Structured Network Learning

Stuart Andrews
Computer Science Dept.
Columbia University
New York, NY 10027

Tony Jebara
Computer Science Dept.
Columbia University
New York, NY 10027

Abstract

Man-made or naturally-formed networks typically exhibit a high degree of structural regularity. In this paper, we introduce the problem of structured network prediction: given a set of n entities and a desired distribution for connectivity, return a likely set of edges connecting the entities together in a network having the specified degree distribution. Prediction is useful for initializing a network, augmenting an existing network, and for filtering existing networks, when the structure of the network is known. In order to capture the inter-dependencies amongst pairwise predictions to learn parameters of our model, we build upon recent structured output models. Novel in our approach is the use of partially labeled training examples, and a network structure sensitive loss function. We present encouraging results of the model predicting equivalence graphs and links in a social network.

1 Introduction

Man-made or naturally-formed networks typically exhibit a high degree of structural regularity. A social network represents friendship or professional affilation and reflects the fabric of society. Complex biological systems, such as the regulatory networks, metabolic networks, and signal transduction pathways are composed of frequently occuring motifs [Kashtan et al., 2004]. One can even regard an equivalence relation over n entities as a structured graph that is a union of vertex disjoint complete subgraphs.

In this paper, we introduce the problem of *structured network prediction*: given a set of n entities and a desired distribution for connectivity, return a likely set of edges connecting the entities together in a network having the specified degree distribution. If a network adheres to the given distribution, we say that it is *structured*. Prediction is useful for initializing a network, augmenting an existing network, and for filtering existing networks, when the structure of the network is known. While there is existing research dealing with the empirical analysis of existing networks [Barabasi and Oltvai, 2004] the artificial simulation of network formation [Albert and Barabasi, 2002] and the extraction of network structures at various scales [Kashtan et al., 2004] there has been little attention paid to the prediction of structured networks in a data driven manner [Culotta et al., 2004, Rabbat et al., 2006].

The main difficulty of this problem is the management of inter-dependencies of edge predictions. The presence of one edge (j, k) in a network is dependent on the presence and/or absence of the other edges $(j, \cdot)$ and $(\cdot, k)$ incident to the j-th and k-th vertices. Researchers have previously extended example-based learning methods [Kondor and Jebara, 2006, Shalev-Shwartz et al., 2004] to pairwise data. However, these methods assume that all pairs of entities are independent and identically distributed (i.i.d.). Due to the i.i.d. assumption, these approaches are not sensitive to the overall network structure generated by independent pairwise predictions. Another difficulty with this approach is the necessity for managing the bias of the classifier which is affected by the strong asymmetry of the edge versus non-edge classes. Since many networks have far fewer edges than the maximum n^2 edges, the tendency of i.i.d. methods is to classify all pairs as non-edges.

We adopt a uniform connectivity hypothesis derived from the observation that many real-world networks arise in the presense of resource constraints or costs distributed among edges [Pennock et al., 2002]. For instance, on a social network such as the type maintained at a university, members typically interact with a small number of friends compared to the overall size of the network. Likewise, professors can advise only a limited number of graduate students. Our predictive model thus ensures that we obtain a network with the expected structure.

In order to capture the inter-dependencies amongst pairwise predictions to learn parameters of our model, we consider the application of structured output models [Tsochantaridis et al., 2004, Taskar et al., 2003, 2005a] to the network prediction problem. To facilitate learning over inter-dependent output spaces, structured output models generalize the empirical risk minimization framework following the approach of [Crammer and Singer, 2001] by defining the margin as the functional distance between the true output and the highest-ranked alternative. By focussing effort on the margin, these formulations provide a tractable alternative to conditional likelihood modeling which would otherwise depends on the computation of a partition function [Lafferty et al., 2001]. 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., 2004, 2005b].

Our method for structured learning is novel in that we consider partially labeled networks as inputs to the learning algorithm, to handle applications such as initialization, augmentation and filtering in a transductive fashion. Partial labeling refers to the connectivity; only a subset of the potential n^2 edges are observed. In order to apply the structured output model of [Taskar et al., 2005a] to our problem, we first show how to parameterize a combinatorial family of graphs using a linear model. In particular, we introduce a novel loss function for evaluating predictions in this transductive setting, and derive an upper bound that can be minimized using a variant of the dual extragradient algorithm.

2 Structured networks

In general, the most common type of structural characterization of large networks is made in terms of their degree distributions. This is the distribution over the *degrees* of the vertices, where the degree $\delta(v)$ of a vertex v is the number of edges incident to that vertex. Alternatively, networks can often be classified into well-defined combinatorial families of graphs, such as chains, trees, forests, generalized matchings or unions of disjoint complete subgraphs.

In Figure 1, we compare the degree distributions of three structured networks, going from the least to most constrained. The first example is the degree distribution from a protein-protein interaction network. The degree distribution for this network exhibits scale-free behaviour, appearing as a line with negative slope. A second example is a social network comprised of links that members create with friends. The degree distribution in this case is nearly uniform on a bounded interval, and zero elsewhere. This pattern is a characteristic of social networks with limited enrollment, for example, within an academic institution or private company. A final example is an equivalence graph for a collection of $m = 300$ face images of $m = 30$ individuals, each observed under 10 different viewing conditions. A complete graph connects the images of each person, yielding a degree distribution that is a Dirac delta function at the value 9.

In order to express the degree distribution in a convenient form, we need the following definitions. Consider a set of vertices $\mathcal{V} = \{1, \ldots, n\}$ and the complete set of edges connecting pairs $\mathcal{E} = \{(j,k) \,|1 \leq j < k \leq n\}$. Let $e(j,k)$ be the index of edge (j,k) in an enumeration of $\mathcal{E}$. In our current research, we deal with graphs that have undirected edges, although the techniques can be adapted to the directed case. A complete vertex-edge incidence matrix on n vertices, denoted $\mathbf{A}_n$, has elements $a_{i,e(j,k)}$, $1 \leq i,j,k \leq n$, $j < k$. The i-th row indicates which edges are incident to vertex i in a complete graph.[1] Thus, we have

$$\mathbf{A}_5 = \left[\begin{array}{cccc|ccc|cc|c} 1 & 1 & 1 & 1 & & & & & & \\ 1 & & & & 1 & 1 & 1 & & & \\ & 1 & & & 1 & & & 1 & 1 & \\ & & 1 & & & 1 & & 1 & & 1 \\ & & & 1 & & & 1 & & 1 & 1 \end{array}\right] \tag{1}$$

[1] Since edges are undirected, we have $a_{j,e(j,k)} = a_{k,e(j,k)}$.

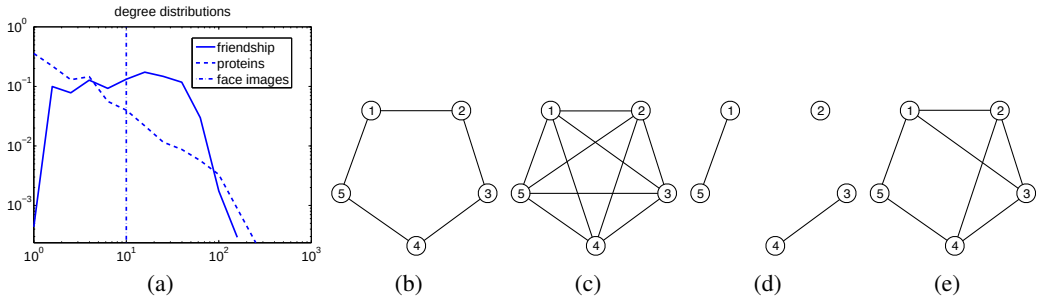

Figure 1: (a) A log-log plot showing degree distributions for a social network (friendship), a protein-protein interaction network (protein), and an equivalence graph (face images). (b) B-matchings with $b = 2$, and (c) $b = 4$. (d) When $b = 1$ the best we can do is a near-perfect 1-matching, where vertex 2 has $\delta(2) = 0 < 1$, and similarly for (e), when $b = 3$, we obtain the near-perfect 3-matching $\delta(5) = 2 < 3$.

which corresponds to the incidence matrix for the graph in Figure 1(b).

Now, let a subset of edges $\mathbf{G}_\mathbf{y} \subseteq \mathcal{E}$ be parameterized using a 0-1 vector $\mathbf{y} = \left(y_{e(j,k)}\right)$, $1 \leq j, k \leq n$, where an edge (j, k) is in $\mathbf{G}_\mathbf{y}$ if $y_{e(j,k)} = 1$.[2] The following relation holds $\delta(v_i) = (\mathbf{A}_n\mathbf{y})_i$ which says that the degrees of vertices in a graph with edges $\mathbf{G}_\mathbf{y}$ are given by the components of the matrix product $\mathbf{A}_n\mathbf{y}$.

At this time, we will focus on a combinatorial family of graphs called b-matchings, which have specific degree distributions, are flexible, and lead to efficient algorithms. The methods of this paper can be extended directly to alternative combinatorial families of graphs.

For a given collection of vertices and fixed b, we define a *b-matching* to be any subset of undirected edges connecting vertices such that each vertex v has exactly b neighbours ($\delta(v) = b$). The set of b-matchings is a combinatorial family of networks having uniform connectivity over the vertices; the resulting network has a degree distribution that is a Dirac delta function on the value b. Strictly speaking, to align ourselves with graph theoretic and network flow terminology, these are *k-factors* [Schrijver, 2003].[3] When there does not exist a b-matching, we also consider a *near-perfect* b-matching in which one vertex has $b - 1$ neighbours. Figure 1 depicts a graph having a 2 and a 4-matching, and a near-perfect 1 and 3-matching. In the future, we plan to study more general types of b-matchings. We can characterize b-matchings as 0-1 vectors $\mathbf{y}$ that satisfy the degree constraints

$$\mathbf{A}_n\mathbf{y} = b\mathbf{1}\,. \tag{2}$$

2.1 Structured network prediction

We propose to use *maximum-weight b-matching* as a predictive model. Given a set of vertices and their attributes, the model assigns weights to edges and returns the b-matching that has the most weight as the prediction. Like the edge indicator vector $\mathbf{y}$, the edge weights are represented by a vector $\mathbf{s} = \left(s_{e(j,k)}\right)$, however $s_{e(j,k)} \in \mathbb{R}_+$ can be any positive real-value. The *weight* of $\mathbf{G}_\mathbf{y} \subseteq \mathcal{E}$ is defined as the sum of the weights of its edges and equals $\mathbf{s}^T\mathbf{y}$. Letting $\mathbf{z}$ be a real-valued vector variable whose components correspond to those of $\mathbf{y}$, the maximum-weight b-matching problem can be formulated as an LP

$$\max_{\mathbf{z} \in \mathbf{P}^n} \mathbf{s}^T\mathbf{z}\,, \tag{3}$$

in terms of the *b-matching polytope* $\mathbf{P}^n$ which is convex-hull of b-matchings [Schrijver, 2003]. A description of this polytope is included in the Appendix. Fortunately, even though the $\mathbf{P}^n$ is

[2] We often write $y_{j,k}$ in place of $y_{e(j,k)}$, with the understanding that this still refers to a component of the *vector* $\mathbf{y}$.

[3] The general definition of a b-matching allows vertices to have individually specified b_i, and any number of incident edges *up to* b_i. Moreover, the definition allows edges to be used *multiple* times. A b-matching is capacitated with capacity $c \geq 0$ if edges are used at most c times, and *simple* if $c = 1$. It is perfect if each vertex has *exactly* b_i edges for each vertex i. A *k-factor* is a simple and perfect b-matching where $b_i = k$, $\forall i$.

described by an exponential number of linear constraints, there exist polynomial-time combinatorial algorithms for finding the maximum. Gabow [Gabow, 1983] presents an efficient technique for reducing b-matchings to 1-matchings, for which Edmund's algorithm can be applied. Fremuth-Paeger et al. [Fremuth-Paeger and Jungnickel, 1998] provide a balanced network flow model and an implementation that solves a variety of related matching and flow problems. Both methods are specializations of the primal-dual methodology for solving linear programs with exponentially many constraints [Papadimitriou and Steiglitz, 1982].

There are special cases, however, when the representation of the convex hull is much simpler. For example, if b is even and if we consider *uncapacitated* b-matchings where an edge can be chosen multiple times, the resulting LP is $\max_{\mathbf{z}\geq 0,\, \mathbf{A}_n\mathbf{z}=b\mathbf{1}} \mathbf{s}^T\mathbf{z}$, which can be solved efficiently. Moreover, we may choose to relax the b-matching criteria by simply omitting the constraints in Equation (14).

3 Learning the weights

The key question remains: how does one assign weights to the edges? In a typical b-matching application, the weights $\mathbf{s} = (s_{j,k})$ are engineered in advance from the attributes $\mathbf{x}_j, \mathbf{x}_k \in \mathbb{X}$ associated with the vertices. This approach requires domain knowledge, and becomes overly complicated when there are a large number of attributes. Our main contribution is a method that learns a function for assigning weights in a given domain, based on a training data set $\mathcal{D} = \{(\mathbf{X}_i, \mathbf{y}_i) \,|\mathbf{X}_i \in \mathbb{X}^{n_i},\, \mathbf{y}_i \in \mathbf{P}^{n_i}\}$, where $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{X}^n$ is the set of attribute vectors $\mathbf{x}_k \in \mathbb{X} \subseteq \mathbb{R}^d$ for $\mathcal{V}$. The training data may consist of a single network, or several independent but related networks defined over disjoint sets of vertices. The attributes of vertices are specified in $\mathbf{X}_i$, and the partially observed structure in $\mathbf{y}_i$.[4]

In particular, we consider a semi-supervised setting where we have incompete knowledge of the structure for the network(s) in the training data. We assume that the attributes of each vertex are known, but only a subset of the network edges have been observed. For a given example $(\mathbf{X}, \mathbf{y}) \in \mathcal{D}$ the observed network structure consists of a set of known edges $\{(j,k) \in \mathcal{E} | y_{j,k} = 1\}$, known non-edges $\{(j,k) \in \mathcal{E} | y_{j,k} = 0\}$, and unknown edges $\{(j,k) \in \mathcal{E} | y_{j,k} = ?\}$. The first objective in assigning the weights is to ensure, for each instance, that the structure predicted by our model (a maximum-weight b-matching over the vertices specified by $\mathbf{X}$) matches the subset of observed edges specified in $\mathbf{y}$. The second objective is to ensure that the predictive model is able to generalize by obtaining the correct structure over the unobserved edges.

We consider a parametric family $s_{j,k} = (t - d_Q(\mathbf{x}_j, \mathbf{x}_k))$, defined in terms of a pre-specified constant threshold $t \in \mathbb{R}$, and an adaptable Mahalanobis distance metric $d_Q(\mathbf{x}_j, \mathbf{x}_k) = (\mathbf{x}_j - \mathbf{x}_k)^T Q (\mathbf{x}_j - \mathbf{x}_k)$, defined in terms of a positive semidefinite (p.s.d.) matrix Q. For high dimensional feature spaces, we also consider p.s.d. Q that are diagonal. We adopted the Mahalanobis distance that has previously been used for metric learning [Xing et al., 2003, Goldberger et al., 2004, Shalev-Shwartz et al., 2004] so that we have a well-known baseline for comparison. Additionally, the weights obtained using the Mahalanobis distance are easy to interpret. Since Q is symmetric and positive semidefinite, we know that $Q = P^T P$ for some P. Therefore, the weight $s_{j,k}$ for an edge (j,k) depends on $d_Q(\mathbf{x}_j, \mathbf{x}_k) = \|P\mathbf{x}_j - P\mathbf{x}_k\|_2^2$ which is the Euclidean distance between the points in a suitably scaled and rotated space. From this perspective, we are finding the best transformation P of the feature space to facilitate b-matching.

3.1 I.I.D. learning

One approach to learning the weight function is to treat the edges from all input b-matchings as independent and identically distributed (i.i.d.). In other words, we assume we have observed a training data set $\mathcal{D}_{\text{i.i.d.}} = \left\{((\mathbf{x}_j, \mathbf{x}_k), y_{j,k}) \,|\, (\mathbf{x}_j, \mathbf{x}_k) \in \mathbb{X}^2,\, y_{j,k} \in \{0,1\}\right\}$. We can then learn the Mahalanobis distance and threshold using max-margin learning [Kondor and Jebara, 2006, Shalev-Shwartz et al., 2004] to robustly predict the presence of an edge between vertex pairs.

[4] The approach can also be applied when attributes are instead associated with edges.

3.2 Inter-dependent outputs learning

In order to apply the structured output model of [Taskar et al., 2005a] to our problem, we first show that the parametric family of edge weights can be expressed as a linear model. We then proceed to develop this model while accounting for the specifics of semi-supervised structured network learning. In particular, we introduce a novel loss function for evaluating predictions in the transductive setting, and derive an upper bound that can be minimized. Like [Taskar et al., 2005a], we apply the dual extragradient algorithm of [Nesterov, 2003], however some modification is required.

Let $\mathbf{f}(\mathbf{x}_j, \mathbf{x}_k) = \mathrm{vec}\left((\mathbf{x}_j - \mathbf{x}_k)(\mathbf{x}_j - \mathbf{x}_k)^T\right)$ denote the vector of elements from the outer product matrix of $\mathbf{x}_j - \mathbf{x}_k$ with itself. We call $\mathbf{f}(\mathbf{x}_j, \mathbf{x}_k)$ the *features* of the edge (j, k). Also, define the matrix $\mathbf{F}$ having $(-1)\mathbf{f}(\mathbf{x}_j, \mathbf{x}_k)$ as columns. Finally, letting $\mathbf{w} = \mathrm{vec}(Q)$ denote the vector of elements from the p.s.d. Q, we have the following equivalent expression for the weight of a b-matching $\mathbf{y}$

$$\mathbf{s}^T\mathbf{y} = \sum_{j<k} y_{j,k}\left(t - \mathbf{w}^T\mathbf{f}(\mathbf{x}_j, \mathbf{x}_k)\right) \quad (4)$$

$$= \frac{1}{2}nbt + \mathbf{w}^T\mathbf{F}\mathbf{y}\,. \quad (5)$$

In the sequel, we will omit the first term, since it does not depend on $\mathbf{w}$ or $\mathbf{y}$. Note that due to symmetry, one needs only the upper triangular elements of matrix Q, and the outer product matrices $(\mathbf{x}_j - \mathbf{x}_k)(\mathbf{x}_j - \mathbf{x}_k)^T$ comprising edge features. Therefore, by defining the input-output feature vector $\mathbf{f}(\mathbf{X}, \mathbf{y}) = \mathbf{F}\mathbf{y}$, we can write our linear predictive model as follows

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}' \in \mathbf{P}^n} \mathbf{w}^T\mathbf{f}(\mathbf{X}, \mathbf{y}')\,. \quad (6)$$

We derive the max-margin learning problem following [Taskar et al., 2005a]. The key insight that enables max-margin learning for multiclass, and in particular, structured outputs, is that one can define the margin in terms of the functional distance $\mathbf{w}^T\Delta\mathbf{f}(\mathbf{y}') = \mathbf{w}^T(\mathbf{f}(\mathbf{X}, \mathbf{y}) - \mathbf{f}(\mathbf{X}, \mathbf{y}'))$ between the true output and the highest ranked alternative structure. However, since the functional distance scales with the number of prediction errors on individual labels $y'_{j,k}$, a modified definition called the *inverse-loss weighted margin* is used instead

$$\gamma(\mathbf{X}, \mathbf{y}, \mathbf{y}') = \Delta(\mathbf{y}')^{-1}\mathbf{w}^T\Delta\mathbf{f}(\mathbf{y}')\,, \quad (7)$$

where $\Delta(\mathbf{y}') = \Delta(\mathbf{y}, \mathbf{y}')$ quantifies the cost of predicting $\mathbf{y}'$ when the true network is $\mathbf{y}$.

We modify the standard Hamming loss for semi-supervised network structure learning in two ways. First, since we have only observed the true network structure $y_{j,k}$ for a subset of edges, we only evaluate the loss on these elements. Secondly, for structured network prediction, the number of individual label errors $\Delta(\mathbf{y}')$ can grow to $\mathcal{O}(n^2)$. For this reason, we define the loss to be $\Delta(\mathbf{y}') = \min\{\mathbf{L}, h(\mathbf{y}, \mathbf{y}')\}$ where $h(\mathbf{y}, \mathbf{y}') = \mathbf{y}^T\mathbf{1} + (\mathbf{1} - 2\mathbf{y})^T\mathbf{y}'$ is the Hamming distance between $\mathbf{y}$ and $\mathbf{y}'$. Figure 2 displays the hinge loss for increasing values $\Delta(\mathbf{y}')$ and compares the Hinge losses used by $\mathrm{SVM}_1^{\Delta s}$ of [Tsochantaridis et al., 2004] and [Taskar et al., 2005a].

After a change of variables and the introduction of slack variables, the primal max-margin optimization problem is

$$\min_{\mathbf{w} \in \mathcal{W}, \xi \geq 0} \quad \frac{1}{2C}\|\mathbf{w}\|^2 + \sum_i \xi_i \quad (8)$$

$$\text{s.t.} \quad \xi_i \geq \max_{\mathbf{y}' \in \mathbf{P}^{n_i}} \Delta_i(\mathbf{y}') - \mathbf{w}^T\Delta\mathbf{f}_i(\mathbf{y}')\,, \forall i \quad (9)$$

where $\mathcal{W} = \{\mathbf{w} = \mathrm{vec}(Q) \,|\, Q \succeq 0\}$. In [Taskar et al., 2005a], solutions to this problem are related[5] to solutions of the following, where $\mathcal{W}^\gamma = \{\mathbf{w} \in \mathcal{W} | \, \|\mathbf{w}\|_2 \leq \gamma\}$ for an appropriate choice of γ

$$\min_{\mathbf{w} \in \mathcal{W}^\gamma} \quad \sum_i \max_{\mathbf{y}' \in \mathbf{P}^{n_i}} \Delta_i(\mathbf{y}') - \mathbf{w}^T\Delta\mathbf{f}_i(\mathbf{y}')\,. \quad (10)$$

[5] For the 2-norm, for any solution to the former problem, there is a γ that yields the same solution on the latter.

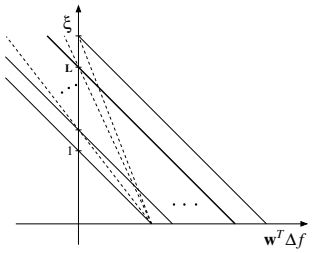

Figure 2: The effect of the Hamming loss on the Hinge loss functions used by (1) $\text{SVM}_1^{\Delta s}$ of [Tsochantaridis et al., 2004] (dashed lines, decreasing in slope), (2) the dual extragradient technique of [Taskar et al., 2005a] (solid lines, shifting right), and (3) our method (solid lines, shifting right until the bold line intersecting the vertical axis at $\mathbf{L}$).

Finally, by defining the joint output space $\mathbf{Z} = \mathbf{P}^{n_1} \times \ldots \times \mathbf{P}^{n_m}$, and noting that the Hamming distance can be expressed as $h(\mathbf{y}, \mathbf{y}') = a + \mathbf{c}^T\mathbf{y}'$ for appropriate $a \in \mathbb{R}$ and $\mathbf{c} \in \mathbb{R}^n$, we can write the following equivalent learning problem

$$\min_{\mathbf{w}\in\mathcal{W}^\gamma} \max_{\mathbf{z}\in\mathbf{Z}} \quad \sum_i \mathbf{w}^T\mathbf{F}_i\mathbf{z}_i + \min\left\{\mathbf{c}^T\mathbf{z}_i, \mathbf{L}\right\} - \mathbf{w}^T\mathbf{F}_i\mathbf{y}_i\,. \tag{11}$$

Due to the clamping of the loss, the gradient of this objective is not Lipschitz continuous.[6] Fortunately, this does not preclude the use of the dual extragradient algorithm, because the gradient operator still has bounded variation [Nesterov, 2003].

3.3 Implementation

We implemented the memory efficient version of the dual extragradient outlined in Figure 5 of [Taskar et al., 2005a] using our definitions from above; we refer the reader to this paper for details. We chose $\dot{\mathbf{w}} = \text{vec}(\mathbb{I})$ equal to the vector representation of the identity matrix, and $\dot{\mathbf{z}}_i = \frac{b}{n}\text{vec}(\mathbf{1}^{n\times n})$ equal to our expectation of where the "average" b-matching solution lies. The gradient step-size for the algorithm is weighted by a factor of $\frac{1}{\sqrt{t}}$ where t is the iteration number, as described in Algorithm (3.4) in [Nesterov, 2003]. As shown in [Taskar et al., 2005a], one can approximate the solution of Equation (10) for a range of γ from a single run of the dual extragradient algorithm by simply removing the constraint $\|\mathbf{w}\|_2 \leq \gamma$.

The basic update in the dual extragradient algorithm involves a gradient step in variables $\mathbf{w}$ and $\mathbf{z}_i$ followed by a Euclidean projection onto their respective feasible sets. Projecting onto $\mathcal{W}$ is performed by dropping the negative eigenvalues from the eigendecomposition of Q. On the other hand, we have not found an efficient method to project $\mathbf{z}_i$ onto the $\mathbf{P}^{n_i}$. Since the solution may lie on a face of the polytope, b-matching algorithms do not apply.

One solution is to relax the problem, and projecting onto a larger convex hull, for example the hull formed by points that satisfy Equations (12-13) in the Appendix. Another method that we emply is to decompose each network into small overlapping subnetworks and perform polynomial-time 0-1 b-matching on these. An advantage of this approach is that it is efficient and can be scaled-up to large networks. Empirically, this method gives promising results. We are working on bounds to justify this approximation scheme.

4 Related work

In [Barabasi and Bonabeau, 2003, Albert and Barabasi, 2002], network structure is characterized by measuring the distribution of vertex connectivity in the network. They document scale-free, or power-law, behaviour in the degree distribution. Several models have been proposed (rich-get-richer) that simulate the formation of networks with these properties, however edge formation is driven by random processes that do not take into account attributes at the nodes. In [Kleinberg,

[6] We take one-sided limits of the component derivatives when the gradient is not defined.

1998, Kleinberg et al., 1999], hubs and authorities are analyzed. Recent efforts to detect graph structures do so by analyzing overlapping community structures, subgraph concentrations, and frequently occuring subgraphs [Derenyi et al., 2005, Palla et al., 1969, Kashtan et al., 2004, Koyutürk et al., 2004]. These methods help to understand an overall network structure, but do not perform prediction of network structures.

Recently, [Culotta et al., 2004] presented an end-to-end system that constructs an individualized social network based on a user's email inbox. The system identifies people in emails, searches for their Web-presence and links between people together. Another recently published work [Rabbat et al., 2006] analyzes weak co-occurance data to predict a likely network.

5 Experiments

We used a collection of face images from the Olivetti Research Laboratory[7] to validate our approach. The collection contains $m = 300$ face images of $m = 30$ individuals, each observed under 10 different viewing conditions that included pose, eyewear, expression and illumination variations. The images are 92 x 112 gray value. We centered the data and applied PCA to reduce the featue dimensionality.

We first partitioned the 300 vertices into subsets: 200 training vertices, 50 test vertices and 50 validation vertices. The edges between testing points, and betwen test points and training or validation points, are not used by the algorithm. The edges between validation points, and between validation points and training points, are used by the algorithm for selecting parameters. We then sample 50 overlapping subnetworks of size 50, so that each subnetwork contained points from the training, testing and validation sets.

We evaluate this method and two variants that do not use b-matching. The first variant uses a simpler predictive model that selects the top $b\frac{n}{2}$ edges, without constraining the degree distribution. The second variant uses k-nn as a predictive model by connecting each vertex to its b nearest-neighbours. Figure 3 shows the results. Also included in the plots are the results obtained by the Euclidean distance metric. Both validation and testing set results are plotted. In these plots, "raw dist." refers to the Euclidean distance metric, while "learn dist." refers to the metric learned by the modified dual extragradient algorithm for a specific variant. We have also compared receiver-operator curves for the respective methods. These curves are obtained by adding $\epsilon d_Q(j,k)$ to the integral values of the structural predictions in $y_{j,k}$. The threshold points are marked on each curve. Notice that while the edge prediction accuracy is nearly constant for all variants, the variants that ensure the degree distribution is uniform (k-nn / b-matching) obtain higher recall rates.

6 Conclusions

We have outlined a novel structured network prediction problem, developed an learning algorithm that attempts to solve it, and shown encouraging results on a controlled example involving an equivalence graph over images. It is possible to learn a modified distance 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.

References

R. Albert and A. L. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 2002.

A. L. Barabasi and E. Bonabeau. Scale-free networks, 2003.

A. L. Barabasi and Z. Oltvai. Network biology: Understanding the cell's functional organization. *Nature Genetics*, 5, 2004.

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

[7] http://www.uk.research.att.com/facedatabase.html

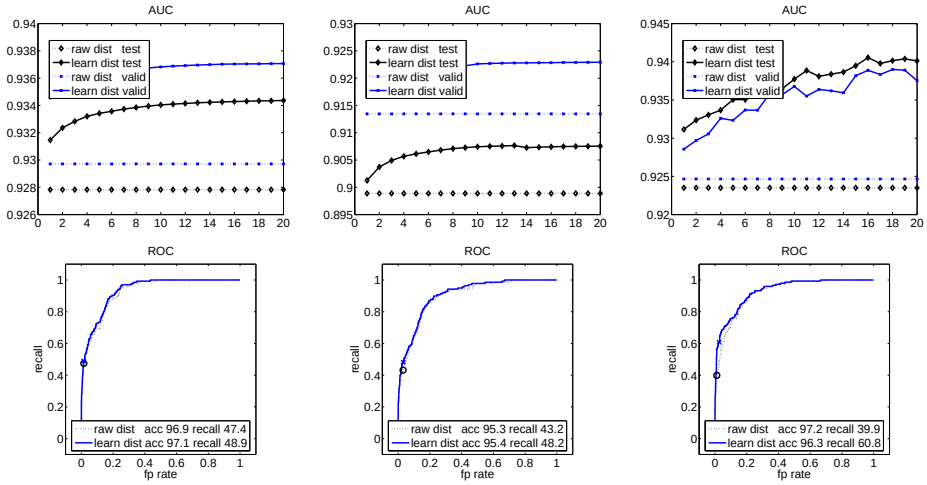

Figure 3: Predictive model with (left) no constraints on edge predictions, (middle) using k-nn edge predictions, and (right) using b-match. (Top) AUC versus iteration, (bottom) ROC curves.

A. Culotta, R. Bekkerman, and A. McCallum. Extracting social networks and contact information from email and the web. In *AAAI*, Aron Culotta and Ron Bekkerman and Andrew McCallum 2004.

I. Derenyi, G. Palla, and T. Vicsek. Clique percolation in random networks. *Physical Review Letters*, 2005.

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

H. N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *ACM Theory of Computing*, 1983.

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

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

J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *ACM*, 1998.

J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins. The web as a graph: Measurements, models, and methods. In *COCOON*, 1999.

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

M. Koyutürk, A. Grama, and W. Szpankowski. An efficient algorithm for detecting frequent subgraphs in biological networks. *BIOINFORMATICS*, 2004.

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

Y. Nesterov. Dual extrapolation and its applications for solving variational inequalities and related problems. Technical report, Catholic University of Louvain, 2003.

G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 1969.

C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Prentice-Hall, Englewood Cliffs, N.J., 1982.

D. M. Pennock, G. W. Flake, S. Lawrence, E. J. Glover, and C. L. Giles. Winners don't take all: Characterizing the competition for links on the web. *PNAS*, 2002.

M. Rabbat, M. Figueiredo, and R. Nowak. Network inference from co-occurrences. Technical report, University of Wisconsin, 2006.

A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer-Verlag, 2003.

Shai Shalev-Shwartz, Yoram Singer, and Andrew Y. Ng and. 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*, 2004.

B. Taskar, S. Lacoste-Julien, and M. I. Jordan. Structured prediction, dual extragradient and bregman projections. Technical report, U. C. Berkeley, 2005a.

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

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning*, 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.

A The b-matching polytope

A common misconception is that the b-matching polytope is obtained simply by relaxing the integrality constraints on $\mathbf{y}$ to the range $0 \leq \mathbf{y} \leq 1$, while still satisfying the degree constraints in Equation (2). However, this relaxation is a superset of convex-hull of b-matchings, and in particular contains extreme points that are not integral. To describe the b-matching polytope, we denote for each subset $U \subseteq \mathcal{V}$ the subset of edges $\Delta(U) = \{(j,k) \,|j \in U,\, k \notin U\}$. Also, let the indicator vector $\mathbb{1}_F = (\mathbb{1}[(j,k) \in F])\,,\ 1 \leq j < k \leq n$, and note that $\mathbf{y} = \mathbb{1}_{\mathbf{G_y}}$. The b-matching polytope $\mathbf{P}^n$ may then be characterized by the following linear equations

$$0 \leq z_j \leq 1,\ \forall j \tag{12}$$

$$\mathbf{A}_n \mathbf{z} = b\mathbf{1} \tag{13}$$

$$\mathbf{z}^T \mathbb{1}_{\Delta(U)\setminus F} - \mathbf{z}^T \mathbb{1}_F \geq 1 - |F|\,,$$
$$\forall U \subseteq \mathcal{V}, F \subseteq \Delta(U)\ ,\ \text{and}\ b\,|U| + |F|\ \text{odd}. \tag{14}$$

Equation (12) relaxes the integrality of variable $\mathbf{y}$. Following that, Equation (13) enforces that the degree of each vertex is b. The constraints in Equation (14) are required to exclude from the polytope all non-integral extreme points $\mathbf{z}$ satisfying Equations (12) and (13). Clearly, it is not possible to work with this representation of the convex hull directly, since there are an exponential number of constraints in Equation (14).