Link Prediction via Matrix Factorization

Aditya Krishna Menon and Charles Elkan

University of California, San Diego
La Jolla, CA 92093
{akmenon,elkan}@cs.ucsd.edu

Abstract. We propose to solve the link prediction problem in graphs using a supervised matrix factorization approach. The model learns latent features from the topological structure of a (possibly directed) graph, and is shown to make better predictions than popular unsupervised scores. We show how these latent features may be combined with optional explicit features for nodes or edges, which yields better performance than using either type of feature exclusively. Finally, we propose a novel approach to address the class imbalance problem which is common in link prediction by directly optimizing for a ranking loss. Our model is optimized with stochastic gradient descent and scales to large graphs. Results on several datasets show the efficacy of our approach.

Keywords: Link prediction, matrix factorization, side information, ranking loss.

1 The Link Prediction Problem

Link prediction is the problem of predicting the presence or absence of edges between nodes of a graph. There are two types of link prediction: (i) *structural*, where the input is a partially observed graph, and we wish to predict the status of edges for unobserved pairs of nodes, and (ii) *temporal*, where we have a sequence of fully observed graphs at various time steps as input, and our goal is to predict the graph state at the next time step. Both problems have important practical applications, such as predicting interactions between pairs of proteins and recommending friends in social networks respectively. This document will focus on the structural link prediction problem, and henceforth, we will use the term "link prediction" to refer to the structural version of the problem.

Link prediction is closely related to the problem of collaborative filtering, where the input is a partially observed matrix of (user, item) preference scores, and the goal is to recommend new items to a user. Collaborative filtering can be seen as a bipartite weighted link prediction problem, where users and items are represented by nodes, and edges between nodes are weighted according to the preference score. More generally, both problems are instances of *dyadic prediction*, which is the problem of predicting a label for the interaction between pairs of entities (or *dyads*) [17]. Despite this connection, there has been limited interaction between the link prediction and collaborative filtering literatures, aside from a few papers that propose to solve one problem using models from the other [7,23].

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part II, LNAI 6912, pp. 437–452, 2011.

1.1 Challenges in Link Prediction

We point out three challenges in link prediction that a model should ideally address. First, in addition to the topological information of the graph, we sometimes have extra *side information* or covariates for the nodes. For example, in a graph of interaction between pairs of proteins, we might have features describing the biological properties of each protein. This information can be useful in predicting links, especially when a node is only sparsely connected. Since the graph topology and the side information potentially encode different types of information, combining them is expected to give the best performance. However, it is not obvious how best to do this in general. Further, to be flexible, we would like to make the side information only an *optional* component of a model.

Second, link prediction datasets are characterized by extreme *imbalance*: the number of edges known to be present is often significantly less than the number of edges known to be absent. This issue has motivated the use of area under the ROC curve (AUC) as the *de facto* performance measure for link prediction tasks, as, unlike standard 0-1 accuracy, AUC is not influenced by the distribution of the classes. However, the class imbalance still hampers the effectiveness of many models that would otherwise be suitable on balanced data.

Third, it is imperative that models be computationally efficient if they are to scale to graphs with a large number of nodes and/or edges. Such large-scale graphs are characteristic of many real-world applications of link prediction, such as the aforementioned friend-recommendation in social networks.

1.2 Our Contributions

This paper studies the effectiveness of matrix factorization techniques for the structural link prediction problem, inspired by their success in collaborative filtering [20]. We first make a case for matrix factorization to serve as a foundation for a general purpose link prediction model. We explain how it may be thought of as learning *latent features* from the data, and why it can be expected to be more predictive than popular unsupervised scores. We show how latent features may be combined with explicit features for nodes and edges, and in particular how the factorization can be combined with the outputs of other models. Further, we propose a novel mechanism to allow factorization models to overcome the imbalance problem, based on the idea of optimizing for a *ranking loss*. Experimentally, we first show that factorization significantly outperforms several popular unsupervised scores. Next, we demonstrate that while explicit features are usually able to provide better estimates of linking behaviour than implicit features, combining the two can further improve performance. Finally, we show that optimizing for a ranking loss can improve AUC by up to 10%.

Before proceeding, we define the link prediction problem more formally and fix the notation used in this paper.

1.3 Problem Definition and Notation

Formally, structural link prediction has as input a partially observed graph $G \in \{0, 1, ?\}^{n \times n}$, where 0 denotes a *known absent* link, 1 denotes a *known present* link, and ? denotes an unknown status link. Our goal is to make predictions for the ? entries. The set of observed dyads is denoted by $\mathcal{O} = \{(i,j) : G_{ij} \neq ?\}$, and we use $\mathcal{O}_i$ to mean the observed dyads involving the ith node. In some cases, we may have additional features (or covariates) for dyads and/or individual nodes. We call such features *side information.*

We will use capital variables (e.g. X) to denote matrices and lower-case variables (e.g. x) to denote vectors. We will use x_i to mean the ith row of the matrix X. The Frobenius norm of the matrix X is denoted by $||X||_F^2 = \sum_i ||x_i||_2^2$.

2 Existing Link Prediction Models

At a high level, existing link prediction models fall into two classes: unsupervised and supervised. Unsupervised models compute scores for pairs of nodes based on topological properties of the graph. For example, one such score is the number of common neighbours that two nodes share. Other popular scores are the Adamic-Adar [1] and Katz score [22]. These models use predefined scores that are invariant to the specific structure of the input graph, and thus do not involve any learning. Supervised models, on the other hand, attempt to be directly predictive of link behaviour by learning a parameter vector θ via

$$\min_{\theta} \frac{1}{|\mathcal{O}|} \sum_{(i,j) \in \mathcal{O}} \ell(G_{ij}, \hat{G}_{ij}(\theta)) + \Omega(\theta), \tag{1}$$

where $\hat{G}_{ij}(\theta)$ is the model's predicted score for the dyad (i,j), $\ell(\cdot,\cdot)$ is a *loss function*, and $\Omega(\cdot)$ is a *regularization* term that prevents overfitting. The choice of these terms depends on the type of model. We list some popular approaches:

1. **Feature-based models.** Suppose each node i in the graph has an associated feature vector $x_i \in \mathbb{R}^d$. Suppose further that each dyad (i,j) has a feature vector $z_{ij} \in \mathbb{R}^D$. Then, we may instantiate Equation 1 via

$$\hat{G}_{ij}(w,v) = L(f_D(z_{ij}; w) + f_M(x_i, x_j; v)) \tag{2}$$

for appropriate $f_D(\cdot), f_M(\cdot,\cdot)$ acting on dyadic and monadic features respectively, and a link function $L(\cdot)$. Both linear [33] and nonlinear [14,5] choices of $f_D(\cdot), f_M(\cdot,\cdot)$ have been considered. In the linear case, it is standard to let $f_D(z_{ij}; w) = w^T z_{ij}$ and $f_M(x_i, x_j; v) = (v^{(1)})^T x_i + (v^{(2)})^T x_j$, where $v^{(1)} = v^{(2)}$ iff the graph is undirected. The loss is typically either square- or log-loss, and the regularizer typically $\frac{\lambda_w}{2}||w||_2^2 + \frac{\lambda_v}{2}||v||_2^2$. Note also that we can compute multiple unsupervised scores between pairs of nodes (i,j), and treat these as comprising a feature vector z_{ij}.

2. **Graph regularization models.** Here, we assume the existence of node features $x_i \in \mathbb{R}^d$, based on which we construct a kernel $K_{ii'jj'}$ that compares the node pairs (i,j) and (i',j'). We compute the predicted adjacency matrix $\hat{G}$ by constraining that values in this matrix should vary smoothly according

to K. Thus K acts as a graph regularizer, a popular approach in semi-supervised learning [39]. In the framework of Equation 1, we have

$$\Omega(\hat{G}) = \frac{\lambda}{2} \sum_{i,i',j,j'} K_{ii'jj'} (\hat{G}_{ij} - \hat{G}_{i'j'})^2 + \frac{\mu}{2} \sum_{(i,j) \notin \mathcal{O}} \hat{G}_{ij}^2$$

The above is called *link propagation* [19,26]. The performance of such methods depends on the choice of kernel K, which is pre-specified and not learned from the data.

3. **Latent class models.** These models assign each node of the graph to a class, and use the classes to predict the link structure. [4] assumes that nodes interact solely through their class memberships. It is possible to extend this to allow nodes to have membership in multiple classes [3]. These models are largely Bayesian, and so are not directly expressible in the empirical loss framework of Equation 1. Nonetheless, they do learn a matrix of probabilities $P \in \{0,1\}^{C \times C}$, where C is the number of classes, and this is done by placing appropriate priors on P, which may be viewed as a form of regularization.
4. **Latent feature models.** Here, we treat link prediction as a matrix completion problem, and factorize $G \approx L(U \Lambda U^T)$ for some $U \in \mathbb{R}^{n \times k}$, $\Lambda \in \mathbb{R}^{k \times k}$ and link function $L(\cdot)$. Each node i thus has a corresponding *latent vector* $u_i \in \mathbb{R}^k$, where k is the number of *latent features.* In the setup of Equation 1, we have
$$\hat{G}_{ij}(U, \Lambda) = L(u_i^T \Lambda u_j).$$
The regularizer $\Omega(U, \Lambda) = \frac{\lambda_U}{2}||U||_F^2 + \frac{\lambda_L}{2}||\Lambda||_F^2$ usually. Such models have been successful in other dyadic prediction tasks such as collaborative filtering [20]. This approach has not been studied as extensively in the link prediction literature as it has in the collaborative filtering literature. Bayesian versions of these methods using a sigmoidal link function have been studied in statistics [16] and political science [34], where they are sometimes referred to as *bilinear random effects* models. These fields have focussed more on qualitative analysis of link behaviour than predictive performance and scalability. In the machine learning literature, computationally efficient frequentist versions of these latent feature models have been studied in [23,37], and Bayesian models have also been extended to allow for an infinite number of latent features [24,25].

2.1 Do Existing Methods Meet the Challenges in Link Prediction?

We recall the three challenges in link prediction from Section 1.1, and study how they are handled by current models.

- **Incorporating topological and side information.** Existing models typically use a nonlinear classifier such as a kernel SVM [14] or decision tree [21] to combine topological and explicit features. However, the topological structure is exploited by just learning weights on unsupervised scores. We will show that this is potentially limiting, since unsupervised scores have limited expressivity. Latent feature methods like [16] do incorporate side information, but we will subsequently describe a limitation of this approach.

- **Overcoming imbalance.** Relatively little attention has been paid to modifying models so as to account for the class imbalance. One solution is undersampling the set of training dyads [14,21], but this has the disadvantage of necessarily throwing away information. [9] addresses imbalance by casting the problem as being cost-sensitive with unknown costs, and tunes the costs via cross-validation.
- **Scaling to large graphs.** Methods based on topological scores generally scale to large graphs, by virtue of mostly requiring only simple operations on the adjacency matrix. Some scores that look at long-range relationships between node pairs, such as the Katz measure, can be approximated in order to run on large graphs [10]. For methods based on explicit features, undersampling to overcome imbalance also reduces the number of training examples [21]. Latent class and latent feature methods based on Bayesian models do not scale to large graphs due to the cost of MCMC sampling.

We summarize representative link prediction methods in Table 1, and note whether they meet the three challenges. We would like a model that has the same expressiveness as existing methods, while directly addressing the above challenges. Our proposal is to extend matrix factorization to this end. The next section proposes the model, and argues why it is an suitable foundation for a link prediction model.

Table 1. A comparison of various link prediction methods. The *'s for "Large graphs?" indicate methods that perform some pre-processing on the data.

Class	Method	side information?	Imbalance?	Large graphs?
Unsupervised	Adamic-Adar [22]	No	No	Yes
	Katz [22]	No	No	Yes
Feature-based	Toplogical feats [14]	Optional	No	Yes
	CCP [9]	Optional	Yes	Yes*
	HPLP [21]	Optional	Yes	Yes*
Graph regularizer	LP [19]	Required	No	No
	ELP [26]	Required	No	Yes
Latent class	MMSB [3]	No	No	No
	IBP [24]	Optional	No	No
	IMRM [25]	Optional	No	No
Latent feature	Random effects [16]	Optional	No	No
	LFL [23]	Optional	No	Yes
	Our model	Optional	Yes	Yes

3 Extending Matrix Factorization for Link Prediction

A basic matrix factorization model for link prediction involves optimizing

$$\min_{U,\Lambda,b} \frac{1}{|\mathcal{O}|} \sum_{(i,j)\in\mathcal{O}} \ell(G_{ij}, L(u_i^T \Lambda u_j + b_i + b_j)) + \Omega(U,\Lambda) \tag{3}$$

for some appropriate link function $L(\cdot)$, loss function $\ell(\cdot)$ and regularizer $\Omega(\cdot, \cdot)$. As explained in the previous section, we interpret each $u_i \in \mathbb{R}^k$ as being a latent vector for each node, and so this is also called a *latent feature* approach. The b_i and b_j terms are node-specific *biases*, which are analogous to the intercept term in standard supervised learning. If the graph is undirected, then we can absorb Λ into the U matrix. For directed graphs, we can let Λ be an arbitrary asymmetric matrix, following [38,23].

3.1 Why is the Factorization Approach Appealing?

One nice property of the above model is that it can be trained using stochastic gradient descent, where we repeatedly draw a random $(i, j) \in \mathcal{O}$ and update u_i and u_j based on the corresponding gradients. A sweep over all observed (i, j) is known as an epoch, and often only a small constant number of epochs is needed for convergence. Training is thus linear in the number of observed dyads. As noted earlier, latent feature models for link prediction have been considered previously [16,34], but are typically trained with MCMC sampling, thus limiting analysis to small graphs. By contrast, with stochastic gradient descent we can handle graphs with several thousands of nodes and millions of edges.

Of course, scalability would be useless if the model were insufficiently rich. In fact, latent feature models can be seen as a generalization of latent class models, which may be thought of as learning a binary matrix $U \in \{0,1\}^{n \times C}$, where C is the number of classes, and predicting UWU^T for a matrix W of inter-class link scores. Latent features can also be viewed as a much richer way of exploiting topological information than popular unsupervised measures described in the previous section. By construction, the latent feature approach exploits the graph topology so as to be maximally predictive of link behaviour. Thus in general, one would expect its scores to be more accurate than any single unsupervised method. A further conceptual advantage over unsupervised scores is that the learned u_i's let us make qualitative analyses of the nodes in the graph; for example, they may be used to visualize the structure of the graph.

Note that basic latent feature models are *not* the same as just computing the singular value decomposition (SVD) of the adjacency matrix with unknown status and known absent edges collapsed into a single class. Latent feature methods only attempt to model the known present and known absent edges in the graph, with regularization to prevent overfitting; no effort is spent in modelling the unknown status edges. The solutions of the two models are thus very different.

The other appealing property about matrix factorization is that there are intuitive ways to extend the model to incorporate side information and overcome the imbalance problem. We now describe these extensions in turn.

3.2 How Do We Combine Explicit and Latent Features?

Suppose we have explicit features $x_i \in \mathbb{R}^d$ for the ith row (or column) of the data, and features $z_{ij} \in \mathbb{R}^D$ for each dyad. A standard way to incorporate these features with the latent features is via a linear combination:

$$\min_{U,\Lambda,w,v,b} \frac{1}{|\mathcal{O}|} \sum_{(i,j)\in\mathcal{O}} \ell(G_{ij}, L(u_i^T \Lambda u_j + b_i + b_j + f_D(z_{ij}; w) + f_M(x_i, x_j; v))) + \Omega(U, \Lambda, w, v), \tag{4}$$

where $f_D(z_{ij}; w) = w^T z_{ij}$ and $f_M(x_i, x_j; v) = v^T x_i + v^T x_j$. This underpins the approaches of [24,16,15,23]. There is a subtle point regarding the choice of f_M. For the monadic features x_i, choosing $f_M(x_i, x_j; v) = v^T x_i + v^T x_j$ corresponds to forming a vector $x_{ij} = \left[x_i \; x_j\right]$ for the pair (i, j), and using a standard linear model. However, a drawback of this approach is that it only learns the *propensity* of i and j for the outcome G_{ij} [32]. Specifically, if we did not have the latent and dyadic features, for a fixed user i, the model would produce an identical ranking of link scores to every other user. This is obviously very restrictive.

An alternative is to use the prediction function $f_M(x_i, x_j; V) = x_i^T V x_j$, where $V \in \mathbb{R}^{d\times d}$. This does not suffer from the propensity issue, and is known as a *bilinear regression* model [11]. (To contrast, we will refer to the previous model as *unilinear* regression.) We let V be symmetric iff the graph G is. In a collaborative filtering context, such bilinear models have been used in [8,37]. Note that we need to learn d^2 parameters, which may be prohibitive. In such cases, we can either perform dimensionality reduction on the x_i's, or constrain V to be a diagonal plus low-rank matrix, $V = D + A^T B$, and learn the factors A, B, D.

The ability to augment latent with explicit features has a pleasant consequence, namely that we can combine latent features with the results of any other link prediction model. Suppose another model returns scores $\hat{G}_{ij}$ for the dyad (i, j). Then, we can treat this as being a dyadic feature z_{ij} in the above framework, and learn latent features that fit the residual of these scores. In general then, the latent feature approach has a natural mechanism by which any predictive signal can be incorporated, whether it is an explicit feature vector or model predictions. However, a caveat is in order: it is not necessary that combining latent features with another model will improve performance on test data. If the latent features learn similar structure to the other model, then combining the two cannot be expected to yield better results.

As a final remark, we note that the linear combination of latent and explicit features is not the only way to incorporate side information. This issue has been studied in the context of the *cold-start* problem [29] in collaborative filtering. Recent advances in this literature are based on inferring reasonable values of latent features by falling back to the side information as a prior [2,12]. However, unlike most collaborative filtering applications, in link prediction we are mostly interested in using side information to improve predictions, rather than dealing with cold-start nodes. Therefore, we expect it to be most useful to directly augment the latent feature prediction with one based on side information.

3.3 How Do We Overcome Imbalance?

Imbalanced classes pose a problem for at least two reasons: (i) with fewer examples of one class, it is more difficult to infer reliable patterns (ii) it is standard to train models to optimize for 0-1 accuracy, which can be made very high by trivially predicting everything to belong to the dominant class; hence, most models are prone to yield biased results. Our matrix factorization model is not immune to these problems when trained with square- or log-loss, which may be seen as convex approximations to the 0-1 loss. Therefore, we must consider how to modify the model to account for imbalance. A standard strategy to overcome imbalance in supervised learning is undersampling [6], where we randomly throw out examples from the dominant class till the classes are more reasonably balanced. A disadvantage of undersampling is that it necessarily throws out information in the training set. Further, it is not clear to what ratio we can undersample without compromising the variance of our learned model.

An alternative is based on the following observation: in imbalanced classification problems, we often measure performance using AUC, which is agnostic to the distribution of classes. Intuitively, to get good test set AUC, it makes sense to *directly optimize* for AUC on the training set. This implicitly overcomes the imbalance problem, while simultaneously attempting to get the best AUC score on test data. This idea appears under-explored in the supervised learning literature, possibly due to the perceived complexity of optimizing for AUC. However, it is possible to optimize for AUC even on very large datasets [30]. To do this, we begin with the pairwise SVMRank framework [18]. Consider a binary classification scenario with training set $\{(x_i, y_i)\}$, and let $P = \{i : y_i = 1\}$ and $N = \{i : y_i = 0\}$. The empirical AUC of a linear classifier with weight w is

$$\hat{\mathcal{A}} = \frac{1}{|N||P|} \sum_{i \in P} \sum_{j \in N} \mathbf{1}[w^T x_i > w^T x_j].$$

The problem of maximizing AUC can be cast as

$$\min_w \frac{1}{|N||P|} \sum_{i \in P} \sum_{j \in N} \mathbf{1}[w^T (x_i - x_j) < 0].$$

The above is a binary classification task on $\{(x_i - x_j, 1)\}_{(i,j) \in \mathcal{Q}}$, where $\mathcal{Q} = \{(i,j) : y_i = 1, y_j = 0\}$. Thus, to maximize the AUC, we can replace the indicator function above with a regularized loss function:

$$\min_w \frac{1}{|N||P|} \sum_{(i,j) \in \mathcal{Q}} \ell(w^T (x_i - x_j), 1) + \Omega(w).$$

The above can be optimized efficiently using stochastic gradient descent, where at each iteration we randomly pick a *pair* of examples and compute the gradient with respect to them [30]. We can directly translate this idea to matrix factorization for link prediction. However, there is a subtlety in how we decide what pairs of examples to consider. Suppose $\mathcal{O}^+, \mathcal{O}^-$ are the sets of known present and known absent dyads respectively. Then, there are two ways in which we can construct pairs of nodes.

Per-node pairs. Here, we consider (known present, known absent) pairs that share one node in common. This can be thought as applying the AUC loss for every row of the G matrix. The optimization is

$$\min_{U,\Lambda} \frac{1}{|\mathcal{D}|} \sum_{i=1}^{n} \frac{1}{|\mathcal{O}_i^+||\mathcal{O}_i^-|} \sum_{j \in \mathcal{O}_i^+, k \in \mathcal{O}_i^-} \ell(u_i^T \Lambda (u_j - u_k), 1) + \Omega(U),$$

where $\mathcal{D}$ is set of all (i, j, k) triplets where $j \in \mathcal{O}_i^+, k \in \mathcal{O}_j^-$. In the case of logistic loss, the above model is similar to BPR [27], although the motivations of the two models are different: BPR was proposed to deal with implicit feedback (or positive only) collaborative filtering datasets. In the above, we are treating the known present links as the "positive feedback", and only links in $\mathcal{O}_i^-$ as being "unspecified feedback". Further, we combine the learned latent features with side information via bilinear regression.

All pairs. Here, we consider (known present, known absent) pairs that *need not* share a node in common. This can be thought as applying the AUC loss globally on every dyad in G. The optimization is

$$\min_{U,\Lambda} \frac{1}{|\mathcal{O}^+||\mathcal{O}^-|} \sum_{(i,j) \in \mathcal{O}^+, (i',j') \in \mathcal{O}^-} \ell(u_i^T \Lambda u_j - u_{i'}^T \Lambda u_{j'}, 1) + \Omega(U).$$

The choice between the two schemes depends on whether we ultimately evaluate AUC on a per-node or global basis. This choice in turn is largely problem specific; for example, in a social network we would like each individual user to have a good ranking, whereas in a protein-protein interaction network, our interest is in which unobserved dyads are worth performing further analysis on. Regardless of the choice of scheme, we are faced with the problem of having to learn from a large number of examples, one that is superlinear in the number of observed dyads. However, in practice, only a fraction of a single epoch of stochastic gradient descent is needed to achieve good results; we will discuss this more in our experiments.

3.4 The Final Model

Our final model optimizes for AUC directly, with regularization to prevent overfitting. We linearly combine side information via a bilinear regression model. Assuming we follow the per-node ranking scheme, and assuming per-node and per-dyad side information x_i and z_{ij} respectively, the objective is:

$$\min_{U,\Lambda,V,w,b} \frac{1}{|\mathcal{O}|} \sum_{i=1}^{n} \sum_{j \in \mathcal{O}_i^+, k \in \mathcal{O}_i^-} \ell(L(u_i^T \Lambda (u_j - u_k) + b_i + b_j + x_i^T V x_j + w^T z_{ij}), 1) + \frac{\lambda_U}{2} ||U||_F^2 + \frac{\lambda_\Lambda}{2} ||\Lambda||_F^2 + \frac{\lambda_V}{2} ||V||_F^2 + \frac{\lambda_w}{2} ||w||_2^2, \quad (5)$$

where the link function $L(\cdot)$ and loss $\ell(\cdot, \cdot)$ are user-specified, and $\Lambda = I$ if the graph is symmetric. Further, if required, we factorize the bilinear weights $V = D + A^T B$ for diagonal D and arbitrary A, B, and learn A, B, D.

4 Experimental Design

We present an experimental comparison of our model to other link prediction methods. We used datasets from a range of applications of link prediction:

- **Computational biology**
    - Protein-protein interaction data from [31], denoted "Prot-Prot". Each protein has a 76 dimensional feature vector.
    - Metabolic pathway interaction data for *S. cerevisiae* provided in the KEGG/PATHWAY database [36], denoted "Metabolic". Each node has three sets of features: a 157 dimensional vector of phylogenetic information, a 145 dimensional vector of gene expression information, and a 23 dimensional vector of gene location information.
- **Bibliographic networks**
    - The co-author network of authors at the NIPS conference [28], denoted "NIPS". Each node has a 14035 dimensional bag-of-words feature vector, being the words used by the author in her publications. We performed latent semantic indexing (LSI) to reduce the number of features to 100.
    - The co-author network of condensed-matter physicists [21], denoted "Condmat". Following [21], we consider node-pairs that are 2 hops away in the first five years of the data. There is no side information in this problem.
- **Other networks**
    - A network of military disputes between countries [13], denoted "Conflict". Following [34], we considered all disputes in the period 1990–2000. The graph is directed, with an edge originating from the conflict initiator. Due to time constraints, we only report results on the symmetrized version of the data, whether two countries have a link if either initiated conflict with the other. Each node has 3 features, comprising the country's population, GDP and polity, and additionally each dyad has 6 features, including e.g. the countries' geographic distance.
    - The US electric powergrid network [35], denoted "PowerGrid". There is no side information in this problem. This dataset is challenging because of the extreme imbalance (1 link for every 1850 non-links), and the nature of its linking behaviour: we expect two nodes to be linked if they are nearby geographically, which is a latent feature that may be difficult to infer.

To evaluate the models, we kept aside a fixed fraction of the observed dyads $\mathcal{O}$ for training various models, and evaluated AUC on a test set comprising the remaining dyads. This process was repeated 10 times, and we report the average test set AUC. We used two training split ratios: for the datasets with side information (Prot-Prot, Metabolic, NIPS and Conflict), we used 10% of dyads for training, and for the others (Condmat, PowerGrid), we used 90% of dyads for training. On the latter two datasets a 10% split would cause most nodes to have no known present links in the training set, making it difficult to make predictions based solely on the topological structure.

Table 2. Properties of datasets used in experimental comparison

Dataset	Nodes	$\lvert\mathcal{O}^+\rvert$	$\lvert\mathcal{O}^-\rvert$	+ve:−ve ratio	Average degree
Prot-Prot	2617	23710	6,824,979	1 : 300	9.1
Metabolic	668	5564	440,660	1 : 80	8.3
NIPS	2865	9466	8,198,759	1 : 866	3.3
Condmat	14230	2392	429,232	1 : 179	0.17
Conflict	130	320	16580	1 : 52	2.5
PowerGrid	4941	13188	24,400,293	1 : 1850	2.7

We swept over a grid of regularization parameters and learning rates for stochastic gradient descent, and evaluated the average AUC across the 10 splits. We report results for the parameters selected by the grid search. The number of latent features, k, was informally picked to be 30 for all datasets except Conflict, where only $k = 5$ were needed to achieve good performance. Our MATLAB scripts are available for download at `http://cseweb.ucsd.edu/~akmenon/code`.

Table 2 summarizes the dataset sizes in terms of number of nodes and known present/known absent dyads. Condmat is the only dataset where some edges have genuinely missing status even at test time (corresponding to node pairs more than 2 hops away). Note that we do not undersample any of the datasets. We see that PowerGrid is the most imbalanced dataset, while also having a small average degree.

5 Experimental Results

We divide the experimental results into three parts, each considering a different type of method. Methods with scores in **bold** have the highest score amongst methods being compared in that table. Methods that additionally have a **star** * are the best performing across all tables. In both cases, we only consider differences in AUC that are greater than the standard deviation across the splits.

Do latent features improve on unsupervised scores? We first report results on the following methods, which only exploit topological information:

- *Unsupervised scores.* We used Adamic-Adar (AA), Preferential Attachment (PA), Shortest-Path (SHP) and Katz, which are popular scores that perform well on a range of graphs [22]. We also ran linear regression on all unsupervised scores (Sup-Top), which attempts to find a weighted combination of the scores with better performance.
- *Raw SVD.* We computed the raw SVD on the adjacency matrix, treating known absent and unknown status edges as being one and the same.
- *Factorization.* We used the factorization model of Equation 3 using square-loss and identity link (Fact-Sq), and log-loss with sigmoid link (Fact-Log).
- *Unsupervised scores as input to factorization.* Here, we used all the unsupervised scores as features for each dyad, and fed them into a factorization model trained with square-loss (Fact+Scores).

Table 3. Test AUC scores for methods based on topological features alone

Dataset	AA	PA	SHP	Katz	Sup-Top
Prot-Prot	0.564 ± 0.005	0.750 ± 0.003	0.726 ± 0.005	0.727 ± 0.005	0.754 ± 0.003
Metabolic	0.524 ± 0.005	0.524 ± 0.005	0.626 ± 0.004	0.608 ± 0.007	0.628 ± 0.001
NIPS	0.512 ± 0.002	0.543 ± 0.005	0.517 ± 0.003	0.517 ± 0.003	0.542 ± 0.007
Condmat	0.567 ± 0.014	0.716 ± 0.026	0.673 ± 0.018	0.673 ± 0.017	0.720 ± 0.020
Conflict	0.507 ± 0.008	0.546 ± 0.024	0.512 ± 0.014	0.512 ± 0.014	**0.695 ± 0.076**
PowerGrid	0.589 ± 0.003	0.442 ± 0.010	0.659 ± 0.015	0.655 ± 0.016	**0.708 ± 0.062***

Dataset	SVD	Fact-Sq	Fact-Log	Fact+Scores
Prot-Prot	0.635 ± 0.003	**0.795 ± 0.005**	**0.793 ± 0.002**	**0.793 ± 0.005**
Metabolic	0.538 ± 0.017	**0.696 ± 0.001**	**0.695 ± 0.001**	**0.696 ± 0.002**
NIPS	0.512 ± 0.031	**0.612 ± 0.007**	**0.610 ± 0.008**	**0.613 ± 0.019**
Condmat	0.629 ± 0.051	**0.810 ± 0.020***	**0.822 ± 0.025***	**0.812 ± 0.020***
Conflict	0.541 ± 0.094	**0.692 ± 0.040**	**0.692 ± 0.039**	**0.689 ± 0.042**
PowerGrid	0.691 ± 0.026	0.637 ± 0.012	0.675 ± 0.017	**0.751 ± 0.020***

Our results are given in Table 3. (The table is split into two halves for visual clarity.) We make the following observations:

- Individual unsupervised scores are always outperformed by factorization methods by around 5%–25%. In most cases, factorization also outperforms a supervised combination of such scores. This suggests that in general, latent features better exploit topological information by virtue of directly optimizing to be predictive of link behaviour.
- In some cases, combining multiple topological features worsens test set AUC. The reason is likely twofold: first, a linear combination of weights may be too simplistic to leverage their combined power. Second, linear regression may underperform due to the imbalance of the data, as noted in Section 3.3.
- In most cases, combining latent features and unsupervised scores does not improve performance. This indicates that the unsupervised scores do not capture sufficiently complementary information to the latent features.
- As conjectured in Section 3.1, raw SVD performs much worse than the factorization methods on the datasets using 10% of dyads for training, as it treats all missing edges as being known absent.
- Amongst the two factorization approaches, the choice of loss function generally does not influence results significantly. For both losses, we required no more than 10 epochs to converge on any dataset.

How predictive is side information? Next, we tried several methods that use side information $x_i \in \mathbb{R}^d$ for each node i:

- *Raw similarity.* As a baseline, we use the cosine similarity $\frac{x_i^T x_j}{||x_i|| ||x_j||}$ as the predicted score for the node-pair (i, j).
- *Link propagation.* We use Link Propagation (LP) with the "sum kernel" as defined in [19], using cosine similarity as our base measure. We specifically use a special case of LP described in [19] that can be efficiently implemented in MATLAB using Lyapunov functions. We also tried an approximation to this method, Exact Link Propagation (ELP) [26].

Table 4. Test AUC scores for methods based on explicit features. Condmat and PowerGrid are not included because they do not have side information.

Dataset	Similarity	LP	ELP	ULR	BLR	Fact+LP	Fact+BLR
Prot-Prot	0.680 ± 0.002	0.771 ± 0.002	0.740 ± 0.003	0.670 ± 0.002	0.776 ± 0.006	0.789 ± 0.003	**0.813 ± 0.002***
Metabolic	0.605 ± 0.002	0.719 ± 0.001	0.659 ± 0.010	0.694 ± 0.007	0.725 ± 0.012	0.701 ± 0.002	**0.763 ± 0.006***
NIPS	**0.953 ± 0.000**	0.767 ± 0.004	0.929 ± 0.010	0.611 ± 0.007	**0.951 ± 0.002**	0.885 ± 0.032	0.945 ± 0.003
Conflict	0.577 ± 0.008	0.614 ± 0.016	0.648 ± 0.029	**0.869 ± 0.029***	**0.891 ± 0.017***	0.693 ± 0.046	**0.890 ± 0.017***

– *Regression.* We apply unilinear (ULR) and bilinear (BLR) regression on the feature vectors, corresponding to Equation 4 with the two choices of prediction function $f_M(\cdot, \cdot)$ discussed in Section 3.2, and where the latent features are omitted. Both methods did *not* use unsupervised scores as input, and were trained with square-loss.

– *Combinations.* We combined the factorization model with Link Propagation (Fact+LP) and bilinear regression (Fact+BLR), the latter being the model given in Equation 4 for bilinear $f_M(\cdot, \cdot)$.

We present the results in Table 4. (Condmat and PowerGrid are not included because they do not possess side information.) We observe the following:

– Bilinear regression is better than plain factorization on all datasets but Prot-Prot, which indicates that it is difficult to infer latent structure from the observed data that is more predictive than the given side information. This is not surprising given the sparsity of known present edges in the datasets.

– On Prot-Prot and Metabolic, jointly learning latent features and a bilinear regression model gives better performance than doing either individually. This suggests that despite the general superiority of explicit over latent features, the two can have complementary characteristics.

– The factorization model does not benefit from incorporating the output of LP. In fact, we find the test set AUC decreases on the NIPS dataset. On most datasets, LP had training AUC close to 1, suggesting that it is difficult to learn latent features on top of these scores without overfitting.

– Unilinear regression is always outperformed by bilinear regression, usually by a significant margin. This shows that the propensity problem with unilinear regression, discussed in §3.2, has important practical implications.

– Bilinear regression always outperforms both variants of Link Propagation.

Does optimizing for a ranking loss overcome imbalance? Finally, we check whether directly optimizing for AUC helps overcome imbalance. We apply the model of Equation 5 using square-loss (Fact-Rank), and consider an alternative where the ranking is defined over all dyads (Fact-Rank-Global) (see Section 3.3 regarding the per-node versus global ranking). We also optimize the BLR and Fact+BLR models of the previous section with the per-node form of ranking loss (BLR-Rank and Fact+BLR-Rank). On datasets with many dyads, the ranking losses only required a small fraction of a single epoch to converge (e.g.

Table 5. Test AUC scores for methods optimized with ranking loss

Dataset	Fact-Rank	Fact-Rank-Global	BLR-Rank	Fact+BLR-Rank
Prot-Prot	0.798 ± 0.001	0.794 ± 0.001	0.785 ± 0.003	**0.806 ± 0.003**
Metabolic	0.705 ± 0.007	0.706 ± 0.006	**0.764 ± 0.007***	**0.765 ± 0.007***
NIPS	0.609 ± 0.008	0.605 ± 0.007	0.949 ± 0.002	**0.956 ± 0.002***
Condmat	**0.814 ± 0.019***	**0.826 ± 0.019***	N/A	N/A
Conflict	0.690 ± 0.042	0.686 ± 0.042	**0.885 ± 0.018***	**0.886 ± 0.021***
PowerGrid	0.723 ± 0.015	**0.754 ± 0.014***	N/A	N/A

on PowerGrid, 1%–5% of an epoch for per-node ranking, and 0.01%–0.05% for global ranking). From the results in Table 5, we note that:

- For factorization methods, the ranking loss is dramatically superior to the regression losses on the PowerGrid dataset, which is the most imbalanced and has a small average degree. On other datasets, the differences are more modest, but the ranking loss is always competitive with the regression losses, while requiring significantly fewer dyads for convergence.
- For bilinear regression, optimizing with a ranking loss gives better performance than square loss on Prot-Prot and Metabolic.
- Jointly learning latent features and a bilinear regression model with a ranking loss performs at least as well as optimizing with square loss.

6 Conclusion

In the paper, we proposed a model that extends matrix factorization to solve structural link prediction problems in (possibly directed) graphs. Our model combines *latent features* with optional explicit features for nodes and edges in the graph. The model is trained with a *ranking loss* to overcome the imbalance problem that is common in link prediction datasets. Training is performed using stochastic gradient descent, and so the model scales to large graphs. Empirically, we find that the latent feature approach significantly outperforms popular unsupervised scores, such as Adamic-Adar and Katz. We find that it is possible to learn useful latent features on top of explicit features, which can give better performance than either model individually. Finally, we observe that optimizing with a ranking loss can improve AUC performance by around 10% over a standard regression loss. Overall, on six datasets from widely different domains, some possessing side information and others not, our proposed method (Fact-BLR-Rank from Table 5 on datasets with side information, Fact-Rank on the others) has equal or better AUC performance (within statistical error) than previously proposed methods.

References

1. Adamic, L.A., Adar, E.: Friends and neighbors on the web. Social Networks 25(3), 211–230 (2003)
2. Agarwal, D., Chen, B.-C.: Regression-based latent factor models. In: KDD 2009, pp. 19–28. ACM, New York (2009)
3. Airoldi, E., Blei, D.M., Fienberg, S.E., Xing, E.P.: Mixed membership stochastic blockmodels. In: NIPS, pp. 33–40 (2008)
4. Batagelj, V., Ferligoj, A., Doreian, P.: Generalized blockmodeling. Informatica (Slovenia) 23(4) (1999)
5. Beck, N., King, G., Zeng, L.: Improving quantitative studies of international conflict: A conjecture. American Political Science Review 94(1), 21–36 (2000)
6. Chawla, N.V., Japkowicz, N., Kotcz, A.: Editorial: special issue on learning from imbalanced data sets. SIGKDD Explor. Newsl. 6, 1–6 (2004)
7. Chen, H., Li, X., Huang, Z.: Link prediction approach to collaborative filtering. In: Joint Conference on Digital Libraries, vol. 7, pp. 141–142 (2005)
8. Chu, W., Park, S.-T.: Personalized recommendation on dynamic content using predictive bilinear models. In: WWW 2009, pp. 691–700. ACM, New York (2009)
9. Doppa, J.R., Yu, J., Tadepalli, P., Getoor, L.: Learning algorithms for link prediction based on chance constraints. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010. LNCS, vol. 6321, pp. 344–360. Springer, Heidelberg (2010)
10. Dunlavy, D.M., Kolda, T.G., Acar, E.: Temporal link prediction using matrix and tensor factorizations. ACM Transactions on Knowledge Discovery from Data (in Press, 2011)
11. Ruben Gabriel, K.: Generalized bilinear regression. Biometrika 85, 689–700 (1998)
12. Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., Schmidt-Thieme, L.: Learning attribute-to-feature mappings for cold-start recommendations. In: ICDM, pp. 176–185 (2010)
13. Ghosn, F., Palmer, G., Bremer, S.: The MID3 data set, 1993-2001: Procedures, coding rules, and description. Conflict Management and Peace Science 21, 133–154 (2004)
14. Hasan, M.A., Chaoji, V., Salem, S., Zaki, M.: Link prediction using supervised learning. In: SDM Workshop on Link Analysis, Counterterrorism and Security (2006)
15. Hoff, P.: Modeling homophily and stochastic equivalence in symmetric relational data. In: NIPS (2007)
16. Hoff, P.D.: Bilinear mixed effects models for dyadic data. Journal of the American Statistical Association 32, 100–286 (2003)
17. Hofmann, T., Puzicha, J., Jordan, M.I.: Learning from dyadic data. In: NIPS II, pp. 466–472. MIT Press, Cambridge (1999)
18. Joachims, T.: Optimizing search engines using clickthrough data. In: KDD 2002, pp. 133–142. ACM, New York (2002)
19. Kashima, H., Kato, T., Yamanishi, Y., Sugiyama, M., Tsuda, K.: Link propagation: A fast semi-supervised learning algorithm for link prediction. In: SDM, pp. 1099–1110 (2009)
20. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42, 30–37 (2009)
21. Lichtenwalter, R., Lussier, J.T., Chawla, N.V.: New perspectives and methods in link prediction. In: KDD, pp. 243–252 (2010)

22. Lu, L., Zhou, T.: Link prediction in complex networks: A survey (2010), http://arxiv.org/abs/1010.0725
23. Menon, A.K., Elkan, C.: A log-linear model with latent features for dyadic prediction. In: ICDM (2010)
24. Miller, K., Griffiths, T., Jordan, M.: Nonparametric latent feature models for link prediction. In: NIPS, vol. 22, pp. 1276–1284 (2009)
25. Mørup, M., Schmidt, M.N., Hansen, L.K.: Infinite multiple membership relational modeling for complex networks. In: NIPS Workshop on Networks Across Disciplines in Theory and Application (2010)
26. Raymond, R., Kashima, H.: Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010. LNCS, vol. 6323, pp. 131–147. Springer, Heidelberg (2010)
27. Rendle, S., Freudenthaler, C., Gantner, Z., Lars, S.-T.: BPR: Bayesian personalized ranking from implicit feedback. In: UAI 2009, pp. 452–461. AUAI Press, Arlington (2009)
28. Roweis, S.: NIPS dataset (2002), http://www.cs.nyu.edu/~roweis/data.html
29. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: SIGIR 2002, pp. 253–260. ACM, New York (2002)
30. Sculley, D.: Large scale learning to rank. In: NIPS Workshop on Advances in Ranking (2009)
31. Tsuda, K., Noble, W.S.: Learning kernels from biological networks by maximizing entropy. Bioinformatics 20, 326–333 (2004)
32. Vert, J.-P., Jacob, L.: Machine learning for in silico virtual screening and chemical genomics: New strategies. Combinatorial Chemistry & High Throughput Screening 11(8), 677–685 (2008)
33. Wang, C., Satuluri, V., Parthasarathy, S.: Local probabilistic models for link prediction. In: ICDM, pp. 322–331 (2007)
34. Ward, M.D., Siverson, R.M., Cao, X.: Disputes, democracies, and dependencies: A reexamination of the Kantian peace. American Journal of Political Science 51(3), 583–601 (2007)
35. Watts, D.J., Strogatz, S.H.: Collective dynamics of "small-world" networks. Nature 393(6684), 440–442 (1998)
36. Yamanishi, Y., Vert, J.-P., Kanehisa, M.: Supervised enzyme network inference from the integration of genomic data and chemical information. In: ISMB (Supplement of Bioinformatics), pp. 468–477 (2005)
37. Yang, S.H., Long, B., Smola, A., Sadagopan, N., Zheng, Z., Zha, H.: Like like alike – joint friendship and interest propagation in social networks. In: WWW (2011)
38. Zhu, S., Yu, K., Chi, Y., Gong, Y.: Combining content and link for classification using matrix factorization. In: SIGIR 2007, pp. 487–494. ACM, New York (2007)
39. Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. Technical report, CMU (2002)