Learning Spectral Graph Transformations for Link Prediction

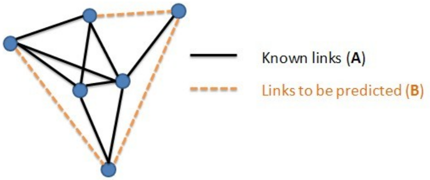
J. Kunegis, A. Lommatzsch. ICML, 2009

Presented by Wei Bi
Apr.21st, 2011

Outline

- The Problem
  - Link Prediction
  - Known Solutions
- Proposed Generalized Formalism
- Experiment Evaluation
- Conclusions and Discussions

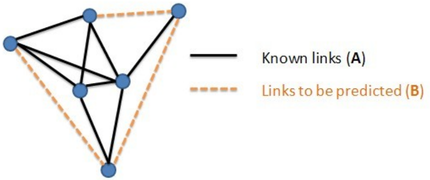
Link Prediction



- Motivation: Recommend connections in a social network
- Predict links in an undirected, unweighed network
- Objective: Using the adjacency matrices $\mathbf{A}$ and $\mathbf{B}$, find a function $F(\mathbf{A})$ giving prediction values corresponding to $\mathbf{B}$:

$$F(\mathbf{A}) = \mathbf{B}$$

Path Counting



- Follow paths
- Number of paths of length k given by $\mathbf{A}^k$
- Nodes connected by many paths
- Weight powers of $\mathbf{A}$: $\alpha A^2 + \beta A^3 + \gamma A^4 + \cdots$
- Examples:
    - Exponential graph kernels: $e^{\alpha \mathbf{A}} = \Sigma_i \frac{\alpha^i}{i!} \mathbf{A}^i$
    - Von Neumann kernel: $(\mathbf{I} - \alpha \mathbf{A})^{-1} = \Sigma_i \alpha^i \mathbf{A}^i (0 < \alpha < 1)$

Laplacian Link Prediction Functions

- Graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$, $D_{ii} = \Sigma_j A_{ij}$
  - Resistance Distance: L^+
  - Regularized Laplacian: $(\mathbf{I} + \alpha \mathbf{L})^{-1}$
  - Heat diffusion kernel: $e^{-\alpha \mathbf{L}}$

Computation of Link Prediction Functions

Observation: eigenvalue decomposition of $\mathbf{A}/\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^T$

Matrix polynomial	$\Sigma_i \alpha_i \mathbf{A}^i$	$= \mathbf{U}(\Sigma_i \alpha_i \Lambda^i)\mathbf{U}^T$
Matrix exponential	$e^{\alpha\mathbf{A}}$	$= \mathbf{U}e^{\alpha\Lambda}\mathbf{U}^T$
Von Neumann kernel	$(\mathbf{I} - \alpha\mathbf{A})^{-1}$	$= \mathbf{U}(\mathbf{I} - \alpha\Lambda)^{-1}\mathbf{U}^T$
Rank reduction	$\mathbf{A}_{(k)}$	$= \mathbf{U}\Lambda_{(k)}\mathbf{U}^T$
Resistance distance	L^+	$= \mathbf{U}\Lambda^+\mathbf{U}^T$
Regularized Laplacian	$(\mathbf{I} + \alpha\mathbf{L})^{-1}$	$= \mathbf{U}(\mathbf{I} + \alpha\Lambda)^{-1}\mathbf{U}^T$
Heat diffusion kernel	$e^{-\alpha\mathbf{L}}$	$= \mathbf{U}e^{-\alpha\Lambda}\mathbf{U}^T$

Spectral Transformation

Learning Spectral Transformations

- Link prediction functions are spectral transformations of **A**/**L**

$$F(\mathbf{A}) = UF(\Lambda)U^T$$
$$F(\Lambda)_{ii} = f(\Lambda_{ii})$$

- A spectral transformation F corresponds to a function of reals f

Matrix polynomial	$\Sigma_i \alpha_i \mathbf{A}^i$	$f(x) = \Sigma_i \alpha_i x^i$
Matrix exponential	$e^{\alpha \mathbf{A}}$	$f(x) = e^{\alpha x}$
Matrix inverse	$(\mathbf{I} - \alpha \mathbf{A})^{-1}$	$f(x) = \frac{1}{(1-\alpha x)}$
Rank reduction	$\mathbf{A}_{(k)}$	$f(x) = x \text{when} \lvert x \rvert \geq x_0, 0$ otherwise
Pseudoinverse	$\mathbf{A}^+$	$f(x) = \frac{1}{x} \text{when} x > 0, 0$ otherwise

Finding the Best Spectral Transformation

- Find the best spectral transformation on test set **B**

$$\min_F \|F(\mathbf{A}) - \mathbf{B}\|_F,$$
$$\text{where} \quad \|X\|_F \quad \text{denotes} \quad (\Sigma_{ij}|X_{ij}|^2)^{1/2}$$

- Equivalent minimization problem

$$\begin{aligned} & \min_F \|\mathbf{U}F(\Lambda)\mathbf{U}^T - \mathbf{B}\|_F \\ = \quad & \min_F \|F(\Lambda) - \mathbf{U}^T\mathbf{B}\mathbf{U}\|_F \end{aligned}$$

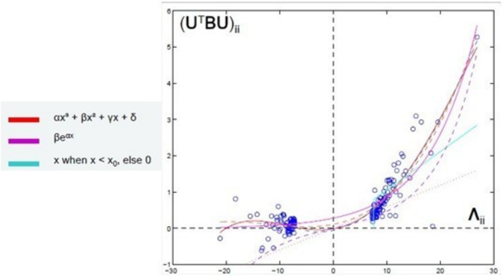
- Reduce to diagonal, because off-diagonal in $F(\Lambda)$ is constant zero

$$\min_f \Sigma_i (f(\Lambda_{ii}) - (\mathbf{U}^T\mathbf{B}\mathbf{U})_{ii})^2$$

- The best spectral transformation is given by a one-dimensional least-squares problem

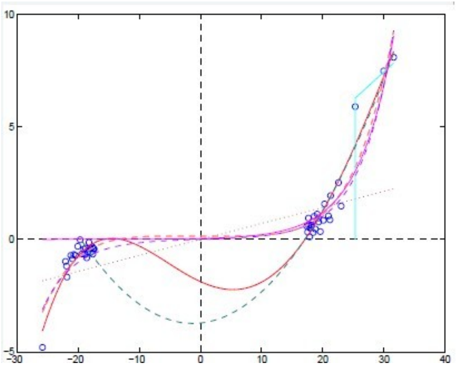
Example: DBLP Citation Network (undirected)

- DBLP citation network
- Symmetric adjacency matrices $\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^T, \mathbf{B}$

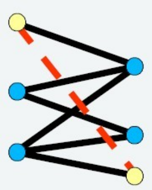


Variants: Weighted and Signed Graphs

- Weighted undirected graphs: use $\mathbf{A}$ and $\mathbf{L} = \mathbf{D} - \mathbf{A}$
- Signed graphs: use signed graph Laplacian with $\mathbf{D}_{ii} = \Sigma_j |\mathbf{A}_{ij}|$
- Example: Slashdot Zoo (social network with negative edges)



Variants: Bipartite Graphs



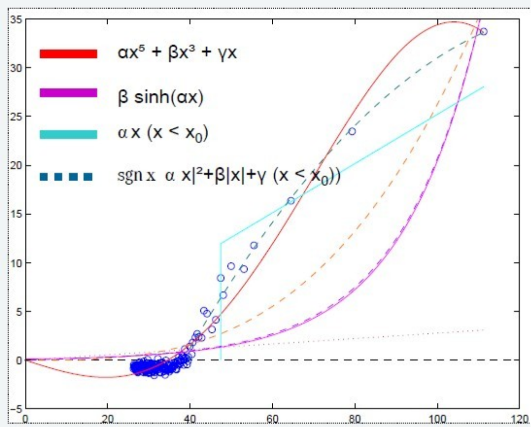
- Bipartite graphs: paths have **odd length**
- Compute sum of odd powers of $\mathbf{A}$
- The resulting polynomial is odd $\alpha\mathbf{A}^3 + \beta\mathbf{A}^5 + \cdots$
- For other link prediction functions, use the odd components

Variants: Bipartite Graphs

- How to compute the odd powers of $\mathbf{A}$ efficiently?
- $\mathbf{A}^{2n+1} = [\mathbf{0}\ \mathbf{R}; \mathbf{R}^T\ \ \mathbf{0}]^{2n+1} = [\mathbf{0}\ (\mathbf{R}\mathbf{R}^T)^n\mathbf{R}; \mathbf{R}^T(\mathbf{R}\mathbf{R}^T)^n\ \mathbf{0}]$
- Singular value decomposition of $\mathbf{R} = \mathbf{U}\Sigma\mathbf{V}^T$
- $(\mathbf{R}\mathbf{R}^T)^n\mathbf{R} = (U\Sigma V^T V\Sigma U^T)^n U\Sigma V^T = (U\Sigma^2 V^T)^n U\Sigma V^T = U\Sigma^{2n+1}V^T$
- Odd powers of A are given by odd spectral transformations of $\mathbf{R}$

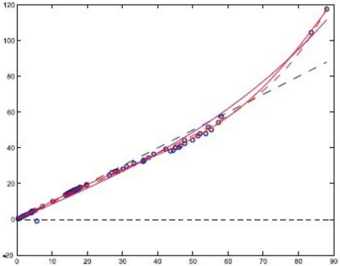
Variants: Bipartite Graphs

- Example: MovieLens rating graph
- Rating values: {-2,-1,0,+1,+2}

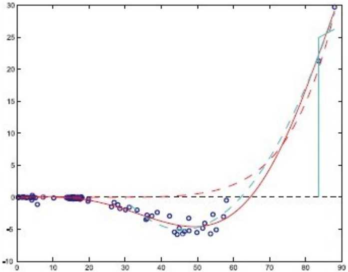


Variants: Bipartite Graphs

- Example: jester



(a) jester($\mathbf{A} \to \mathbf{A} + \mathbf{B}$)



(c) jester($\mathbf{A} \to \mathbf{B}$)

Experiments

- 2/3 edges for training, 1/3 edges for testing.
- Learn $F(A)$ using the proposed method.
- Use the prediction function to compute predictions for edges in the test set.
- Evaluation: Pearson correlation coefficient.

Experiments

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

Experiments

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

Conclusions

- Many link prediction functions are spectral transformations
- Spectral transformations can be learned

Discussions

- Minimize some other norms
- Eigenvalue decomposition and SVD are expensive.
- The evolution of the graph: time series problem, probabilistic model?