mathbf{Z}\in\mathcal{B}}{\mathrm{argmax}} \sum_{j,k\in\mathcal{O}\cup\mathcal{U}} \left(\mathbf{w}^T\mathbf{x}_{j,k} + \Delta_{j,k}\right) z_{j,k}\,. \tag{14}$$

However, since the entries of $\mathbf{Y}_\mathcal{U}$ are not known, the loss terms $\Delta_{j,k}$ can not be computed for $(j,k) \in \mathcal{U}$. Our proposed solution is to first predict values $\mathbf{V}_\mathcal{U} = \hat{\mathbf{Y}}_\mathcal{U}$ using the current scores

$$\mathbf{V}_t = \underset{\mathbf{V}\in\mathcal{B},\, \mathbf{V}_\mathcal{O}=\mathbf{Y}_\mathcal{O}}{\mathrm{argmax}} \sum_{j,k\in\mathcal{O}\cup\mathcal{U}} \mathbf{w}^T\mathbf{x}_{j,k}v_{j,k}\,, \tag{15}$$

and then to compute the loss terms $\Delta_{j,k}$ using our predicted labels from $\mathbf{V}_\mathcal{U}$. Substituting these $\Delta_{j,k}$ back into Equation (14), we obtain an alternative structure $\mathbf{Z}_t$ that approximates the optimal loss augmented generalized matching. As far as the cutting-plane algorithm is concerned upon the receipt of $\mathbf{Z}_t$, it has obtained another violated constraint corresponding to a valid generalized matching structure; only the observed component $\mathbf{Z}_\mathcal{O}$ is considered further.

5. Implementation Details

This section discusses a number of issues that arise when implementing the cutting-plane algorithm for learning to predict generalized matchings.

5.0.1 Score Computation

A significant computational aspect of CUT-SVM is the calculation, each iteration, of the edge scores $s_{j,k} = \mathbf{w}^T\mathbf{x}_{j,k}\,, \forall j,k$, which requires $\mathcal{O}\left(n^2 d\right)$ operations, where d the feature dimension. This issue is compounded by that fact that d may be large, and so the complete set of features may not fit in memory. For the large graphs that we studied, we were able to store attributes of each node, and recompute the edge features $\mathbf{x}_{j,k} = \mathbf{f}\left(\mathbf{x}_j, \mathbf{x}_k\right)$ on the fly.

5.0.2 NEGATIVE SCORES

It is easy to show that exact inference will never select edges that have negative scores; omitting the edge would result in a higher scoring graph that also satisfies the upper degree constraints. Therefore, we could remove edges with negative scores in order to speed-up certain generalized matching solvers. However, such thresholding complicates CUT-SVM learning because the structures $\mathbf{Z}_t$ will have varying numbers of edges, which in turn introduces uneven biases through the linear terms $\sum_{j,k} \mathbf{x}_{j,k} \left(y_{j,k} - z_{j,k}\right)$ of the constraints.

Ideally, we would like the matching solver to be invariant to positive or negative shifts of the score values. One solution is to change the definition of generalized matchings to use equality constraints instead of upper degree bounds, thus ensuring that all structures $\mathbf{Z}_t$ have the same number of edges even though some of their scores are negative. Two difficulties of this approach are first, that the matching solver may not support equality constraints, and second, the degree constraints may become impossible to satisfy if the degrees are estimated poorly. Moreover, in practice it is much more reasonable to derive an upper bound on the degree, rather than its exact value. Hence, the alternative solution that is adopted here is to constrain the scores to be positive, and use upper degree bounds in the definition of the generalized matchings as in Equation (6). One method to ensure positive scores is to use only positive features, and additionally, to constrain the weight vector to be positive. An easier approach is to add a sufficiently large constant to the scores.

5.0.3 GREEDY APPROXIMATE INFERENCE

The degree-constrained subgraph algorithm has time complexity of $\mathcal{O}\left(n^3\right)$. We measured the runtime of the GOBLIN solver on a number of real-world generalized matching problem instances using random score matrices, and report the results in Figure 3. Times were measured on a 2GHz Intel Core Duo processor. Since this is the state of the art, it is impractical to use exact inference within the cutting-plane algorithm for more than a few iterations on graphs with $n > 300$. To address this issue, we considered greedy approximations to GM. One extremely simple and fast greedy algorithm was proposed recently by (Mestre, 2006) which has theoretical running time bound of $\mathcal{O}\left(\max_i \boldsymbol{\delta}_i \left|\mathcal{E}\right|\right)$. While it is not always able to find the optimal generalized matching, this algorithm has a 1/2-factor approximation guarantee. The greedy algorithm is replaced by the exact algorithm whenever the cut-generation fails, and for testing.

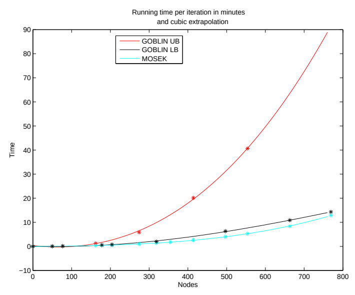


Figure 3: Inference times

Other approximations based on loopy belief propagation techniques have been investigated recently in (Huang and Jebara, 2007, Sanghavi et al., 2007). While these approximations should be used with caution, as demonstrated in (Kulesza and Pereira, 2007), their efficiency makes them very attractive for cutting-plane approaches.

5.0.4 DEGREE CONSTRAINTS FOR TESTING

Testing can be performed in two different ways. The first method solves the GM problem over $\mathcal{O} \cup \mathcal{U}$ with the provided degrees $\boldsymbol{\delta}_i$, and subsequently returns the labels from $\mathbf{Y}_{\mathcal{U}}$. The second method solves the GM over $\mathcal{U}$ using the residual degrees after subtracting the training edges $\boldsymbol{\delta}'_j =$

$\delta_j - \sum_{k:(j,k)\in\mathcal{O}} y_{j,k}$, and subsequently returns the result $\mathbf{Y}_{\mathcal{U}}$. In practice was have found that the second method works better, probably because the resulting GM inference problem is more constrained.

6. Evaluation

One of the motivations of this work was to understand if, when, and where topological constraints could be used to improve the prediction of missing edges in a partially observed graph. For this reason, we tried to design a system that was capable of working with a wide range of network data sets. This required flexibility in handling directed and undirected edges as well as various sorts of node and edge attribute data.

6.1 Connectivity and Attribute Data

The graph data we considered included:

- Simple directed graphs embedded in $\mathbb{R}^2$ and $\mathbb{R}^3$: these networks were synthetically generated and included a circle, torus, star, etc.
- A directed cell-signaling network (Sachs et al., 2005): for this network, the expression levels of 11 signaling molecules subjected to external stimuli were measured using a technology called flow cytometry. The network is defined by the causal influences of one molecule on another. These have been determined through extensive and time consuming laboratory study.
- An undirected protein-interaction graph (Yamanishi et al., 2004), and an undirected enzyme-interaction graph (Yamanishi et al., 2005): these data sets are described by several $n \times n$ kernels, each encoding similarity of a biological attribute. The edges, which correspond to physical or chemical interactions, are determined experimentally.
- A directed citation network from CoRA (McCallum et al., 2000): this database contains document attributes (e.g. abstract, authors' names, date of publication, topic classification), in addition to a citation network (edges from one paper to another) for several thousand papers on machine learning.
- An undirected friendship graph from a social network: web pages and the accompanying "friendship" structure were manually downloaded in a neighbourhood a seed page.

We have tailored our model to handle both directed and undirected graphs independently, but we do not handle graphs that have both directed and undirected arcs. While self loops are handled, we did not find an example where we needed to use them.

In all cases except the protein-interaction and enzyme-interaction examples, the attribute data we collected were measurements associated with the nodes. For the embedded geometric graphs, we used the coordinates in space for the node features. The node attribute data for the cell-signaling network are the expression levels of various molecules. For the citation and social network, the attribute data for the nodes consists of formatted text. In order to compress the bag-of-words representations from the citation network and the social network, we use latent Dirichlet allocation (LDA) (Griffiths and Steyvers, 2004). This resulted in a vector of 111 dimensions describing the documents in CoRA, and a 171 dimensional feature vector describing an individual user's page.

k	1	2	3	4	5	6	7	8	9	10	11	12	14
n	668	667	608	483	407	326	287	243	174	81	75	41	15

Table 1: The relationship between the size of the core and degree k.

The protein and enzyme data sets were instead described by several $n \times n$ kernels each. While the kernel values themselves could be used as edge features, we adopted the empirical feature map approach in order to extract relevant features for the nodes. This was accomplished by decomposing the kernel $K = (k_{j,k})$ via eigendecomposition. The resulting features $\mathbf{x}_j$ measure the similarity of the j-th node to a set of prototypes, like the empirical feature map (Tsuda, 1999).

In all cases except the signaling network, given a pair of nodes (j, k) we defined the features for edge $\mathbf{x}_{j,k} \triangleq \text{vec}\left(\mathbf{x}_j \mathbf{x}_k^T\right)$ using an outer product of the node features. The result is a high dimensional representation for edges that includes all pairwise products between a feature from j and a feature from k. In general $\mathbf{x}_{j,k} \neq \mathbf{x}_{k,j}$ which provides an opportunity for the model to discriminate edge directionality, if required by the task. For the signaling network, the edge features were constructed by hand for interpretability of the learned weight vector, and consistency with previous work.

As mentioned earlier, we have assumed that accurate upper degree bounds can be estimated in each application. Thus, for these preliminary experiments, we have used the true degree $\boldsymbol{\delta}_i$ calculated over the entire graph. In the subsequent sections, we describe experiments on geometric graphs, protein-interaction graphs and cell-signaling graphs.

6.2 Experimental Protocol

We used the notion of the k-core of a graph to select subgraphs of manageable size for development purposes and to observe behaviour of CUT-SVM closely. A core of a graph of degree k is the set of nodes and edges that remain after recursively pruning any node (and its edges) with degree $\boldsymbol{\delta} < k$. The cores of a graph form a nested sequence, much like the various cores of an onion. The relationship between the degree k and the size of the core is shown in Table 1 for the enzyme-interaction graph from (Yamanishi et al., 2005).

We perform 5-fold cross-validation to evaluate and compare CUT-SVM and *i.i.d.* SVM using one of the bipartite schemes from Figure 1. The bipartite partition of edges $\mathcal{O} \cup \mathcal{U}$ is determined from the train-test partition of the nodes on each fold. Thus, we first divide the n nodes into 5 subsets of roughly equal size. Then, on the k-th fold, the testing subset of nodes is determined by the k-th subset, while the remaining nodes comprise the training subset.

The algorithms train on data $(\mathbf{X}, \mathbf{Y}_{\mathcal{O}})$, and are evaluated based on their predictions for the unknown labels $\mathbf{Y}_{\mathcal{U}}$. We look at the area under the curve (AUC), accuracy and recall. For a fair comparison with CUT-SVM, the threshold of the *i.i.d.*-SVM classifier is chosen so that it predicts $N_{tot} = \sum_i \boldsymbol{\delta}_i / 2$ edges, which is the same number of edges predicted by CUT-SVM. Since CUT-SVM predicts 0-1 outputs, the threshold is alway fixed at 0.5. The cross validated performance for a given metric, is taken to be the average performance measured on the testing edges $\mathcal{U}$ across folds.

6.3 Results

6.3.1 Completing the Circle Experiments

The purpose of this example is to validate our algorithms. To generate the graph, we placed $M = 24$ points on a circle centered on the origin with even spacing. These were connected using directed edges in counter-clockwise order. The x-y coordinates of the points were used as node attributes. To make this example challenging, we used a random 2:3 partition of the nodes as can be seen in the top row of the Figure. 4. If we use larger training sets, then each method described below obtains near-perfect performance.

We compared *i.i.d.*-SVM, CUT-SVM and CUT-SVM with transduction, using a slack penalty $C = 1$. We used a fairly loose tolerance $\epsilon = 0.1$. The bottom row of the figure shows the completions predicted by these algorithms. The *i.i.d.*-SVM in (d) ignores the degree constraints and appears to have been confused by the 87 negative edges which far out-number the 3 positive ones. The CUT-SVM algorithm without transduction (d) does somewhat better primarily due to the use of the degree constraints. The training set is still too small to learn an accurate model, especially given the loose tolerance ϵ. Finally, in (f) we see that the CUT-SVM with transduction has predicted nearly all edges correctly. This is primarily due to the fact that the transductive model adapts its learning to the presence of the test points.

6.3.2 Enzyme-Enzyme Interactions

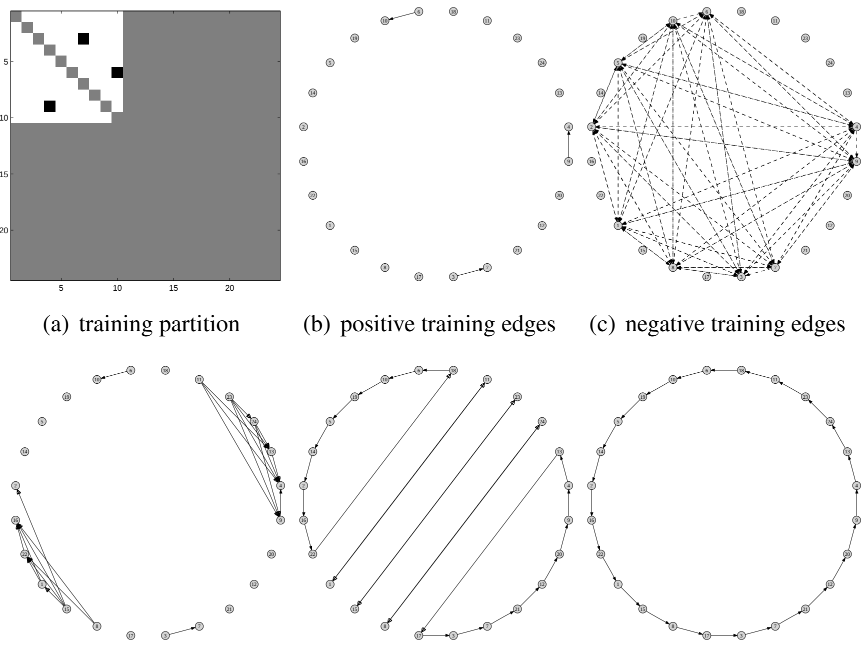
We focus next on a completion task involving the enzyme-interaction network from (Yamanishi et al., 2005). We performed an eigendecomposition on the “gene expression”, “localization”, and “phylogenetic profile” kernels, keeping enough eigenbases to explain 50% of the variance of each kernel, resulting in a 44 dimensional feature vector for each node. The outer product $\mathbf{x}_j\mathbf{x}_k^T$ has 1936 elements, however, since the network is undirected, symmetry constraints are used to reduce this dimension to 990.

The core of size $n = 81$ was used for this experiment. We set $\epsilon = 0.05$ and varied the slack parameter across the following values $C = \{1e2, 1e4, 1e6, 1e8, 1e10, 1e12, 1e14\}$. We initialized the weight vector of the cutting-plane algorithm using the solution from *i.i.d.*-SVM.

In the following discussion, we describe in detail several aspects of the model, making reference to the following figures.

- Figure 5 displays learning curves for just one of the many runs of CUT-SVM.
- Figure 6 compares the results of CUT-SVM with that of *i.i.d.*-SVM in terms of the learned scores, the predictions and the ROC curves.
- Figure 7 details the behaviour of the algorithms across the range of C values.

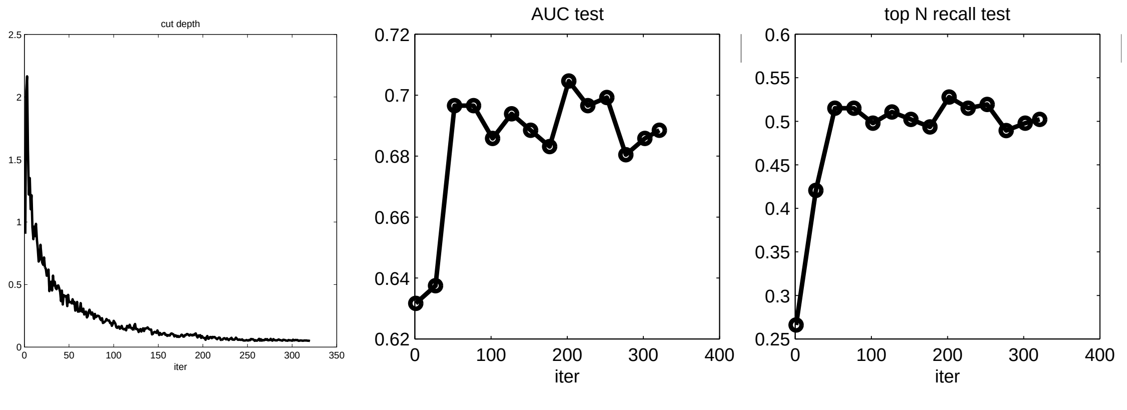
To begin, observe how in Figure 5 (a) , the depth of the cutting-planes converges quickly in the first iterations and yet remains positive with a long tail. This is a typical profile for a problem with many similar constraints, most of which are not active at the optimum. This behaviour also validates the finite convergence theorems of (Tsochantaridis et al., 2004, Joachims, 2006). In (b) and (c), we see the performance on the test edge, measured every 25 iterations. While these curves quickly level off, in the first few iterations there is significant improvement over the *i.i.d.*-SVM algorithm that is shown at iteration 0.



(a) training partition (b) positive training edges (c) negative training edges

(d) i.i.d. SVM (e) CUT SVM (f) transductive CUT SVM

Figure 4: Completing the connectivity of a circle. The training subset, generated from a 2:3 split of the nodes, is depicted in (a). The black entries indicate positive edges which are also shown in (b). The white entries indicate negative edge which are also shown in (c). The bottom row shows the predicted completions; the edges from the test set that were predicted to be positive, in addition to the positive edges from the training set. False-positives are demarked by open arrows, which can be seen by magnifying the individual figures. Results include (d) *i.i.d.* SVM, (e) CUT-SVM without using transduction, and (f) CUT-SVM using transduction.



(a) cut depth (b) area under curve (c) recall

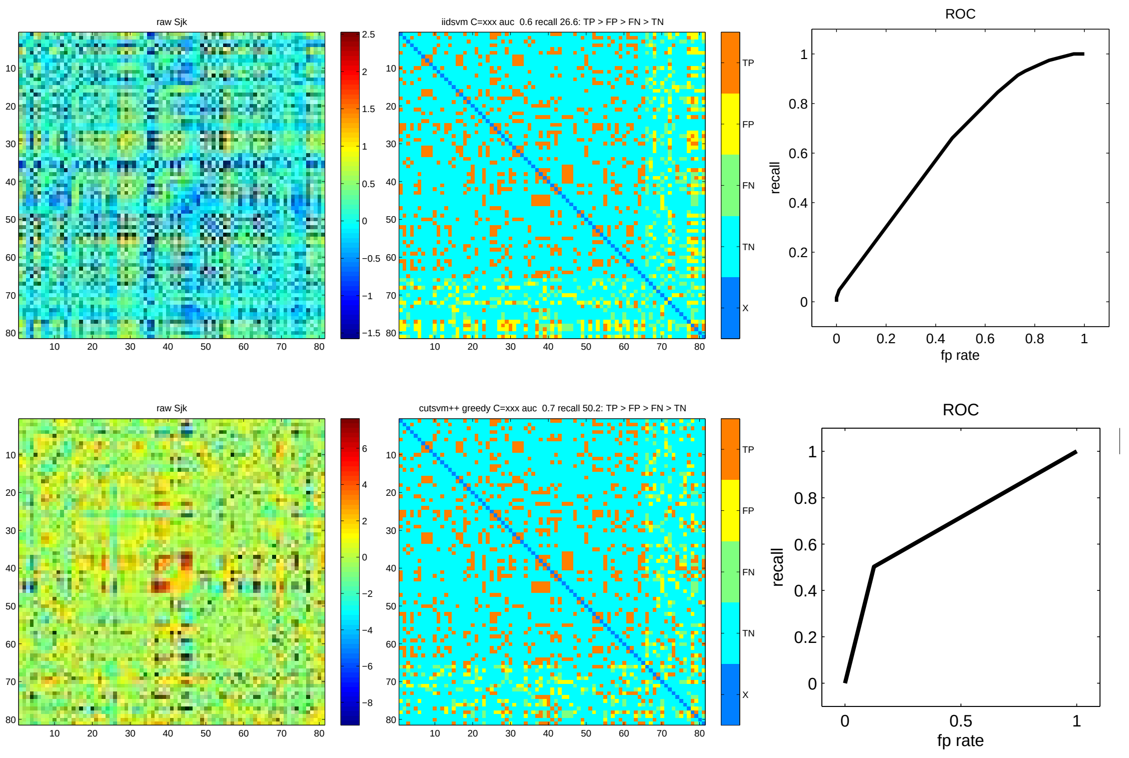
Figure 5: Learning curves for CUT-SVM on the size $n = 81$ core enzyme-interaction network.

Clearly, the degree constraints are helping CUT-SVM. This effect is also evident in the score and prediction matrices in Figure 6 (a-b). The prediction matrix uses color entries to indicate how an edge is classified, with colors for true-positives (tp), true-negatives (tn), false-positives (fp), and false negatives (fn). The *i.i.d.*-SVM algorithm cannot help but learn a banded score matrix, and it will make predictions that are similarly banded, which is evident from the Figure 6 (b). Figure 7 provides a more complete view of the performance of both algorithms across a range of C values. The difference in the recall measure of the two algorithms is striking. For example, at the largest C value the average recall of *i.i.d.*-SVM is 27.3 ± 3.9, while that of CUT-SVM is 46.4 ± 5.3, which is significant at $p \ll 0.001$. This demonstrates the significant effect that topology has on improving prediction.

We have also run CUT-SVM on larger core examples adapting parameters to ensure timely convergence. For example, on the size 174 core, using a relatively large tolerance of $\epsilon = 0.2$, we were still able to obtain impressive results (cf. Figure 8). In order to save on the training time for the models that use large C values, we initialized with the final cutting-plane models with cuts runs with smaller C values. This explains why the average training time of the CUT-SVM model decreases for the largest C value. The ability to initialize CUT-SVM with a arbitrary collections of cutting-planes in this manner can be of great importance in practical applications, when training must be repeated in a dynamic or online setting, or when using multiple processors to find cuts.

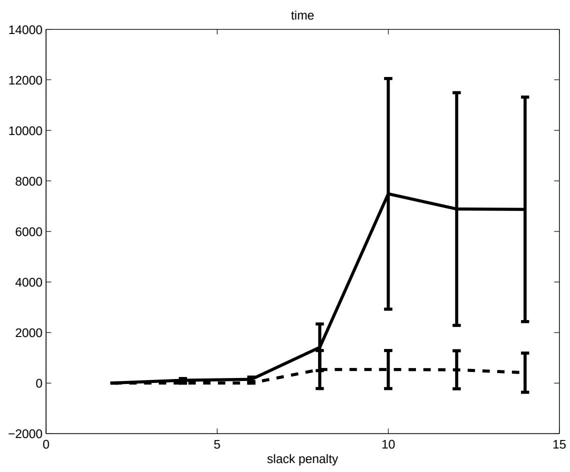
It is interesting to note that while CUT-SVM outperforms *i.i.d.*-SVM in terms of accuracy and recall, *i.i.d.*-SVM performs better in terms of AUC (cf. Figures 6 and 8, right side). In fact, for the size 174 core with a slack penalty of $C = 10^8$, the average AUC is 0.67 ± 0.06 for *i.i.d.*-SVM, and 0.64 ± 0.01 for CUT-SVM. Part of the reason for this, is that CUT-SVM has 0-1 outputs, giving rise to a piecewise linear ROC curve which has less area than the rounded curve generated by *i.i.d.*-SVM. However, it is also by design that the CUT-SVM algorithm focuses on predicting the most likely matching, and places less emphasis on the overall ordering of the edge scores.

We were able to run CUT-SVM on the core of size $n = 326$ in roughly 8 hours. At test time, it takes 1 hour to solve GM using the exact inference algorithm. A complete set of results for this and larger networks are expected for the final version of this paper.

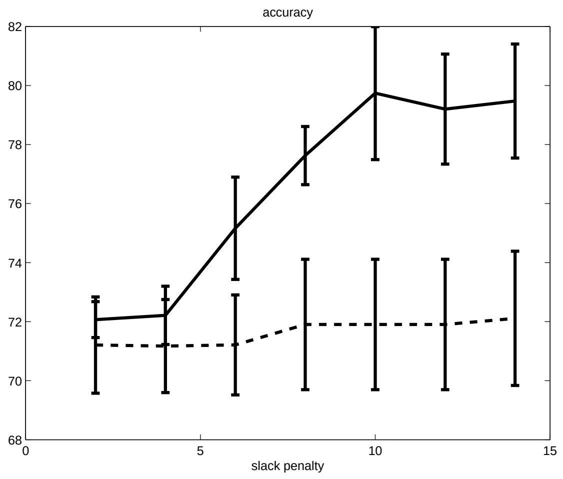


(a) score matrix (b) predictions (c) ROC

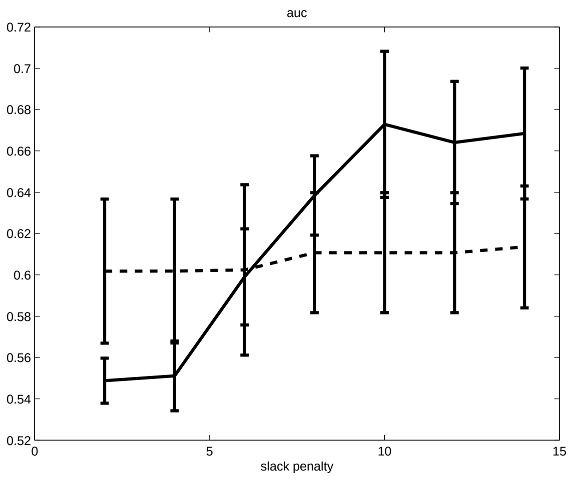
Figure 6: A comparison of score matrices (column a), predictions matrices (column b), and receiver-operator curves (ROC) for (top) *i.i.d.*-SVM and (bottom) CUT-SVM, on the size 81 core enzyme-interaction network.



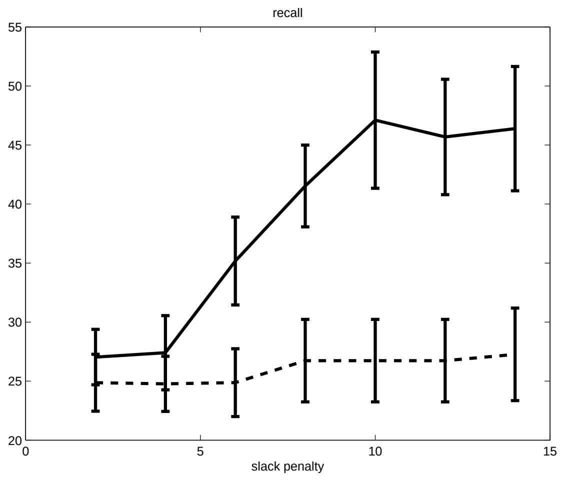
(a) time (sec.)



(b) accuracy

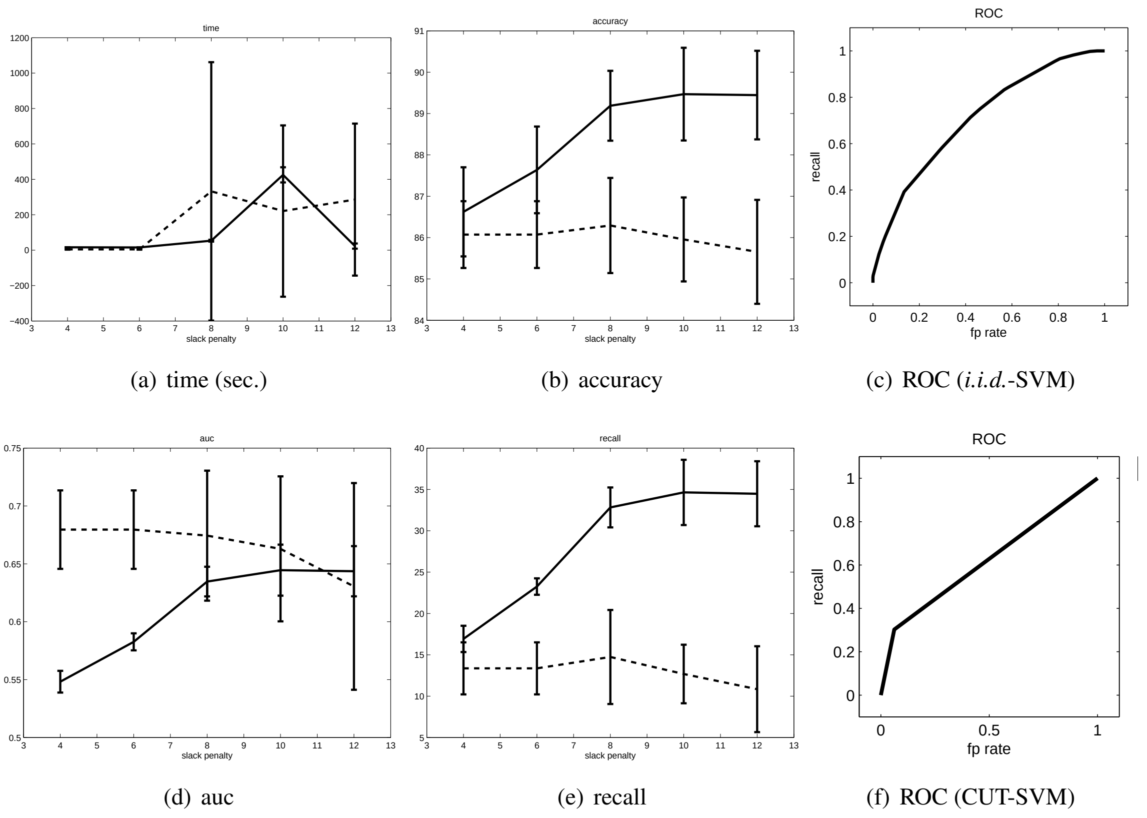


(c) auc



(d) recall

Figure 7: A comparison of running time (seconds), accuracy, area under the curve and recall, while varying the slack parameter C for (dashed line) *i.i.d.* SVM and (solid line) CUT-SVM, on the size 81 core enzyme-interaction network. The x-axis reports $\log_{10}(C)$.



(a) time (sec.)

(b) accuracy

(c) ROC (*i.i.d.*-SVM)

(d) auc

(e) recall

(f) ROC (CUT-SVM)

Figure 8: (Right: a,b,d,e) A comparison of running time (seconds), accuracy, area under the curve and recall, while varying the slack parameter C for (dashed line) *i.i.d.* SVM and (solid line) CUT-SVM, on the size 174 core enzyme-interaction network. The x-axis reports $\log_{10}(C)$. (Left: c,f) A comparison of receiver-operator curves for (top) *i.i.d.* SVM and (bottom) CUT-SVM, on the size 174 core enzyme-interaction network. This comparison is for $\log_{10}(C) = 8$, where the average AUC for the *i.i.d.*-SVM algorithm is greater than that of CUT-SVM. Notice, however, that at the same C value, the ordering of the two algorithms in terms of average recall is reversed.

T	2700	1350	675	364	*i.i.d.* SVM T=2700
AUC (recall)	0.97 (80)	0.96 (79)	0.96 (77)	0.96 (77)	0.94 (73)

Table 2: Predicting causal influence between molecules in a signaling network with limited data (T measurements). Results of CUT-SVM averaged over 5 repeated trials. A trial randomly selects two disjoint $11 \times T$ slices of the expression data for training and for testing. The final column is the performance of an *i.i.d.* support vector machine trained on the edge features.

6.3.3 PREDICTING CAUSAL RELATIONSHIPS IN A CELL

As a final example, we consider the cell signaling network previously studied in (Sachs et al., 2005, Eaton and Murphy, 2007). Our goal was to address two of the limitations of Bayesian networks outlined by (Sachs et al., 2005): 1) the inferred dependency graph is acyclic, and 2) effective inference requires many observations. Using CUT-SVM in this framework allows us to learn a function that maps intervention data into a causal graphs. However, being a supervised method, CUT-SVM requires that some of the causal relationships are known a priori. It is remarkable that the Bayesian methods operate in an unsupervised fashion.

Flow-cytometry data was collected and discretized into 3 levels by (Sachs et al., 2005). The data is summarized by two 11×5400 matrices. The first contains discrete expression levels 1-3, and the second contains the assumed perfect intervention state. We use pairwise flow-cytometry measurements to describe each possible directed edge $j \to k$ by a feature vector $\mathbf{x}_{j,k}$. Our representation, which is based on contingency tables of the expression levels, is constructed to be: 1) invariant to the ordering and number of flow-cytometry measurements; 2) sensitive to correlations between levels but invariant to their sign; and 3) sensitive to both parent and child interventions, under the assumption that they are not coincident (Andrews and Jebara, 2007).

For our first experiment, the goal was to see if we could generalize across different subsets of the expression data. We randomly sampled two disjoint feature subsets of size $11 \times T$ from the original 11×5400 data matrices, to be used for training and testing. We then duplicated the 11 nodes, and assigned one set of features to each copy, thereby creating two disjoint and independent networks, one for training and one for testing. This procedure creates a graph completion task similar to that shown in Figure 1 column (b). The results included in Table 2 show that we can generalize well from sample to sample, and that performance does not degrade for small sample sizes. For comparison, we have included the performance of an *i.i.d.* SVM on subsets of size T=2700. These results show the benefit of using supervision in reconstructing biological networks, and furthermore the advantage of using generalized matchings within a structured prediction framework for this task.

For our second experiment, depicted in Figure 9, we tested whether we could complete a network that was partially observed, this time using all 5400 samples. We created a graph completion task as in Figure 1 column (a), using a 2:1 split of the nodes. Here, we discovered that we could complete the network with an average area under the curve (AUC) of 0.93 and recall of 80%.

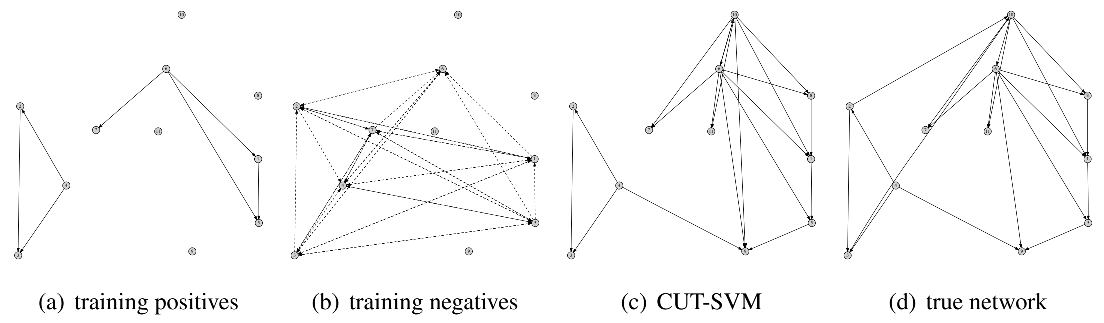


Figure 9: Predicting causal influence between molecules in a signaling network. Positive and negative edges used for training are shown in (a) and (b). The completed graph after learning is shown in (c) and the ground truth in (d). False-positives are demarked by open arrows.

7. Related Work

Graphs appear in a variety of supervised and unsupervised machine learning techniques using a combination of node and edge attributes for enhanced modeling. Many methods leverage a relational structure to learn classifications, clusterings, and/or rankings for individual nodes (e.g. (Weston et al., 2004)(Kleinberg, 1998)(Page et al., 1999)), and for entire graphs (Tsuda and Kudo, 2006),(Borgwardt et al., 2005),(Kudo et al., 2005)). To save space, we mention briefly only methods that perform relational learning of some form. Existing machine learning methods for predicting missing edges in a network can be categorized roughly as follows:

1. Pairwise methods, such as metric, kernel and distance learning such as (Xing et al., 2003, Goldberger et al., 2004, Shalev-Shwartz et al., 2004, Kondor and Jebara, 2006, Lanckriet et al., 2002, Alfakih et al., 1999) or (Vert and Yamanishi, 2004, Ben-Hur and Noble, 2005) mentioned above.

2. Topological methods like (Liben-Nowell and Kleinberg, 2003).

3. Methods for structure learning in Bayesian networks, such as (Jaimovich et al., 2006) and (Sachs et al., 2005).

4. Learning to predict structured output variables (Bakir et al., 2007).

Structure learning in Bayesian networks is concerned with the recovery of dependencies amongst a collection of random variables, given multiple independent observations of the random variables. All of the data contributes to the reconstruction of the network. Our work is similar to structure learning in Bayesian networks, because we are seeking a single structure. However, the nodes and the feature vectors in our work, do not correspond to repeated observations of random variables. Structure learning in a Bayesian network setting usually proceeds by a greedy hill search until no improvements can be made to the graph likelihood or posterior, at a (local) maximum. There are also many instances of more elaborate probabilistic models dealing with relational structure (Getoor et al., 2001, Friedman and Koller, 2003, Taskar et al., 2004b, Pe'er et al., 2006, Airoldi et al., 2007) although we are unaware that any of these use topological constraints such as ours.

The task of learning to predict structured output variables has received a great deal of attention recently (see (Bakir et al., 2007) for a broad overview), with applications from natural language processing (Koo et al., 2007) to biological sequence analysis (Rätsch et al., 2007). Here the goal is to find a function that maps multivariate structured inputs to multivariate structured outputs, given multiple independent examples of this mapping. The task involves many related structures (many sentences or many mRNA sequences), instead of completing just a single structure as we consider in this paper. To our knowledge, no researchers have coupled structured output learning with degree-constrained subgraphs.

While there exist several discriminative methods for learning to predict structured outputs, we prefer the cutting-plane method because of its relative simplicity, ability to generalize, and dependence on standard solvers. Moreover, our algorithm is closely related to SVM-perf, to which we compare our results. We did not use the perceptron because it was not competitive with CUT-SVM. We tried extragradient approach, but ran into computational difficulty solving $\mathcal{O}\left(n^3\right)$ quadratic flow problems. On the other hand, CUT-SVM, with its two step loop that solves loss-augmented inference and then resolves a quadratic program, allowed us to plug-n-play with various inference algorithms in a modular fashion.

8. Conclusions

This work introduces a structured-output model for learning to predict generalized matchings based on attribute and topology data that are available in real-world settings. First, we propose a probabilistic model for generalized matchings. Then, adopting a discriminative approach based on empirical risk minimization, we derive an optimization problem for learning the model parameters. A conceptually simple cutting-plane procedure has been optimized to approximate the solution efficiently.

The main challenge as far as efficiency is concerned is solving the GM inference problem. During the development of the cutting-plane algorithm, we explored several approximation methods for this step; trading-off the quality (i.e. depth) of the cut with its generation time. We are currently exploring automated schedulers that attempt to maximize progress in the minimum amount of time.

A transductive extension of the algorithm is proposed that has been shown to generalize better than the non-transductive version. In practice, the transductive version is slower because it requires two calls to the GM subproblem each iteration, and usually requires more iterations to converge. Developing a more robust and efficient version is a goal for our future work.

Evaluation on several graph completion tasks demonstrate the versatility of our model and implementation. The results demonstrate that it is possible to learn a metric that facilitates prediction of structured networks. These network predictions are useful because they not only predict edges accurately, but do so with high recall. We also provide encouraging results that demonstrate it is possible to use structured learning with generalized matchings to learn to predict graphs with several hundreds of nodes.

References

E.M. Airoldi, D.M. Blei, S.E. Fienberg, and E.P. Xing. Mixed membership stochastic blockmodels. *URL http://arxiv. org/abs/0705.4485. Manuscript under review*, 2007.

A.Y. Alfakih, A. Khandani, and H. Wolkowicz. Solving Euclidean Distance Matrix Completion Problems Via Semidefinite Programming. *Computational Optimization and Applications*, 12(1): 13–30, 1999.

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

Y. Altun, T. Hofmann, and A.J. Smola. Gaussian process classification for segmenting and annotating sequences. *ACM International Conference Proceeding Series*, 2004.

Y. Altun, D. McAllester, and M. Belkin. Maximum margin semi-supervised learning for structured variables. *Advances in Neural Information Processing Systems*, 18, 2005.

S. Andrews and T. Jebara. Graph reconstruction with degree-constrained subgraphs stuart andrews. In *Workshop on Statistical Models of Networks - NIPS*, 2007.

G. Bakir, T. Hofmann, B. Schölkopf, and S. V. N. Vishwanathan. *Predicting Structured Data*. MIT Press, Cambridge, Massachusetts, 2007.

P. Baldi, S. Brunak, P. Frasconi, G. Soda, and G. Pollastri. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15(11):937–946, 1999.

A. Ben-Hur and W.S. Noble. Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21(Suppl 1):i38–i46, 2005.

K. Bennett and A. Demiriz. Semi-supervised support vector machines. *Advances in Neural Information Processing Systems*, 11:368–374, 1998.

K. Bleakley, G. Biau, and J.P. Vert. Supervised Reconstruction of Biological Networks with Local Models. *Bioinformatics*, 23(13):57–65, 2007.

K.M. Borgwardt, C.S. Ong, S. Schoenauer, SVN Vishwanathan, A. Smola, and H.P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(Suppl 1):i47–i56, 2005.

V. Chatalbashev, B. Taskar, and D. Koller. Disulfide connectivity prediction via kernelized matching. In *RECOMB*, 2005.

G. Chechik, G. Heitz, G. Elidan, P. Abbeel, and D. Koller. Max-margin classification of incomplete data. In *Advances in Neural Information Processing Systems*, 2007.

R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive svms. *Journal of Machine Learning Research*, 2005.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, pages 265–292, December 2001.

D. Eaton and K. Murphy. Exact Bayesian structure learning from uncertain interventions. In *Artificial Intelligence and Statistics, AISTATS*, March 2007.

E. Even-Dar and M. Kearns. A small world threshold for economic network formation. In *NIPS*, 2006.

C. Fremuth-Paeger and D. Jungnickel. Balanced network flows. a unifying framework for design and analysis of matching algorithms. *Networks*, 1998.

N. Friedman and D. Koller. Being bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Machine Learning*, 2003.

A. Gammerman, V. Vovk, and V. Vapnik. Learning by transduction. In *UAI*, pages 148–155, 1998. URL `citeseer.ist.psu.edu/gammerman98learning.html`.

B. Gassend, C. W. O'Donnell, W. Thies, A. Lee, M. van Dijk, and S. Devadas. Learning biophysically-motivated parameters for alpha helix prediction. *BMC Bioinformatics*, 2007.

L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of relational structure. In *ICML*, 2001.

A. Globerson and S. Roweis. Metric learning by collapsing classes. In *NIPS*, 2006.

J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *NIPS*, 2004.

T. L. Griffiths and M. Steyvers. Finding scientific topics. *PNAS*, 2004.

M.J. Herrgaard, M.W. Covert, and BO Palsson. Reconstruction of microbial transcriptional regulatory networks. *Current Opinion in Biotechnology*, 15(1):70–77, 2004.

B. Huang and T. Jebara. Loopy belief propagation for bipartite maximum weight b-matching. *Artificial Intelligence and Statistics (AISTATS)*, 2007.

A. Jaimovich, G. Elidan, H. Margalit, and N. Friedman. Towards an integrated protein-protein interaction network: A relational markov network approach. *Journal of Computational Biology*, 2006.

T. Joachims. Training linear SVMs in linear time. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, 2006.

T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, 1999. URL `citeseer.nj.nec.com/joachims99transductive.html`.

N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 2004.

J. E. Kelley. The cutting-plane method for solving convex programs. *Industrial and Applied Mathematics*, 1960.

P.M. Kim, L.J. Lu, Y. Xia, and M.B. Gerstein. Relating Three-Dimensional Structures to Protein Networks Provides Evolutionary Insights. *Science*, 314(5807):1938, 2006.

J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

R. Kondor and T. Jebara. Gaussian and wishart hyperkernels. In *NIPS*, 2006.

T. Koo, A. Globerson, X. Carreras, and M. Collins. Structured Prediction Models via the Matrix-Tree Theorem. *Proc. EMNLP*, 2007.

T. Kudo, E. Maeda, and Y. Matsumoto. An application of boosting to graph classification. *Advances in Neural Information Processing Systems*, 17, 2005.

A. Kulesza and F. Pereira. Structured Learning with Approximate Inference. *Advances in Neural Information Processing Systems*, 2007.

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

Gert R.G. Lanckriet, N. Cristianini, and P. Bartlett. Learning the kernel matrix with semi-definite programming. In *ICML*, 2002.

D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM*, 2003.

A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval Journal*, 3:127–163, 2000. URL `www.research.whizbang.com/data`.

J. Mestre. Greedy in Approximation Algorithms. *LECTURE NOTES IN COMPUTER SCIENCE*, 4168:528, 2006.

M. Middendorf, A. Kundaje, C. Wiggins, Y. Freund, and C. Leslie. Predicting genetic regulatory response using classification. *Bioinformatics*, 20(1):232–240, 2004.

L. Page, S. Brin, R. Motwani, and T. Winograd. The page rank citation ranking: Bringing order to the web. Technical report, Stanford University, 1999.

D. Pe'er, A. Tanay, and A. Regev. MinReg: A Scalable Algorithm for Learning Parsimonious Regulatory Networks in Yeast and Mammals. *The Journal of Machine Learning Research*, 7: 167–189, 2006.

M. Punta and B. Rost. PROFcon: novel prediction of long-range contacts. *Bioinformatics*, 2005.

G. Rätsch, S. Sonnenburg, J. Srinivasan, H. Witte, K.R. Müller, R.J. Sommer, and B. Schölkopf. Improving the Caenorhabditis elegans Genome Annotation Using Machine Learning. *PLoS Computational Biology*, 3(2):313–322, 2007.

K. Sachs, O. Perez, D. Pe'er, D.A. Lauffenburger, and G.P. Nolan. Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data. *Science*, 308(5721):523–529, 2005.

S. Sanghavi, D.M. Malioutov, and A.S. Willsky. Linear Programming Analysis of Loopy Belief Propagation for Weighted Matching. *Advances in Neural Information Processing Systems*, 2007.

Shai Shalev-Shwartz, Yoram Singer, and Andrew Y. Ng. Online and batch learning of pseudo-metrics. In *ICML*, 2004.

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

B. Taskar, D. Klein, M. Collins, D. Koller, and C. D. Manning. Max-margin parsing. In *EMNLP*, 2004a.

B. Taskar, M. F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *NIPS*, 2004b.

B. Taskar, S. Lacoste-Julien, and D. Klein. A discriminative matching approach to word alignment. In *EMNLP*, 2005.

B. Taskar, S. Lacoste-Julien, and M. I. Jordan. Structured prediction, dual extragradient and bregman projections. *JMLR*, 2006.

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

K. Tsuda. Support Vector Classifier with Asymmetric Kernel Functions. *European Symposium on Artificial Neural Networks (ESANN)*, pages 183–188, 1999.

K. Tsuda and T. Kudo. Clustering graphs by weighted substructure mining. *Proceedings of the 23rd international conference on Machine learning*, pages 953–960, 2006.

J.P. Vert and Y. Yamanishi. Supervised graph inference. *Advances in Neural Information Processing Systems*, 17:1433–1440, 2004.

D. Vitkup, P. Kharchenko, and A. Wagner. Influence of metabolic network structure and function on enzyme evolution. *Genome Biol*, 2006.

J. Weston, A. Elisseeff, D. Zhou, C.S. Leslie, and W.S. Noble. Protein ranking: From local to global structure in the protein similarity network. *Proceedings of the National Academy of Sciences*, 101 (17):6559–6563, 2004.

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *NIPS*, 2003.

Y. Yamanishi, J. P. Vert, and M. Kanehisa. Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, 2004.

Y. Yamanishi, J.P. Vert, and M. Kanehisa. Supervised enzyme network inference from the integration of genomic data and chemical information. *Bioinformatics*, 21(Suppl 1):i468–i477, 2005.

A. Zien, U. Brefeld, and T. Scheffer. Transductive support vector machines for structured variables. *Proceedings of the 24th international conference on Machine learning*, pages 1183–1190, 2007.