PLUMS: Predicting Links Using Multiple Sources

Karthik Subbian* Arindam Banerjee* Sugato Basu†

Abstract

Link prediction is an important problem in online social and collaboration networks, for recommending friends and future collaborators. Most of the existing approaches for link prediction are focused on building unsupervised or supervised classification models based on the availability of accepts and rejects of the past recommendations. Several of these methods are feature-based and they construct a large number of network-level features to make the prediction more effective. A more flexible approach is to allow the model to learn the required features from the network for a specific task, rather than explicit feature engineering. In addition, most of the social and collaboration relationships do not happen instantly and rather build slowly over time through several low cost interactions, such as email and chat. The existing approaches often ignore the availability of such auxiliary networks to make link prediction more robust and effective. The main focus of this work is to build a robust and effective classifier for link prediction using multiple auxiliary networks. We develop a supervised random walk model, that does not require any explicit feature construction, and can be personalized to each user based on the past accept and reject behavior. Our approach consistently outperforms several popular baselines in terms of precision and recall in multiple real-life data sets. Also, our approach is robust to noise and sparsity in auxiliary networks, while several popular baselines, specifically feature-based ones, are inconsistent in their performance under such conditions.

1 Introduction

The problem of link prediction [1, 2] is to predict the edges (or links) in a graph that will form in the future, given all previous edge interactions. This problem is fundamental to several online social and collaboration networks [1]. It is used in recommending friends in social networks and collaborators in academic collaboration networks [3]. These relationships often do not form instantaneously; they rather slowly build over time, using several low risk interactions [4], such as informal chats, emails, and meetings. Given these multiple sources of other interactions, we hypothesize that the task of predicting friendship or collaboration in a network can be improved using these multiple sources of interaction.

In online interactions, single sign-on has made it possible to have access to a person's explicit interactions over multiple sources (e.g., email, calendar, chat), which can be tracked over time. The interactions could also be sometimes implicit, where two users may have high similarity profiles or interests. We have listed below a few scenarios where such implicit and explicit networks may be available for link prediction.

- Researchers from several highly related sub-fields in computer science, such as artificial intelligence, machine learning, computer vision and data mining, may chose to collaborate with their peers in the other sub-field. The relationships of an author in multiple related sub-fields act as a supporting evidence for predicting their future collaborations.
- In networks like Twitter one can construct retweet and mention networks, where repeated sharing and mentioning of user posts occur frequently. These networks can serve as additional evidence to predict new follower relationships.

These additional interaction networks (often referred to as *auxiliary networks*) can be used to understand the relationship similarities between users, in order to predict their future links that may form in the *primary network* of interest. Currently there is one work [5], as per our knowledge, that uses these auxiliary sources of information for link prediction. While this work explores the problem of link prediction from multiple networks, it restricts itself to path-based features and constructs paths of length up to three — the authors use these features for classification using L_1 and composite norm regularizers. In contrast, we want to explore the possibility of constructing a classifier with *no explicit feature engineering*, as several features found interesting in one application may not be effective in another. We show this in our experiments — several of these feature-based classifiers perform differently in different time periods, even on the same data set.

Our focus in this work is on supervised link prediction. The problem of link prediction can be addressed in both unsupervised [6, 1] and supervised [5, 7] scenarios. When the graph is specified until time t_1 and the links are predicted for time after t_1, the task is unsupervised. In a supervised scenario, the graph is given until time t_1 and the accepted and rejected user link information (e.g., friend invitations accepted or rejected by a user in a social network) between time t_1 and t_2 are used to supervise or train the model. The

*University of Minnesota, Minneapolis, MN 55455. {karthik,banerjee}@cs.umn.edu

†Google, Mountain View, CA. sugato@google.com

trained model is then used to predict the edges that will form after time t_2, using the graph information available until time t_2. In this paper, we focus on supervised link prediction, where the information about past user accepts and rejects is available.

In summary, we address the link prediction problem in the following setting: (1) using multiple auxiliary networks, (2) with no explicit feature engineering, and (3) supervised using the past accepts and rejects. To realize our goal of "no explicit feature engineering", we need a way to explore the network structure without having to explicitly construct features like common neighbors, number of possible two-hop paths, etc. A random walk on the graph is a natural choice. In a traditional random walk with restart [8], the walker is allowed to explore the network structure, given the edge probabilities. However, in our scenario, the walker must be made aware of users' accepts and rejects, such that walker decides to walk along paths that closely reflect the user behavior and interests. Moreover, we need to accommodate *multiple sources* in the random walk process, where the random walker is allowed to traverse across multiple graphs though different types of edges connecting a pair of nodes (for instance, a pair of nodes may have chat and email interactions).

Interestingly, there is a recent work on supervised random walk model [8], using the accepts and rejects of past link recommendations. The focus of this work is to learn the edge propagation probabilities using node and edge level features for a single network, using a random walk model. However, our approach uses multiple auxiliary networks and learns to combine the information from multiple such networks to perform the task of link prediction.

We set up the link prediction problem in an optimization setting, where the objective is to train a model that maximizes the user accepts and minimizes the user rejects (i.e. maximize the accuracy of link recommendation), while preserving the random walk constraint. Each user is different and can behave in a unique way — to address that, we need a personalized version of our approach, where we train the model specific to individual users. We develop both a general and personalized random walk approach — in the general approach we learn model parameters at a network-level, while in the personalized approach we learn parameters at the user-level.

The key contributions of this paper are as follows. We develop a *supervised* random walk model for *multiple networks* to perform link prediction, with *no explicit feature engineering*. We also have developed a *personalized* version of the model, where the model trained is specific to each user. We show that our approach is a *generalization of Katz centrality*. We empirically demonstrate the *superiority of our approach in terms of precision and recall*. We show that our approach is *robust to sparse and noisy auxiliary networks*, compared to several recent link prediction approaches [5, 9] that use explicit features and multiple sources.

1.1 Related Work Networks are often heterogeneous in terms of nodes and edges properties, several techniques have been proposed to address link prediction for heterogeneous networks [10, 11, 12, 7]. Several other related works discuss friendship recommendation using auxiliary sources of information [13, 14, 15, 16]. In [13] location based recommendation is performed using multiple network interactions, while assuming the target network is new and sparse. There are other related works [17, 8] on random walk based approaches for neighborhood search and recommendation in heterogeneous networks. Some of these works focus on using edge or node attributes [8], while others [18] make use of the sparsity of the multiple networks. There are a few data-driven approaches [19], to predict links in bibliographic networks using meta-paths. However, none of these approaches tend to learn by supervising a random walk model based on the past accepts and rejects of links in the networks, especially with no strong assumptions on the underlying network structure. A detailed survey of link prediction techniques can be found in [9, 20, 2]. More discussion on link prediction for evolving networks can be found in [21].

In a more recent work [5], Lu et al. considers explicit feature-based classification with L_1 and composite norm regularizers for the problem of link prediction in multiple networks. The L_1 regularizer does feature selection while performing classification. On the other hand, composite norm knocks out features and its subsets based on an underlying overlapping group structure [5]. In this approach, as the length of path k or the number of auxiliary network n increases, the number of features increases significantly, in the order of $O(n^k)$. In our approach, we avoid explicit extraction of features and do a network-level training. As we demonstrate through experiments, our approach is robust to noise as we do not specify any features explicitly.

2 Problem Formulation

Let $\mathbf{A_1}, \ldots, \mathbf{A_n}$ be the (weighted) adjacency matrix of n networks with m nodes, where $\mathbf{A}_k \in \Re_+^{m \times m}$. These networks represent different interactions between the users. For instance, $\mathbf{A}_1$ could represent email interactions, $\mathbf{A}_2$ could represent calendar invite interactions, and $\mathbf{A}_3$ represent chat interactions, etc. Let $\mathbf{P}_k$ denote the normalized row stochastic version of $\mathbf{A}_k$, where $\mathbf{P}_k(u, v)$ can be computed as:

$$\mathbf{P}_k(u, v) \triangleq \mathbf{P}_k(v|u) = \frac{\mathbf{A}_k(u, v)}{\sum_{w \in N(u)} \mathbf{A}_k(u, w)} \tag{2.1}$$

The set of neighbors of node u is denoted by $N(u)$. A random walk with restart can be performed using $\mathbf{P_k}$ with a restart probability of α — the resulting stationary distribution of such a random walk, for network k, is given

in (2.2).

$$\boldsymbol{\pi}^{(k)} = (1-\alpha)\mathbf{P}_k^T\boldsymbol{\pi}^{(k)} + \alpha\boldsymbol{\gamma} \tag{2.2}$$

where the bias vector $\boldsymbol{\gamma} \in \Delta_m$ contains probabilities of restarting at individual nodes at each restart, and Δ_m denotes the m-dimensional probability simplex. For our purposes, we will consider a bias vector specific to each user. In particular, for user i, we will have $\boldsymbol{\gamma} = \mathbf{e_i}$, where $\mathbf{e}_i$ is a vector with the i-th element equal to 1 and the other elements set to 0. Then, for user i, the stationary distribution is given by

$$\boldsymbol{\pi}_i^{(k)} = (1-\alpha)\mathbf{P}_k^T\boldsymbol{\pi}_i^{(k)} + \alpha\mathbf{e_i}\ . \tag{2.3}$$

Note that $\mathbf{e_i}$ is the i-th corner of the Δ_m simplex, and the random walk with restart corresponding to π_i always restarts from this corner.

Given n networks, with each network containing m users, the goal of our approach is to make meaningful recommendations by finding similar users, traversing along different network paths. This can be done by combining these n different networks using a convex combination of row stochastic matrices, equivalently, $\mathbf{P}(\mathbf{x}) = \sum_{k=1}^n x_i\mathbf{P}_i$, where $x_i \geq 0$ and $\sum_i x_i = 1$. The random walk with restarts are performed over this convex combination $\mathbf{P}$, and the resulting random walk probabilities span multiple networks. In particular, for user i, the stationary distribution satisfies the following equation: $\boldsymbol{\pi}_i = (1-\alpha)\sum_{k=1}^n x_k\mathbf{P}_k^T\boldsymbol{\pi}_i + \alpha\mathbf{e_i}$.

As an alternative, one might consider doing independent random walks on n different networks and consider them as features in a classification model (e.g., logistic regression). However, this independent treatment may not achieve the goal of capturing cross-network interactions. Consider an example where nodes i and j are connected via a chat interaction and nodes j and k are connected an email interaction. Using n independent random walks the two hop closeness of i to k can never be realized. However, if we overlap the networks and combine the edge weights using a convex combination, the random walker can traverse from i to k in two hops. Formally, the convex combination of stationary distributions for multiple network is given by $\pi_i = \sum_k x_k\pi_i^{(k)}$ and by substituting (2.3) in this equation we can easily see $\mathbf{P}^T\pi_i \neq \sum_k x_k\mathbf{P}_k^T\pi_i^{(k)}$. Hence it is important to consider a combination of these networks, rather than treating them independently.

2.1 Exact and Approximate Formulations Online social and collaboration networks often have sufficient ground truth of past accepts of recommendations in the network. Let us denote this with $\mathbf{y}_i \in \{-1,1\}^{m\times 1}$, where m is the number of users in the network, $\mathbf{y}_i(j) = 1$ denotes the *accept* of a recommendation of user j by user i, and $\mathbf{y}_i(j) = -1$ denotes the rejection. Our model is general enough where $\mathbf{y}_i$ can denote accepts, rejects and did-not-care cases with 1, -1 and 0 respectively. In this paper, for simplicity, we deal with 1 and -1 case only. The goal of our link recommendation is to be consistent with past recommendations, while making effective future predictions. So, we learn the weights for the convex combination $\mathbf{x} = [x_1,\ldots,x_n]$ such that our new model is consistent with previous interactions. We set this up as the following optimization problem, which we refer to as the exact formulation since (2.2) is used as a constraint:

$$\min_{\mathbf{x},\boldsymbol{\Pi}} \quad -\sum_{i=1}^m \mathbf{y}_i^T\boldsymbol{\pi}_i$$

subject to

$$\boldsymbol{\pi}_i = (1-\alpha)\left(\sum_{k=1}^n x_k\mathbf{P}_k^T\right)\boldsymbol{\pi}_i + \alpha\mathbf{e}_i,\ \forall i \tag{2.4}$$

$$\sum_{k=1}^n x_k = 1\ ,$$

$$x_k \geq 0\ ,\ k = 1,\ldots,n.$$

One way to make the top-k recommendations for user i is to select leading k nodes that have highest k values in terms of the stationary probabilities in π_i. As our model recommendations must also be inline with the previous user accepts and rejects, we consider a linear objective as shown in (2.4). The objective of the formulation (2.4) ensures that the accepted recommendations receive a higher personalized random walk score compared to the ones not accepted and the random walk constraint ensures the coefficients $x_1,\ldots,x_k$ are adjusted such that the accepted friends are treated closer than the ones not accepted.

The objective of formulation (2.4) is linear, for a particular value of x. However, we do not know x and learning it alongside π_i makes the random walk constraint bi-linear and the underlying problem more challenging. We relax this problem further, by considering an approximate version of π_i where, $\boldsymbol{\pi}_i \approx (1-\alpha)\left(\sum_{k=1}^n x_k\mathbf{P}_k^T\right)\boldsymbol{\pi}_i + \alpha\mathbf{e}_i,\ \forall i$. We consider a specific instantiation of such an approximate formulation as follows:

$$\min_{\mathbf{x},\boldsymbol{\Pi}} \quad -\sum_{i=1}^m \mathbf{y}_i^T\boldsymbol{\pi}_i \tag{2.5}$$

subject to

$$\sum_{i=1}^m \left\|\boldsymbol{\pi}_i - (1-\alpha)\left(\sum_{k=1}^n x_k\mathbf{P}_k^T\right)\boldsymbol{\pi}_i - \alpha\mathbf{e}_i\right\|^2 \leq \epsilon$$

$$\sum_{k=1}^n x_k = 1\ ,$$

$$x_k \geq 0\ ,\ k = 1,\ldots,n.$$

In the inequality constraint, ϵ determines how strictly the constraint is enforced, and $\epsilon = 0$ yields the formulation in (2.4) as a special case. The approximate formulation has more flexibility, and a non-zero value of ϵ may lead to better predictive performance in practice. We revisit the approximate formulation in the context of algorithm development in Section 3 and subsequent empirical evaluation in Section 4.

2.2 Relationship to Katz Similarity The Katz similarity matrix is defined as $\sum_{j=0}^{\infty} (\beta \mathbf{A})^j$ [1], where β is a decay factor that lies between 0 and 1, $\mathbf{A}$ is the adjacency matrix of the underlying graph. An entry in the Katz similarity matrix $\mathbf{K}_{ij}$ for a pair of nodes i and j is the weighted average of different path lengths reachable from node i to j. The weights assign decreasing importance to paths of increasing length.

We show that the objective of the original formulation in (2.4) reduces to $-\operatorname{Tr}\left(\mathbf{Y}^T \sum_j (1-\alpha)^j \left(\mathbf{P}^T\right)^j\right)$, where $\mathbf{P} = \sum_k x_k \mathbf{P}_k^T$, using the closed form solution of $\mathbf{\Pi}$. When $\mathbf{A} = \mathbf{P}_i, \forall i$ and $\beta = (1-\alpha)$, in this reduced form, Katz similarity matrix becomes a special case of the objective used in our approach. This implies when all the n networks are exactly same as each other which is equal to $\mathbf{A}$ and the decay factor equal (1 - restart probability) then supervised random walk similarity measure is same as the Katz similarity. But, in our case we have different networks and we wish to learn the importance of each network from the underlying ground truth. Thus, our approach uses a generalized version of Katz similarity measure, for the multiple network scenario.

3 Algorithms and Analysis

The formulation in (2.4) has a linear objective and a bi-linear constraint containing $\mathbf{x}$ and $\mathbf{\Pi}$. One can consider a few different approaches to solve the problem, e.g., considering the Lagrangian relaxation by introducing dual variables and doing dual ascent [22] [23], writing $\boldsymbol{\pi}$ in closed form in terms of $\mathbf{x}$ and subsequently using a projected gradient descent method on $\mathbf{x}$ [22]. For algorithm development, we focus on the approximate formulation in (2.5), which is arguably more general. The problem in (2.5) can be equivalently characterized as follows:

(3.6)
$$\min_{\mathbf{x},\mathbf{\Pi}} L_\lambda(\mathbf{x},\mathbf{\Pi}) = -\sum_{i=1}^{m} \mathbf{y}_i^T \boldsymbol{\pi}_i + \frac{\lambda}{2} \sum_{i=1}^{m} \left\| \left(\boldsymbol{\pi}_i - (1-\alpha) \left(\sum_{k=1}^{n} x_k \mathbf{P}_k^T \right) \boldsymbol{\pi}_i - \alpha \mathbf{e}_i \right) \right\|_2^2 ,$$

where $\lambda = \lambda(\epsilon)$ is a suitable non-negative constant corresponding to ϵ. Here λ can be viewed as a Lagrangian multiplier so that high values of ϵ correspond to low λ and low/zero values of ϵ corresponds to large values of λ. For our algorithms, we consider λ to be a constant which will be determined in practice by cross-validation on predictive performance (see Section 4).

3.1 PLUMS The problem in (3.6) is not jointly convex in $(\mathbf{x}, \Pi)$, but is convex in each variable while the other is fixed. Based on such structure, we focus on developing an alternating minimization algorithm which updates Π and $\mathbf{x}$ alternately. In particular, we focus on an alternating mirror descent type updates till convergence. For suitable proximal functions d_1, d_2 we consider the following alternating updates:

(3.7)
$$\mathbf{\Pi}^{t+1} = \operatorname*{argmin}_{\mathbf{\Pi}} \eta \langle \nabla_{\mathbf{\Pi}} L_\lambda(\mathbf{\Pi}^t, \mathbf{x}^t), \mathbf{\Pi} \rangle + d_1(\mathbf{\Pi}, \mathbf{\Pi}^t)$$
$$\mathbf{x}^{t+1} = \operatorname*{argmin}_{\mathbf{x} \in \Delta} \eta \langle \nabla_{\mathbf{x}} L_\lambda(\mathbf{\Pi}^{t+1}, \mathbf{x}^t), \mathbf{x} \rangle + d_2(\mathbf{x}, \mathbf{x}^t) ,$$

where $\eta > 0$ is the learning rate. Note that the updates on $\mathbf{\Pi}$ are unconstrained whereas $\mathbf{x} \in \Delta$, the unit simplex. For the purposes of our experiments, we choose both the proximal functions to be the squared Euclidean distance, which yields a simple gradient descent update for $\mathbf{\Pi}$ and a projected gradient descent update for $\mathbf{x}$:

$$\mathbf{\Pi}^{t+1} = \mathbf{\Pi}^t - \eta \nabla_{\mathbf{\Pi}} L_\lambda(\mathbf{\Pi}^t, \mathbf{x}^t) \tag{3.8}$$
$$\mathbf{x}^{t+1} = \operatorname{Proj}_\Delta(\mathbf{x}^t - \eta \nabla_{\mathbf{x}} L_\lambda(\mathbf{\Pi}^{t+1}, \mathbf{x}^t)) . \tag{3.9}$$

The projection step for $\mathbf{x}$ is simply an Euclidean projection to the simplex, for which efficient algorithms exist [24]. The gradients for $\mathbf{\Pi}$ and $\mathbf{x}$ are listed below in (3.10) and (3.11) respectively.

$$\begin{aligned} \nabla_{\mathbf{\Pi}} L_\lambda(\mathbf{x},\mathbf{\Pi}) = {} & \lambda \Pi - \lambda(1-\alpha)(\mathbf{P}(\mathbf{x})^T + \mathbf{P}(\mathbf{x}))\mathbf{\Pi} \\ & + \lambda(1-\alpha)^2 \mathbf{P}(\mathbf{x})\mathbf{P}(\mathbf{x})^T \mathbf{\Pi} \\ & + \lambda\alpha(1-\alpha)\mathbf{P}(\mathbf{x}) - \lambda\alpha\mathbb{I} - \mathbf{Y} . \end{aligned} \tag{3.10}$$

$$\begin{aligned} \nabla_{x_k} L_\lambda(\mathbf{x},\mathbf{\Pi}) = {} & \lambda(1-\alpha)^2 \operatorname{Tr}(\mathbf{\Pi}^T \mathbf{P}(\mathbf{x}) \mathbf{P}_k^T \mathbf{\Pi}) \\ & - \lambda(1-\alpha) \operatorname{Tr}(\mathbf{\Pi}^T \mathbf{P}_k^T \mathbf{\Pi}) \\ & - \lambda\alpha(1-\alpha) \operatorname{Tr}(\mathbf{P}_k^T \mathbf{\Pi}) . \end{aligned} \tag{3.11}$$

Putting $\mathbf{x} = \mathbf{x}^t$ and $\mathbf{\Pi} = \mathbf{\Pi}^t$ in (3.10) will give the desired gradient for the $\mathbf{\Pi}$-update in (3.8). Similarly, setting $\mathbf{x} = \mathbf{x}^t$ and $\mathbf{\Pi} = \mathbf{\Pi}^{t+1}$ in (3.11) will give the desired gradient for the $\mathbf{x}$-update in (3.9)

3.2 pPLUMS: Personalized PLUMS The PLUMS approach discussed in the previous section may be limited in terms of its ability to learn specific user likes and dislikes, as we learn parameters only at a network-level. Consider

a scenario where a user is actively using the email service more heavily than the chat service. In such case, the significance of email network for that user should be higher compared to the chat service. For this purpose, we consider a personalized PLUMS approach, and refer to it as *pPLUMS*. The formulation follows from (3.6) except that the coefficients x_k corresponding to each network are replaced by diagonal matrix $\mathbf{D}_k$. The entry $\mathbf{D}_k(i,i)$ denotes the importance of the node i on the network k, and we assume $D_k(i,i) \geq 0, \sum_k D_k(i,i) = 1$ for all i. The corresponding objective is given by

(3.12)
$$\min_{\mathbf{D}_k, \mathbf{\Pi}} L_\lambda(\mathbf{D}, \mathbf{\Pi}; \lambda) = -\sum_i \mathbf{y}_i^T \pi_i + \frac{\lambda}{2} \sum_i \left\| \left(\pi_i - (1-\alpha) \left(\sum_k D_k \mathbf{P}_k^T \right) \pi_i - \alpha \mathbf{e_i} \right) \right\|_2^2$$

We use the same alternating (projected) gradient descent approach to solve (3.12). Let $\mathbf{P}(\mathbf{D}) = \sum_k \mathbf{D}_k \mathbf{P}_k$. Then, following a similar analysis as earlier, the gradient w.r.t. $\mathbf{\Pi}$ is given by

$$\begin{aligned} &\nabla_{\mathbf{\Pi}} L_\lambda(\mathbf{D}, \mathbf{\Pi}) \\ &= \lambda \Pi - \lambda(1-\alpha)(\mathbf{P}(\mathbf{D})^T + \mathbf{P}(\mathbf{D}))\mathbf{\Pi} \\ &\quad + \lambda(1-\alpha)^2 \mathbf{P}(\mathbf{D})\mathbf{P}(\mathbf{D})^T \mathbf{\Pi} \\ &\quad + \lambda\alpha(1-\alpha)\mathbf{P}(\mathbf{D}) - \lambda\alpha\mathbb{I} - \mathbf{Y} \,. \end{aligned} \tag{3.13}$$

A comparison between (3.10) and (3.13) illustrates the mild difference between the two updates, where $\mathbf{P}(\mathbf{x})$ is replaced by $\mathbf{P}(\mathbf{D})$.

Also, a similar analysis also yields the gradient w.r.t. the diagonal matrix $\mathbf{D}_k$ to be

$$\begin{aligned} \nabla_{D_k} L_\lambda(\mathbf{D}, \mathbf{\Pi}) &= \lambda(1-\alpha)^2 \operatorname{diag}\left(\mathbf{\Pi}\mathbf{\Pi}^T \mathbf{P}(\mathbf{D}) \mathbf{P_k^T}\right) \\ &\quad - \lambda\alpha(1-\alpha)\operatorname{diag}\left(\mathbf{\Pi}\mathbf{P}_k^T\right) \\ &\quad - \lambda(1-\alpha)\operatorname{diag}\left(\mathbf{\Pi}\mathbf{\Pi}^T \mathbf{P}_k^T\right). \end{aligned} \tag{3.14}$$

The key difference between (3.14) for pPLUMS and its simpler counterpart (3.11) in PLUMS is the following: PLUMs considers an inner product between factors leading to a scalar gradient for x_k, whereas pPLUMS considers an element-wise product between factors leading to a diagonal gradient for $\mathbf{D}_k$. The factors involved in the two updates are the same, except for $\mathbf{P}(\mathbf{x})$ being replaced by $\mathbf{P}(\mathbf{D})$ and $\mathrm{Tr}(.)$ operator is replaced by the $\mathrm{diag}(.)$ operator. The algorithm for PLUMS is shown in Fig. 1. Replace $\mathbf{x}$ with $\mathbf{D}$ and corresponding gradients, in Fig. 1, for a personalized version of this algorithm.

4 Experimental Results

In this section, we describe the data sets, baselines, evaluation measures and results of our experiments.

Algorithm *PLUMS*$(P_1, \ldots, P_n)$
begin
  Initialize $x_k^{(0)} = 1/n$, $\Pi^{(0)} = 1/n \cdot \mathbf{1}$, $t = 0$, λ, max_t
  Initialize $\mathbf{P} = \sum_k x_k \mathbf{P}_k$, ϵ_1, ϵ_2, converged = **false**
  $L^{(t)} = L_\lambda(\mathbf{\Pi}^{(t)}, \mathbf{x}^{(t)})$
  do
    % Compute$\nabla_{\mathbf{x}} L_\lambda(\mathbf{\Pi}^{(t)}, \mathbf{x}^{(t)})$
    $\mathbf{x}^{(t+1)} = \mathrm{Proj}_\Delta(\mathbf{x}^{(t)} - \eta \nabla_{\mathbf{x}} L_\lambda(\mathbf{\Pi}^{(t)}, \mathbf{x}^{(t)}))$
    % Compute $\nabla_{\mathbf{\Pi}} L_\lambda(\mathbf{\Pi}^{(t)}, \mathbf{x}^{(t+1)})$
    $\mathbf{\Pi}^{(t+1)} = \mathbf{\Pi}^{(t)} - \eta \nabla_{\mathbf{\Pi}} L_\lambda(\mathbf{\Pi}^{(t)}, \mathbf{x}^{(t+1)})$
    $L^{(t+1)} = L_\lambda(\mathbf{\Pi}^{(t+1)}, \mathbf{x}^{(t+1)})$
    if $((\left\| x^{(t+1)} - x^{(t)} \right\|_1 \leq \epsilon_1)$ **or**
      $(L^{(t)} - L^{(t+1)} \leq \epsilon_2)$ **or** $(t >= max\_t))$
        converged = **true**
    endif
    $t = t + 1$
  while(**not** converged)
end

Figure 1: PLUMS Algorithm

4.1 Data Sets We used three kinds of data sets for our evaluation purposes: (1) DBLP, (2) Protein-Protein (P-P) interaction and (3) synthetic networks.

4.1.1 Synthetic Data Set We have prepared the synthetic data set by first constructing the primary network using a power law graph, as most of the social and biological networks are scale-free. We used the BTER [25] tool to construct the synthetic graph for the power law exponent of 2.5 and 3, as the typical value in actual networks is in the range of 2 to 3. We refer to these two data sets as Synthetic A and Synthetic B respectively. We then generate two Auxiliary networks, representing email and chat interaction. These networks are simulated using a conditional probability table, where the presence or absence of an edge in the primary network is the conditioning variable. We generated graphs with 1000 nodes each. We randomly timestamped each edge in the all networks in to time bins t_1, t_2 and t_3. The edges in time t_1 are used as the training network and t_2 as the corresponding ground truth for the training period. Similarly, all edges in t_1 and t_2 form the testing network, while edges in t_3 are used as the ground truth for the testing network. We sampled the negative training data uniformly at random from all possible negative edges available. We maintained equal number of positive and negative samples while constructing negative samples [26].

4.1.2 DBLP Data Set In DBLP we extracted all the publications from 1996 to 2011 for top 5 venues, as listed in `http://academic.research.microsoft.com/`, for four different sub-fields of computer science: Machine Learning (ML), Information Retrieval (IR), Data Mining (DM), and Operating Systems (OS). We construct two

subgroups of data set from these fields: (1) IR (Primary), ML (Auxiliary 1), DM (Auxiliary 2) and (2) IR (Primary), DM (Auxiliary 1), OS (Auxiliary 2). Here, "Primary" indicates the network for which prediction is performed, while "Auxiliary" indicates the supporting networks that are used to improve the link prediction performance. For each of these subgroups, we created partitions of data set over three time periods: (a) 2001-2005, (b) 2006-2008 and (c) 2009-2011. For five year windows we split them in to 2, 2, and 1 years for time bins t_1, t_2, and t_3 respectively and for three year window all time bins are of equal size of one year each. Once the edges are assigned to one of these time bins, the training and test data preparation follows the same procedure as described for the synthetic data set. To deal with non-common authors in each sub-field, we consider the union of authors in all three networks as the complete set of authors in our experiments. We refer to the IR*-ML-DM data set over the three time periods 2001-2005, 2006-2008 and 2009-2011 as DBLP1a, DBLP1b and DBLP1c respectively. Similarly, we refer the IR*-DM-OS data set for three time periods as DBLP2a, DBLP2b, DBLP2c respectively. We use the superscript '*' to denote the primary network.

4.1.3 Protein-Protein Interaction Data Set We downloaded the protein-protein interaction data set for the *african aquatic frog* (Xenopus laevis) from `thebiogrid.org`. We used the following three interaction types to construct the primary and two Auxiliary networks: Physical Association (Primary), Direct Interaction (Auxiliary 1), and Colocalization (Auxiliary 2). We hypothesize that genetic interactions in Auxiliary networks can be helpful in finding unknown interactions in primary network. The edges here are undirected, unweighted and do not have timestamps. We split them randomly in ratios 40:30:30 to three bins t_1, t_2 and t_3 respectively. Then, the training and test data is prepared using the same procedure as described for the synthetic data.

4.2 Evaluation Measure We use precision, recall and area under the *precision-recall* (P-R) curve to measure the effectiveness of our approach. The threshold curves, such as P-R curves, are found to more fair and effective in measuring the classifier performance for an entire operating range than looking at a point-wise precision or recall measures [26]. We compute the precision and recall measures as commonly used in information retrieval applications. We sweep the *top-k* from 1 to 500 for each user (i.e. node) to obtain the top-k predictions (retrieved) for the proposed approach and each baseline. The relevant predictions are available from the ground truth. Given the relevant and retrieved predictions, the precision and recall measures are averaged over 10 different runs.

4.3 Baselines Our choice of baselines is based on popularity and relevance, for the problem of link prediction using multiple sources.

LOGISTIC REGRESSION: Each training data point for the logistic regression classifier is an edge and the features include both independent and combined network features. The independent network features are outdegree, common neighbors, outdegree edge weight, and personalized page rank scores normalized over all outgoing neighbors. The normalized edge weight score is zero if the members of the edge are not immediate. The independent network features are constructed for both primary and Auxiliary networks. The combined network features are constructed by combining the networks with equal weights, and the features include, common neighbors, number of two hop paths, outdegree, and personalized page rank score. The personalized page rank is constructed with the restart probability of 0.15.

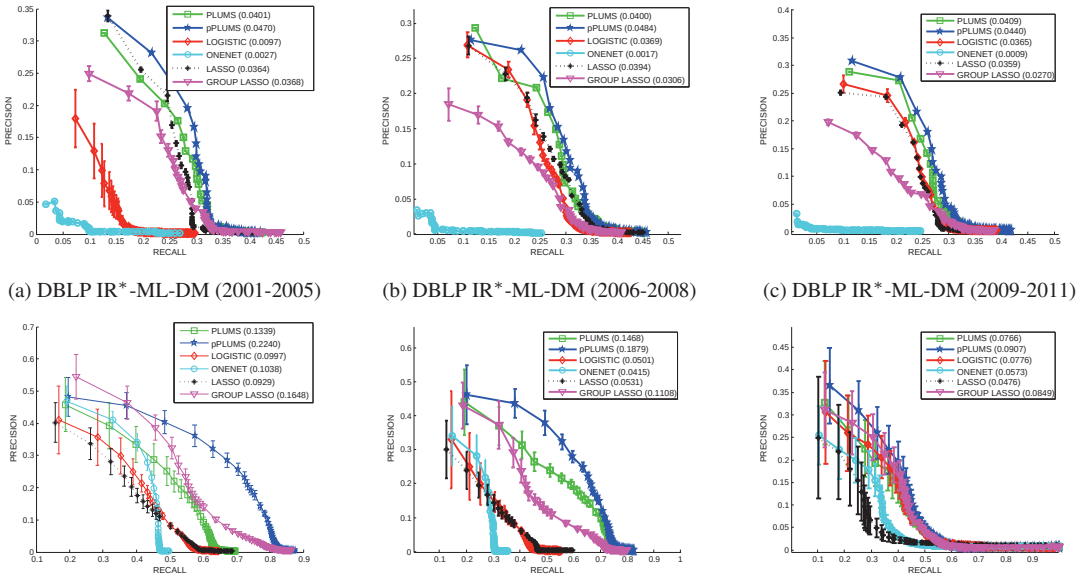
LASSO: A data point in the Lasso classifier is an edge, where the features are constructed as described in [5]. The features constructed were for one, two and three hop paths across different permutations of the network. For one primary and two auxiliary networks, there are 39 features that were constructed for this classifier. In addition, we also used all the combined network features we created for logistic classifier.

GROUP LASSO: The group lasso classifier uses the same set features as LASSO. The composite norm used for this classifier uses the overlapping group structure, where a subset of path patterns found not useful is used to knock out supersets containing those subsets. This hierarchical sparsity is implemented using a Directed-Acyclic-Graph (DAG) structure as described in [5].

ONENET: This baseline uses the personalized random walk scores computed on the primary network structure, with no auxiliary network information. This is used to evaluate the benefit obtained by using the additional network structures. The restart probability of random walk was set to 0.15.

4.4 Experimental Results In this section, we empirically evaluate the baselines and our approach for model effectiveness and robustness.

4.4.1 Effectiveness Analysis The effectiveness of our approach is measured in terms of precision-recall (P-R) curves. The P-R curves for Synthetic, DBLP and BIOGRID are shown in Figure 2, where x-axis measures the recall and y-axis the precision. We computed the values for various points by sweeping the top-k predictions for each user from 1 to 500. As the number of predictions for each user increases, the total number of predictions that overlap with ground truth increases and hence the recall increases. Meanwhile, the total number of positive predictions increases faster compared to the true positives and the precision decreases. We averaged the precision recall values for ten runs and show the deviation as error bars in these charts. The *Area Under the*



(a) DBLP IR*-ML-DM (2001-2005) (b) DBLP IR*-ML-DM (2006-2008) (c) DBLP IR*-ML-DM (2009-2011)

(g) Synthetic A (h) Synthetic B (i) BioGrid PA*-DI-CO (FROG)

Figure 2: Experimental results for DBLP, Synthetic and BIOGRID data sets. The primary network is denoted by a superscript * and the Auxiliary network 1 and 2 are listed next to primary network in that order. The years used for experiments for DBLP data set is listed in the brackets. For synthetic data set A, the power law exponent was set to 2.5 and for B it was set to 3. The species *Xenopus Laevis* (Frog), is used for protein-protein interaction networks.

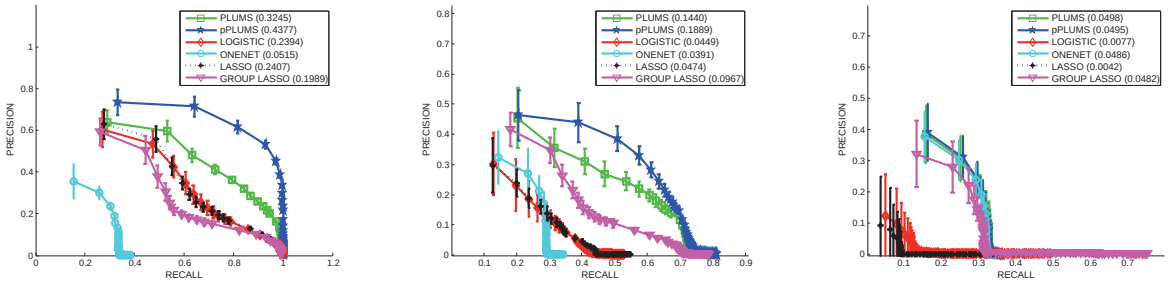
Curve (AUC) for the P-R curve is shown in brackets in the legend for each method. Higher the value of AUC, better the performance of the approach. The maximum value of AUC is 1 and minimum is 0.

The DBLP1 data set considers IR co-authorship as the primary network and the auxiliary networks are ML and DM co-authorship networks. We find that in all our DBLP experiments *pPLUMS approach consistently outperforms all baseline methods*, including our non-personalized version, PLUMS. We ordered the baselines for DBLP data set using the average AUC performance and across all three time periods. Our pPLUMS and PLUMS methods take the first and second spot respectively, followed by the Lasso and Group-Lasso baselines. The *best performing baseline Lasso is* 26% *lesser than pPLUMS in terms average AUC* measure. The Logistic has an average AUC of 0.0365 and using random walks with restart only in the primary network performs the worst with an average AUC of 0.0009 — the main reason for this poor performance is the number of new connections that are formed in the primary network due to collaborations in auxiliary interactions are much more stronger. As preferential attachment plays an important role in these networks, the leading scientists or researchers attract a majority of collaborators in the auxiliary network where they are well-known. The random walk performed across multiple networks brings together these central people under one network, in order to make new predictions more effective. Using one network misses out on several of these leading researchers that are actually driving the collaboration through other auxiliary networks, which are not explicitly visible in the primary network.

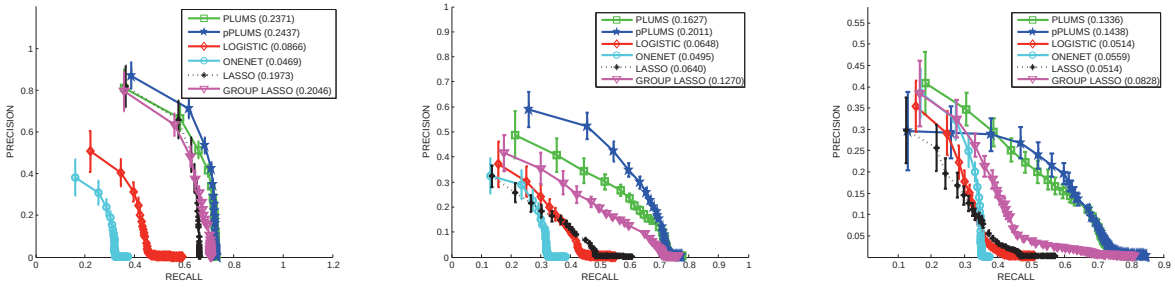
In the synthetic data set, we created different networks each time we ran the algorithm using the BTER model [25]. In Synthetic-A the power law exponent was set to 2.5 and in B it was set to 3. We see that the performance of pPLUMS and other baselines considerably decrease, in terms of the AUC value, as the exponent value increases. The reason for this is number possible future connections becomes lower as the graph becomes denser in close neighborhoods. Thus, the precision drops sharply as we sweep through the threshold of more than 4 or 5. pPLUMS consistently performed best in all runs and in average, compared to all baselines, in terms of AUC.

For protein-protein interaction networks, *Group-Lasso closely competes with our pPLUMS* approach — all other baselines performed poorly compared to pPLUMS. The deviation of AUC values in this data set was higher compared to other data sets — due to the small size and sparsity of the network, the addition of random edges for negative examples makes the network structure significantly different in each run. In spite of that, our approach *pPLUMS is considerably robust with smallest average deviation* across all points as 0.048, while Group-Lasso has 0.060. In comparison for DBLP, pPLUMS or PLUMS has almost close to zero deviation compared to significantly larger deviations for Group-Lasso and Lasso.

4.4.2 Robustness Analysis We analyze the robustness of our approach using two types of experiments: (a) by reducing the information available in auxiliary networks and (b)



(a) Low sparsity (b) Medium Sparsity (c) High Sparsity



(a) Low Noise (b) Medium Noise (c) High Noise

Figure 3: (top row) As the auxiliary network becomes highly sparse, there is very little or no auxiliary information available for prediction. However, our approach is quite robust and performs the best compared to all baselines, using only the information in the primary network. (bottom row) We analyze the robustness, by varying the number of random edges in the auxiliary network. When the auxiliary network is complete, it becomes uninformative for prediction. While the several baselines suffer due to noisy conditions, our approaches are quite robust and pPLUMS performs the best in all scenarios.

by increasing noise in the auxiliary networks. The first type of experiment indirectly induces sparsity in the network, as it reduces the number of edges in the auxiliary network. In this case, when the network is empty, it has no auxiliary information to support the prediction in primary network. In the latter case, we implicitly make the network dense, by adding more random edges to the auxiliary network to increase the noise. Under high noise conditions, the auxiliary network is complete, as every node is connected to every other node — hence the resulting network is uninformative. We perform both these experiments using our synthetic data set.

For increasing the sparsity in the auxiliary networks, we reduce the conditional probabilities of generating an edge in the auxiliary network, given that it is preset in the primary network. As the probabilities decrease the auxiliary network becomes empty. Similarly, for increasing the noise in the auxiliary networks, we randomly sample edges from pairs of nodes in the primary network and apply a conditional distribution for that edge to be present in the auxiliary network. As the conditional probabilities increase more number of random edges begin to appear in the auxiliary network.

In Figure 3 we have shown the robustness of our approach measured in terms of P-R curves. *Under low sparsity conditions, when the auxiliary network is highly informative, our methods pPLUMS and PLUMS perform at their best.* More importantly, as we increase the sparsity levels, they continue to perform the best. As one can see, PLUMS, pPLUMS and ONENET all begin to perform very similarly in high sparsity conditions, as there is no auxiliary information. Note that Group-Lasso is quite robust and performs consistently well in high sparsity conditions, even though the overall performance is lower than our approach. This is due to the flexibility of the Group-Lasso structure to knock down specific path patterns and subsets that are not useful for prediction.

Similarly, as we increase the noise levels, in terms of the number of random edges in the auxiliary networks, our approach is consistently the best. Under low noise conditions all approaches perform equally well, except logistic regression and ONENET. *As the noise level increases*, all the methods reduce in their overall performance, but *the amount of decrease for Lasso is the highest*, as the number of features containing path level information from only the primary network is very limited. When the noise levels are extremely high, all methods further drop in performance. As seen in the sparsity variation experiment, the Group-Lasso [5] baseline is quite robust to noise perturbations also. In comparison, Lasso and Logistic are intermittent in their performance, under high noise and sparsity conditions. *pPLUMS performs the best in all noise scenarios.*

Scalability Discussion: The current paper does not focus on scalability related issues, however, our approach is designed with scalability in mind. The basic operation needed for our algorithm is a matrix-vector multiplication (for computing the gradients in the PLUMS algorithm). This operation can be done in an extremely scalable way

using Map-Reduce (Hadoop) [27], or multi-core frameworks (OpenMP) [28].

5 Conclusions

The problem of link prediction is fundamental to social and collaboration networks. Most of the existing collaboration mediums consists of several modes of interactions. These multiple interaction networks can be used to make a effective and robust link prediction model by combining multiple sources of information. We developed a supervised random walk model that can be both personalized and general, depending on the needs of the application. We showed that our approach is a generalization of Katz similarity for multiple networks. Our approach outperforms several popular and relevant baselines in terms of precision and recall curves. Our approach is also robust under sparse and noisy conditions, while the popular baselines can have inconsistent performance depending on the data set.

Acknowledgements

The research was supported in part by NSF grants IIS-1447566, IIS-1422557, CCF-1451986, CNS-1314560, IIS-0953274, IIS-1029711, and by NASA grant NNX12AQ39A, DARPA grant W911NF-12-C-0028 and IBM Ph.D. fellowship award. Arindam Banerjee also acknowledges the generous support from IBM and Yahoo. The authors thank the anonymous reviewers for their valuable comments.

References

[1] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[2] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[3] G. R. Lopes, M. M. Moro, L. K. Wives, and J. P. M. De Oliveira, "Collaboration recommendation on academic social networks," in *Advances in Conceptual Modeling–Applications and Challenges*, 2010, pp. 190–199.

[4] A. Singhal, K. Subbian, T. G. Kolda, A. Pinar, and J. Srivastava, "On reciprocity in massively multi-player online game networks," in *ASONAM*, 2013.

[5] Z. Lu, B. Savas, W. Tang, and I. S. Dhillon, "Supervised link prediction using multiple sources," in *ICDM*, 2010, pp. 923–928.

[6] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki, "Link prediction using supervised learning," in *SDM06: Workshop on Link Analysis, Counter-terrorism and Security*, 2006.

[7] D. Davis, R. Lichtenwalter, and N. V. Chawla, "Supervised methods for multi-relational link prediction," *Social Network Analysis and Mining*, pp. 1–15, 2013.

[8] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in *WSDM*. ACM, 2011, pp. 635–644.

[9] M. A. Hasan and M. J. Zaki, "A survey of link prediction in social networks," in *Social Network Data Analytics*, 2011, pp. 243–275.

[10] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla, "New perspectives and methods in link prediction," in *ACM SIGKDD*, 2010, pp. 243–252.

[11] D. Davis, R. Lichtenwalter, and N. V. Chawla, "Multi-relational link prediction in heterogeneous information networks," in *ASONAM*, 2011, pp. 281–288.

[12] J. Tang, T. Lou, and J. Kleinberg, "Inferring social ties across heterogenous networks," in *WSDM*, 2012, pp. 743–752.

[13] J. Zhang, X. Kong, and P. S. Yu, "Transferring heterogeneous links across location-based social networks," in *WSDM*. ACM, 2014, pp. 303–312.

[14] E. Zhong, E. W. Xiang, W. Fan, N. N. Liu, and Q. Yang, "Friendship prediction in composite social networks," *arXiv preprint arXiv:1402.4033*, 2014.

[15] L. M. Aiello, A. Barrat, R. Schifanella, C. Cattuto, B. Markines, and F. Menczer, "Friendship prediction and homophily in social media," *ACM Transactions on the Web (TWEB)*, vol. 6, no. 2, p. 9, 2012.

[16] E. Zheleva, L. Getoor, J. Golbeck, and U. Kuter, "Using friendship ties and family circles for link prediction," in *SNAKDD*, 2008, pp. 97–113.

[17] N. Lao and W. W. Cohen, "Relational retrieval using a combination of path-constrained random walks," *Machine learning*, vol. 81, no. 1, pp. 53–67, 2010.

[18] Y. Dong, J. Tang, S. Wu, J. Tian, N. V. Chawla, J. Rao, and H. Cao, "Link prediction and recommendation across heterogeneous social networks," in *ICDM*. IEEE, 2012, pp. 181–190.

[19] Y. Sun, R. Barber, M. Gupta, C. C. Aggarwal, and J. Han, "Co-author relationship prediction in heterogeneous bibliographic networks," in *ASONAM*. IEEE, 2011, pp. 121–128.

[20] L. Getoor and C. P. Diehl, "Link mining: a survey," *SIGKDD Explorations Newsletter*, vol. 7, no. 2, pp. 3–12, 2005.

[21] C. C. Aggarwal and K. Subbian, "Evolutionary network analysis: A survey," *CSUR*, vol. 47, no. 1, p. 10, 2014.

[22] D. P. Bertsekas, "Nonlinear programming," 1999.

[23] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[24] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l 1-ball for learning in high dimensions," in *ICML*, 2008, pp. 272–279.

[25] C. Seshadhri, T. G. Kolda, and A. Pinar, "Community structure and scale-free collections of erdős-rényi graphs," *Physical Review E*, vol. 85, no. 5, p. 056109, 2012.

[26] R. Lichtnwalter and N. V. Chawla, "Link prediction: fair and effective evaluation," in *ASONAM*, 2012, pp. 376–383.

[27] B. Bahmani, K. Chakrabarti, and D. Xin, "Fast personalized pagerank on mapreduce," in *SIGMOD*. ACM, 2011, pp. 973–984.

[28] R. A. Van De Geijn and J. Watts, "Summa: Scalable universal matrix multiplication algorithm," *Concurrency-Practice and Experience*, vol. 9, no. 4, pp. 255–274, 1997.