חיזוי קשרים ברשתות חברתיות בעזרת למידה מובנית

ניר רוזנפלד

הוגש כמילוי חלקי של החובות לתואר מוסמך במדעים

מנחים: ד"ר אמיר גלוברזון
פרופ' יעקב גולדנברג

אייר תשע"ג

בית הספר להנדסה ומדעי המחשב
על שם רחל וסלים בנין
האוניברסיטה העברית בירושלים
ישראל

תקציר

בשנים האחרונות, רשתות חברתיות זכו להתיחסות מחקרית מקשת רחבה של תחומים. כאובייקטים מורכבים ודינמיים, רשתות חברתיות הן כר פורה לאתגרים רבים בתחום המערכות הלומדות. אחת המטלות הבולטות בתחום היא *חיזוי קשרים* – בהינתן רשת אשר רק חלק מהקשרים בה נצפים, יש לחזות את הקשרים החסרים או את העתידים להתוסף. לרשתות חברתיות תכונות מבניות עשירות המשרות תלות על הקשרים הנחזים. תכונות מבניות אלו יכולות, כידע מוקדם על הבעיה, יכולות לסייע בבניית חוזים בעלי שגיאה נמוכה. עם זאת, רב העבודות המוכרות כיום העוסקות בחיזוי קשרים אינן מתחשבות בתכונות מבניות ובתלות הנוצרת בין הקשרים, או שאינן מאכוונות לבניית חוזים. בנוסף, למרות שמדד ה-AUC הוא השכיח בתחום, מעטות הן השיטות הבנויות במיוחד עבורו.

בעבודה זו אנו ניגשים למטלת חיזוי הקשרים דרך המסגרת של *למידה מפוקחת מובנית*. על ידי שימוש בכלים כגון dual decomposition ותכנון לינארי, אנו מראים כיצד לשלב פקטורים חדשניים מסדר גבוה המסוגלים לבטא את המבניות המורכבת הקיימת ברשתות חברתיות. בנוסף, אנו מציעים שיטה חדשה המאפשרת ללמוד מודלים מורכבים ישירות עבור מדד ה-AUC. אנו מציגים תוצאות ניסוייות עבור רשתות חברתיות סינטטיות ואמיתיות.

Structured Learning for Link Prediction

Nir Rosenfeld

Submitted in partial fulfilment of the requirements
of the degree of Master of Science

Under the supervision of

Dr. Amir Globerson

and

Prof. Jacob Goldenberg

April 2013

Rachel and Selim Benin
School of Computer Science and Engineering
The Hebrew University of Jerusalem
Israel

Acknowledgments

I would like to thank my advisors Amir Globerson and Jacob Goldenberg. This thesis would bot have been made possible without Amir's open mindedness, insightful guidance, and patient explanations, nor without Jacob's contagious enthusiasm and granted feeling of competence. I feel privileged to have worked with both, separately and together.
I would also like to thank Ofer Meshi for teaching me the insides and outsides of the upsides and downsides of research, and my machine learning lab partners for all the support, fun, and noise they constantly produce.
Finally, I thank my girlfriend Nitzan for her never-ending moral support and genuine ideas, and my family for their incessant encouragement and sincere interest.

Abstract

In recent years, social networks have become the focus of research throughout multiple domains. As complex dynamic objects, they give rise to many interesting challenges for machine learning. One of the most prominent tasks is that of *link prediction*: given a network with partially observed edges, predict the set of missing or future edges. Social networks contain rich structural regularities which induce dependencies between predicted edges. These structural regularities could serve as prior knowledge for constructing low-error predictors. However, most current link prediction methods either do not take structure and dependencies into account, or do not directly train a predictor. Moreover, although the AUC is the most common performance measure, few methods optimize directly over it.

In this work, we approach the task of link prediction through the framework of supervised *structured learning*. By using relaxation schemes such as dual decomposition and linear programming, we show how to incorporate novel high-order factors which express social network structural regularities into structured models. Furthermore, we propose a novel method for optimizing structured models directly over the AUC loss. We include experimental results on synthetic and real world social networks.

Contents

Chapter 1

Introduction

1. Overview

Humans, as a social species, interact and form both passing and persistent connections between individuals. A group of interacting people form a *social network*, which can be modeled as a graph $G = (V, E)$ whose nodes represent the individuals and whose edges represent their interactions. Many interesting questions and problems arise in the study of social networks. One of the most fundamental problems is that of *link prediction*. Given a network G with observed edges E, the task it to predict the missing edges E'. In *temporal* link prediction, the input is a time series of networks $G_1, \ldots, G_T$, where the goal is to predict the future network G_{T+1}. We approach this problem from a machine learning perspective, where we try to learn a linear model over a sample set of networks representing the linkage history of a fixed set of nodes.

Our approach is based on the insight that social networks have *structure*, which may be expressed by inter-edge dependencies. In other words, the existence of edge (i, j) at time $t+1$ is dependent not only on edges at time t, but also on predicted edges at time $t + 1$, where this dependence follows some structure. A learning framework which is suiting to such structural dependence is *structured learning*. In structured learning, the score of a prediction is a function of an assignment to *all* variables (as opposed to over a single variable, as in the case of non-structured learning). In our work, the score of an assignment is given by its value in a Markov Random Field (MRF), whose factors can incorporate the rich structural regularities found in social networks.

While in non-structured learning the prediction rule is usually straightforward, in structured learning it comprises of solving non-trivial combinatorial optimization problems, which are NP-hard in the general case. By applying relaxation techniques such as dual-decomposition and linear programming to our learning objective, we reduce the large problem into smaller combinatorial optimization problems corresponding to the factors in our model. These factors, defined appropriately, can encode prior

knowledge we hold regarding interaction between edges.

In our work, we aim to make use of theoretical results regarding social networks from the social sciences (such as a scale-free distribution of node degrees, 'small world' effects, community structure etc.) as prior knowledge in the link prediction task. We do this by proposing tractable factors for MRFs, which encode this domain-specific structural prior knowledge.
However, before integrating these factors into the task, it is necessary to adjust the loss function in the learning objective to suit the nature of the learning problem. Specifically, since the common performance measure in link prediction is the *AUC score*, it is appealing to incorporate a surrogate 'AUC-loss' in the learning objective. In this work we suggest a method which allows direct optimization over the AUC loss for structured models.

The AUC, defined to be the area under the ROC curve (a function of true and false positive rates of a predictor), is a common performance measure for dlassifiers and ranking algorithms. As opposed to the hamming loss, it is a function of all variables in the model, and therefore adds structure to the learning task in itself. Joachims (2005) provides an elegant method for learning a linear model which factorizes over the variables. However, it is not clear how his model can be extended to structured models.
In our work we provide a method for learning *structured* models over the AUC loss, so that high order factors can easily be integrated into the model. This allows for a much richer class of models to be handled. We apply our method to the task of link prediction, and show that adding structure to the model boosts AUC performance.

The AUC measures how well an ordering of the variables fits a given binary assignment. The AUC loss is therefore a function of a binary-vector ground truth and a predicted *ordering*. Therefore, our learning framework is a hybrid: its input is that of a classification task, but its output is that of a ranking task. If a binary assignment of the variables is thought of as a partial ordering ($x_i > x_j \Rightarrow i \prec j$), the ground truth can be thought of as a partial observation. This partiality adds to the difficulty of learning, and induces a trade-off between the convexity and the tightness of the surrogate loss.

2. Main Contributions

Our main contributions are as follows:

1. We construct a novel link prediction structured model over orderings of edges where the AUC loss can be naturally incorporated alongside other high-order factors.

2. We propose three such high-order factors specific for the task of link prediction, based on the notions of clustering, small-world effects, and scale-free degree distribution.

3. We show how to efficiently solve the combinatorial optimization problems induced by the proposed high-order factors, common multi-variate losses, and the proposed AUC loss.

4. We suggest several solutions to the problems arising from the partiality of observations, and discuss optimization techniques relevant to the task.

5. We run experiments on synthetic and real-world networks and compare our results to those obtained by other algorithms.

The incorporation of the AUC loss into a structured learning framework can serve as a basis for further research. With a tractably learnable model in hand, we can now construct and incorporate complex, structural factors which can express rich prior knowledge regarding social networks and their formation dynamics. This general AUC framework can easily be applied to other domains in which AUC is a common performance measure.

3. Outline

The chapters in this work are organized as follows. We begin in chapter 2 where we describe *link prediction*, a longstanding prediction task in the domain of complex networks. Next, in chapter 3 we formalize link prediction as a machine learning task, and survey relevant works in the field. By analyzing the strengths and weaknesses of current methods, we build a case for approaching link prediction from a *structured learning* perspective. Chapter 4 contains an introduction to structured learning, and several handy tools which we use within this framework - Markov Random Fields, dual decomposition, and LP relaxations. In chapter 5 we elaborate on link prediction as a structured learning task, propose several novel social-network related high-order features, and discuss relevant multi-variate loss functions. In chapter 6, we focus on the most common link prediction measure - the AUC - and propose a novel model for learning directly over the AUC loss and several of its extensions. Finally, in chapter 7 we present some experimental results for learning with the AUC loss.

Chapter 2

Link Prediction

1. Overview

Scientific interest in social networks dates back to the early 30-s. Pioneering works by sociologists such as Parsons (1968), Simmel and Wolff (1950) and social psychologist Moreno (1934) focused their research on the structure of interpersonal relationships and how it affects social affairs. Later, Milgram's famous small-world experiment shed a light on the connectedness of society as a whole (Milgram, 1967). A mathematical formulation of these new approaches began in the 50-s and led to new theories and principled research methods (Luce and Perry, 1949; White, 1970; Fienberg and Keith Lee, 1975). Over the years, works of researchers from diverse fields such as sociology, psychology, economics, ecology, biology, physics, mathematics, computer science, and others have led to the creation of the academic field of network science. Many interesting questions in the field can be formulated as algorithmic problems, and thus fitting for research in computer science. Issues such as the computation of network properties, the generation of networks, and the understanding of network dynamics, are examples of important network-related questions which have been treated in the computer science literature. One of the first and most prominent algorithmic tasks in network science is that of link prediction - given a partial network, return the set of missing edges.

Since a graph makes for a good abstraction of a social network, the general task of link prediction can be formalized as follows. Let $G = (V, E)$ be some graph representing a social network, where the nodes V represent the people in the network and the edges E represent links between them. As input, we are given V and a subset of *observed* edges $E_{obs} \subset E$. Our task is to produce as output the set of *missing* links $E_{miss} = E \setminus E_{obs}$. While this problem is mathematically well defined, if G is just any arbitrary graph, it is not clear how it can be solved. Luckily, extensive research in the network sciences has shown that social networks contain many regularities which can be exploited. Furthermore, since the task of *exactly* retrieving E_{miss} is formidable, most formulations of the link prediction problem require as output a subset of edges

which maximize some performance measure (for which E_{miss} is a maximizer), or an ordering over the edges (in which edges in E_{miss} should be highly ranked). Many different approaches have lead to many different solutions to the problem of link prediction. In the following sections we further describe the task of link prediction, and survey machine learning solutions found in the literature.

2. The Missing Link(s)

But why should edges be missing in the first place? To answer this, we will describe some real-world settings of link prediction. Each setting gives rise to a different reason for links to be missing, and a different goal for link prediction.

2.1 Data collection

For various reasons, be it academic or business-related, people or institutions often collect social network data. This data can be sampled by methods as diverse as handing out friend-naming questionnaires, or crawling on-line social networks. Often these methods produce a small sample of the network, where many edges (and possibly nodes) are missing. These samples are often biased, and tend to be noisy. Therefore, the goals of link prediction in this case are to enrich the sample with probable, unsampled edges, remedy the sampling biases, and reduce the errors caused by noise.

2.2 Multi-model and partial networks

Many different networks can simultaneously exist amongst the same group of people. Such networks are called multi-modal. It is often the case that while some types of networks are accessible, others are not. For instance, while friendships are often visible, enmities tend to reside under the surface. A special case of simultaneous networks is when one network is a subset of the other. For instance, a network linking people who frequently phone each other is probably a partial to the network of people who socialize in general, and can be used as a proxy.
In this setting, the observed and missing edges are of a different *types*. The goal of link prediction here is to predict the unseen network, given the observed one.

2.3 Dynamic networks

In many setting, networks are actually dynamic objects, where nodes and edges constantly appear and disappear. Since these dynamics contain regularities as well, a natural link prediction task that arises is: given the history of the network, what is its future? What links are bound to form?
Link prediction in this setting is used to predict the appearance of new or recurring edges, given past connections. The 'missing' links are in fact yet to be created.

In settings where networks are maintained by some outer source (such as online social networks), link recommendation can be thought of as an extension of the above.

2.4 Anomaly Detection

In some scenarios, it is useful to report false connections, for instance to high-status peers. In others, such as covert networks, some edges are best kept secret. Observed links which are not real and real links which are hidden can be thought of as *anomalies*. Link prediction in this setting can be used to detect these anomalies.

3. Performance measures

As noted before, instead of requiring for all missing links to be retrieved, most link prediction tasks define a measure of how good a predicted edge set or an ordering of the edges is. In this section we present the most common measures used in link prediction. Consider a network $G = (V, E)$, where E are the true edges. We denote by E_{obs} the set of observed links, by E_{miss} the set of true missing links, and by D the set of predicted edges (we assume that $D \subseteq \overline{E}_{obs}$). A performance measure estimates how well a prediction D relates to the missing edges E_{miss}.

The measures we present are a function of the type I and type II errors of a prediction. More specifically, they are a function of the true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) of a prediction. Each edge in D is counted as a true positive if it is also in E_{miss}, or a false positive if it is not. Likewise, each potential edge *not* in D (namely every $e \in \overline{E}_{obs} \setminus D$) is counted as a false negative if it is in E_{miss}, and as a true negative if not. We shall denote by P (for positive) the number of edges in E_{miss}, and by N (for negative) the number of potential edges which are not (namely $\overline{E}_{obs} \setminus E_{miss}$).

3.1 Accuracy

Accuracy, the simplest measure of how good a prediction is, uses the following idea - gain a point for every right guess ('hit'), and lose a point (or rather gain none) for every wrong one ('miss'). Formally,

$$accuracy = \frac{TP + TN}{P + N} \tag{2.1}$$

It is easy to see that E_{miss} is indeed a maximizer of the above measure. We also note that this measure decomposes over D, in the sense that it can be written as a sum over all $e \in D$. This is easy to see when accuracy is thought of as the hamming distance between the indicator vectors v, u of E_{miss} and D, respectively:

$$hamming(v, u) = \sum_i \mathbb{1}_{\{v_i \neq u_i\}} \tag{2.2}$$

Social networks are known to be sparse - usually the number of edges is in the order of the number of nodes ($|E| = O(|V|)$). This leads to great imbalance between the two classes of existent and nonexistent edges. Therefore, the accuracy measure is not very effective - a predictor which predicts all nodes to be nonexistent will achieve very high accuracy.
Accuracy is rare in the link prediction literature - in most works other measures are used. In works that *do* focus on accuracy, the setting is usually set such that classes are balanced (for instance in Al Hasan et al., 2006).

3.2 Recall, Precision, and F1

More sophisticated error measures are functions of the contingency table of the type of errors. The most common ones in link prediction are precision, recall, and F1 (which is a function of precision and recall). Intuitively, recall states how many of the missing edges have been retrieved, and precision states how many of the predicted edges were indeed missing. The F1 score is the harmonic average of the recall and precision of a prediction.

Recall, precision and F1 are formally defined as following:

$$recall = TP/(TP + FN) \tag{2.3}$$

$$precision = TP/(TP + FP) \tag{2.4}$$

$$F1 = \frac{2TP}{2TP + FP + FN} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.5}$$

To battle class imbalance, in some settings the number of predicted links (in other words, the size of D) is predefined to be some constant k. In these, precision and recall are referred to as $precision@k$ and $recall@k$. The k edges are usually attained by ranking all edges and choosing the top k, but can also be chosen by a model constraining its output to hold exactly k edges. However, if an algorithm outputs a ranking, the AUC measure is usually considered more adequate.

3.3 AUC

In certain settings, the output is *required* to be an ordering of the potential edges (and not a subset of them). Edges are usually ordered by the belief of them truly existing, so that if one wants to predict k edges, he should choose the top k. A common performance measure for an ordering output over binary ground-truth inputs is the area under the the Receiver-Operating Characteristic (ROC) curve, coined AUC for 'area under curve'. The ROC is attained by plotting the true positive rate (TP/P, namely recall) and false positive rate (FP/N, also known as 'fall-out') on the x and y axis (respectively) as the number of predicted edges k is increased (see fig. 6.1). The

ROC captures the essence of what happens when more and more edges are added to a predicted set, based on the output ordering. The AUC, simply being the area under the ROC curve, measures how good an ordering is regarding the above intuition. Since the ROC is normalized, the AUC ranges between 0 and 1. An AUC of 0 implies a bad ordering - one that when edges are selected by it, the recall is always 0; an AUC of 1 implies a good ordering - one that for each added edge, the recall is always 1. In other words, AUC=0 implies that all true missing edges are placed at the bottom of the ordering, while AUC=1 implies that all of them are at the top. Note that the internal ordering of the missing or non-missing edges is of no significance. The expected AUC of a random ordering is 0.5.

The AUC measure is by far the most prevalent in the link prediction literature. Since the it is central in this work, it well discussed in detail in chapter 6.

Chapter 3

Link Prediction as a Learning Task

As we saw, link prediction is in essence a prediction problem over highly regularized data with well defined performance measures. Therefore, approaching it from a machine learning perspective is natural. Furthermore, the abundance of rich social network data made available in recent years has boosted the quantity and quality of work in the field. In this section we survey machine learning approaches to link prediction found in the literature. More thorough surveys can be found mainly in Liben-Nowell and Kleinberg (2007) and in Lü and Zhou (2011), and also in Xiang (2008) and in Hasan and Zaki (2011).

In the coming section we use the following notation. A learning task is defined by a sample set $\boldsymbol{x} = \left(x^1, \ldots, x^M\right) \in \mathcal{X}^M$ and (optionally) a corresponding label set $\boldsymbol{y} = \left(y^1, \ldots, y^M\right) \in \mathcal{Y}^M$, where each sample-label pair (x^m, y^m) is assumed to be drawn from some joint distribution $D_{\mathcal{X}\mathcal{Y}}$. Since in most settings a partial network G (or a set of networks) is given as input, the sample $\boldsymbol{x}$ contains information regarding the network (or part of it). A label $\boldsymbol{y}$ is a binary vector, with an entry y_e for each potential edge e, indicating its existence ($y_e = 1$) or non-existence ($y_e = 0$). Features $\phi_m = \phi(x^m; G)$ are computed form the topological properties of the sample network(s). In some settings, other information (such as nodal or edge attributes, timestamps, or semantic side-data) is available and can be incorporated into features as well.

1. Unsupervised learning

In *unsupervised learning*, the input data consists of samples alone, with no corresponding labels. In many of the following settings, the input data is a network, where each sample is focused on an edge, and features are extracted from the edge's neighborhood, or from the whole network. Therefore, features are topological, in the sense that they are a function of only the topology of the network (and not of other side-information). Predictions are made for all potential edges (namely those absent from the given network), where features for them are calculated in the same manner.

In essence, most of the proposed unsupervised methods consist of a heuristic score function, on the basis of which potential edges are ordered. The score function is often interpreted as a *proximity measure*, where close nodes are assumed to have a high probability of linkage. An algorithm's output is either a full ordering over the potential edges, or an edge set containing the top k scoring potential edges. If a full ordering is returned, the common performance measure is the AUC. If a set of edges is returned, common performance measures are recall, precision, and F1.
Albeit being amazingly simple, these unsupervised methods preform rather well on real data in various domains, scale adequetly, and can easily be applied to the set of all potential edges (usually on the order of $O(|V|^2)$). They are often used as building blocks for more sophisticated supervised methods. However, they are entirely heuristic, and do not make use of the knowledge of existing edges in the given network, which could potentially improve performance.

We now present several well studied unsupervised methods common in the link prediction literature. These methods regularly serve as baselines for more advanced learning techniques. For empirical comparisons of the following methods and others, see Liben-Nowell and Kleinberg (2007); Lü and Zhou (2011). In this section, we denote by $\Gamma(v)$ the set of neighbors of a node v.

1.1 Common Neighbors (CN)

Many proximity measures are functions of the set of joint neighbors. The most straightforward score based on this is the number of common neighbors. Formally:

$$\mathrm{CN}(u, v) = |\Gamma(u) \cap \Gamma(v)| \tag{3.1}$$

for any nodes u, v in the network. This score was initially used by Newman (2001) as a proximity measure in collaboration networks.

1.2 Jaccard Coefficient (JC)

Other proximity measures based on common neighbors where quick to follow, most being differently normalized variations. The most notable is Jaccard's Coefficient (Jaccard, 1901), borrowed from its modern use in the field of information retrieval (Salton and Mcgill, 1986). The measure is defined by:

$$\mathrm{JC}(u, v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|} \tag{3.2}$$

or in other words, the number of common neighbors divided by the number of overall neighbors.

1.3 Adamic-Adar (AA)

A more sophisticated useage of common neighbors is that proposed by Adamic and Adar (2003). Although originally used in the context of linked web pages, it has pro-

vided superb empirical results in many link prediction settings over several domains, especially considering its simplicity.
The logic behind this measure is to give weights to common neighbors as a function of their degree. Perhaps counterintuitively, the weights are *inversely* proportional to the degree - high degree nodes contribute small weights, and low degree nodes contribute large weights. Adamic and Adar (2003) propose to use the inverse of the logarithm of the degree:

$$\mathrm{AA}(u,v) = \sum_{z \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log(|\Gamma(z)|)} \tag{3.3}$$

1.4 Preferential Attachment (PA)

Real world networks across various domains tend to have a scale-free degree distribution (Barabási and Albert, 1999; Mitzenmacher, 2004; Clauset et al., 2009). One of the first network evolution models to present this kind of global statistical behavior was the *Preferential Attachment* model (Barabási and Albert, 1999), where nodes are added one by one to the network, and linked to m existing nodes with probability proportional to their degree. In a similar manner, the preferential attachment proximity measure (Newman, 2001; Barabâsi et al., 2002) is defined to be the product of the degrees of both nodes:

$$\mathrm{PA}(u,v) = |\Gamma(u)| \cdot |\Gamma(v)| \tag{3.4}$$

1.5 Katz (KZ)

So far, the proximity measures presented where only a function of a node's neighborhood. In contrast, the Katz and PageRank measures make use of the whole network to calculate the score of a potential edge. Specifically, the distance between nodes u and v relies on the ensemble of all paths between them.
In the Katz measure (Katz, 1953), the score of an edge is a weighted sum of the lengths of all paths between them, where the weights decay exponentially with the path's length:

$$\mathrm{KZ}(u,v) = \sum_{p \in P(u,v)} \beta^{|p|} |p| \tag{3.5}$$

where $P(u,v)$ is the set of all paths between u and v, $|p|$ is the length of path p, and β is an exponentially decaying parameter. A nice feature of the Katz measure is that it can be efficiently computed by the following closed form matrix formula:

$$\mathrm{KZ} = (I - \beta A)^{-1} - I \tag{3.6}$$

where A is the adjacency matrix of the network, and I is the identity matrix. As with the Adamic-Adar measure, the Katz measure has also shown fair predictive qualities over various domains and settings. Recently a generalization of the Katz measure for temporal networks has been proposed (Dunlavy et al., 2011).

1.6 PageRank (PR)

Another way to measure proximity in a network is to use *random walks*. As in the original PageRank algorithm (Page et al., 1998), the basic measure is defined by:

$$\mathrm{PR}(u, v) = H(u, v) \tag{3.7}$$

where $H(u, v)$ is the *hitting time* from u to v in the network Nodes are traversed by randomly selecting neighbors with probability $1 - \alpha$, and the walk is restarted at a random node with probability α. The hitting time $H(u, v)$ is the expected number of steps it takes a walk started at u to reach v.
Other variants include the commute time $C(u, v) = H(u, v) + H(v, u)$ for directed networks, normalized PageRank $\mathrm{nPR}(u, v) = H(u, v) \cdot \pi_v$ (where π is the stationary distribution), and others. Hit times with random restarts can be computed efficiently with an algorithm proposed by Tong et al. (2006).

2. Supervised learning

Supervised learning is a popular, principled way of learning a classifier over labeled data, namely a set of sample-label pairs $\{(x^m, y^m)\}_{m=1}^{M}$. In most methods, a mapping $f : X \to Y$ from a family of hypotheses H is learned by training on the input data, and is then used to create predictions for new, unseen data. The performance of a learned model is evaluated according to its predictions on a held-out test set, according to some performance measure. In the heart of most supervised methods lies a loss function corresponding (as best as possible) to the desired performance measure. In this way, the mapping f is chosen as to best fit the data with regards to the performance measure, and regularization is often applied to reduce overfitting.
While unsupervised methods can easily handle all potential edges, for supervised methods this task can be daunting. This drawback makes supervised methods work in some settings, but not in others. However, many supervised methods have both strong theoretical guarantees and superior empirical results, which more than compensates for their computational cost.

2.1 Plugging in network features

Prolific research in supervised learning has amounted to a solid set of algorithms with theoretical guarantees and empirical success. As in many new domains, the first supervised link prediction works (such as Al Hasan et al., 2006) consisted of constructing simple task-specific features and applying well known algorithms such as SVMs (Cortes and Vapnik, 1995), decision trees (Quinlan, 1986), and neural networks (Hopfield, 1982). The features used in these early works were mostly simple topological network attributes such as degree and distance (Lichtenwalter et al., 2010; Al Hasan et al., 2006) and similarity measures used in unsupervised settings, such as Adamic-Adar (1.3), Katz (1.5), and Common Neighbors (1.1) (see Benchettara et al., 2010;

Fire et al., 2011). When available, semantic side information was also used (Al Hasan et al., 2006). Next, new task-specific features where constructed for particular settings. For instance, Lichtenwalter et al. (2010) presented a modified PageRank feature fit for weighted networks, and Benchettara et al. (2010) constructed features specific for bi-partite networks. Most works of this genre supply an empirical comparison between different supervised methods, alongside information-theoretic analysis of the contribution of different features. Other well-known classification schemes such as probabilistic classification have also been applied (Wang et al., 2007).

2.2 Matrix completion and factorization

Another useful representation of a network is the *adjacency matrix* of the network graph, where the (i, j)-th entry of a matrix A corresponds to the edge between nodes i and j. Several link prediction techniques are based on this representation. In the most basic setting, $A_{ij} = 1$ if edge (i, j) is known to exist, 0 if it is known to *not* exist, and left blank if no information is available. The task is then to fill in the unobserved (blank) entries, based on the observed ones. This task is often reffered to as *matrix completion*, and has been addressed by general methods such as collaborative filtering (Koren et al., 2009).

In (Menon and Elkan, 2011), each node i is assumed to have a latent feature vector u_i. In turn, these vectors induce an approximation of the adjacency matrix A by the decomposition $U\Lambda U^\top$, where U is a concatenation of the latent feature vectors u_i and Λ is an arbitrary low-rank asymmetric matrix. The objective is then to find an assignment to the feature vectors which minimizes some distance measure between the original adjacency matrix A and the decomposition (plus some regularization term).

Another example of using matrix factorization techniques can be found in Tang et al. (2013). While not solving the link prediction problem per se, the authors model the social network notion of homophily (McPherson et al., 2001) in order to predict trust between actors. In their proposed solution, a low rank decomposition of a trust-matrix is learned, where sparsity and homophily serve as regularizers.

Research in the network sciences has shown that real-world networks tend to have a fractal-like, recursive structure (Ravasz and Barabási, 2003). In (Leskovec et al., 2010), this idea is captured by constraining the adjacency matrix to be a k-th order Kronecker product of some small initial matrix. While learning in this setting is not done by minimizing a loss function (but rather by maximizing the likelihood of the model parameters), it is still an interesting example of the merits of modeling a network by its adjacency matrix.

2.3 Non-topological features

Online social networks contain rich, abundant, and attainable nodal attributes (e.g. personal attributes, interests, geographic location) and edge attributes (e.g. tie strength, sign, type, time of creation). Many social networks are also related to rich textual data (e.g. email content in communication networks, scientific articles in collaboration networks), therefore allowing for the extraction of semantic features. Hence, recent works in link prediction have focused on creating classifiers which incorporate both topological network features and other side information.

In (Backstrom and Leskovec, 2011), the authors frame an extension of the PageRank algorithm (Page et al., 1998) into a supervised learning task which takes into account node and edge attributes. A random walk is used to define a proximity measure between nodes, and is biased according to node and edge attributes, therefore taking them into account. The learning objective contains soft constraints over the PageRank scores as a function of the labeled input.

In some settings, labeled data is rare or costly, but unlabeled data can easily be acquired. Semi-supervised learning, a cross-breed between supervised and unsupervised learning, gives a formal learning framework for problems in such settings. Given a social network, known edges and known non-edges can be considered labeled data, and edges whose status is unknown can be considered as unlabeled. In fact, this is exactly the *transductive* learning setting, since all samples are known in advance. In Kashima et al. (2009), the authors apply a variant of label-propagation, a well known semi-supervised algorithm (Zhu et al., 2003), for estimating the strength of networks with many link types given a partial network and nodal attributes.

3. Generative models

As opposed to discriminative models which aim at learning a classification rule, the goal in *generative learning* is to learn a joint distribution over the space of sample-label pairs. Once attained, the generative model can be used for classification by maximizing conditional probabilities. If the model is parameterized, the learning objective is usually to find a set of parameters which best explain the data, often done by maximizing the data's likelihood.
Generative models are very expressive and can easily incorporate rich global structural features. However, they may be hard to learn, and inference is often intractable. Hence, when the task is merely classification, they may be too general, and therefore require more data. In the following section we present some generative solutions to the link prediction task found in the literature. For more complete surveys, see Goldenberg et al. (2010) and Airoldi et al. (2007).

3.1 Topological models

Inspired by the classic $G(N, p)$ random graph model proposed by Erdős and Rényi (1960), statistical models for social networks started appearing in the 80's, and are still popular in the literature today. These models often encode parameterized probability distributions which express topological properties of graphs. First attempts included modeling outgoingness, popularity and reciprocation (the p_1 and p_2 models, Holland and Leinhardt, 1981; Duijn et al., 2004, respectively). Exponential Random Graph models (a.k.a p^* models, see Frank and Strauss (1986); Wasserman and Pattison (1996)) are Markov Networks which contain factors relating to various structures such as k-stars and triangles, and have been successfully applied to social networks (Snijders et al., 2006; Robins et al., 2007). Other models giving rise to scale-free and small-world networks have also been proposed (Barabási and Albert, 1999; Watts and Strogatz, 1998).
More recent works include a temporal variation of the p^* model (Hanneke and Xing, 2007), a model which includes cycles which reproduce real-world clustering coefficients (Huang, 2006), a method in which a scale free degree distribution is attained by means of regularization (Liu and Ihler, 2011), and a non-parametric model which focuses on neighborhoods by allowing each to have different dynamics (Sarkar et al., 2012).

While the above works do not necessarily focus on solving the link prediction problem, the models they suggest and their learning and inference methods can easily be used for link prediction. The classic works have also influenced most contemporary generative link prediction papers, and are therefore worth mentioning.

3.2 Hierarchical models

One of the most notable traits of social networks is that they often exhibit *communities* (Clauset et al., 2004), highly-clustered regions of tightly-knit relationships. Clustering however is not constrained to a local scale. Community-like structures appears in multiple scales, and can by thought of as hierarchy of groups (Ravasz and Barabási, 2003). This self-similar fractal-like behavior has been observed in various networks (Song et al., 2005).

The most notable link prediction work based on a hierarchical model can be found in Clauset et al. (2008). The authors proposed a probabilistic model of hierarchical organization. Once learned, networks generated by this model exhibit other known real-world network properties such as a skewed degree distribution, short path lengths, and high clustering. The model can be used to predict missing links by predicting the existence of unobserved links having high probability.
Another work (presented in 2.2) which models networks with a recursive structure by

fitting their adjacency matrix to a Kronecker product of some small matrix can be found in Leskovec et al. (2010).

3.3 Evolutionary and dynamical models

Networks, by nature, are dynamic objects (Leskovec et al., 2008). Links form and break, attributes change, and structures appear and disappear over time. In several works an *evolutionary process* is modeled, and new or missing links are predicted based on the learned evolutionary mechanism. In Kashima and Abe (2006), the authors construct a link-copying evolutionary Markov Process, where a node i copies a link to some node j from node k with probability w_{ik}. Links are predicted based on their probability in a stationary state.

While dynamics do exist in networks, they do not have to be modeled. Several works aim at the task of *temporal* link prediction without assumptions regarding the evolutionary process of the network. For instance, Tylenda et al. (2009) assume that linking patterns are preserved but may vary over time. They model this by posing linking patterns as soft constraints.

3.4 Latent feature models

One of the strengths of probabilistic models is that they can easily incorporate *latent* variables. Since in most cases not all existing information is observed, latent features can be encoded into the model, and their interaction with the observed variables can result in better learning and prediction.
In Miller et al. (2009), the authors propose a model with both latent and observed features for nodes and edges. As in Leskovec et al. (2010), they take as input the observed edges and non-edges, and predict the existence of unobserved potential edges. In Menon and Elkan (2010), the authors allow for ordinal and nominal labels, by creating a model with nodal latent feature vectors, where the interaction term is given by an inner product of two such vectors.

4. A case for structured learning

As we examine the aforementioned machine learning approaches to link prediction, two main trends emerge - a local trend and a global trend. Unsupervised proximity measures such as common neighbors and Adamic Adar are in principle *local*, in the sense that they focus on a individual links. Even the Katz and PageRank measures, though making use of information from the whole network, are ultimately local. Most supervised methods build on this notion of locality, and treat different potential edges as different, unrelated samples; predictions made for different edges are independent. However, the *existence* of these edges may be inter-dependent. In these methods, global properties of networks can not be taken into account, and there is no guaran-

tee that the predicted network will adhere to them.
Generative models on the other hand focus on *global* statistical traits of networks, where 'global' is interpreted as describing neighborhoods, communities, or the whole network. Since a distribution over entire networks is modeled, global properties are naturally incorporated into the model. However, *local* information does not always play a role in the process. Moreover, these global properties are encoded into the distribution over *networks*, and not into the prediction task per se.

It seems that in order to predict links, either a local or a global approach can be taken, each with its pros and cons. But can *both* roads be taken simultaneously? *Structured learning* is learning framework fit for the integration of both approaches. It allows for the principled, supervised training of classifiers which can incorporate rich structural features. Moreover, by generalizing standard supervised learning approaches, structured learning allows for the loss function itself to contain structure. Using *dual decomposition*, structural regularities can be easily incorporated into the structured learning framework.
There are only two structured-learning based link predictions works we are aware of in the literature, both by the same authors. Andrews and Jebara (2006) suggested a method for predicting a network whose degree distributions attains to that of a $b-matching$. Later, Andrews and Jebara (2008) present a structured link prediction method in which the in and out-degrees of nodes can be constrained. This is fitting for setting where the number of friends a person has is known, but their identity is not.

In the following sections, we will briefly overview structured learning fundamentals, and present dual decomposition as a framework under which structured learning can be practiced. In the next chapter we elaborate on the case of link prediction as a structured learning task by portraying structural regularities in social networks, and showing how they can be incorporated into a learning framework.

Chapter 4

Structured Learning

In this chapter we present the fundamentals of structured learning. We begin with some motivation and examples, proceed with a formal representation of the structured learning task, and present Markov Random Fields (MRFs) and Dual Decomposition as two important tools we use for modeling structural regularities within this learning framework.

1. Motivation

In supervised learning, a predictor is learned from a set of sample-label pairs (x^m, y^m) with the objective of returning good outputs y for new, unseen samples x, where 'good' is defined with regard to some loss function. In the basic supervised learning framework, it is implicitly assumed that the output is 'simple', in the sense that a single output has no internal structure, nor that there exist dependencies between outputs. However, in many real world settings, this is clearly not the case. Many domains have rich structural regularities, whether within an output or between a set of outputs. Problems with structural properties are generally referred to as *structured output prediction* (Bakir et al., 2007), while the general learning task of data with structured output is referred to as *structured learning*. As we shall see, link prediction is indeed a task containing high structural regularities, for which structured learning is a most suitable a approach.

Standard supervised methods are inadequate for structured learning. Binary classification approaches are not fit for modeling interdependencies, while classic multi-class methods may suffer from an over-representation of an unnecessarily exponentially-large output space. Moreover, the standard 0-1 classification loss function does not capture the structured essence of the output, deeming the surrogate hinge-loss inadequate. These drawbacks of standard methods have led to a multitude of approaches for incorporating structure into learning. We adopt the general approach and notation of Tsochantaridis et al. (2004), which we will formally introduce in the next

section. But before delving into formalities, we present two motivating examples.

Image segmentation

Consider a simple prediction task, where noisy images are given as input, and as output each pixel is assigned to either the 'foreground' or 'background' class, depicting its role in the image. This task is widely known as *image segmentation*, since the image is segmented into foreground and background. In this task, a binary label is created for every pixel.
Treating each pixel independently often leads to inadequate results. However, real-world images are highly structured, and contain strong dependencies amongst subsets of pixels. For instance, one such regularity relates to adjacent pixels. If one assumes that two adjacent pixels are dependent (for instance, tend to be of similar color), adding *pairwise factors* over these pixel pairs can express this belief. In this model, each image is treated as a sample, and its label is a binary vector with an entry for each pixel. The parameters of these factors can be learned over a training set of noisy images and segmentation ground truths.

Parsing trees

In the task of natural language parsing, a sentence is given as input, and a predictor outputs a *parse tree* for that sentence - a directed tree representing its syntactic structure. In the parse tree, leafs represent the words in the sentence, internal nodes represent grammatical categories (such as noun, verb, determiner, etc.), and edges depict relations between and within words and categories. The output of this prediction task - a directed tree - is by any means a complex structured object with strong regularities, both in the topological features of the directed tree and in the frequency and position of grammatical sub-structures.

2. Formal representation

Consider the standard supervised learning setting where the learner is presented with a set of M sample-label pairs $\{(x^m, y^m)\}_{m=1}^M$, assumed to be drawn from some joint distribution $D_{\mathcal{X}\mathcal{Y}}$. The goal of supervised learning is to find a mapping $f : \mathcal{X} \to \mathcal{Y}$ from a family of hypotheses H. This mapping is evaluated with regard to a loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, where a low cost is attributed to similar labels. Therefore, we assume here that $\Delta(y, y) = 0$ and $\Delta(y, y') \geq 0$ for all $y, y' \in \mathcal{Y}$. The learning objective is often to find a mapping f which minimizes the expected loss over P, and is in practice approximated by minimizing the empirical loss:

$$L(f) = \frac{1}{M} \sum_{m=1}^{M} \Delta\left(f(x^m), y^m\right) \tag{4.1}$$

This formulation, however, does not allow for the exploitation of the output's structural regularities. Therefore, in structured learning, the mapping f is generalized in the following manner: a discriminant function $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is defined, inducing the prediction rule:

$$f(x) = \arg\max_{y \in Y} F(x, y) \tag{4.2}$$

F can be thought of as assessing how much a label y is compatible to a sample x. Since the prediction rule is now mediated by a function of both x and y, F can be used to express the structural qualities of the data. However, learning is now penalized by the computational complexity of inferring a prediction (solving the $\arg\max$), which can be hard in general. Nonetheless, in many structured learning problems the $\arg\max$ can be solved or approximated efficiently, making the computational cost noticed but worthwhile.

In this work, as in many others, we focus on linear discriminant functions of the form:

$$F(x, y; w) = \langle w, \phi(x, y) \rangle = w^\top \phi(x, y) \tag{4.3}$$

where w is some weight vector, $\phi = \phi(x, y)$ is a feature vector computed from a sample-label pair, and $\langle \cdot, \cdot \rangle$ denotes the standard inner product. For convenience, we will also assume that y is in the form of a binary vector. Since ϕ is a function of y as well as x, it can be made to express the structure of the output. For instance, in the image segmentation examples, ϕ will likely take into account assignments to pairs of adjacent pixels; in the parsing tree example, ϕ could take into account a tree's depth.

In the above setting, the $\arg\max$ problem is of a combinatorial flavor, since it consists of an optimization problem over a combinatorial body (namely all assignments to $y \in Y$). In many cases the objective function is accompanied by constraints over y, sometimes as a function of x. For instance, in the parsing tree example, y is constrained to represent some arborescence. Since we are interested in maximizing F, these constraints can by enforced by adding a function $g(x, y)$:

$$F(x, y; w) = \langle w, \phi(x, y) \rangle + g(x, y) \tag{4.4}$$

such that $g(x, y) = -\infty$ for any (x, y) pairs not adhering to the constraints.

In terms of a learning objective, Tsochantaridis et al. (2004, 2006) suggest solving the following minimization problem:

$$\min_{w} \frac{\lambda}{2}\|w\|^2 + \frac{1}{M}\sum_{m=1}^{M} \max_{y \in Y} H^m(y) \tag{4.5}$$

where:

$$H^m(y) = w^\top \phi(x^m, y) + \Delta(y, y^m) - w^\top \phi(x^m, y^m) \tag{4.6}$$

$H^m(y)$ serves as a surrogate to the loss Δ, and is in fact a generalization of the binary hinge-loss. Intuitively, the standard margin is replaced by a general notion of the distance between the true output and the highest-scoring alternative (Andrews and Jebara, 2006). As for classification, the prediction rule is set to be:

$$\hat{y} = h(x^m) = \arg\max_{y \in Y} w^\top \phi(x, y) \tag{4.7}$$

Many algorithms for minimizing over w, such as the Structured Perceptron (Collins, 2002), structured SVM (Tsochantaridis et al., 2006), and structured stochastic subgradient descent variants (Ratliff et al., 2006; Meshi et al., 2010) require computing the maximizing assignment (or max-marginals) of $H^m(y)$ as intermediate steps, and computing $h(x^m)$ is always required for classification. Therefore, the theoretical challenge in structured learning often boils down to efficiently solving the combinatorial problems induced by the model. Hence, in order to apply structured learning to link prediction, one must construct features which not only express the structural regularities of the domain, but which can also be computed efficiently.

In our work, we express structural regularities by constructing a Markov Random Field (MRF) over a social network. MRFs allow for a natural formulation of structural prior knowledge in the form of factors, and fit well in the above structured learning setting. In the following section we briefly describe MRFs and their role in structured learning.

3. Markov Random Fields

Markov Random Fields (MRFs) are a class of graphical models representing joint distributions over a set of random variables $X = X_1, \ldots, X_n$ in a compact manner (Kindermann et al., 1980). They are a tool for expressing a joint probability P as a product of factors $\pi_c(x_c)$ over subsets of variables ($c \subseteq [n]$). Random variables are modeled as nodes in a graph, and edges express conditional independencies. In this

graph, factors are associated with maximal cliques of variables, which define their scope.

Formally, a joint distribution modeled by an MRF can be expressed as follows:

$$P(X = x) = \frac{1}{Z} \prod_c \pi_c(x_c) \tag{4.8}$$

where Z, the *partition function*, is a normalizing constant defined by:

$$Z = \sum_{x \in \Omega} \prod_c \pi_c(x_c) \tag{4.9}$$

and Ω is the set of all legal assignments to X.

Common tasks performed on MRFs are estimating probabilities, conditional probabilities, and computing the *maximum a-posteriori* (MAP) - the maximizing assignment to a set of unobserved variables (possibly conditioned on a set of observed variables). While estimating probabilities requires the calculation of the partition function (which is in general a hard task), computing the MAP does not. This trait leads the MAP problem to be tractable for a wide range of models, and approximable for many others.

Setting $\theta_c = \log \pi_c$, the joint log-distribution can be rewritten as:

$$\log P(X = x) = \sum_c \theta_c(x_c) - \log(Z) \tag{4.10}$$

Since the log function is monotonically increasing, and since $\log(Z)$ is constant, in order to compute the MAP it suffices to solve:

$$\arg\max_x \sum_c \theta_c(x_c) \tag{4.11}$$

If the variables are discrete, for many models computing the expression in (4.11) boils down to solving a combinatorial optimization problem.

MRFs are a strong an expressive tool for casting structural regularities into a probabilistic model. They have been studied extensively and applied successfully in fields such as natural language processing, computational biology, and machine vision.
In the core of many generative approaches to link prediction found in the literature lies an MRF or another graphical model (for instance Taskar et al., 2003; Wang et al., 2007; Huang, 2006), and the learning goal is to estimate its parameters. This however usually requires the estimation of the partition function in some form, and can make learning and inference hard. In contrast, structured learning requires only

the computation of the MAP assignment; by focusing efforts on minimizing the loss function, the need to estimate the partition function is bypassed. In this framework an MRF can serve as a score function, where an assignment's score is the value of its unnormalized probability. A prediction in this model is defined to be the MAP (4.11), and the loss if a function of the model's prediction and the ground truth.

In our work, we model a joint distribution over the set of potential edges of a network. Variables are defined for every potential edge, and the unnormalized log-value of an assignment to these variables is taken as the assignment's score. The model is parametrized by some weight vector w, and the learning task is to find a w which minimizes the loss over the training set.
In the general case, as in our work, exactly solving the MAP problem is far from trivial, and is in usually NP-hard. However, many approximation schemes exist. In the following section we introduce one such scheme called *dual decomposition*.

4. Dual Decomposition for MRFs

Recall that in order to find the MAP value, one must solve the following optimization problem:

$$\mathrm{MAP}(\theta) = \max_{x \in \mathcal{X}} \sum_{i=1}^{n} \theta_i(x_i) + \sum_{f \in F} \theta_f(x_f) \tag{4.12}$$

where F denotes the set of high-orders factors in the model. Notice that we have explicitly differentiated the factors into singleton scores over the n individual variables x_i and factors f over sets of variables x_f (where f's notation is abused to denote both the factor and its scope).
In general, computing the maximal value or inferring the maximal assignment in an MRF is NP-hard. Therefore, many approximation schemes have been proposed. Two such schemes - dual decomposition (Komodakis et al., 2011; Johnson, 2008) and linear programming (LP) relaxations over marginal polytopes (Wainwright and Jordan, 2008) - are closely related, and are in fact dual to one another. In both, constraints enforced in the original problem are relaxed, giving rise to tractable optimization problems which form an upper bound on the original problem. A nice artifact of dual decomposition is that an optimality certificate is attainable - if a certain condition is met, the solution of the relaxed problem is guaranteed to hold for the original problem as well. In this section we give a preliminary introduction to dual decomposition and its counterpart LP relaxation. For a more detailed analysis see Sontag, Globerson, and Jaakkola (2010).

4.1 Dual Decomposition

The main difficulty in solving the MAP problem is that x is shared across factors; a single variable x_i may affect not only the value of θ_i, but also that of several potentials θ_c. Dual decomposition addresses this difficulty be treating subproblems as independent, thus forming a relaxation of the original problem.
Later to be justified, we start by defining variables $\delta_{fi}(x_i)$ for every $f, i \in f, x_i$, and insert them into (4.12) in the following manner:

$$\text{MAP}(\theta) = \max_{x \in \mathcal{X}} \sum_{i=1}^{n} \left(\theta_i(x_i) + \sum_{f:i \in f} \delta_{fi}(x_i) \right) + \sum_{f \in F} \left(\theta_f(x_f) - \sum_{i \in f} \delta_{fi}(x_i) \right) \tag{4.13}$$

Notice that since each δ was added and subtracted once, the MAP has not changed. Next, we rewrite (4.13) by creating a duplicate of each variable for every factor it is part of, and enforcing agreement among the duplicates. This allows us to push the max into the sums. Denote by x_f^f the copy of variables x_f for the factor f, then:

$$\begin{aligned} \text{MAP}(\theta) = & \sum_{i=1}^{n} \max_{x_i} \left(\theta_i(x_i) + \sum_{f:i \in f} \delta_{fi}(x_i) \right) + \\ & \sum_{f \in F} \max_{x_f^f} \left(\theta_f(x_f^f) - \sum_{i \in f} \delta_{fi}(x_i^f) \right) \\ & \text{s.t.} \quad x_i = x_i^f \quad \forall\, i \in f,\ f \in F \end{aligned} \tag{4.14}$$

Now that we have formulated the agreement between factors as constraints, it is straightforward to relax the MAP problem by discounting these constraints, or in other words, no longer demanding agreement. Since we have only eliminated constraints, the resulting expression is an upper bound on the original MAP problem. Furthermore, since it is an upper bound for *any* value of δ, we can best approximate the original MAP by minimizing over δ. Thus, we are presented with a dual problem, where the δ-s play the role of dual variables (for clarity, since there is no ambiguity we have replaced x_f^f with x_f):

$$\min_{\delta} L(\delta), \tag{4.15}$$

$$\begin{aligned} L(\delta) = \quad & \sum_{i=1}^{n} \max_{x_i} \left(\theta_i(x_i) + \sum_{f:i \in f} \delta_{fi}(x_i) \right) + \\ & \sum_{f \in F} \max_{x_f} \left(\theta_f(x_f) - \sum_{i \in f} \delta_{fi}(x_i) \right) \end{aligned}$$

Notice that the global optimization problem in (4.11) has been broken up into smaller problems over subsets of variables. Solving the set of smaller problems is

potentially an easier task. However, a necessary condition for this relaxation to be worthwhile is that each local max operation can be efficiently solved; this is a central assumption of the dual decomposition framework. Nonetheless, factors which *do* have tractable solutions can easily be integrated into the model.
In the context of link prediction, dual decomposition lets predictive models contain high-order factors which express structural regularities of networks, given that maximizing over their reparametrization is tractable. If the variables in the model represent the set of potential edges, high-order factors can model the interaction between the existence and nonexistence of sets of edges. When plugged into a structured learning framework, dual decomposition allows us to tractably run inference over rich probabilistic models, while still directly optimizing a classifier over a designated loss function. In chapter 5 we go into detail about how this can be done, and propose several concrete, novel, high-ordered features for link prediction.

4.2 The δ-shifting trick

In many factors, tractable solutions can be applied only when $\delta_{fi}(x_i)$ can by written as $\delta'_{fi} \cdot x_i$ for some δ'. This is usually the case when x is thought of as an indicator vector of some set of objects, and the goal is to find a maximizing set. Using the δ' formulation allows for gaining δ'_{fi} when i is chosen and 0 if it is not. Fortunately, a simple $\arg\max$-preserving transformation from δ to δ' exists. Setting $\delta'_{fi} = \delta_{fi}(0) - \delta_{fi}(1)$ gives:

$$-\sum_{i \in f} \delta_{fi}(x_i) = -\left(\sum_{\substack{i \in f: \\ x_i = 1}} \delta_{fi}(1) + \sum_{\substack{i \in f: \\ x_i = 0}} \delta_{fi}(0) + \sum_{\substack{i \in f: \\ x_i = 1}} \delta_{fi}(0) - \sum_{\substack{i \in f: \\ x_i = 1}} \delta_{fi}(0) \right) \tag{4.16}$$

$$= \sum_{\substack{i \in f: \\ x_i = 1}} (\delta_{fi}(0) - \delta_{fi}(1)) - \sum_{i \in f} \delta_{fi}(0) \tag{4.17}$$

$$= \sum_{i \in f} \delta'_{fi} x_i - \sum_{i \in f} \delta_{fi}(0) \tag{4.18}$$

Since weight has been shifted from $\delta(0)$ to δ', we call this the *δ-shifting trick*, or simply the *shifting trick*. We will use this trick on several occasions in this work. When we do, this means that we replace $\delta(x)$ with $\delta' x$ and discard the constant $c = -\sum_{i \in f} \delta_{fi}(0)$.

4.3 An equivalent LP relaxation

Recall the original optimization problem given in (4.12). We focus now on the following relaxation:

$$\max_{\mu \in M_L} \sum_i \sum_{x_i} \mu_i(x_i) \theta_i(x_i) + \sum_f \sum_{x_f} \mu_f(x_f) \theta_f(x_f) \tag{4.19}$$

where M_L is the *local marginal polytope*, forcing μ to correspond to legal local probabilities, and contains the following constraints:

$$\sum_{x_f \setminus i} \theta_f(x_f) = \theta_i(x_i) \qquad \forall\, f, i \in f, x_i$$

$$\sum_{x_i} \mu_i(x_i) = 1 \qquad \forall\, i$$

$$\mu \geq 0$$

Had we forced μ to be integral, the optimization problem in (4.19) would be equivalent to the original combinatorial problem in (4.12). Therefore, the above relaxation is obtained by dropping the inetgrality constraints, allowing variables to be in the range $[0, 1]$. The μ-s can be thought of as weights in a normalized average over the assignments; constraining them to the local marginal polytope ensures agreement in their marginal probabilities.
Although the number of constraints in M_L may be exponential in the size of the largest factor, many problems (e.g. shortest paths, matchings, network flows) are known to have a compact representation in which the number of constraints is polynomial.

In fact, taking the Lagrange dual of (4.19) results in exactly the dual decomposition relaxation of (4.15). This shows a tight connection between the dual decomposition framework and LP relaxations, and allows us to use whichever frameworks best suits our needs.

Chapter 5

Structured Learning for link Prediction

In this chapter we propose a general structured model for link prediction. We begin by formally defining the model and discussing its qualities. Thereafter, we present some well known structured regularities found in real world networks, and propose novel ways of incorporating them into the model.

1. A structured link prediction model

At the base of a structured learning scheme lies a score function $score : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, which for a given sample $x \in \mathcal{X}$ gives a value to every assignment to the label variables $y \in \{0,1\}^n$ (we assume labels are binary vectors). In our work we assume the score function is *linear* in the sense that it is an inner product between a weight vector w and a feature vector ϕ, namely:

$$score(x, y) = w^\top \phi(x, y) \tag{5.1}$$

The prediction rule for a given sample x is set to:

$$\arg\max_{y \in \{0,1\}^n} w^\top \phi(x, y) \tag{5.2}$$

And the learning objective is given by:

$$\min_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{M}\sum_{m=1}^{M} \max_{y \in \{0,1\}^{n_m}} \left\{ w^\top \phi(x^m, y) + \Delta(y, y^m) - w^\top \phi(x^m, y^m) \right\} \tag{5.3}$$

Therefore, in order to learn within the structured learning framework, we must be able to solve the arg max in (5.2) and handle the sample-specific optimization problems given in (5.3).

We now construct a general model for the link prediction task. We set the score function *score* to be the unnormalized log-probability of a *log-linear* MRF. Recall that an MRF is a graphical model whose nodes are variables and whose edges define conditional dependencies. Since we are interested in predicting links in a network, a binary variable $y_e \in \{0, 1\}$ is created for each potential edge e. Instead of explicitly defining edges for the MRF, we define *factors* θ_f over subsets of variables which implicitly define the edges. In a *log-linear model*, factors are assumed to be linear in some parameter w, namely $\theta_f(y) = w_f^\top \phi_f(y_f)$. The induced probability function is of the form:

$$P(Y = y; w) = \frac{1}{Z} \exp\left\{\sum_{f \in F} w_f^\top \phi_f(y_f)\right\} \propto \exp\left\{w^\top \phi(y)\right\} \tag{5.4}$$

Therefore, its unnormalized log-probability (discarding the constant) is:

$$\log \tilde{P}(Y = y; w) = w^\top \phi(y) \tag{5.5}$$

which is exactly of the form of the score function *score*.

A note: having one graph representing a network and another representing a graphical model can be confusing, especially since edges in the first are mapped to nodes in the second. To avoid this confusion, nodes *and* edges *will always refer to the network, and* variables *and* factors *will be used to refer to the MRF.*

In our proposed model, the MRF includes *singleton factors* parametrized by shared weights w_s, high order factors $f \in F$ parametrized by weights w_f, and non-parametrized high-order factors $g \in G$. We will be interested in computing MAP assignments of the form:

$$\arg\max_{y \in \{0,1\}^n} \sum_{i=1}^{n} w_s^\top \phi_i(y_i) + \sum_{f \in F} w_f^\top \phi_f(y_f) + \sum_{g \in G} \theta_g(y_g) \tag{5.6}$$

Since in most settings we are given a network (or networks) as input, features are a function of the given network and (and not of sample x per se). Therefore, the x variable has been omitted from the feature function ϕ.

Singleton factors are used to bias the prediction of individual edges according to some prior knowledge. The singleton feature functions are usually given as $\phi_i(y_i) = \phi_i \cdot y_i$, where ϕ_i is a feature vector composed of different local scores. These local scores are often based on network topology (node degree, graph distance, etc.) and heuristics from unsupervised link prediction methods (see chap. 3 sec. 1). The shared weight parameter w_s is used to weight between the different local scores across all edges.
Parametrized high-order features are the main focus of this section and are the prime

reason for using structured learning for link prediction. Since the features' scopes can be over multiple variables, they can be used to encode structural regularities in networks; learning will then output the best weighing scheme over these factors, in the form of w. In the next sections we describe some well-known network structural regularities, and suggest novel high-order factors for them.
The non-parametrized high-order features have two main roles. The first is to encode hard constraints. This is done by setting $\theta_g(y_g) = 0$ for all y_g adhering to the constraints, and $-\infty$ for all others. The second and more important role is as the label loss, namely $\theta^m_\Delta(y) = \Delta(y, y^m)$. This will allow us to use multi-variate losses such as recall, precision, and F_1, and be the basis for constructing a model for the AUC loss, which a primary focus of this work and is detailed in chapter 6.

As mentioned, solving combinatorial optimization problems like (5.6) is in general hard. Hence, we often resort to some relaxation scheme. Since in most presented cases we use *dual decomposition*, we will focus on solving problems of the form:

$$\max_{y_f} \left\{ w_f^\top \phi_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right\} \tag{5.7}$$

Where $\phi_f(y_f)$ is some structure-encoding feature function. When possible, we will also show how to solve these problems with an *LP relaxation* by suggesting appropriate variables, weights, and linear constraints.

2. High order network features

In this section we focus on three of the main structural regularities consistently found in social networks - *high clustering*, *small-world effects*, and *scale free degree distributions*. We describe these network phenomena, propose novel features which encode the structural regularities they entail, and elaborate on solving the derived optimization problems.
We motivate this section by first presenting a simpler structural regularity found in directional networks - *reciprocity*. Throughout this section, we will use the letters i, j (and sometimes s, t) to denote nodes and e, f to denote edges in the network.

2.1 Reciprocity

Consider a directed social network, where edges (i, j) and (j, i) are distinct. One of the first structural regularities explored by network science researchers was that of *reciprocity* or *mutuality* - the tendency for same-nodes edge pairs to be bi-directional (Katz and Powell, 1953; Wellman, 1997).
Friendship networks are a classic example of directed networks. In these networks, even though edges do not *have* to be bi-directional, reciprocity is often abundant. In

other networks, like formal employee relations in a company or a network of financial loans, edges are almost never reciprocated. Yet in others, such as social microblogging or e-mail correspondence networks, some edge pairs are bound to be reciprocal (for instance between two real people), while others are not (for instance between a spam address and a real person). By building a model able to express mutuality, we can *learn* the prevalence of reciprocity in the data as a function of features over pairs of corresponding edges. Given this learned pairwise model, we can then output predictions which take into account reciprocal regularities, thus hopefully gaining higher accuracy.

How can we model reciprocation in link prediction? It is clear that if the edge variables are binary, singleton factors alone cannot model the interaction between edges (i, j) and (j, i). Therefore, we suggest enhancing a simple decomposable model with *pairwise factors* - factors over pairs of edges. Our proposed score function is:

$$score(y) = \sum_e w_s^\top \phi_e(y_e) + \sum_{i<j} w_p^\top \phi_{ij}(y_{ij}, y_{ji}) \tag{5.8}$$

$$= \sum_e \theta_e(y_e) + \sum_{i<j} \theta_{ij}(y_{ij}, y_{ji}) \tag{5.9}$$

The left addend is straightforward - w_s is a shared singleton weight vector, and $\phi_e(y_e)$ is some singleton feature vector for edge e. Combined, they form singleton potentials $\theta_e(y_e)$, each containing a value for $y_e = 0$ and for $y_e = 1$. The right addend is the interesting addition, and is defined as follows. Denote the number of pairwise features by d. Then, w_p is a shared pairwise weight vector of size $4d$; it contains d entries for each of the four assignments to (y_{ij}, y_{ji}). If ϕ_{ij} is a feature vector of size d for the pair of nodes i, j, then $\phi_{ij}(y_{ij}, y_{ji})$ returns a vector of size $4d$ with ϕ_{ij} in the entries corresponding to (y_{ij}, y_{ji}) as in w_p, and zeros elsewhere. Combined, they form a *pairwise potential* - a table of size 2×2, with a value for each assignment to (y_{ij}, y_{ji}). This model is known as a *pairwise MRF* (Kindermann et al., 1980), and is a generalization of the renowned Ising model (Ising, 1925).
By shifting mass from the pairwise potentials into the singleton potentials and adding a constant, a more compact model with the same maximizing assignment can be defined by:

$$score(y) = \sum_e \theta'_e y_e + \sum_{i<j} \theta'_{ij} y_{ij} y_{ji} \tag{5.10}$$

Recall that using dual decomposition, we need to be able to solve the following optimization problem:

$$\arg\max_{y_{ij}, y_{ji}} \theta_{ij}(y_{ij}, y_{ji}) - \delta_{ij}(y_{ij}) - \delta_{ji}(y_{ji}) \tag{5.11}$$

This is in fact easily achieved by enumeration - since the number of assignments is only four, we can compute the expression for each of them (by simply adding up the

relevant values in θ_{ij}, δ_{ij}, and δ_{ji}), and choose the maximizing assignment.

A contemporary method for *reciprocity prediction* can be found in Cheng et al. (2011).

2.2 Clustering

One of the most prominent attributes of social relations is that people tend to *cluster*. Social networks often display tightly-knit communities with many internal connections, which are sparsely connected to other communities (Holland and Leinhardt, 1971; Watts and Strogatz, 1998). Interestingly, this community structure appears at several scales (Ravasz and Barabási, 2003).
Many clustering indices try to capture the essence of this social tendency. Most of them are based on the notion of *closed triads* - triples of nodes with all three edges between them existent. The nodes of a closed triad are often of the same community, and communities in turn are constructed of many overlapping closed triads. In dynamic networks, *triadic closure* is the tendency of node triads to close (Granovetter, 1973). For instance, if edges (i, j) and (j, k) exist, it is somewhat likely that the edge (i, k) will eventually appear. Triadic closure is a mechanism which can lead to a high degree of local clustering in a network (see section 3.1 in Easley and Kleinberg, 2010).

If the task at hand is predicting new edges, we can express local clustering by using a pairwise model (5.9) and choosing as pairs all edge pairs $(i, k), (k, j)$ for observed edges (i, j):

$$score(y) = \sum_e \theta'_e y_e + \sum_{ij} \sum_k \theta'_{ijk} y_{ik} y_{kj} \tag{5.12}$$

In a decomposable model, triadic closure can be expressed by how likely a potential edge (i, j) is to close a triad with existing edges (i, k) and (k, j). This kind of reasoning is often used in many unsupervised scores (see chap. 3 sec. 1). In contrast, a pairwise model can express triadic closure even for triads with only *one* existing edge. This can be meaningful if we assume the network evolves over time and tends to triadic closure.

In section (5.11) we suggested solving a pairwise model with dual-decomposition by breaking up the problem into many small subproblems, each containing a single pair. Having many such pairs can negatively affect the convergence time and the tightness of the relaxation. Fortunately, under reasonable assumptions, all pair factors can be integrated into one single factor, whose induced maximization problem is tractable. Denote by P the set of edge pairs. If the pairwise potentials are *super-*

modular, then the pairwise term:

$$\arg\max_y \sum_{(e,f)\in P} (\theta_{ef}(y_e, y_f) - \delta_{ef\to e}(y_e) - \delta_{ef\to f}(y_f)) = \tag{5.13}$$
$$\arg\max_y \sum_{(e,f)\in P} \theta_{ef}(y_e, y_f) - \sum_e \sum_{f:(e,f)\in P} \delta_{ef\to e}(y_e)$$

can be solved with min-cut algorithms (Kolmogorov and Zabin, 2004). A pairwise potential is super-modular if its pairwise potentials satisfy $\theta_e(0,1) + \theta_e(1,0) \leq \theta_e(0,0) + \theta_e(1,1)$. This is a reasonable assumption in our case if we believe that triads containing a single edge or all three edges are more likely than triads containing two edges.

We can further expand our model to include factors over not only pairs, but *triples* of edges; if we choose triads as the triples, our model can genuinely express *clustering*, since triads are its building blocks. Also, we no longer need any edges in the triad to exist, therefore making triad closure applicable to link prediction settings where there are no observable edges in the target network. Factors over three (or more) variables can be solved in a bundle if they can be written as a linear multinomial of a *pseudo-boolean function* with positive high-order coefficients:

$$f(x_1 \dots x_n) = \sum_{c\subseteq[n]} \alpha_c \prod_{i\in c} x_i, \qquad \forall\, c : |c| > 1 \quad \alpha_c \geq 0 \tag{5.14}$$

It is well known that such functions can be maximized in polynomial time (Grötschel et al., 1981), and are in fact a generalization of concept of super-modularity presented above. For a survey on pseudo-boolean functions see Boros and Hammer (2002).

2.3 Small world

One of the most surprising traits of social networks is that they are in fact a *small world* - average distances between nodes in the network are surprisingly short. In his classic experiment, Milgram (1967) showed that Nebraskan farmers could deliver a package to an unknown, random stockbroker in Boston in as few as six first-name-basis hops on average. A contemporary large-scale replication of the small-world experiment was run by Dodds, Muhamad, and Watts (2003) on an e-mail network, giving similar results. Small world behavior has been empirically found in many real-world networks, and is the focus of many theoretical and applied research in the network sciences (see Kleinberg, 2000; Watts and Strogatz, 1998, for classic examples).

To express an incline towards relatively short shortest-path lengths, we suggest adding the following *short-path* global factor:

$$\theta_{SP}(y; G, P) = \beta \cdot \sum_{(s,t)\in P} L(y, s, t; G) \tag{5.15}$$

where G is the given network, P is a set of node pairs, $\beta \leq 0$ is a penalizing scalar weight (and a part of the weight vector w), and $L(y, s, t; G)$ is the length of the shortest-path between nodes s and t in a network obtained by adding edges y to the observed network G.
Intuitively, since $\beta \leq 0$, shortest-paths between pairs $(s, t) \in P$ are penalized proportianly to their length. Thus, an optimal assignment should contain relatively short shortest-paths between node pairs in P.

As before, in order to incorporate this structural factor into an MRF we apply dual decomposition. It is tempting to add θ_{SP} as a single factor to the MRF. However, maximizing the relevant optimization problem:

$$\arg\max_{y} \theta_{SP}(y; G, P) - \sum_{e} \delta_{SP \to e}(y_e) \tag{5.16}$$

is known to be NP-hard, since it is equivalent to the Generalized Network Steiner problem (Agrawal et al., 1995), a variant of the Steiner Tree problem, one of the seven problems shown to be NP-complete by Karp (1972). Therefore, we further decompose θ_{SP} into distinct global factors $\theta^{st}_{SP}(y) = \beta L(y, s, t; G)$ for each node pair $(s, t) \in P$. This induces the following optimization problems:

$$\arg\max_{y} \theta^{st}_{SP}(y) - \sum_{e} \delta^{st}_{e}(y_e) \qquad \forall\, (s, t) \in P \tag{5.17}$$

Solving such combinatorial optimization problems is nontrivial. We propose a polynomial algorithm for cases where $\beta \leq 0$ (when $\beta > 0$ the problem can be shown to be NP-hard by a simple reduction from the Longest Path problem).

Solving the short-paths factor

Consider a pair of nodes $(s, t) \in P$. First, we shift the weights of the δ-s as in (4.2) and define $W_e = \delta^{st}_{SP \to e}(0) - \delta^{st}_{SP \to e}(1)$, attaining:

$$\arg\max_{y} \sum_{e} W_e y_e + \beta L(y, s, t; G) \tag{5.18}$$

Under this formulation, the problem can be described as follows. We are given a graph $G' = (V, E \cup E')$, source and target nodes s, t, and and edge weight function $W : E \to \mathbb{R}$. Edges in E' are fixed, and our goal is to choose a subset of edges from E which maximize the expression in (5.18), where the y_e-s are indicator variables for edges $e \in E$.
It is well known that the shortest-path problem $L(y, s, t)$ with positive weights $-\beta$ can

be solved exactly by the following linear program form of a a network flow problem:

$$\begin{aligned} \min_{z} \quad & -\beta \sum_{ij} z_{ij} \\ \text{s.t.} \quad \forall i \quad & \sum_{j} z_{ij} - \sum_{j} z_{ji} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = t \\ 0 & \text{o.w.} \end{cases} \\ & z \geq 0 \end{aligned} \tag{5.19}$$

where the variables z are directional indicators on the edges used in the shortest path, and are constrained to enforce a legal flow. Now we can rewrite our original problem as:

$$\begin{aligned} \arg\max_{y,z} \quad & \sum_{ij} w_{ij} y_{ij} + \beta \cdot \sum_{ij} z_{ij} \\ \text{s.t.} \quad \forall i \quad & \sum_{j} z_{ij} - \sum_{j} z_{ji} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = t \\ 0 & \text{o.w.} \end{cases} \end{aligned} \tag{5.20}$$

$$\begin{aligned} & z \leq y \\ & z \geq 0 \end{aligned} \tag{5.21}$$

where the additional constraint $z \leq y$ allows the path fragment z_{ij} to appear only if its corresponding edge y_{ij} does (this is a fractional extension to the integral case).

Let y^* be an optimal assignment to our problem. Consider variables y_e with $w_e > 0$. These variables *must* be on in y^*, since by choosing them we can only gain (both by adding w_e to our score and by potentially shortetning the shortest path), and by not choosing them we can only lose (by the same logic).
Therefore, we are left with the task of choosing the participating variables y_e with $w_e < 0$. Notice that such variables y_e will be chosen only if z_e is also chosen; choosing only y_e will result in losing $|w_e|$, without any benefit the cost of the shortest path. Since y_e and z_e both refer to the same edge, we may view the negative w_e as a penalty on choosing z_e (for the positive w_{ij}-s there is no punishment, since we choose their corresponding ys anyway). Now our problem is reduced to finding the shortest weighted path in a graph with weights u such that:

$$u_e = -\left(\min(0, w_e) + \beta\right) \tag{5.22}$$

Now, the $\arg\max$ of our original problem is the union of the y_e-s with positive w_e and the y_e-s that correspond to the shortest path found in the graph weighted by u. Note that in this formulation it is easy to incorporate separate penalty terms β_s^{st} for each pair (s, t) and edge e, instead of a single global β.

2.4 Scale free

Considered by some to be the 'signature trait' of complex networks, scale free degree distributions appear in a multitude of natural and engineered networks. Pioneering work by Albert and Barabási (2002) led to a multitude of research on the mechanisms causing this emergent phenomenon, as well as justification for its abundance (for instance Barabási and Albert, 1999; Mitzenmacher, 2004; D'Souza et al., 2007; Clauset et al., 2009). Networks with heavily skewed degree distributions have been found to be robust, versatile, and scalable, and are in some sense optimal for communications and collective problem solving.

Scale free distributions are only a subclass of *positively skewed* distributions - distribution whose skewness (the normalized third central moment) is positive. Intuitively, in a positively skewed distribution most of the mass is concentrated in a long, heavy right tail, in which the mean resides. Scale free distributions are extremely right skewed, and most real world degree distributions are assumed to theoretically have infinite skewness (since the scaling exponent α is usually between 2 and 3, see Albert and Barabási (2002)). However, there are many other distribution classes which display extreme positive skewness. In fact, some researchers claim that the degree distribution of real world networks is not scale free but rather *log-normal*, which can also be extremely positively skewed (Pennock et al., 2002; Mitzenmacher, 2004). Moreover, small networks do not necessarily display a scale free degree distribution, which is considered an asymptotic trait. However, small networks' degree distributions are almost always heavily skewed. Hence, we suggest a way of expressing a tendency towards positive *skewness* of the degree distribution.

The skewness of a sample is a non-linear function, and is non trivial to optimize over. A much simpler proxy to the actual skewness is *Cyhelsk's skewness coefficient*, given by:

$$\text{skew}(x) = \frac{|\{i : x_i < \bar{x}\}| - |\{i : x_i \geq \bar{x}\}|}{n} \tag{5.23}$$

for a sample set $x = x_1, \ldots, x_n$ whose average is $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. In the context of degree distributions, a high skewness coefficient would imply that the degree of most nodes is lower than the average degree. Since in a given network the number of nodes is constant, encouraging nodes to have a degree lower than average should yield a positively skewed degree distribution. Therefore, we propose to add the following factors:

$$\theta_D(y, \eta) = \begin{cases} 0 & \text{if} \quad 2\sum_{ij} y_{ij} = \eta \\ -\infty & \text{o.w.} \end{cases} \tag{5.24}$$

$$\theta^i_{SK}(y_{i,\Gamma(i)}, \eta) = \begin{cases} \gamma & \text{if} \quad \sum_{j\in\Gamma(i)} y_{ij} < \frac{\eta}{N} \\ 0 & \text{o.w.} \end{cases} \tag{5.25}$$

where $N = |V|$, $\Gamma(i)$ denotes the set of i's neighbors and $y_{i,\Gamma(i)} = \{y_{ij} : j \in \Gamma(i)\}$. A supplementary variable η is introduced - it is forced by θ_D to hold the sum of all degrees in the network, thus allowing to compare the degree d_i of any node i to the average degree η/N. Note that since $\sum d_i = 2\sum y_{ij} \leq 2n$, η takes values in $0, 2, 4, \ldots, 2n$. For $\gamma > 0$, the θ_{SK} factors encourage nodes to have a degree lower than the average, thus providing a higher score for networks with a positive skewness coefficient. Having γ as a part of the model's weight vector allows us to learn the amount of skewness in the training set.

We focus on solving the induced optimization problems:

$$\arg\max_{y,\eta} \theta_D(y,\eta) - \sum_{ij} \delta_{D\to ij}(y_{ij}) - \delta_{D\to\eta}(\eta) \tag{5.26}$$

$$\arg\max_{y_{i,\Gamma(i)},\eta} \theta^i_{SK}(y_{i,\Gamma(i)},\eta) - \sum_{j\in\Gamma(i)} \delta_{SK\to ij}(y_{ij}) - \delta_{SK\to\eta}(\eta) \tag{5.27}$$

We first resolve to solving (5.26). As before, assume we have applied the shifting trick (4.2) to the $\delta(y_{ij})$-s. Notice that for every value of k that η can take, exactly $k/2$ y_{ij}-s must be on in order to prevent the $-\infty$ penalty in θ_D. Hence, the optimal assignment would be to turn on the $k/2$ y_{ij}-s with the highest $\delta_{ij}(y_{ij})$ values, adding their score to $\delta_\eta(k)$ to compute the objective. The overall optimal assignment can be found by repeating the above procedure for all legal assignments of k, and selecting the overall best. This can be done efficiently by first sorting the y_{ij}-s according to $\delta_{ij}(y_{ij})$ and keeping a cumulative sum over iterations. Thus, the cost of solving this problem is $O(n + |\Gamma(i)| \log |\Gamma(i)|)$, where n is the total number of variables/potential edges.

To solve (5.27), we notice two things. First, for any fixed value of η, θ_{SK} depends only on the *number* of y_{ij}-s which are on. Hence, if k y_{ij}-s are on, it is always better to select those with the highest δ scores. Second, for any fixed number k of y_{ij}-s which are on, θ_{SK} scores γ if $\eta > Nk$, and 0 otherwise. Hence, the optimal η is either the one with the highest δ value for which $\eta > Nk$, or the one with highest δ value for which $\eta \leq Nk$, depending on the value of γ. This gives rise to the following algorithm. First, we sort the y_{ij} variables by their δ value, and gradually turn them on. For every k which are on, we compute the objective for both of the above η candidates, and select the one which results in a better objective. As an assignment, we choose the k which attained the overall best score, and assign y and η accordingly. A simple precomputation can significantly reduce the cost of finding the η candidates. We create a table T with an entry for each value of η. Then, we go over $\delta_\eta(k)$ for all $k = 0, 2, \ldots, 2n$, and store in $T(k)$ the index $k' \leq k$ for which $\delta_\eta(k')$ has been highest so far. This procedure allows us to query the optimal $\eta \leq j$ for any j in $O(1)$. The same can be done for $\eta > j$ by going over all k-s from $2n$ down to 0. With these precomputational steps, the cost of solving this problem is $O(n)$.

3. Multi-variate structured loss functions

In binary classification, the standard loss for a prediction y over the true label y^m is the *0/1 loss*, defined by:

$$\ell_{0/1} = \mathbb{1}_{\{y \neq y^m\}} \tag{5.28}$$

The natural extension of the 0/1 loss for binary *vector* labels is the *hamming distance*:

$$\ell_{hamm}(y, y^m) = \sum_i \mathbb{1}_{\{y_i \neq y_i^m\}} \tag{5.29}$$

and is an appropriate loss for the accuracy performance measure (chap. 2 sec. 3.1). The hamming loss is *simple* in the sense that it decomposes over the elements of the label vector, therefore having no *structure*. However, other performance measure call for appropriate multivariate losses that *do* contain structure, such as the recall@k, precision@k, F_1, and the AUC (see chap. 2 sec. 3).

Multi-variate losses have been attended to in the literature. Specifically, Joachims (2005) has provided an SVM variant which can learn decomposable models over the aforementioned measures. However, his method does not apply to structured models containing high order factors. In this section, we show how to incorporate a structured loss function into a structured learning framework. We solve the optimization problems given by the learning objective (chap. 4 eq. 4.6):

$$\arg\max_y \; w^\top \phi(x^m, y) + \Delta(y, y^m) - w^\top \phi(x^m, y^m) \tag{5.30}$$

by treating the label loss $\Delta(y, y^m)$ as a high-order factor $\theta_\Delta(y; y^m)$, and applying dual decomposition. Therefore, we focus on solving:

$$\arg\max_y \; \theta_\Delta(y; y^m) - \sum_i \delta_{\Delta \to i}(y_i) \tag{5.31}$$

for the various losses previously discussed. As before, we assume the 'shifting trick' is applied (4.2).

3.1 Recall and precision

When using the recall or precision performance measures, it is common practice to constrain the prediction to have exactly k variables on for some predefined parameter k. For a binary prediction vector y and a true label y^m, the corresponding losses in this case are defined by:

$$\ell_{recall@k}(y, y^m) = 1 - \frac{1}{k^m} \sum_{i: y_i^m = 1} y_i \tag{5.32}$$

$$\ell_{prec@k}(y, y^m) = 1 - \frac{1}{k} \sum_{i: y_i = 1} y_i^m \tag{5.33}$$

where $k^m = |\{i : y_i^m = 1\}|$. For these losses, structure comes in the form of constraining y to have exactly k entries which are on (other than this constraint, the loss decomposes). Integrating these losses into the relevant optimization problem given in (5.31) and rearranging, we get:

$$\arg\max_y \theta_{on}(y;k) - \sum_i \left(\delta_i + \frac{y_i^m}{k'}\right) y_i \tag{5.34}$$

where θ_{on} is a cardinality potential (that is, $\theta_{on}(y;k) = 0$ if y has exactly k entries which are on, and $-\infty$ otherwise). The expression for precision is similar.
Denote $u_i = (\delta_i' + y_i^m/k')$. Problem (5.34) can be solved in time $O(n \log n)$ by sorting the values u_i in descending order and choosing the top k indices i to be on in y.

3.2 $\mathbf{F_1}$

The F_1 loss is defined by:

$$\ell_{F_1}(y, y^m) = 1 - F_1(y, y^m) \tag{5.35}$$

(see chap. 2 eq. 2.5 for the definition of the F_1 function). The F_1 score is a more complex measure, and no longer seems to decompose, even under constraints. However, we can use the same technique given by Joachims (2005), replacing the model's score with $\delta_i y_i$.

The algorithm Joachims (2005) suggests iterates over all possible combinations of the contingency table of true positives, false positives, true negatives and false negatives. At each iteration, it computes the maximizing assignment under the constraints given by the current contingency table, keeping record of past maximizing assingment and returning the overall best. Under *these* constraints, the F_1 loss *does* decompose. Since these constraints are simply cardinality constraints, each maximizing assignment can be solved by sorting the variable weights as in recall@k and precision@k.

3.3 AUC

As opposed to the previous error measures, the AUC is a function of a *binay* vector label, but a predicted *ranking*. Therefore, it is not even clear how a *model* can be defined for learning over this measure, let alone a structured one.
Therefore, the AUC loss requires special attention. Since it is by far the most common preformance measure for link prediction, this attention is justified. In the next chapter we elaborate on how to learn a structured model over the AUC loss (which entails structure in itself). We will see that by *duplicating* a standard MRF and imposing some constraints, the AUC loss *can* decompose over the model's variables, and can be optimzed efficiently as a structured factor.

Chapter 6

The AUC Loss

In the task of binary classification, a predictor must label instances as belonging to one of two classes. However, the prediction rule often relies on the relation of a sample's score to an intrinsic discriminating threshold. For example, in linear classifiers, the prediction rule is set to:

$$\text{sign}\langle w, x\rangle - b \tag{6.1}$$

where $\langle w, x\rangle$ is the score and b is the learned threshold. However, in some settings the threshold is inconsistent between samples, or is bound to be controlled by an end user. In others, scores are used to construct an implicit ranking of the samples, thus discarding the threshold altogether. In these settings, standard measures of binary outputs are inadequate, and other measures must be used.
The *AUC* is such a measure, and is popular in many learning tasks, and in particular in link prediction. In this chapter we present the AUC measure and AUC loss, and suggest a novel framework for extending a given MRF so that learning over it can be tuned to optimize the AUC.

In this chapter we use the following notation. Consider a sample set $\{(x^m, y^m)\}_{m=1}^M$. In univariate cases, where labels y^m are binary, we denote by *pos* the set of samples whose labels are *positive* or *true* (namely $y^m = 1$), and by *neg* those who are *negative* or *false* ($y^m = 0$). Accordingly, we denote $P = |pos|$ and $N = |neg|$. Similarly, in multivariate cases, the labels y^m are binary vectors of size n_m. For each sample m, we will define $pos_m = \{i : y_i^m = 1\}$ and $neg_m = \{i : y_i^m = 0\}$, and hence $P_m = |pos_m|$, $N_m = |neg_m|$. When a single sample is discussed, we will drop the m sub/superscript. Sometimes we will relate to all labels as a single binary vector $y' = \left(y^1, \ldots, y^M\right)$. When discussing ranking models, we will use π to denote a ranking over a set of variables, and z to denote a nested ranking matrix. As a convention we state that i is ranked higher than j is $\pi(i) < \pi(j)$.

1. The ROC plot and the AUC measure

The *ROC* (receiver operating characteristic) curve (Metz, 1978) is a plot used for depicting the tradeoff between the false positive rate (FPR) and the true positive rate (TPR) of binary classifiers. The ROC is produced by varying the discriminating threshold of a classifier over a plot with FPR and TPR on its x and y axes, respectfully. This can be thought of as gradually 'turning on' variables according to their *rank*, (implicitly) given by the classifier. If the variable which has just been turned on corresponds to a true label, the TPR increases; if the corresponding label is false, the FPR increases. Therefore, the ROC is a monotonically (weakly) increasing step-plot. While the ROC plot is helpful in analyzing the performance of a predictor, it is not a performance measure per se, since it cannot be used to directly compare two or more predictors. A popular method for aggregating the information given in the ROC into a single measure is integrating over the plot, or more simply put, calculating the area under it. The area under the ROC, simply known as the *AUC*, is a common performance measure of binary classifiers and ranking predictors (Provost et al., 1997; Fawcett, 2006). It is widely used in many machine learning settings, and is by far the most common measure in link prediction.

Since the ROC is normalized, the AUC is always in the range $[0, 1]$, with higher AUC corresponding to better performance. A classifier which ranks all true labels higher than false labels gets an AUC of 1; a classifier ranking all false labels higher than true labels gets an AUC of 0. The expected AUC of a random classifier is 0.5.

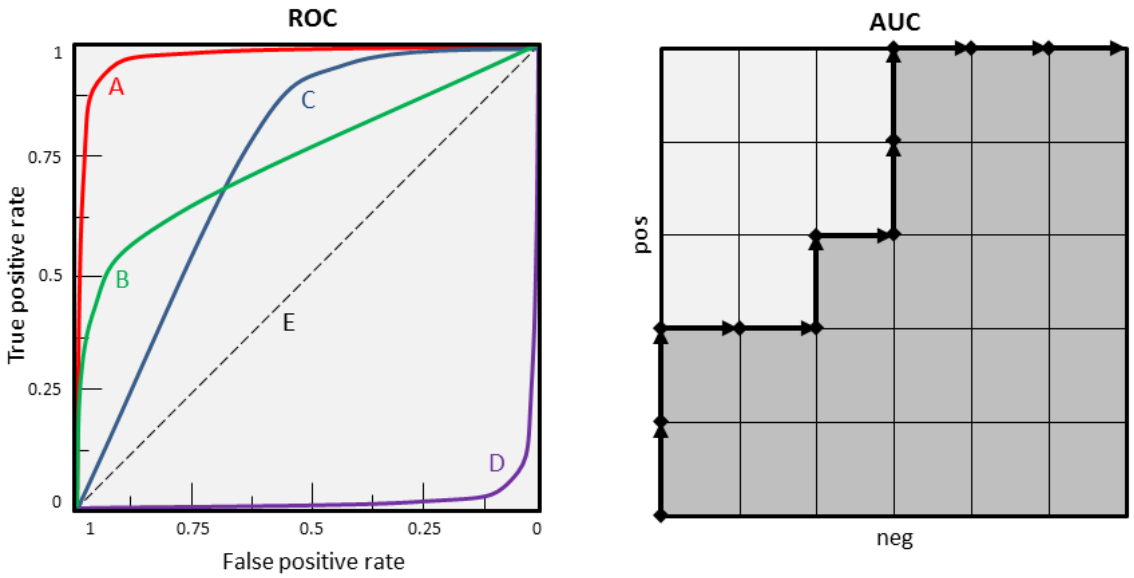

Figure 6.1: **Left:** An ROC plot for 5 predictors, with very high AUC (A), high TPR (B), high FPR (C), low AUC (D), and random (E). **Right:** An AUC plot for the ordering $\pi = 3, 6, 10, 5, 1, 8, 9, 7, 2, 11, 4$ over the label $y_m = 10100110100$, giving an AUC value of 0.733.

2. AUC, learning, and link prediction

In the link prediction task, it often makes sense to predict a *ranking* over potential edges. This is true for various reasons. First, in most realistic settings, end-users prefer a ranking of potential edges over a binary prediction. This allows *them* to decide how many edges will augment the observed network by taking the k top-ranked edges. Furthermore, since the true missing edges constitute only a small fraction of the potential edges (or in other words, labels are grossly imbalanced), binary predictions may be too harsh to be useful. In contrast, a ranking can be 'good' just by placing the true missing edges at high enough spots. Finally, the performance of supervised link prediction methods are usually compared to unsupervised baselines. Since these unsupervised methods output a ranking, for proper comparison it is only reasonable that supervised methods do the same.
While these arguments imply that the output of link predictors should be a ranking of the potential edges, the ground truth is often *binary* - edges are either truly missing, or they are not. The AUC is a common measure for exactly this setting - predicted *rankings* over *binary* ground truths. Hence, it is a fitting measure for link prediction, and is therefore popular in the literature.

Nevertheless, in almost all supervised link prediction works, learning is performed by optimizing over the *error rate*, a loss fit for the performance measure of *accuracy* (chap. 2 sec. 3.1). Since the AUC is the measure used to evaluate performance in practice, it only makes sense to optimize it directly. To do this, two things should be taken into account. First, the model used should preferably be a *ranking model*, in the sense that its output is a ranking over all variables (as opposed to a ranking based on independent variable scores, see Joachims (2002)). Using the AUC as a performance measure for binary classifiers has recently been criticized in the literature (Lobo et al., 2007; Hand, 2009). However, as a performance measure of *rankings*, it is most adequate. This implies that dependencies between variables should be taken into account. Second, the learning objective should be directly related to the AUC. Minimizing the *AUC loss*, namely $\Delta_{\text{AUC}} = 1 - \text{AUC}$, achieves just this (Joachims, 2005). The only link prediction paper we are aware of in which the AUC is directly minimized is Menon and Elkan (2011).

In most supervised link prediction methods, local features are integrated into the model. Since each such feature conveys information regarding only a single edge, such models cannot express the edge interdependence required for ranking. Joachims (2005) suggests an elegant way of optimizing such models over the AUC loss. The original problem is replaced by a classification problem over pairs $(x_i, x_j) \in pos \times neg$ (that is, $y'_i = 1, y'_j = 0$). Based on the intuition that positive samples should be ranked higher than negative samples, the features in the new problem are defined by $w^\top (\phi(x_i) - \phi(x_j))$, and the corresponding label is $y'_{ij} = 1$. This is a sufficient representation since the AUC is indifferent to all other pairs (namely $pos \times pos$, $neg \times neg$,

and $neg \times pos$). A linear predictor is then implicitly learned over this new extended sample space. In practice, the learning algorithm utilizes the nested structure of the AUC loss to solve a problem over a compact, linear representation of the (otherwise quadratic) $pos \times neg$ space.

In the above formulation, a ranking is predicted for given binary ground truths, as desired for the link prediction setting. However, even though the AUC loss can express multivariate structure, the model itself does not; features are still defined over single edges. While technically this method can be applied to multivariate examples, it does not allow for complex structured features.
In our work, we propose a framework for optimizing complex *structured* models directly over the AUC loss. We do this by modeling *all* pairs of samples, and constraining the output to represent a ranking. Moreover, our model allows for the integration of local features found in unsupervised methods with high-order features, such as those proposed in the previous chapter or in the generative link prediction literature. Hence, our learning framework is designed for learning structural link prediction models, optimized for high AUC.

In the next section, we describe the general proposed model. Then, we further analyze the AUC loss and show how to solve the induced optimization problem both by dual decomposition and LP relaxations. Finally, we elaborate on extensions to the standard AUC loss easily attained in our model.

3. Structured learning with the AUC loss

The solution proposed by Joachims (2005) is fit for *decomposable models*; it relies heavily on the fact that the score can be written as a sum over local singleton scores. When this is the case, in the extended learning problem over $pos \times neg$ pairs, the AUC decomposes as well. Furthermore, the loss is independent of w, and can be separated from the model, allowing for consistency in predictions between train and test times. Finally, the resulting learning problem is convex, which is a desirable trait.
Unfortunately, these properties do not necessarily hold when the score does *not* decompose, as in the case of structured models. It is not clear how to naturally generalize the score of a sample pair in a structured model, whether consistency can be preserved, or whether learning can still be tractable.

Recall that the AUC loss is a function of a predicted ranking π and a binary vector label y' of size n. We denote this by $\Delta_{\text{AUC}}(\pi, y')$. Assume we are given a structured MRF in the form of $w^\top \phi(x, y)$. Our objective is to learn a weight vector w which minimizes the expected AUC loss over a training set. However, it is not clear how this can be done, since the model is over binary vectors y, while the loss is over a ranking π.

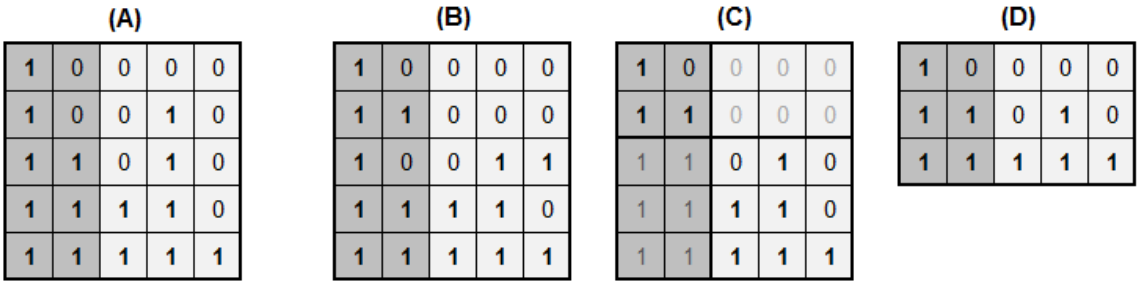

Figure 6.2: **(A)** A nested ranking matrix z for the ranking $\pi = 1, 3, 4, 2, 5$ over the label $y_m = 11000$; **(B)** A ranking matrix with no nestedness constraints; **(C)** A ranking matrix consistent with y_m; **(D)** A binned nested ranking matrix for bins $\{1, \{2, 3\}.\{4, 5\}\}$.

Earlier we claimed that the ROC can be attained by gradually turning on variables according to their score. However, in a structured model it is not clear how this can be done, since variables are dependent, and do not necessarily have a score of their own. Nonetheless, the above notion can be extended in the following manner: instead of gradually turning on variables, for every rank $k = 1 \dots n$ we can output a prediction y constrained to have exactly k entries which are on. This gives us a series of predictions, each for every rank. In order to preserve consistency between the different predictions, we constrain a prediction of rank $k+1$ to include the prediction of rank k. In other words, previous ranks are *nested* in the current rank.
Combined, these constraints result in a natural ordering over variables - the k^{th} variable in the ranking is the one added in rank k. However, since these constraints create dependency both within and between ranks, it is required that all variables of all ranks be optimized simultaneously.

In order to do this, we create n independent copies of the original model, one for each rank k. Note that w is shared across all duplicates. Each model of rank k has a duplicate set of variables $y^{(k)}$. We can think of the set of variables $y^{(1)}, \dots, y^{(n)}$ as a binary matrix z, where $z_{ki} = y_i^{(k)}$. In other words, each row k of the matrix z holds the n variables of the rank k model (see figure 6.2). Due to independency, the model's corresponding score can now be written as:

$$score(z) = w^\top \phi(x, z) = w^\top \left(\sum_{k=1}^{n} \phi(x, z_{k\cdot}) \right) \tag{6.2}$$

To each model of rank k we add a *cardinality constraint*, forcing its output to have exactly k entries on. This is expressed by the row constraints:

$$\sum_i z_{ki} = k \qquad \forall\, k \in [n] \tag{6.3}$$

Finally, we add consistency constraints between consecutive models - if i is turned on at rank k, it should also be on in all ranks $k' > k$. In other words, the columns of z

should be nested, which can be expressed by:

$$z_{ki} \leq z_{k+1,i} \qquad \forall\, i \in [n], k \in [n-1] \tag{6.4}$$

Given these constraints, z is now a *nested ranking matrix*, and can easily be mapped to and from a ranking π. Therefore, the AUC loss can be expressed as function of z, just as the duplicated model. Finally, each hinge-loss term in the structured learning objective (chap. 4 eq. 4.6) can be written as:

$$H^m(z) = w^\top \phi(x^m, z) + \Delta_{\text{AUC}}(z, y^m) - w^\top \phi(x^m, z^m) \tag{6.5}$$

As before, the prediction rule is set by $\arg\max_{z \in R} w^\top \phi(x^m, z)$, which is consistent with the learning objective. Moreover, it is a generalization of Joachims (2005) in the sense that for a given decomposable model, both methods produce the same predicted ranking.

However, the above formulation is a bit misleading, since it assumes the existence of a ranking label z^m, which is usually not the case in link prediction and in numerous other tasks. To overcome this difficulty, we will replace the unattainable 'true' ranking z^m with a ranking z' *consistent* with y^m, in the sense that if $y^m_i = 1$ and $y^m_j = 0$ then i is ranked higher than j in z'. In other words, the full ranking in z' is consistent with the partial ranking induced by y^m. We denote this consistency by $z' \sim y^m$.
Notice that $\arg\max H^m(z)$ is an upper bound on Δ_{AUC} for *any* legal $z' \sim y^m$. However, using just an arbitrary such z' inserts redundant noise into the system. As is often done with *latent variables*, the model itself can be used to better select z'. Since the hinge loss term is derived for the general constraint:

$$w^\top \phi(x^m, z') > w^\top \phi(x^m, z) + \Delta(z, y^m) \qquad \forall\, z \neq z' \tag{6.6}$$

we will consider this notion in our selection of z'.

If we require that *all* ranking which are consistent with y^m have higher scores than all non-consistent ones, the hinge loss term becomes:

$$\begin{aligned} &\max_{z \in R} w^\top \phi(x^m, z) + \Delta_{\text{AUC}}(z, y^m) - \min_{z' \sim y^m} w^\top \phi(x^m, z') \\ &= \max_{\substack{z \in R \\ z' \sim y^m}} w^\top \left(\phi(x^m, z) - \phi(x^m, z')\right) + \Delta_{\text{AUC}}(z, y^m) \end{aligned} \tag{6.7}$$

where R is the set of nested ranking matrices. Since this term is a piecewise linear convex function, the resulting objective is convex, which is a desirable trait since learning can be done using standard convex optimization techniques. However, it constitutes a loose bound on Δ_{AUC} and may impose too harsh a demand on learning.

In the other extreme, if we require that there *exists* a consistent ordering which scores higher that all non-consistent orderings, the hinge term becomes:

$$\max_{z \in R} w^\top \phi(x^m, z) + \Delta_{\text{AUC}}(z, y^m) - \max_{z' \sim y^m} w^\top \phi(x^m, z') \tag{6.8}$$

This results in a tight bound on Δ_{AUC} (since 0 loss can be attained), and is often used in settings with latent variables (in fact, the inner rankings of *pos* and *neg* can be viewed as such). Unfortunately, the induced objective is no longer convex. However, since it is a sum of a convex function and a concave function, optimization methods such as DCA (Tao and Souad, 1986; Tao and An, 1997) can be applied.
As a middle-ground between the previous suggestions, we can require our model to be 'good on average' in the sense that the *average* consistent ranking be better than all inconsistent rankings, giving:

$$\max_{z \in R} w^\top \phi(x^m, z) + \Delta_{\text{AUC}}(z, y^m) - \operatorname*{avg}_{z' \sim y^m} w^\top \phi(x^m, z') \tag{6.9}$$

This gives a loss term which is convex, but not as harsh as the $-\min$ variant. However it is not tight, and may not perform well in settings where averaging is too noisy.

The above variants lead to an interesting tradeoff between the tightness of the bound on the AUC loss and the convexity of the learning problem. In the following section, we propose solutions for the optimization problems induced by the above objectives. For clarity we focus on the variant including an arbitrary $z' \sim y^m$, though the solutions are fit for all of the above.

4. Optimization

Recall that in order to learn our proposed structured model, the following optimization problem needs to be solved:

$$\arg\max_{z \in R} \left\{ w^\top \phi(x^m, z) + \Delta_{\text{AUC}}(z, y^m) - w^\top \phi(x^m, z') \right\} \tag{6.10}$$

Notice that $w^\top \phi(x^m, z')$ is constant, and that Δ_{AUC} can be incorporated into the model and considered as a high order factor, independent of w. Hence, we may view the above problem as an inference task in a structured ranking MRF.
Inference in structured ranking models is in general a hard task. For instance, Mensink et al. (2011) have shown that inference in a pairwise ranking model can be reduced to a *quadratic linear assignment* problem, which is NP-hard (they show however that star configurations are tractable). One way of dealing with such hard problems is by *relaxation*. In chapter 4 we have proposed dual decomposition (sec. 4) and linear programming (sec. 4.3) as two relaxations which are useful in structured learning. We now show how (6.10) can be solved within these relaxations.

4.1 Dual decomposition

Recall that in dual decomposition, in order for learning to be tractable, the following sub-problems must be solved efficiently:

$$\max_{x_f} \theta_f(x_f) - \sum_{i \in f} \delta_{fi}(x_i) \tag{6.11}$$

In the following, we assume the shifting trick (4.2) has been applied. Disregarding other factors in the model, we focus on solving the above for the AUC loss factor θ_Δ and the ranking constraints factor θ_R, given by:

$$\begin{aligned} &\theta_\Delta(z; y^m) = \Delta_{\text{AUC}}(z, y^m) = 1 - \text{AUC}(z, y^m) \\ &\theta_R(z) = \begin{cases} 0 & \text{if } z \in R \\ -\infty & \text{o.w.} \end{cases} \end{aligned} \tag{6.12}$$

As we shall see, θ_Δ can be incorporated either into θ_R or into the other factors.

Ranking constraints

Consider first the ranking constraints. It is clear that the maximizing assignment of the corresponding subproblem is a nested ranking matrix (since other binary matrices get $-\infty$). Denote by D the matrix holding values of $-\delta$, then we can rewrite the problem as:

$$\begin{aligned} &\max_{z \in \{0,1\}^{n \times n}} \theta_R(z) - \sum_{ki} \delta_{ki}(z_{ki}) = \max_{z \in R} \sum_{ki} D_{ki} z_{ki} = \\ &\max_{z \in R} \sum_i \sum_{k \geq \pi_z(i)} D_{ki} = \max_\pi \sum_i D'_{\pi(i),i} \end{aligned} \tag{6.13}$$

where π_z is the ranking corresponding to z and D' is a matrix containing the bottom-up cumulative sums of columns of D. In this formulation, the optimization problem can be thought of as a maximal weighted bi-partite matching problem, where the weight of an edge (i, j) is given by D'_{ij}. This optimization problem, called the *linear assignment problem*, can be solved in $O(n^3)$ by a modified version of Dijkstra's shortest paths algorithm, or by the dedicated Hungarian Algorithm (see Chapelle et al., 2007, for the assignment problem in ranking algorithms).

Though tractable, the cubic complexity of the above algorithms can be significant for large n. A possible solution is to break the constraint factor into n cardinality factors over the rows of z and n nestedness factors over the columns of z. Cardinality factors can be maximized in $O(n \log n)$ by sorting by δ and turning on the

top k variables, and nestedness factors can be solved in $O(n)$ simply by going over all n possible nested assignments. This results in a tradeoff between the computational complexity of a single subproblem and the convergence rate of the whole dual decomposition minimization procedure, since having more factors usually leads to a lengthier convergence process.

The AUC loss

Consider now the AUC loss factor. Let π be a given ranking whose AUC we wish to compute over a binary vector label y^m of size n. It is well known that the AUC is proportional to the number of unswapped $pos \times neg$ pairs:

$$\mathrm{AUC}(\pi, y) = \frac{1}{PN} \left| \left\{ (i,j) : \pi(i) < \pi(j), i \in pos, j \in neg \right\} \right| \tag{6.14}$$

While the AUC does not seem to decompose over π, we show that it *does* decompose over nested ranking matrices z, which are the model's variables. For this, we denote by r the inverse ranking of π, namely $r_i = n - \pi(i) + 1$. For a given $i \in pos = \{p_1, \ldots, p_l\}$, its contribution to the number of unswapped pairs is exactly the number of $j \in neg$ which are ranked lower. The AUC is attained by summing over all such $i \in pos$:

$$\begin{aligned} \mathrm{AUC}(r, y) &= \frac{1}{PN} \sum_{i \in pos} \left| \left\{ j \in neg : r_j < r_i \right\} \right| \\ &= \frac{1}{PN} \sum_{i \in pos} \left| \left\{ j : r_j \le r_i \right\} \right| - \left| \left\{ j \in pos : r_j \le r_i \right\} \right| \\ &= \frac{1}{PN} \sum_{i=1}^{P} (r_{p_i} - i) = \frac{1}{PN} \sum_{i \in pos} r_i - \frac{N+1}{2P} \end{aligned} \tag{6.15}$$

Next, we define a binary matrix $A \in \{0,1\}^{n \times n}$ where $A_{ki} = y^m_i$. A is attained by concatenating n duplicates of ${y^m}^\top$. Given the nested ranking matrix z corresponding to π, since both z and A are binary, we can write:

$$\sum_{ki} A_{ki} z_{ki} = \sum_{i \in pos} \sum_{k \le r_i} 1 = \sum_{i \in pos} r_i \tag{6.16}$$

and therefore the AUC loss is given by:

$$\Delta_{\mathrm{AUC}}(z, y^m) = 1 - \frac{1}{PN} \sum_{ki} A_{ki} z_{ki} + \frac{N+1}{2P} \tag{6.17}$$

Hence, in this formulation, the AUC loss decomposes over z. This allows us to absorb the loss penalties A_{ki} into the singleton factors. However, this is highly undesired

since the maximizing assignment will not necessarily encode a nested ranking matrix, deeming the above equations incongruent. Instead, by merging the loss and constraint factors into a single factor $\theta_{\Delta+R}$, z is forced to be a nested ranking matrix. Hence:

$$\begin{aligned}
&\max_{z\in\{0,1\}^{n\times n}} \theta_{\Delta+R}(z,y) - \sum_{ki} \delta_i^k(z_{ki}) = \\
&\max_{z\in R} 1 - \sum_{ki} (\frac{A_{ki}}{PN} - D_{ki}) z_{ki} + \frac{N+1}{2P} = \\
&\max_{\pi} \sum_i A''_{\pi(i),i} + c
\end{aligned} \tag{6.18}$$

where $A'' = \left(D' - \frac{A'}{PN}\right)$, A' is a matrix containing the bottom-up cumulative sums of columns of A, and $c = 1 + \frac{N+1}{2P}$ is constant. Once more, the above is in fact an assignment problem with weights A''_{ki}, and therefore can be be solved efficiently. If the constraint factor is broken up into row and column factors, the loss factor can be incorporated into the factors of either rows or columns.

4.2 LP relaxation

The dual decomposition analysis points out that the main computational difficulty lies in the ranking constraints. Since these constraints are *linear*, and since Δ_{AUC} decomposes, an LP relaxation naturally arises:

$$\begin{aligned}
\max_{\mu\in M_L} \quad & \sum_{ki}\sum_{z_{ki}} \mu_i^k(z_{ki})\theta_i(z_{ki}) + \sum_{kf}\sum_{z_{kf}} \mu_f^k(z_{kf})\theta_f(z_{kf}) + \Delta_{\text{AUC}}(\mu, y^m) \\
\text{s.t.} \quad & \sum_i z_{ki} = k \qquad \forall\, k \in [n] \\
& z_{ki} \le z_{k+1,i} \qquad \forall\, i = 1 \in [n], k \in [n-1]
\end{aligned} \tag{6.19}$$

where:

$$\Delta_{\text{AUC}}(\mu, y^m) = 1 - \frac{1}{PN}\sum_{ki}\sum_{z_{ki}} \mu_i^k(z_{ki}) A_{ki} z_{ki} + \frac{N+1}{2P} \tag{6.20}$$

Since the nested ranking constraints are enforced globally, and since Δ_{AUC} decomposes over z, it can now be safely absorbed into the singleton potentials. The resulting optimization problem can by solved by any LP solver, and the whole learning objective can be minimized by Quadratic Programming solvers.

5. Extensions

The above model can be easily extended to suit several natural variations of the AUC measure, suited for other practical applications. All of the following extensions build on the idea that the AUC loss is in fact a function of a *partial* ranking over the

variables, defined over $pos \times neg$ pairs alone. By reducing the number of variables, dependencies, or constraints, the learning problems induced by these AUC extensions can be solved in the manners suggested above, though with significantly lower computational complexity. Some examples are given in figure 6.2.

5.1 Local AUC

In many settings, a complex sample m is broken up into smaller disjoint sub-samples $l \in L_m$, and the average AUC over those sub-samples is taken as the performance measure. For instance, if a sample is a social network, predictions can be made over disjoint communities or neighborhoods (though features and factors may be defined over the whole network). To optimize over this *local* AUC loss, each sub-sample $l \in L_m$ can be treated as a sample, and its AUC loss term should be normalized by $|L_m|$.
In practice, treating sub-samples separately causes the loss to be insensitive to the relative rankings of two variables from different sub-samples. This is a generalization of the notion of the AUC as a function of a partial ranking. In the standard AUC, only $pos \times neg$ pairs where considered; in the average AUC, such pairs are considered only if they are in the same sub-sample.
This notion can be taken to the other extreme. If AUC performance is measured across *all* (or distinct sets of) samples, merging them into one (or several) sample(s) will allow for optimizing directly over a single, *global* AUC score.

5.2 Ground-truth consistency

Recall that in one of the hinge loss formulations, the maximizing $z \sim y^m$ of $w^\top \phi(x, z)$ is needed for the learning procedure. Although this problem does not include the AUC loss, it does include the ranking constraints. The additional consistency constraint can be applied by forcing all $i \in pos$ to be ranked higher than all $j \in neg$. This can be viewed as breaking up the sample into *pos* and *neg* sub-samples with unconstrained internal ranking but fixed external relations. In practice, all $\{z_{ki} : i \in pos, k > P\}$ are set to be 1, all $\{z_{ki} : z \in neg, k \leq P\}$ are set to be 0, and $\{z_{ki} : i \in pos, k \leq P\}$ and $\{z_{ki} : i \in neg, k > P\}$ are each constrained to be nested ranking sub-matrices.

5.3 Foregoing nestedness

In the beginning of this chapter, we claimed that in some settings (such as link prediction) the desired output is a ranking over the variables. However, sometimes this ranking is artificial in the sense that an ordering is forced upon the variables just so predictions of all cardinalities can be easily constructed. If we are simply interested in the ability to produce a prediction for every cardinality, a more coherent model would be one in which only these constraints applied. This can be achieved by dropping the nestedness constraints, leaving only the cardinality row-constraints over z.

Since the above proposed model does not output a ranking, its performance cannot be evaluated with the ROC. However, the notion of swapped pairs still applies, leading to a generalization of the AUC measure; swapped pairs are counted separately for each cardinality.

5.4 Binned ranks

In some settings, instead of a full rankings, variables are placed into ranked *bins* $b_1 \prec \cdots \prec b_B$. For instance, web-search query results are binned into pages. In this setting, the goal is to place positive variables in higher-ranked bins; the binned AUC score simply discards intra-bin swapped pairs. Since there is no internal ranking of variables within bins, the AUC can now be extended so that only inter-bin $pos \times neg$ pairs are taken into account. In some sense, this is the opposite of the local AUC case. In the local AUC measure, partial orderings were based on variables' predetermined groupings, while in the binned case rank-groups are defined a-priori.
To handle this setting, we replace the notion of *rank* in our model with that of ranked *bins*. We create a binned ranking model by multiplying the original $w^\top \phi(x, y)$ model B times, one for each bin. In addition to nestedness, we constrain each row k in z to be of cardinality $\sum_{k' \leq k} b_{k'}$. This greatly reduces the number of variables, factors and constraints in the model. For instance, if $B \ll n$ or is constant, solving the assignment problem can be done in $O(n^2)$.

Chapter 7

Experiments

In this chapter we present some empirical results attained by learning with our structured AUC loss framework. First, we present a toy problem in the image segmentation domain which demonstrates the boost in performance attained by adding structure when optimizing over the AUC. Then, we test our model in the genuine task of link prediction over a temporal scientific collaboration network. Since our model utilizes both structural regularities and direct AUC loss minimization (and capitalizes off of their interaction), we compare our results with two baselines: an unstructured model trained for minimizing the AUC loss, and a structured model trained for minimizing the hamming loss.

1. Image segmentation

Image segmentation is a classic example of how simple structure, such as pairwise interactions between adjacent pixels, can tremendously benefit learning. Image segmentation has been studied intensively, and pairwise models are known to achieve higher accuracy than factored models.
To provide a proof of concept to our method, we modify the image segmentation problem so that instead of predicting a segmentation, predictors output a ranking over the pixels, depicting the belief in their being fore or background. Given a ground-truth segmentation, a predicted ranking is measured according to its AUC score over the ground truth. This is exactly the setting for which our model is designated.

As samples, we used 5×5 black and white images of English letters, where each pixel was flipped with probability $p = 0.35$. The original image was used as a label. Training, validation, and test sets where created according to this process. Our objective of choice was the $-\max$ variant (see eq. 6.8). We optimized this objective by running a Difference of Convex functions Algorithm (DCA) algorithm, where each iteration was solved by applying an off-the-shelf Quadratic Programming (QP) solver to the current learning objective. The regularization constant was selected by mini-

mizing the error over the validation set.
A expected, the pairwise model attained an average AUC of 0.7075, whereas the factored model attained only 0.6559, and the structural SVM attained 0.6500.

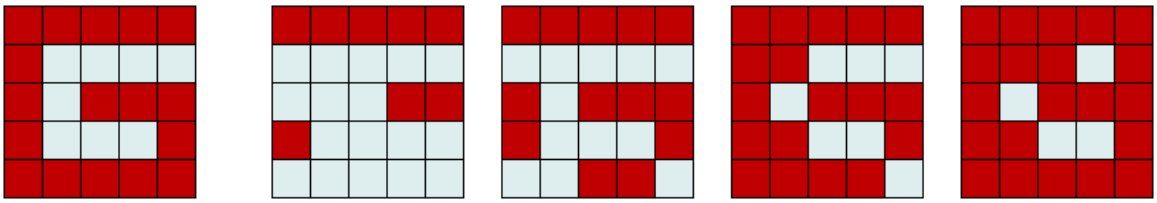

Figure 7.1: Image segmentation experiment - example.
Left: the original letter G. Right: predicted segmentations for $k = 8, 13, 18, 21$ foreground pixels.

2. Link prediction in scientific collaboration networks

To fully test our model, we apply it to the task of link prediction in scientific collaboration networks. A collaboration network contains authors of scientific papers as nodes, and edges are created between coauthors of the same paper. Our input contains networks G_t containing collaborations which took place at timestep t for $t = 1 \dots T$. Therefore, $G_1, \dots, G_T$ represent a timeseries of collaboration networks. In our setting edges are *transient* - an edge (i, j) appears in G_t if and only if i and j collaborated at time t. Hence, this is in fact a *recurring* link prediction task.

Data

Our data consists of collaborations between authors whose papers where accepted to the yearly NIPS conference in the years 1987-2003 (Roweis and Saul, 2000; Globerson et al., 2007). In our experiments, we truncated some of the first years (similar to Wang et al., 2007; Tylenda et al., 2009)) and last 2 years, and defined each timestep to include two consecutive years (as in Sarkar et al., 2012).

Setting

Our setting is similar to that of Sarkar et al. (2012), and is defined as follows. For input, we are given a time series of networks $G_1, \dots, G_T$. The task is then to predict for every node i a ranking π_i over edges $(i, j) \in G_{T+1}$ for every j in i's 2-hop neighborhood (that is, nodes which have been at distance of at most 2 from i sometime until time T). The performance of a predictor is evaluated by its AUC score averaged over all π_i-s for nodes i with a positive degree in G_{T+1}.
The input networks where used to create train, validation, and test sets. For each network G_t, a sample m was created for every focal node $i \in G_t$. Feature vectors ϕ_j^m and ϕ_{jl}^m where extracted from G_t and $\bar{G}_t$ for edges (i, j) and edge pairs $\{(i, j), (i, l)\}$ (respectfully) in i's 2-hop neighborhood. Ground-truth binary vector labels y^m where created according to edges in G_{t+1}. Labels for the test set where taken from G_{T+1}.

Features

As often done in supervised link prediction, unsupervised scores where used as singleton feature vectors. For every network G_t and cumulative network $\bar{G}_t = \bigcup_{t'=1}^{t} G_{t'}$, edge scores where computed for the common neighbors, Adamic-Adar, Katz, and Jaccard coefficient measures. These scores where incorporated into features vectors ϕ_j^m for each edge (i, j) in every sample m.
To demonstrate the power of structure, only simple pairwise factors where used. For every sample m of a focal node i, a factor was defined over a pair of edges $\{(i, j), (i, l)\}$ if the edge (j, l) existed in previous timesteps. We denote the set of pairs for sample m by P_m. This pair selection somewhat reflects the notion of triadic closure. Pairwise feature vectors ϕ_{jl}^m for edge pairs $\{(i, j), (i, l)\}$ contained both unsupervised scores of edges (j, l) and simple functions of the degree of nodes j and l, such as $\min\{d_j, d_l\}$, $\max\{d_j, d_l\}$, $|d_j - d_l|$, and $\sqrt{d_j \cdot d_l}$.

Model and learning objective

As a model, we chose a log-linear MRF over nested ranking matrices $z \in R$, giving rise to a score function of the form:

$$score(z) = w^\top \phi(z) \tag{7.1}$$

For a factored model, the score for a sample m with focal node i is given by:

$$w^\top \phi^m(z) = \sum_k \sum_j w^\top \phi_{ij}^k z_{kj} \tag{7.2}$$

For the pairwise model, we enhance the factored model with factors over pairs:

$$w^\top \phi^m(z) = \sum_k \sum_j w_s^\top \phi_{ij}^k z_{kj} + \sum_k \sum_{(j,l)\in P_m} w_p^\top \phi_{ijl}^k (z_{kj}, z_{kl}) \tag{7.3}$$

Note that in our models, singleton weights w_s and pair weights w_p are shared. Our learning objective was chosen to be:

$$\begin{aligned} &\min_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{M}\sum_{m=1}^{M} \max_{z\in R} H^m(z), \\ &H^m(z) = w^\top \phi^m(z) + \Delta_{\text{AUC}}(z, y^m) - \max_{z' \sim y^m} w^\top \phi^m(z') \end{aligned} \tag{7.4}$$

The validation set was used in order to select the best regularization constant λ by means of a grid search.

Optimization

Since our chosen objective function is non convex but rather a difference of convex functions, we applied the DCA procedure in order to optimize over w. Each DCA

iteration was either solved by running a QP solver over the current objective, or by applying the Frank-Wolfe (FW) algorithm for structured losses (Lacoste-Julien et al., 2012).

Baselines

As baselines, we used the following:

- **Unsupervised methods:** We applied each unsupervised link prediction method used in our model independently to attain a ranking over all edges in G_{T+1}, from which we deduced rankings π_i for the relevant nodes. These methods have been applied to both networks G_T and cumulative networks $\bar{G} = \bigcup_{t=1}^{T} G_t$.
- **Structured model, hamming loss:** To show the contribution of optimizing directly over the AUC loss, we trained a version of our proposed structured model over the hamming loss. A ranking was constructed by sorting the variables according to their singleton beliefs $\theta_i(1) - \theta_i(0)$.
- **Factored model, AUC loss:** To show the contribution of structure, we trained a factored model containing only the singleton factors of our proposed model over the AUC loss using the multivariate-loss structural SVM given by Joachims (2005).

2.1 Synthetic experiments

To portray the enhanced modeling power of structured models, we first ran a series of synthetic experiments. In these experiments, labels where created in accordance to some predetermined weight vector w. In each experiment, orderings π^m where predicted from the model parameterized by the chosen w, and binary vector labels y^m where derived from π^m according to some threshold. Due to computational constraints, only samples with a limited number of 2-hop neighbors where considered. Results are given in table (7.1).

In the first experiment, w_s was set to 0 (thus completely neutralizing the singleton features), and w_p was drawn from a multivariate normal distribution. Both factored and pairwise models where trained over the sample set containing the synthetic labels. Whereas the factored model's AUC is only a tad above that of a random predictor, the pairwise model's AUC is pleasing. In the second experiment, bot w_s and w_p were drawn from a multivariate normal distribution. Moreover, the features too where randomly drawn. In the third experiment, a random weight vector was again used, but now over real features. In these experiments, both models preformed well, though the pairwise model's AUC was significantly better than the that of the factored model (by gaps of 0.07 to 0.15). The fourth synthetic experiment will be discussed later on.

weights	features	#vars	factored	pairwise
$w_s = 0$, random w_p	real data	5	0.5714	**0.8849**
random	random	7	0.8883	**0.9550**
		10	0.8901	**0.9909**
random	real data	5	0.7596	**0.8878**
		10	0.6194	**0.7317**
		20	0.6739	**0.8228**
learned	real data	5	0.9325	**0.9881**
		10	**0.9742**	0.9609*
		20	0.9594	**0.9753**

Table 7.1: Mean AUC scores for the synthetic experiments.
In (*) the pairwise model did not achieve a lower objective value, which can explain the lower AUC.

2.2 Real data experiments

We ran several experiments on the real NIPS dataset as described in earlier sections. For reasons of computational efficiency and to test the generalizability of our model, we examined multiple settings in which only samples with a bounded number of variables (defined by $|y_m|$) where used. In each experiment, the factored model, pairwise model, and aforementioned baselines where compared according to their AUC scores over the test set. Results are given in table (7.2).

As expected, optimizing over the hamming loss resulted in poor performance. Unsupervised methods showed mixed results; some scored horribly, others adequately, and yet others quit well. The Katz measure scored remarkably well, in some settings with an AUC as high as (if not higher than) some supervised methods.
Although having different objective functions, both the factored model and the structural SVM are based on the same singleton features. Since both achieved comparable performance, we conclude that they are altogether similar in their ability to express the data. This is encouraging since the factored model is defined over a larger label space and constitutes a non-convex objective which is harder to optimize.

In all settings but one, the pairwise model optimized over the AUC loss scored highest amongst all methods. This suggests that adding structure to the model, even as simple as pairwise features, can improve performance when measured by AUC. However, the advantageous gap shrinks as the number of variables (and thus the number of samples) is increased, to the point of no gap for all variables. Nevertheless, training on samples with at most 20 variables (which constitute 84% of all samples) but classifying the whole test set maintains the pairwise model's supremacy. Interestingly, training on the restricted sample set results in better performance for all supervised models. This suggests that samples with a high number of variables bias learning against the test set, possibly due to the heavy-tail degree distribution.

max vars	Ham.	SVM	Fact.	Pair.	AA	CN	KZ	JC	AA_C	CN_C	KZ_C	JC_C
5 (40%)	0.667	0.758	0.766	**0.845**	0.452	0.528	0.742	0.444	0.337	0.476	0.825	0.242
10 (64%)	0.609	0.841	0.833	**0.870**	0.53	0.607	0.783	0.408	0.414	0.546	0.835	0.188
20 (84%)	0.572	0.819	0.824	**0.843**	0.613	0.675	0.804	0.461	0.509	0.596	0.806	0.233
all (100%)	0.594	0.813	**0.815**	0.814	0.656	0.704	0.804	0.507	0.556	0.628	0.800	0.242
20→all	-	0.830	0.824	**0.838**	-	-	-	-	-	-	-	-

Table 7.2: Mean AUC scores for the real data experiments.
Supervised methods: Factored MRF over hamming loss (Ham.), Structural SVM (SVM), and factored (Fact.) and pariwise (Pair.) ranking MRFs over the AUC loss . **Unsupervised methods:** Adamic-Adar (AA), common neighbors (CN), Katz (KZ), and Jaccard (JC), run on both G_T and $\bar{G}_T$ (C subscript).

Although adding structure seems to have improved results, the gain in AUC is not as high as we had hoped. This result is perhaps not surprising. First, the overall improvement of *any* supervised method over the unsupervised Katz measure was not significant, while the other unsupervised methods performed poorly. Since these unsupervised scores served as features for the supervised models, it is reasonable that a linear combination of them would not result in significantly better performance. Furthermore, these unsupervised methods have been 'borrowed' from the common task of predicting *new* links; the more recent task of *recurring* link prediction, which has received much less literary attention, may require task-specific features.
Second, the overwhelming majority of samples m with corresponding labels y_m had only one variable which was on (or in other words, $|y_m| = 1$). For these samples, it is unlikely that adding pairwise features should help. Therefore, a pairwise model can potentially improve the AUC of only a limited number of samples. Moreover, since the model included pairwise features for *all* samples, the pairwise signal of samples with $|y_m| > 1$ was probably overwhelmed by the noise of those with $|y_m| = 1$.

It is also possible that in our selected setting, most of the signal can be modeled by singleton features alone. We verified this by creating a synthetic experiment (table 7.1) in which labels where created according to a weight vector w learned by a pairwise model over real data. Results show that while the pairwise model attained slightly better AUC scores, both factored and pairwise models scored very high.
More generaly, adding structure to the model does not always help, as in the case of multi-label prediction (Finley and Joachims, 2008; Dembczynski et al., 2010). The non-convex nature of the learning objective and the intractability of the combinatorial optimization problems may be a setback to the model's true potential. It is also reasonable that simple pairwise features may not be enough to fully model the structural regularities in the data. Hopefully, adding other high-order features such as the task-specific features we suggested in section 2 of chapter 5 will improve performance. Finally, scientific collaborations, and specifically the NIPS dataset, are but one of many social networks available.
Our results and analysis further encourage us to elaborate our model and test it on other data. The promising results of our image segmentation toy example suggests that this model may be beneficial in other domains.

3. Conclusions

In this chapter we have empirically tested the performance of our suggested method for learning structured models over the AUC loss. In a simple image segmentation task, adding pairwise features to the model resulted in significantly better performance. Since it is well known that pairwise features boost performance in image segmentation, we believe that our model will gain superior results for larger and more complex images and image sets.

Our model also displayed superior performance in the task of link prediction. When trained on synthetic network data, our model's performance was better than that of unstructured models by a large margin. When trained on real world collaboration networks, our pairwise model attained higher AUC scores than factored models trained over the AUC loss, structured models trained over the hamming loss, and state-of-the-art unsupervised methods. Since the above improvement in results is attained by adding only simple pairwise features, we believe that adding other high-order features such as those discussed in chapter 5 will further enhance the model's performance.

Altogether, these results suggest that in highly structural domains where performance is evaluated by the AUC, learning structured models over the AUC loss is worthwhile.

Bibliography

Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

Ajit Agrawal, Philip Klein, and R Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

Edoardo M Airoldi, David M Blei, Stephen E Fienberg, Anna Goldenberg, Eric P Xing, and Alice X Zheng. *Statistical Network Analysis: Models, Issues, and New Directions: ICML 2006 Workshop on Statistical Network Analysis, Pittsburgh, PA, USA, June 29, 2006, Revised Selected Papers*, volume 4503. Springer, 2007.

Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *SDM06: Workshop on Link Analysis, Counter-terrorism and Security*, 2006.

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

Stuart Andrews and Tony Jebara. Structured network learning. In *NIPS Workshop on Learning to Compare Examples*, 2006.

Stuart J Andrews and Tony Jebara. Structured prediction of generalized matching graphs. *Journal of Machine Learning Research*, 1:1–26, 2008.

Lars Backstrom and Jure Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 635–644. ACM, 2011.

Gökhan H Bakir, Thomas Hofmann, Bernhard Schölkopf, Alexander J Smola, Ben Taskar, and SVN Vishwanathan. *Predicting structured data*. MIT press, 2007.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

Albert-Laszlo Barabâsi, Hawoong Jeong, Zoltan Néda, Erzsebet Ravasz, Andras Schubert, and Tamas Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3):590–614, 2002.

Nesserine Benchettara, Rushed Kanawati, and Celine Rouveirol. Supervised machine learning applied to link prediction in bipartite social networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*, pages 326–330. IEEE, 2010.

Endre Boros and Peter L Hammer. Pseudo-boolean optimization. *Discrete applied mathematics*, 123(1):155–225, 2002.

Olivier Chapelle, Quoc Le, and Alex Smola. Large margin optimization of ranking measures. In *NIPS Workshop: Machine Learning for Web Search*, 2007.

Justin Cheng, Daniel M Romero, Brendan Meeder, and Jon Kleinberg. Predicting reciprocity in social networks. In *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)*, pages 49–56. IEEE, 2011.

Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 279–286, 2010.

Peter Sheridan Dodds, Roby Muhamad, and Duncan J Watts. An experimental study of search in global social networks. *science*, 301(5634):827–829, 2003.

Raissa M D'Souza, Christian Borgs, Jennifer T Chayes, Noam Berger, and Robert D Kleinberg. Emergence of tempered preferential attachment from optimization. *Proceedings of the National Academy of Sciences*, 104(15):6112–6117, 2007.

Marijtje AJ Duijn, Tom AB Snijders, and Bonne JH Zijlstra. p2: a random effects model with covariates for directed graphs. *Statistica Neerlandica*, 58(2):234–254, 2004.

Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(2):10, 2011.

David Easley and Jon Kleinberg. *Networks, crowds, and markets*, volume 8. Cambridge Univ Press, 2010.

Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl*, 5:17–61, 1960.

Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8): 861–874, 2006.

Stephen E Fienberg and S Keith Lee. On small world statistics. *Psychometrika*, 40 (2):219–228, 1975.

Thomas Finley and Thorsten Joachims. Training structural svms when exact inference is intractable. In *Proceedings of the 25th international conference on Machine learning*, pages 304–311. ACM, 2008.

Michael Fire, Lena Tenenboim, Ofrit Lesser, Rami Puzis, Lior Rokach, and Yuval Elovici. Link prediction in social networks using computationally efficient topological features. In *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)*, pages 73–80. IEEE, 2011.

Ove Frank and David Strauss. Markov graphs. *Journal of the american Statistical association*, 81(395):832–842, 1986.

Amir Globerson, Gal Chechik, Fernando Pereira, and Naftali Tishby. Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research*, 8:2047–2076, 2007.

Anna Goldenberg, Alice X Zheng, Stephen E Fienberg, and Edoardo M Airoldi. A survey of statistical network models. *Foundations and Trends® in Machine Learning*, 2(2):129–233, 2010.

Mark S Granovetter. The strength of weak ties. *American journal of sociology*, pages 1360–1380, 1973.

Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

David J Hand. Measuring classifier performance: a coherent alternative to the area under the roc curve. *Machine learning*, 77(1):103–123, 2009.

Steve Hanneke and Eric Xing. Discrete temporal models of social networks. *Statistical network analysis: Models, issues, and new directions*, pages 115–125, 2007.

MohammadAl Hasan and MohammedJ. Zaki. A survey of link prediction in social networks. In Charu C. Aggarwal, editor, *Social Network Data Analytics*, pages 243–275. Springer US, 2011.

Paul W Holland and Samuel Leinhardt. Transitivity in structural models of small groups. *Comparative Group Studies*, 1971.

Paul W Holland and Samuel Leinhardt. An exponential family of probability distributions for directed graphs. *Journal of the american Statistical association*, 76 (373):33–50, 1981.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8): 2554–2558, 1982.

Zan Huang. Link prediction based on graph topology: The predictive value of the generalized clustering coefficient. In *Workshop on Link Analysis: Dynamics and Static of Large Networks (LinkKDD2006)*, 2006.

Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, 31(1):253–258, 1925.

Paul Jaccard. *Etude comparative de la distribution florale dans une portion des Alpes et du Jura*. Impr. Corbaz, 1901.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384. ACM, 2005.

Jason K Johnson. *Convex relaxation methods for graphical models: Lagrangian and maximum entropy approaches*. PhD thesis, Massachusetts Institute of Technology, 2008.

Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.

Hisashi Kashima and Naoki Abe. A parameterized probabilistic model of network evolution for supervised link prediction. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 340–349. IEEE, 2006.

Hisashi Kashima, Tsuyoshi Kato, Yoshihiro Yamanishi, Masashi Sugiyama, and Koji Tsuda. Link propagation: A fast semi-supervised learning algorithm for link prediction. In *Proceedings of the 2009 SIAM Conference on Data Mining (SDM)*, pages 1099–1110, 2009.

Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18 (1):39–43, 1953.

Leo Katz and JamesH. Powell. A proposed index of the conformity of one sociometric measurement to another. *Psychometrika*, 18:249–256, 1953. ISSN 0033-3123.

Ross Kindermann, James Laurie Snell, et al. *Markov random fields and their applications*. American Mathematical Society Providence, RI, 1980.

Jon M Kleinberg. Navigation in a small world. *Nature*, 406(6798):845–845, 2000.

Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. Mrf energy minimization and beyond via dual decomposition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):531–552, 2011.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Stochastic block-coordinate frank-wolfe optimization for structural svms. *arXiv preprint arXiv:1207.4747*, 2012.

Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 462–470. ACM, 2008.

Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *The Journal of Machine Learning Research*, 11:985–1042, 2010.

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58 (7):1019–1031, 2007.

Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 243–252. ACM, 2010.

Qiang Liu and Alexander Ihler. Learning scale free networks by reweighted l1 regularization. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Jorge M Lobo, Alberto Jiménez-Valverde, and Raimundo Real. Auc: a misleading measure of the performance of predictive distribution models. *Global Ecology and Biogeography*, 17(2):145–151, 2007.

Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

R Duncan Luce and Albert D Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.

Aditya Menon and Charles Elkan. Link prediction via matrix factorization. *Machine Learning and Knowledge Discovery in Databases*, pages 437–452, 2011.

Aditya Krishna Menon and Charles Elkan. A log-linear model with latent features for dyadic prediction. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 364–373. IEEE, 2010.

Thomas Mensink, Jakob Verbeek, Tiberio Caetano, et al. Learning to rank and quadratic assignment. In *NIPS Workshop on Discrete Optimization in Machine Learning*, 2011.

Ofer Meshi, David Sontag, Tommi Jaakkola, and Amir Globerson. Learning efficiently with approximate inference via dual losses. In *ICML*, pages 783–790, 2010.

Charles E Metz. Basic principles of roc analysis. In *Seminars in nuclear medicine*, volume 8, pages 283–298. Elsevier, 1978.

Stanley Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.

Kurt T Miller, Thomas L Griffiths, and Michael I Jordan. Nonparametric latent feature models for link prediction. *Advances in Neural Information Processing Systems (NIPS)*, 22, 2009.

Michael Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet mathematics*, 1(2):226–251, 2004.

Jacob Levy Moreno. *Who shall survive?*, volume 58. Nervous and Mental Disease Publishing Company Washington, DC, 1934.

Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2):025102, 2001.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

Talcott Parsons. *The structure of social action: A study in social theory with special reference to a group of recent European writers*, volume 1. Free Press New York, 1968.

David M Pennock, Gary W Flake, Steve Lawrence, Eric J Glover, and C Lee Giles. Winners don't take all: Characterizing the competition for links on the web. *Proceedings of the national academy of sciences*, 99(8):5207–5211, 2002.

Foster Provost, Tom Fawcett, et al. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the third international conference on knowledge discovery and data mining*, pages 43–48. Amer Assn for Artificial, 1997.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

Nathan Ratliff, J Andrew Bagnell, and Martin Zinkevich. Subgradient methods for maximum margin structured learning. In *ICML Workshop on Learning in Structured Output Spaces*, volume 46. Citeseer, 2006.

Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(2):026112, 2003.

Garry Robins, Tom Snijders, Peng Wang, Mark Handcock, and Philippa Pattison. Recent developments in exponential random graph (¡ i¿ p¡/i¿*) models for social networks. *Social networks*, 29(2):192–215, 2007.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

G. Salton and M. J. Mcgill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

Purnamrita Sarkar, Deepayan Chakrabarti, and Michael I Jordan. Nonparametric link prediction in dynamic networks. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Georg Simmel and Kurt H Wolff. *The sociology of georg simmel*, volume 92892. Simon and Schuster, 1950.

Tom AB Snijders, Philippa E Pattison, Garry L Robins, and Mark S Handcock. New specifications for exponential random graph models. *Sociological Methodology*, 36 (1):99–153, 2006.

Chaoming Song, Shlomo Havlin, and Hernan A Makse. Self-similarity of complex networks. *Nature*, 433(7024):392–395, 2005.

David Sontag, Amir Globerson, and Tommi Jaakkola. Introduction to dual decomposition for inference. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*. MIT Press, 2010.

Jiliang Tang, Huiji Gao, Xia Hu, and Huan Liu. Exploiting homophily effect for trust prediction. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 53–62. ACM, 2013.

Pham Dinh Tao and Le Thi Hoai An. Convex analysis approach to dc programming: theory, algorithms and applications. *Acta Mathematica Vietnamica*, 22(1):289–355, 1997.

Pham Dinh Tao and El Bernoussi Souad. Algorithms for solving a class of nonconvex optimization problems. methods of subgradients. *North-Holland Mathematics Studies*, 129:249–271, 1986.

Ben Taskar, Ming-Fai Wong, Pieter Abbeel, and Daphne Koller. Link prediction in relational data. In *Neural Information Processing Systems*, volume 15, 2003.

Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 613–622, Washington, DC, USA, 2006. IEEE. ISBN 0-7695-2701-9.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(2):1453, 2006.

Tomasz Tylenda, Ralitsa Angelova, and Srikanta Bedathur. Towards time-aware link prediction in evolving social networks. In *Proceedings of the 3rd Workshop on Social Network Mining and Analysis*, page 9. ACM, 2009.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2): 1–305, 2008.

Chao Wang, Venu Satuluri, and Srinivasan Parthasarathy. Local probabilistic models for link prediction. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 322–331. IEEE, 2007.

Stanley Wasserman and Philippa Pattison. Logit models and logistic regressions for social networks: I. an introduction to markov graphs and p^*. *Psychometrika*, 61 (3):401–425, 1996.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

Barry Wellman. Structural analysis: From method and metaphor to theory and substance. In Barry Wellman and S. D. Berkowitz, editors, *Social structures: A network approach*, chapter Structural analysis: From method and metaphor to theory and substance, pages 19–61. JAI Press, Greenwich, CT, 1997.

Harrison C White. Search parameters for the small world problem. *Social Forces*, 49 (2):259–264, 1970.

Evan Wei Xiang. A survey on link prediction models for social network data. *Science And Technology*, 2008.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. *Machine Learning*, 20(2):912, 2003.