Learning Spectral Graph Transformations for Link Prediction

Jérôme Kunegis KUNEGIS@DAI-LAB.DE
Andreas Lommatzsch ANDREAS@DAI-LAB.DE
DAI-Labor, Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany

Abstract

We present a unified framework for learning link prediction and edge weight prediction functions in large networks, based on the transformation of a graph's algebraic spectrum. Our approach generalizes several graph kernels and dimensionality reduction methods and provides a method to estimate their parameters efficiently. We show how the parameters of these prediction functions can be learned by reducing the problem to a one-dimensional regression problem whose runtime only depends on the method's reduced rank and that can be inspected visually. We derive variants that apply to undirected, weighted, unweighted, unipartite and bipartite graphs. We evaluate our method experimentally using examples from social networks, collaborative filtering, trust networks, citation networks, authorship graphs and hyperlink networks.

1. Introduction

In the area of graph mining, several machine learning problems can be reduced to the problem *link prediction*. These problems include the prediction of social links, collaborative filtering and predicting trust.

Approaching the problem algebraically, we can consider a graph's adjacency matrix $\mathbf{A}$, and look for a function $F(\mathbf{A})$ returning a matrix of the same size whose entries can be used for prediction. Our approach consists of computing a matrix decomposition $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ and considering functions of the form $F(\mathbf{A}) = \mathbf{U}F(\mathbf{D})\mathbf{V}^T$, where $F(\mathbf{D})$ applies a function on reals to each element of the graph spectrum $\mathbf{D}$ separately. We show that a certain number of common link and edge weight prediction algorithms can be mapped to this form. As a result, the method we propose provides a mechanism for estimating any parameters of such link prediction algorithms. Analogously, we also consider a network's Laplacian matrix as the basis for link prediction.

Recently, several link prediction methods have been studied: weighted sums of path counts between nodes (Liben-Nowell & Kleinberg, 2003), the matrix exponential (Wu & Chang, 2004), the von Neumann kernel and diffusion processes (Kandola et al., 2002), the commute time and resistance distance kernels (Fouss et al., 2007), random forest kernels (Chebotarev & Shamis, 1997) and the heat diffusion kernel (Ito et al., 2005). Similarly, rank reduction of the adjacency matrix has been proposed to implement edge weight prediction (Sarwar et al., 2000). Our main contribution is to generalize these link prediction functions to a common form and to provide a way to reduce the high-dimensional problem of learning the various kernels' parameters to a one-dimensional curve fitting problem that can be solved efficiently. The runtime of the method only depends on the chosen reduced rank, and is independent of the original graph size.

We show that this generalization is possible under the assumption that the chosen training and test set are simultaneously diagonalizable, which we show to be an assumption made by all link prediction methods we studied. Our framework can be used to learn the parameters of several graph prediction algorithms, including the reduced rank for dimensionality reduction methods. Since we reduce the parameter estimation problem to a one-dimensional curve fitting problem that can be plotted and inspected visually to compare the different prediction algorithms, an informed choice can be made about them without having to evaluate each algorithm on a test set separately.

We begin by describing the method for undirected, unipartite graphs, and then extend it to weighted and bipartite graphs. As an experimental evaluation, we then apply our method to several large network

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

datasets and show which link prediction algorithm performs best for each.

2. Link Prediction in Undirected Graphs

In this section, we review common link prediction methods in undirected graphs that we generalize in the next section.

We will use the term *link prediction* in the general sense referring to any problem defined on a graph in which the position or weight of edges have to be predicted. The networks in question are usually large and sparse, for instance social networks, bipartite rating graphs, trust networks, citation graphs and hyperlink networks. The link prediction problems we consider can be divided into two classes: In unweighted graphs, the task consists of predicting *where* edges will form in the future. In weighted graphs, the task consists of predicting the *weight* of such edges. While many networks are directed in practice, we restrict this study to undirected graphs. Applying this method to directed graphs can be achieved by ignoring the edge directions, or by reducing them to bipartite graphs, mapping each vertex to two new vertices containing the inbound and outbound edges respectively.

Let $\mathbf{A} \in \{0,1\}^{n \times n}$ be the adjacency matrix of a simple, undirected, unweighted and connected graph on n vertices, and $F(\mathbf{A})$ a function that maps $\mathbf{A}$ to a matrix of the same dimension.

The following subsections describe link prediction functions $F(\mathbf{A})$ that result in matrices of the same dimension as $\mathbf{A}$ and whose entries can be used for link prediction. Most of these methods result in a positive-semidefinite matrix, and can be qualified as graph kernels. The letter α will be used to denote parameters of these functions.

2.1. Functions of the Adjacency Matrix

Let $\mathbf{D} \in \mathbb{R}^{n \times n}$ be the diagonal degree matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. Then $\mathcal{A} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is the normalized adjacency matrix. Transformations of the adjacency matrices $\mathbf{A}$ and $\mathcal{A}$ give rise to the exponential and von Neumann graph kernels (Kondor & Lafferty, 2002; Ito et al., 2005).

$$F_{\mathrm{EXP}}(\mathbf{A}) = \exp(\alpha\mathbf{A}) \quad (1)$$
$$F_{\mathrm{EXP}}(\mathcal{A}) = \exp(\alpha\mathcal{A}) \quad (2)$$
$$F_{\mathrm{NEU}}(\mathbf{A}) = (I - \alpha\mathbf{A})^{-1} \quad (3)$$
$$F_{\mathrm{NEU}}(\mathcal{A}) = (I - \alpha\mathcal{A})^{-1} \quad (4)$$

α is a positive parameter. Additionally, the von Neumann kernels require $\alpha < 1$.

2.2. Laplacian Kernels

$\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the combinatorial Laplacian of the graph, and $\mathcal{L} = \mathbf{I} - \mathcal{A} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ is the normalized Laplacian. The Laplacian matrices are singular and positive-semidefinite. Their Moore-Penrose pseudoinverse is called the commute time or resistance distance kernel (Fouss et al., 2007). The combinatorial Laplacian matrix is also known as the Kirchhoff matrix, due to its connection to electrical resistance networks.

$$F_{\mathrm{COM}}(\mathbf{L}) = \mathbf{L}^{+} \quad (5)$$
$$F_{\mathrm{COM}}(\mathcal{L}) = \mathcal{L}^{+} \quad (6)$$

By regularization, we arrive at the regularized Laplacian kernels (Smola & Kondor, 2003):

$$F_{\mathrm{COMR}}(\mathbf{L}) = (\mathbf{I} + \alpha\mathbf{L})^{-1} \quad (7)$$
$$F_{\mathrm{COMR}}(\mathcal{L}) = (\mathbf{I} + \alpha\mathcal{L})^{-1} \quad (8)$$

As a special case, the non-normalized regularized Laplacian kernel is called the random forest kernel for $\alpha = 1$ (Chebotarev & Shamis, 1997). The normalized regularized Laplacian is equivalent to the normalized von Neumann kernel by noting that $(1 + \alpha\mathcal{L})^{-1} = (1 + \alpha)(I - \alpha\mathcal{A})^{-1}$.

(Ito et al., 2005) define the heat diffusion kernel as

$$F_{\mathrm{HEAT}}(\mathbf{L}) = exp(-\alpha\mathbf{L}) \quad (9)$$
$$F_{\mathrm{HEAT}}(\mathcal{L}) = exp(-\alpha\mathcal{L}) \quad (10)$$

The normalized heat diffusion kernel is equivalent to the normalized exponential kernel: $exp(-\alpha\mathcal{L}) = e^{-\alpha}exp(\alpha\mathcal{A})$ (Smola & Kondor, 2003).

2.3. Rank Reduction

Using the eigenvalue decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, a rank-$k$ approximation of $\mathbf{A}$, $\mathbf{L}$, $\mathcal{A}$ and $\mathcal{L}$ is given by a truncation leaving only k eigenvalues and eigenvectors in $\mathbf{\Lambda}$ and $\mathbf{U}$.

$$F_{(k)}(\mathbf{A}) = \mathbf{U}_{(k)}\mathbf{\Lambda}_{(k)}\mathbf{U}_{(k)}^T \quad (11)$$

For $\mathbf{A}$ and $\mathcal{A}$, the biggest eigenvalues are used while the smallest eigenvalues are used for the Laplacian matrices. $F_{(k)}(\mathbf{A})$ can be used for prediction itself, or serve as the basis for any of the graph kernels (Sarwar et al., 2000). In practice, only rank-reduced versions of graph kernels can be computed for large networks.

2.4. Path Counting

Another way of predicting links consists of computing the proximity between nodes, measured by the number and length of paths between them.

One can exploit the fact that powers $\mathbf{A}^n$ of the adjacency matrix of an unweighted graph contain the number of paths of length n connecting all node pairs. On the basis that nodes connected by many paths should be considered nearer to each other than nodes connected by few paths, we compute a weighted mean of powers of $\mathbf{A}$ as a link prediction function.

$$F_{\mathrm{P}}(\mathbf{A}) = \sum_{i=0}^{d} \alpha_i \mathbf{A}^i \tag{12}$$

The result is a matrix polynomial of degree d. The coefficients α_i should be decreasing to reflect the assumption that links are more likely to arise between nodes that are connected by short paths than nodes connected by long paths. Thus, such a function takes both path lengths and path counts into account.

We note that the exponential and von Neumann kernels can be expressed as infinite series of matrix powers:

$$\exp(\alpha\mathbf{A}) = \sum_{i=0}^{\infty} \frac{\alpha^i}{i!}\mathbf{A}^i \tag{13}$$

$$(\mathbf{I} - \alpha\mathbf{A})^{-1} = \sum_{i=0}^{\infty} \alpha^i \mathbf{A}^i \tag{14}$$

We discuss polynomials of weighted graphs later.

3. Generalization

We now describe a formalism that generalizes the link prediction methods of the previous section and introduce an efficient algorithm to choose the optimal method and to estimate any parameter α.

We note that all these link prediction methods can be written as $\mathbf{F} = F(\mathbf{X})$, where $\mathbf{X}$ is one of $\{\mathbf{A}, \mathcal{A}, \mathbf{L}, \mathcal{L}\}$ and F is either a matrix polynomial, matrix (pseudo)inversion, the matrix exponential or a function derived piecewise linearly from one of these. Such functions F have the property that for a symmetric matrix $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, they can be written as $F(\mathbf{A}) = \mathbf{U}F(\mathbf{\Lambda})\mathbf{U}^T$, where $F(\mathbf{\Lambda})$ applies the corresponding function on reals to each eigenvalue separately. In other words, these link prediction methods result in prediction matrices that are simultaneously diagonalizable with the known adjacency matrix. We will call such functions *spectral transformations* and write $F \in \mathcal{S}$.

In the following, we study the problem of finding suitable functions F.

3.1. Finding F

Given a graph G, we want to find a spectral transformation F that performs well at link prediction for this particular graph. To that end, we divide the edge set of G into a training set and a test set, and then look for an F that maps the training set to the test set with minimal error.

Formally, let $\mathbf{A}$ and $\mathbf{B}$ be the adjacency matrices of the training and test set respectively. We will call $\mathbf{A}$ the source matrix and $\mathbf{B}$ the target matrix. The solution to the following optimization problem gives the optimal spectral transformation for the task of predicting the edges in the test set.

Problem 1 *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two adjacency matrices over the same vertex set. A spectral transformation that maps* $\mathbf{A}$ *to* $\mathbf{B}$ *with minimal error is given by the solution to*

$$\begin{aligned} \min_{F} \quad & \| F(\mathbf{A}) - \mathbf{B} \|_F \\ s.t. \quad & F \in \mathcal{S} \end{aligned}$$

where $\| \cdot \|_F$ *denotes the Frobenius norm.*

The Frobenius norm corresponds to the root mean squared error (RMSE) of the mapping from $\mathbf{A}$ to $\mathbf{B}$. While the RMSE is common in link prediction problems, other error measures exist, but give more complex solutions to our problem. We will thus restrict ourselves to the Frobenius norm.

Problem 1 can be solved by computing the eigenvalue decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ and using the fact that the Frobenius norm is invariant under multiplication by an orthogonal matrix.

$$\begin{aligned} & \| F(\mathbf{A}) - \mathbf{B} \|_F \\ = \; & \| \mathbf{U}F(\mathbf{\Lambda})\mathbf{U}^T - \mathbf{B} \|_F \\ = \; & \| F(\mathbf{\Lambda}) - \mathbf{U}^T\mathbf{B}\mathbf{U} \|_F \end{aligned} \tag{15}$$

The Frobenius norm in Expression (15) can be decomposed into the sum of squares of off-diagonal entries of $F(\mathbf{\Lambda}) - \mathbf{U}^T\mathbf{B}\mathbf{U}$, which is independent of F, and into the sum of squares of its diagonal entries. This leads to the following least-squares problem equivalent to Problem 1:

Problem 2 *If* $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ *is the eigenvalue decomposition of* $\mathbf{A}$*, then the solution to Problem 1 is given by* $F(\mathbf{\Lambda})_{ii} = f(\mathbf{\Lambda}_{ii})$*, where* $f(x)$ *is the solution to the following minimization problem.*

$$\min_{f} \quad \textstyle\sum_i (f(\mathbf{\Lambda}_{ii}) - \mathbf{U}_{\cdot i}^T \mathbf{B} \mathbf{U}_{\cdot i})^2$$

Table 1. Link prediction functions and their corresponding real functions. For functions that apply to the adjacency matrix, we show their odd component used for rectangular matrices, as described in Section 5.

Link prediction function	Real function	Odd component
$F_P(\mathbf{A}) = \sum_{i=0}^{d} \alpha_i \mathbf{A}^i$	$f(x) = \sum_{i=0}^{d} \alpha_i x^i$	$f(x) = \sum_{i=0}^{d} \alpha_i x^{2i+1}$
$F_{\mathrm{EXP}}(\mathbf{A}) = \exp(\alpha \mathbf{A})$	$f(x) = e^{\alpha x}$	$f(x) = \sinh(\alpha x)$
$F_{\mathrm{NEU}}(\mathbf{A}) = (\mathbf{I} - \alpha \mathbf{A})^{-1}$	$f(x) = \frac{1}{1-\alpha x}$	$f(x) = \frac{\alpha x}{1-\alpha^2 x^2}$
$F_{\mathrm{COM}}(\mathbf{L}) = \mathbf{L}^{+}$	$f(x) = x^{-1}$ when $x > 0, f(x) = 0$ otherwise	
$F_{\mathrm{COMR}}(\mathbf{L}) = (\mathbf{I} + \alpha \mathbf{L})^{-1}$	$f(x) = \frac{1}{1+\alpha x}$	
$F_{\mathrm{HEAT}}(\mathbf{L}) = \exp(-\alpha \mathbf{L})$	$f(x) = e^{-\alpha x}$	
$F_{(k)}(\mathbf{A}) = \mathbf{U}_{(k)} \mathbf{\Lambda}_{(k)} \mathbf{U}_{(k)}^T$	$f(x) = x$ when $\lvert x \rvert \geq \lvert \mathbf{\Lambda}_{kk} \rvert, f(x) = 0$ otherwise	

This problem is a one-dimensional least-squares curve fitting problem of size n. Since each function $F(\mathbf{A})$ corresponds to a function $f(x)$, we can choose a link prediction function F and learn its parameters by inspecting the corresponding curve fitting problem. This method also allows us to check which graph kernel is best suited to the underlying link prediction problem.

3.2. Curve Fitting

We have thus reduced the general matrix regression problem to a one-dimension least-squares curve fitting problem of size k. In analogy to the various graph kernels and rank reduction methods of the previous section we propose the following curve fitting models to find suitable functions f.

For each link prediction method, we take its matrix function $F(\mathbf{A})$ and derive the corresponding function $f(x)$ on reals. For each function on reals, we additionally insert a multiplicative parameter $\beta > 0$ that is to be learned along with the other parameters. The resulting functions are summarized in Table 1.

The normalized Laplacian can be derived from the normalized adjacency matrix by the spectral transformation $\mathcal{L} = F(\mathcal{A}) = \mathbf{I} - \mathcal{A}$ corresponding to the real function $f(x) = 1 - x$. Thus, the normalized commute time kernel reduces to the normalized von Neumann kernel and the heat diffusion kernel reduces to the normalized exponential kernel. We will therefore restrict the set of source matrices to $\{\mathbf{A}, \mathcal{A}, \mathbf{L}\}$.

Since both $\mathbf{\Lambda}$ and $\mathbf{U}^T\mathbf{B}\mathbf{U}$ are available after having computed the eigenvalue decomposition of A, the main computational part of our method lies in the eigenvalue decomposition of $\mathbf{A}$, which is performed for the usual application of the link prediction methods. Additionally, the computation of $\mathbf{U}^T\mathbf{B}\mathbf{U}$ takes runtime $\mathcal{O}(kr)$ where r is the number of nonzeroes and the curve fitting runtime is only dependent on k.

As the target matrix, we may also use $\mathbf{A} + \mathbf{B}$ instead of $\mathbf{B}$ in the assumption that a link prediction algorithm should return high values for known edges. With this modification, we compute $\mathbf{U}^T(\mathbf{A} + \mathbf{B})\mathbf{U}$ instead of $\mathbf{U}^T\mathbf{B}\mathbf{U}$.

Finally, the problem of scaling has to be addressed. Since we only train F on a source matrix $\mathbf{A}$ but intend to apply the learned function to $\mathbf{A} + \mathbf{B}$, the function F would be applied to spectra of different extent. Therefore, we normalize the spectra we encounter by dividing all eigenvalues by the largest eigenvalue, replacing $\mathbf{\Lambda}$ by $\mathbf{\Lambda}_{11}^{-1}\mathbf{\Lambda}$.

3.3. Illustrative Examples

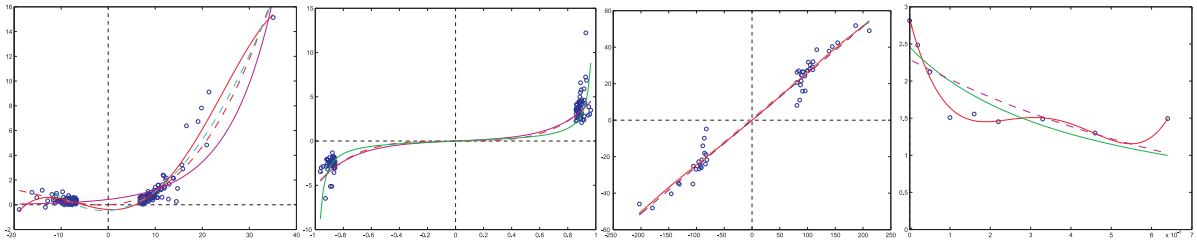
To illustrate our method, we show four examples of finding link prediction functions in unweighted graphs. We use the datasets described in Table 2, with the adjacency matrix, the normalized adjacency matrix and the Laplacian as source matrices.

4. Weighted Graphs

In this section, we show how our method can be extended to graphs with weighted edges, including the case of edges with negative weights.

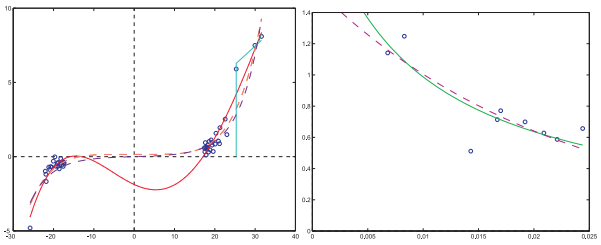
If edge weights are all positive, the method in the previous sections applies trivially. In some networks however, edges have positive as well as negative weights. Such signed graphs arise for instance in social networks where negative edges denotes enmity instead of friendship, or in bipartite rating graphs where the two vertex classes represent users and items, and edges represent ratings that admit positive and negative values.

The link prediction functions based on the adjacency matrix can be interpreted as weighted sums of powers of the adjacency matrix which denote path counts in the network. If some edges have negative weight, the total weight of a path is counted as the product of the edges' weights. This corresponds to the assumption of multiplicative transitivity, which can be sum-



(a) advogato($\mathbf{A} \to \mathbf{B}$) (b) dblp($\mathcal{A} \to \mathbf{A} + \mathbf{B}$) (c) wt10g($\mathbf{A} \to \mathbf{B}$) (d) www($\mathbf{L} \to \mathbf{B}$)

Figure 1. Graphical representation of the one-dimensional curve fitting problem on a selection of datasets from Table 2. The plots show the diagonal elements of $\mathbf{\Lambda}$ (the eigenvalues of the source matrix) on the x-axis and the diagonal elements of $\mathbf{U}^T\mathbf{B}\mathbf{U}$ or $\mathbf{U}^T(\mathbf{A}+\mathbf{B})\mathbf{U}$ on the y-axis. The source and target matrices are shown in parentheses. The result of least-squares curve fitting using several functions f are shown as follows: red: polynomial of degree 4; red dashed: nonnegative polynomial of degree 7; green: the rational function $1/(1-\alpha x)$ in (d) and $\alpha x/(1-\alpha^2 x^2)$ in (b); magenta: the exponential function in (a) and (c), the hyperbolic sine in (b) and the inverse exponential in (d); black: the linear map αx.



(a) slashdot($\mathbf{A} \to \mathbf{A} + \mathbf{B}$) (b) epinions($\mathbf{L} \to \mathbf{B}$)

Figure 2. Curve fitting in networks with signed edge weights. The following functions are estimated: red: polynomial; red dashed: nonnegative polynomial; magenta: exponential function; cyan: rank reduction.

marized as *the enemy of my enemy is my friend* (Hage & Harary, 1983).

The Laplacian graph kernels can also be applied to signed graphs by using the absolute diagonal degree matrix $\mathbf{D}_{ii} = \sum_j |\mathbf{A}_{ij}|$ (Hou, 2005). As the unsigned Laplacian, the signed Laplacian is positive-semidefinite. Unlike the unsigned Laplacian, the signed Laplacian can be positive-definite. This is the case when each connected component of the graph contains a cycle with an odd number of negatively weighted edges.

We also use this alternative definition of $\mathbf{D}$ to define the signed normalized adjacency matrix, giving again $\mathcal{L} = \mathbf{I} - \mathcal{A}$. This definition of the signed normalized adjacency matrix is also justified by interpreting each edge as a *vote* and giving each node's votes equal weight. Figure 2 shows examples of curve fitting for networks with negative edge weights.

5. Bipartite Graphs

In this section, we extend the method of the previous sections to bipartite networks, and show that in such networks we can restrict our curve fitting problem to odd functions.

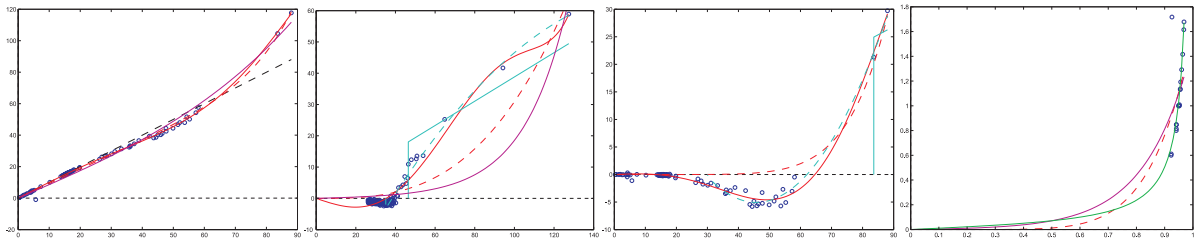
The adjacency matrix $\mathbf{A}$ of a bipartite graph can be written in block form as $\mathbf{A} = [\mathbf{0}\,\mathbf{R}; \mathbf{R}^T\,\mathbf{0}]$ where $\mathbf{R}$ is rectangular. Observing that even powers of $\mathbf{A}$ can be written as $\mathbf{A}^{2n} = [(\mathbf{R}\mathbf{R}^T)^n\,\mathbf{0}; \mathbf{0}\,(\mathbf{R}^T\mathbf{R})^n]$, we see that they result in predictions of zero for edges connecting the two vertex classes. Therefore, we only need to consider odd powers, having the form $\mathbf{A}^{2n+1} = [\mathbf{0}\,(\mathbf{R}\mathbf{R}^T)^n\mathbf{R}; \mathbf{R}^T(\mathbf{R}\mathbf{R}^T)^n\,\mathbf{0}]$. It follows that to predict (bipartite) edges in a bipartite graph, we have to compute $F(\mathbf{R}\mathbf{R}^T)\mathbf{R}$ for a matrix function F. Using the singular value decomposition $\mathbf{R} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$, this can be written as

$$\begin{aligned} F(\mathbf{R}\mathbf{R}^T)\mathbf{R} &= F(\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T\mathbf{V}\mathbf{\Lambda}\mathbf{U}^T)\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \\ &= \mathbf{U}F(\mathbf{\Lambda}^2)\mathbf{\Lambda}\mathbf{V}^T \end{aligned} \tag{16}$$

where $F(\mathbf{\Lambda}^2)\mathbf{\Lambda}$ corresponds to an odd function in $\mathbf{\Lambda}$.

We can therefore restrict the least-squares regression problem to odd functions, as summarized in Table 1. The exponential function is replaced by the hyperbolic sine and $1/(1-\alpha x)$ is replaced by $\alpha x/(1-\alpha x^2)$.

The use of odd functions can also be justified by noting that if $\{\lambda_i\}$ is the set of (positive) singular values of $\mathbf{R}$, then $\mathbf{A}$ has the eigenvalues $\{\pm\lambda_i\}$. By symmetry, the function $f(\lambda)$ has to fulfill $f(-\lambda) = -f(\lambda)$ and thus be odd. Yet another interpretation is given by the observation that in bipartite graphs, paths connecting two vertices in different partitions are necessarily of odd length, and therefore the coefficients of even powers of $\mathbf{A}$ can be ignored in any matrix polynomial used for



(a) jester($\mathbf{A} \to \mathbf{A} + \mathbf{B}$) (b) movielens($\mathbf{A} \to \mathbf{B}$) (c) jester($\mathbf{A} \to \mathbf{B}$) (d) advogato($\mathcal{A} \to \mathbf{A} + \mathbf{B}$)

Figure 3. Fitting of curves in several bipartite datasets. The curves fitted are: red: odd polynomial; red dashed: nonnegative odd polynomial; magenta: the hyperbolic sine; green: the rational function $\alpha x/(1-\alpha^2 x^2)$; cyan: rank reduction.

link prediction.

The method for bipartite graphs cannot be used with the Laplacian $\mathbf{L}$, because the Laplacian has nonzero diagonal entries, and thus its eigenvalues do not correspond to the singular values of $\mathbf{R}$.

Figure 3 illustrates the case of bipartite networks using several example datasets from Table 2. We make the following observations. The plots (b) and (c) computed by using only $\mathbf{B}$ as the target matrix (without $\mathbf{A}$) map many eigenvalues to negative values, and the best curve fitting is attained by a polynomial with negative coefficients. This suggests that the search for link prediction methods should not be reduced to kernels, these being positive-semidefinite. Plot (b) can be interpreted as suggesting to ignore all eigenvalues smaller than a certain threshold, justifying the use of reduction to a smaller rank than would be possible computationally. The advogato dataset in (d) provides an example that justifies the von Neumann kernel.

6. Experimental Evaluation

We investigate eight network datasets and study two separate link prediction tasks: for unweighted networks, we study the task of link prediction and for weighted graphs, the task of predicting the edge weights. The network datasets we inspected are summarized in Table 2. DBLP[1] is a citation graph. Hep-th is the citation graph of Arxiv's high energy physics/theory section (Newman, 2001). Advogato is a trust network with three levels of trust (Stewart, 2005). Slashdot is a social network where users tag each other as friends and foes (Kunegis et al., 2009). Epinions is an opinion site where users can agree or disagree with each other (Guha et al., 2004). WWW and WT10G are hyperlink network datasets extracted from a subset of the World Wide Web (Albert et al., 1999; Bailey et al., 2003). Eowiki is the bipartite user-article authorship graph of the Esperanto Wikipedia[2]. Jester is a dataset of rated jokes (Goldberg et al., 2001). MovieLens is a dataset of movie ratings[3].

6.1. Setup

For each network dataset, we split the set of edges into a test and a training set, with the test set containing a third of all edges. For the bipartite datasets where edge weights correspond to ratings (MovieLens, Jester), we then subtract the mean of all ratings in the training set from the edge weights in the training set. We then reduce the training set of edges to its biggest connected component, and then split the resulting set into source and target adjacency matrices $\mathbf{A}$ and $\mathbf{B}$, where $\mathbf{B}$ contains one third of all edges in the reduced training set. We then use our method to find functions F that minimize the Frobenius norm of the difference between $F(\mathbf{A})$ and $\mathbf{B}$. For the rank-reduced decomposition of $\mathbf{A}$, we use a reduced rank k dependent on the size of the dataset, as given in Table 2. We apply the curve fitting methods described in the previous sections to learn several link prediction kernels for each network. We then use these link prediction functions to compute predictions for the edges in the test set.

As a measure of performance for the link prediction methods, we choose to compute the Pearson correlation coefficient between the predicted and known ratings in the test set. We choose the correlation because it represents a trade-off between the root mean squared error that we minimize and more dataset-dependent measures such as precision and recall, which cannot be applied to all datasets. For the unweighted datasets, we add to the test set a set of non-edges of equal size with weight zero. The results of our evaluation are shown in Table 3.

[1] http://dblp.uni-trier.de/

[2] http://eo.wikipedia.org/

[3] http://www.grouplens.org/node/73

Table 2. Summary of network datasets we used in our experiments and examples.

Name	Vertices	Edges	Weights	k	Description
dblp	12,563	49,779	$\{1\}$	126	Citation graph
hep-th	27,766	352,807	$\{1\}$	54	Citation graph
advogato	7,385	57,627	$\{0.6, 0.8, 1.0\}$	192	Trust network
slashdot	71,523	488,440	$\{-1, +1\}$	24	Friend/foe network
epinions	131,828	841,372	$\{-1, +1\}$	14	Trust/distrust network
www	325,729	1,497,135	$\{1\}$	49	Hyperlink graph
wt10g	1,601,787	8,063,026	$\{1\}$	49	Hyperlink graph
eowiki	2,827+168,320	803,383	$\{1\}$	26	Authorship graph
jester	24,938+100	616,912	$[-10, +10]$	100	Joke ratings
movielens	6,040+3,706	1,000,209	$\{1, 2, 3, 4, 5\}$	202	Movie ratings

6.2. Observations

We observe that the biggest accuracy is generally achieved by source matrices having a more "spread" spectrum: the singular values are further apart relatively, making the curve fitting algorithms more accurate. In some cases however, the curves fit well to a tight spectrum. This is the case for the normalized adjacency matrix of the Advogato dataset.

We also observe that there is not a single graph kernel that performs best for all datasets. Comparing the performance of the various link prediction methods to the curve fitting precision in Figures 1, 2 and 3 we observe that the best link prediction algorithms are those that provide a well-fitting curve. We interpret this as an indication that inspecting the curve fitting plots visually can provide useful insight into each dataset by observing what model for f fits best.

The analysis of bipartite networks, in particular the MovieLens rating dataset, suggests that the matrix hyperbolic sine may be used for collaborative filtering.

7. Related Work

Graph kernels and other link prediction methods are usually defined with parameters, and the choice of any parameter is left to the implementation.

Simultaneously diagonalizable matrices have been considered in the context of *joint diagonalization*, where given a set of matrices $\{\mathbf{A}_i\}$, a matrix $\mathbf{U}$ is searched that makes the matrices $\{\mathbf{U}\mathbf{A}_i\mathbf{U}^T\}$ as diagonal as possible according to a given matrix norm. See for instance (Wax & Sheinvald, 1997). In that context, the task consists of finding an orthogonal matrix $\mathbf{U}$, whereas our approach consists of optimizing the fit by varying the spectrum.

8. Conclusion and Discussion

We have presented a method for learning link and rating prediction functions in large networks. We have shown that a certain number of graph kernels and other link prediction algorithms can be interpreted as the spectral transformation of the underlying graph's adjacency or Laplacian matrix. Restricting our search of accurate link prediction methods to kernels of this form we derived a minimization problem that can be reduced to a one-dimensional least-squares regression problem. This formalism makes it possible to estimate the parameters of the various graph kernels and allows one to compare graph kernels visually using curve fitting.

Although the normalized adjacency matrices have eigenvalues all less than one and very near to each other and one could expect curve fitting to be less precise, the prediction functions learned from the normalized adjacency matrix perform well in many cases, and even very well in a few cases. By applying the exponential graph kernel, we arrived at the matrix hyperbolic sine, which our results suggest performs well for collaborative rating prediction.

By experimenting with various types of network datasets, we found that the various link prediction methods in current use are justified in specific cases. We hope that the method presented here can help make an informed choice as to the right method.

For future work, we intend to use other matrix decompositions, with the constraint that the decomposition include a diagonal matrix that can be transformed. By using another norm than the Frobenius norm, we may extend the method to cases where other norms could be more accurate measures of the prediction error. Finally, other graph kernels and link prediction methods not covered in this paper could be inspected as we did here with the more common ones.

Table 3. The results of our experimental evaluation. For each dataset, we show the source and target matrices, the curve fitting model and the link prediction method that perform best.

Dataset	Best transformation	Best fitting curve	Best graph kernel	Correlation
dblp	$\mathbf{L} \to \mathbf{B}$	Polynomial	Sum of powers	0.563
hep-th	$\mathcal{A} \to \mathbf{B}$	Exponential	Heat diffusion	0.453
advogato	$\mathcal{L} \to \mathbf{B}$	Rational	Commute time	0.554
slashdot	$\mathbf{A} \to \mathbf{B}$	Nonnegative odd polynomial	Sum of powers	0.263
epinions	$\mathbf{A} \to \mathbf{A} + \mathbf{B}$	Nonnegative odd polynomial	Sum of powers	0.354
www	$\mathbf{L} \to \mathbf{A} + \mathbf{B}$	Polynomial	Sum of powers	0.739
wt10g	$\mathbf{A} \to \mathbf{B}$	Linear function	Rank reduction	0.293
eowiki	$\mathbf{A} \to \mathbf{A} + \mathbf{B}$	Nonnegative odd polynomial	Sum of powers	0.482
jester	$\mathcal{A} \to \mathbf{A}$	Odd polynomial	Sum of powers	0.528
movielens	$\mathcal{A} \to \mathbf{A} + \mathbf{B}$	Hyperbolic sine	Hyperbolic sine	0.549

References

Albert, R., Jeong, H., & Barabási, A.-L. (1999). The diameter of the World Wide Web. *Nature*, *401*, 130.

Bailey, P., Craswell, N., & Hawking, D. (2003). Engineering a multi-purpose test collection for Web retrieval experiments. *Information Processing and Management*, *39*, 853–871.

Chebotarev, P., & Shamis, E. (1997). The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control*, *58*, 1505–1514.

Fouss, F., Pirotte, A., Renders, J.-M., & Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *Trans. on Knowledge and Data Engineering*, *19*, 355–369.

Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, *4*, 133–151.

Guha, R., Kumar, R., Raghavan, P., & Tomkins, A. (2004). Propagation of trust and distrust. *Proc. Int. Conf. on World Wide Web* (pp. 403–412).

Hage, P., & Harary, F. (1983). *Structural models in anthropology.* Cambridge University Press.

Hou, Y. (2005). Bounds for the least Laplacian eigenvalue of a signed graph. *Acta Mathematica Sinica*, *21*, 955–960.

Ito, T., Shimbo, M., Kudo, T., & Matsumoto, Y. (2005). Application of kernels to link analysis. *Proc. Int. Conf. on Knowledge Discovery in Data Mining* (pp. 586–592).

Kandola, J., Shawe-taylor, J., & Cristianini, N. (2002). Learning semantic similarity. *Advances in Neural Information Processing Systems* (pp. 657–664).

Kondor, R., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete structures. *Proc. Int. Conf. on Machine Learning* (pp. 315–322).

Kunegis, J., Lommatzsch, A., & Bauckhage, C. (2009). The Slashdot Zoo: Mining a social network with negative edges. *Proc. Int. Conf. on World Wide Web* (pp. 741–750).

Liben-Nowell, D., & Kleinberg, J. (2003). The link prediction problem for social networks. *Proc. Int. Conf. on Information and Knowledge Management* (pp. 556–559).

Newman, M. E. J. (2001). The structure of scientific collaboration networks. *Proc. National Academy of Sciences*, *98*, 404–409.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Application of dimensionality reduction in recommender systems–a case study. *Proc. ACM WebKDD Workshop.*

Smola, A., & Kondor, R. (2003). Kernels and regularization on graphs. *Proc. Conf. on Learning Theory and Kernel Machines* (pp. 144–158).

Stewart, D. (2005). Social status in an open-source community. *American Sociological Review*, *70*, 823–842.

Wax, M., & Sheinvald, J. (1997). A least-squares approach to joint diagonalization. *IEEE Signal Processing Letters*, *4*, 52–53.

Wu, Y., & Chang, E. Y. (2004). Distance-function design and fusion for sequence data. *Proc. Int. Conf. on Information and Knowledge Management* (pp. 324–333).