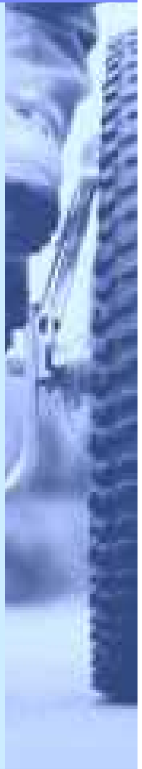
Methods for Network Structure Prediction



Hisashi Kashima

IBM Research / Tokyo Research Laboratory

Methods for network structure prediction

- **Network-structured data**
- **Link prediction problem (= network structure prediction problem)**
- **Link prediction methods based on node information**
- **Link prediction methods based on structural information**
  - [Our contribution] A parameterized model for link prediction

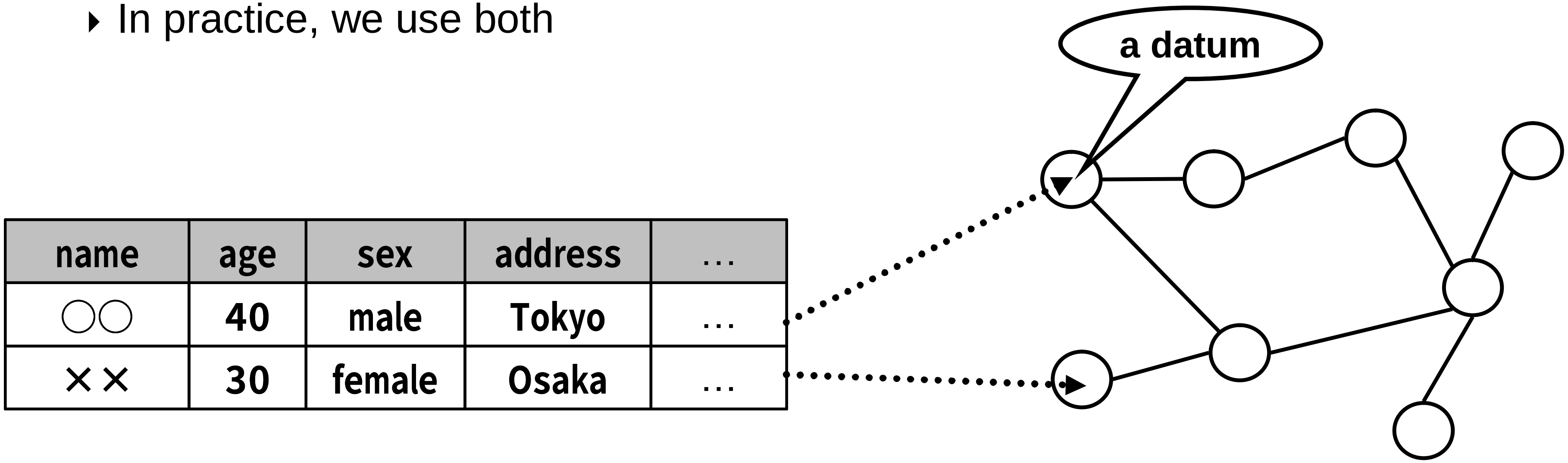
Network-structured data

Network structured data represent relations among data as a graph

- **Relations among data are represented as a graph structure**
  - A node represents a data
  - A link represents a relation between two data

⇔ In standard machine learning setting, data are represented as tables (= feature vectors)

- **Each node can also have an associated vector-structured data**
  - In practice, we use both



name	age	sex	address	…
○○	40	male	Tokyo	…
××	30	female	Osaka	…

table-structured data

Network-structured data

Several network structured data in the real world

- **Nodes represents constituent elements and links represent relations among them**

network structured data	nodes	links
WWW	pages	hyperlinks
SNS	people communities	friendships memberships
biological networks	genes proteins	regulations interactions

- **Not only those static relations, dynamic relations such as**
  - e-mail exchanges
  - cooperation

can be represented by links

Network structure analysis is called “link mining” in data mining

- **Fundamental tasks of link mining defined by Getoor et al.**
  - **Node-related tasks**
    - Node ranking
    - Node classification
    - Node clustering

  - **Structure-related tasks**
    - Link prediction
    - Structured-pattern mining

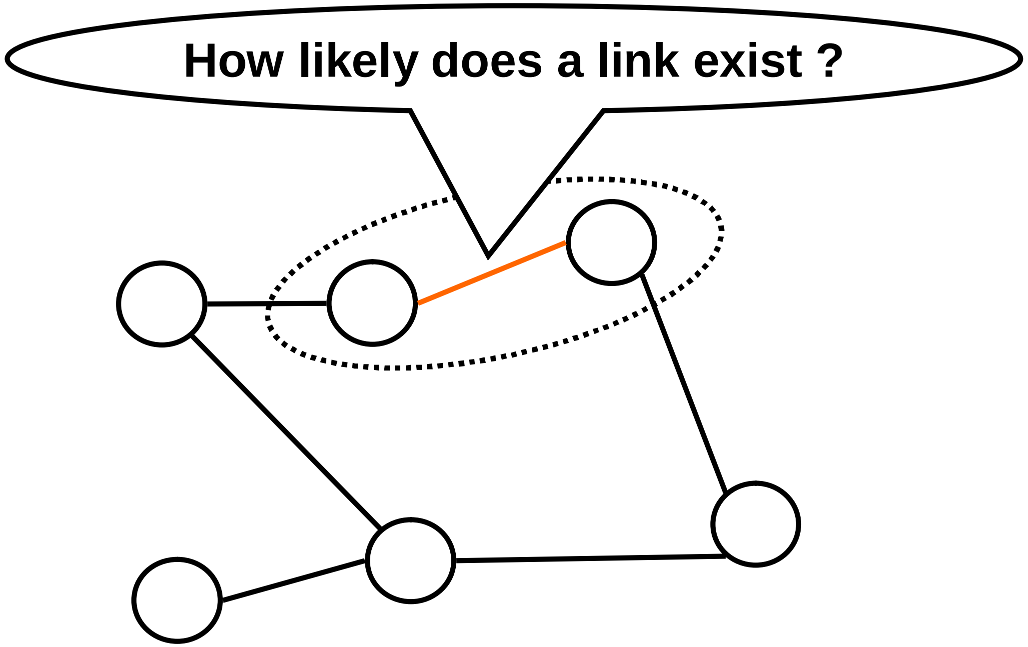
Getoor & Riehl.: *Link Mining: A Survey*, KDD Exploration, vol. 7, 2005

We focus on link prediction in this talk

- **Fundamental tasks of link mining defined by Getoor et al.**
  - **Node-related tasks**
    - Node ranking
    - Node classification
    - Node clustering
  - **Structure-related tasks**
    - Link prediction
    - Structured-pattern mining

Getoor & Riehl.: *Link Mining: A Survey*, KDD Exploration, vol. 7, 2005

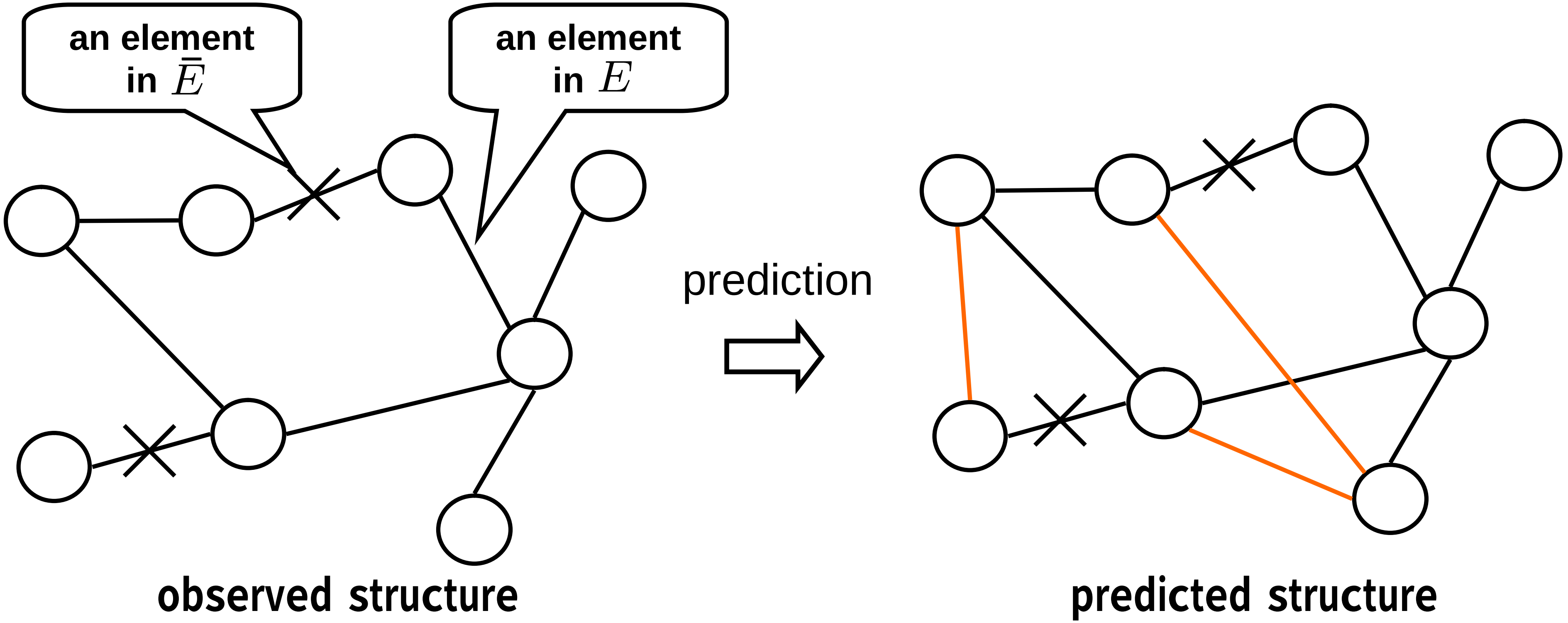
Link Prediction Problem



Link prediction problem as semi-supervised learning: Given partially observed network structure, predict the rest

- **Given**
  - **Some node pairs with links:** E
  - **Some node pairs without links:** $\bar{E}$
- **Predict**
  - **whether links exist or not for the other unknown pairs**

- **Applications**：
  - Prediction of biological networks
  - Recommendation in SNSs
  - Identifying "hidden links" in terrorist networks

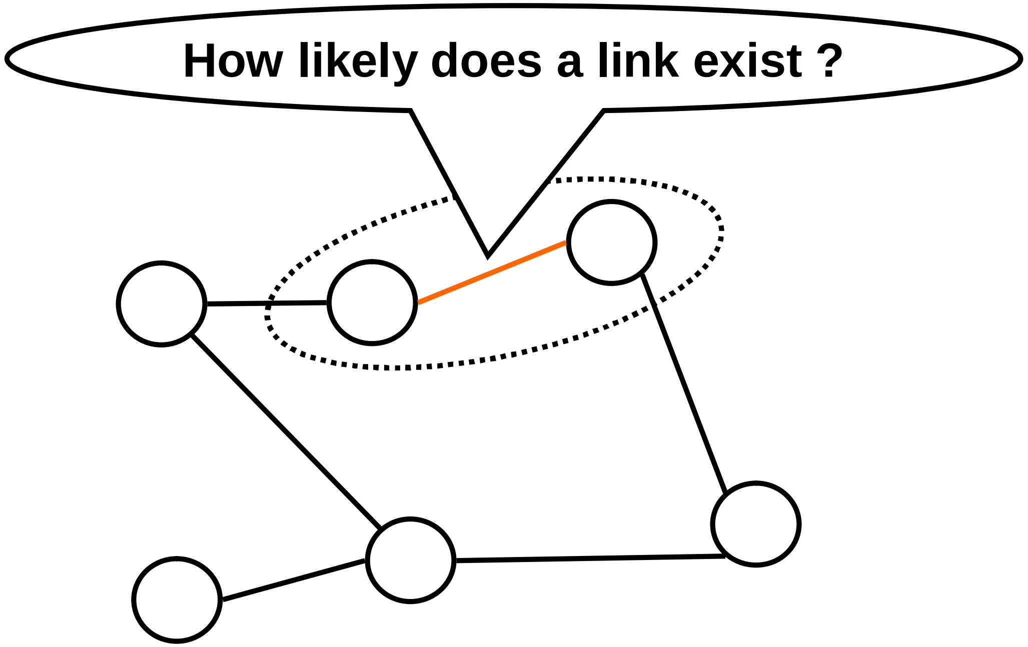


The link prediction problem can be considered as *a ranking problem of node pairs*

- **The link predictor must answer, for a pair of nodes,**
  - **how likely a link exits (= ranking problem), or**
  - **whether or not a link exists (= classification problem)**

  between the nodes

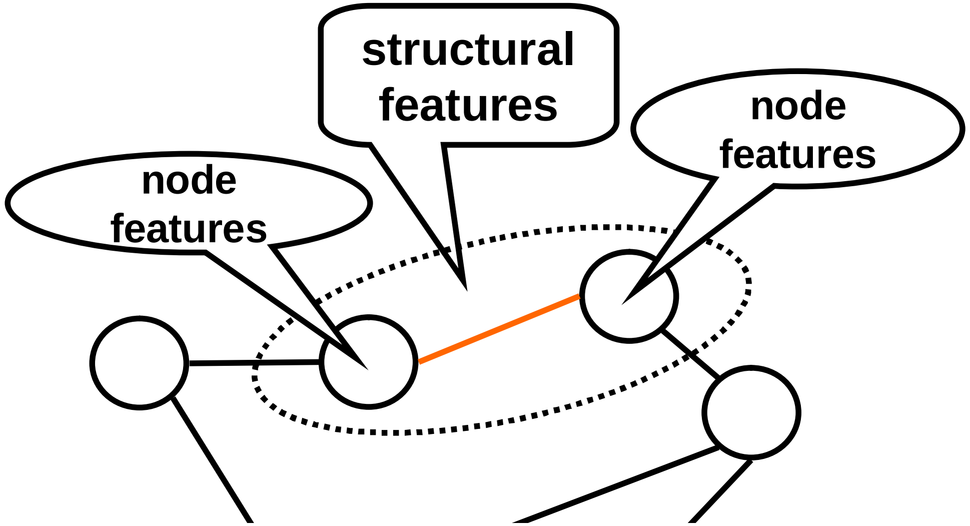
- **Assumption: Existence of link is determined independently**



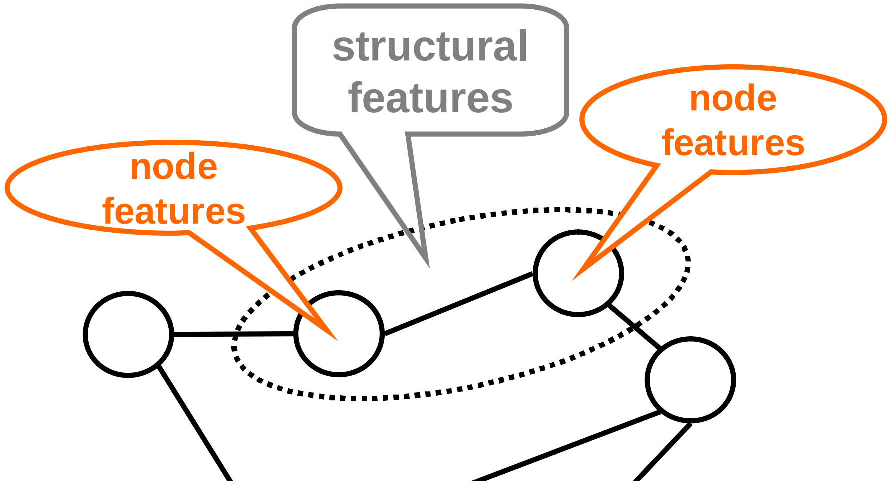
How likely does a link exist ?

Two types of information are available for link prediction: Node features and structural features

- **Node features: information owned by nodes themselves**
  - **Combined to define node-pair features**
  - **Examples:**
    - In SNSs, each person has his/her own personal information such as address, age, ...
    - In protein networks, each protein (= a node) has its own sequence information
- **Structural features: information owned by link structures around node pairs**
  - Usually, inherently defined for node pairs
  - Traditionally proposed in the context of social network analysis in sociometrics
  - Also proposed in information retrieval
- **Link prediction is done based on them**



Link prediction based on node information



Link prediction based on node features uses *feature vectors of node pairs*

- **Ordinary classification is based on the node feature vector x$^{(i)}$ for the *i*-th data**

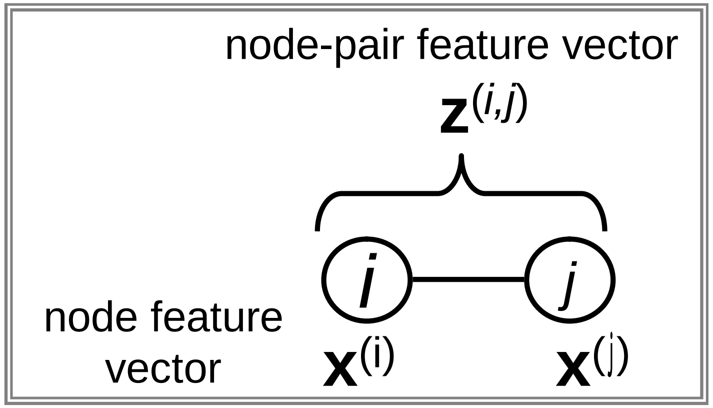
⇕

- **Link prediction is based on the feature vector z$^{(i,j)}$ for node pair (i, j)**
    - z$^{(i,j)}$ is constructed from the pair of node feature vectors **(**x$^{(i)}$, x$^{(j)}$**)**
- **The simplest way to define z$^{(i,j)}$ is to take concatenation of two vectors, or to take element-wise product (or whatever)**

$$\mathbf{z}^{(i,j)} = \left(x_1^{(i)} \cdot x_1^{(j)}\ ,\ x_2^{(i)} \cdot x_2^{(j)}\ ,\ x_3^{(i)} \cdot x_3^{(j)}\ ,\ \ldots \right)$$

, but this is not sufficient ...

- since it can not represent "*i* has a particular feature and *j* has a corresponding (another) feature"



node-pair feature vector

$\mathbf{z}^{(i,j)}$

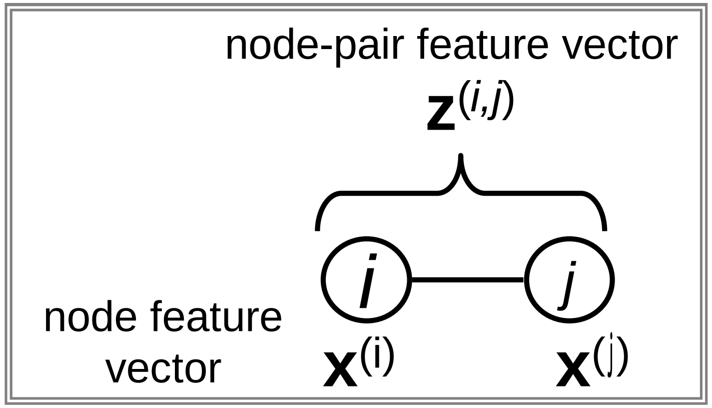
node feature vector

$\mathbf{x}^{(i)}$ $\mathbf{x}^{(j)}$

A more general feature vector for a pair of nodes is defined by tensor products of node feature vectors

- **The feature vector $\mathbf{z}^{(i,j)}$ for node pair (i, j) is defined by tensor product of the pair of node feature vectors $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$**

$$
\begin{aligned}
\mathbf{z}^{(i,j)} &= \mathbf{x}^{(i)} \otimes \mathbf{x}^{(j)} \\
&= \big(x_1^{(i)} \cdot x_1^{(j)},\ x_1^{(i)} \cdot x_2^{(j)},\ x_1^{(i)} \cdot x_3^{(j)},\ \ldots \\
&\qquad x_2^{(i)} \cdot x_1^{(j)},\ x_2^{(i)} \cdot x_2^{(j)},\ x_2^{(i)} \cdot x_3^{(j)},\ \ldots, \\
&\qquad x_3^{(i)} \cdot x_1^{(j)},\ x_3^{(i)} \cdot x_2^{(j)},\ x_3^{(i)} \cdot x_3^{(j)},\ \ldots \big)
\end{aligned}
$$



- **A linear predictor for the node pair (*i, j*) is given as**

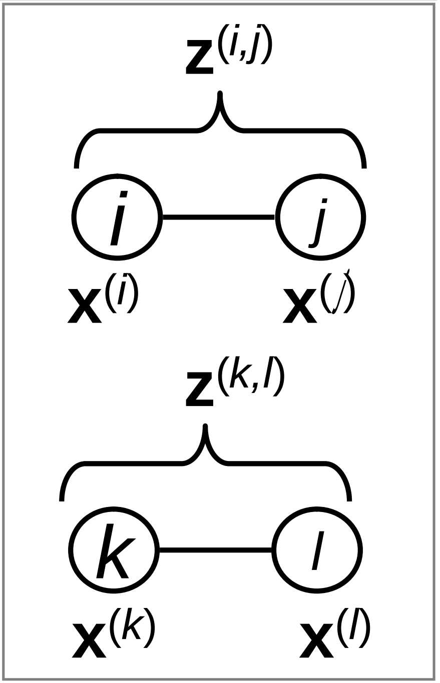
$$\text{prediction}(i,j) := \text{sign} < \mathbf{w}\ ,\ \mathbf{z}^{(i,j)} >$$

  - **w is the parameter**

Oyama & Manning: *Using Feature Conjunctions across Examples for Learning Pairwise Classifiers*, ECML 2004
Ben-Hur & Noble: *Kernel methods for predicting protein–protein interactions*, Bioinformatics, Vol. 21 Suppl. 1, 2005

Kernelization is simple

- **Naive computation of the kernel between two node-pair feature vectors $\mathbf{z}^{(i,j)}$ and $\mathbf{z}^{(k,l)}$ seems to need *O*(#dimensions^2)**



- **But it can be efficiently computed in *O*(#dimensions) by**

$$< \mathbf{z}^{(i,j)}, \mathbf{z}^{(k,l)} > = < \mathbf{x}^{(i)}, \mathbf{x}^{(k)} > \cdot < \mathbf{x}^{(j)}, \mathbf{x}^{(l)} >$$

- **When there is no ordering in a pair of nodes, symmetrize by**

$$< \mathbf{z}^{(i,j)}, \mathbf{z}^{(k,l)} > + < \mathbf{z}^{(i,k)}, \mathbf{z}^{(j,l)} >$$

- **Then, feed them into SVM, and done**

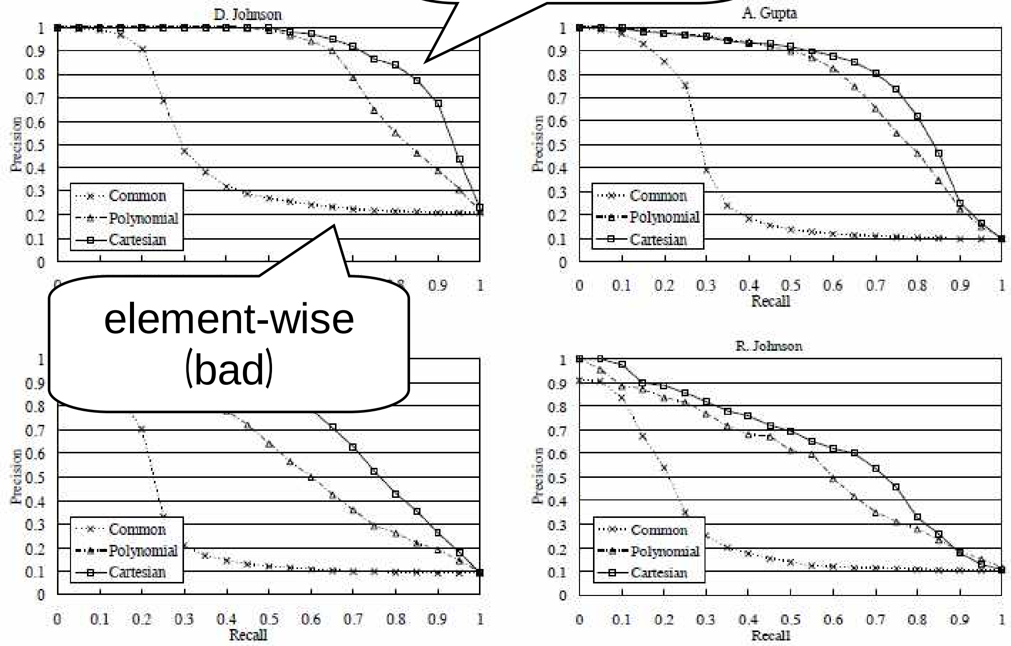
$$\mathrm{prediction}(i,j) := \mathrm{sign} \sum_{k,l} \alpha_{k,l} \left(< \mathbf{z}^{(i,j)}, \mathbf{z}^{(k,l)} > + < \mathbf{z}^{(i,k)}, \mathbf{z}^{(j,l)} > \right)$$

  - α is the parameter

Oyama & Manning: *Using Feature Conjunctions across Examples for Learning Pairwise Classifiers*, ECML 2004
Ben-Hur & Noble: *Kernel methods for predicting protein–protein interactions*, Bioinformatics, Vol. 21 Suppl. 1, 2005

The tensor-product-based feature vector outperforms the element-wise feature vector

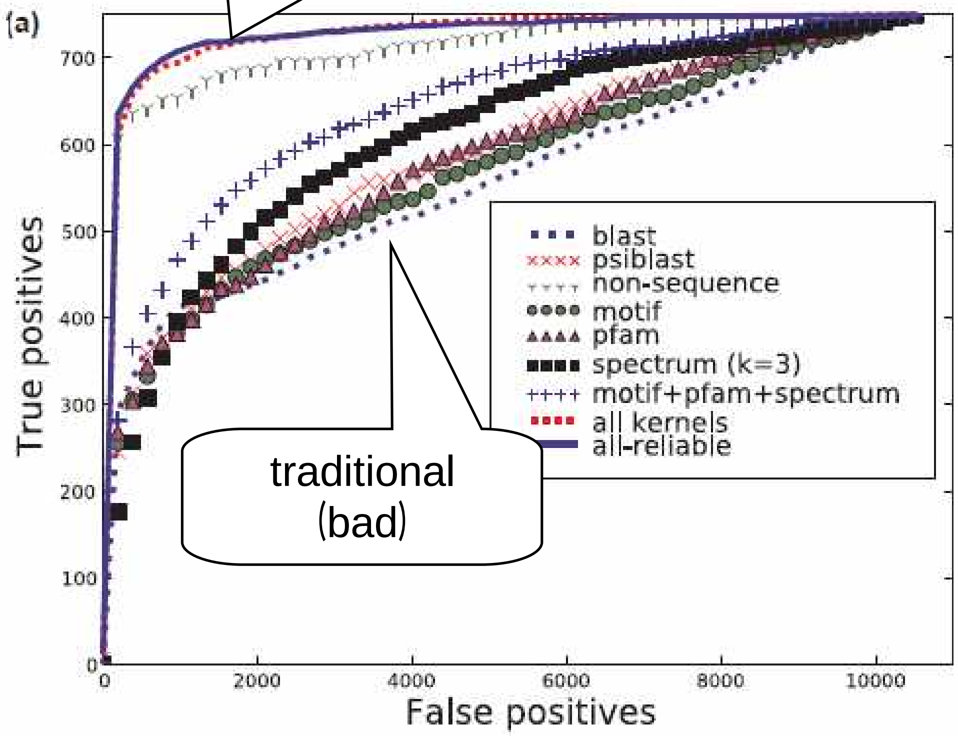
- Author network [Oyama & Manning]



tensor product
(good)

element-wise
(bad)

- Protein network [Ben-Hur & Noble]



proposed
(good)

traditional
(bad)

More efficient alternatives...

- **The kernel method enables computation in *O*(#dimensions), but needs *O*(#nodes^2) parameters**

⇩

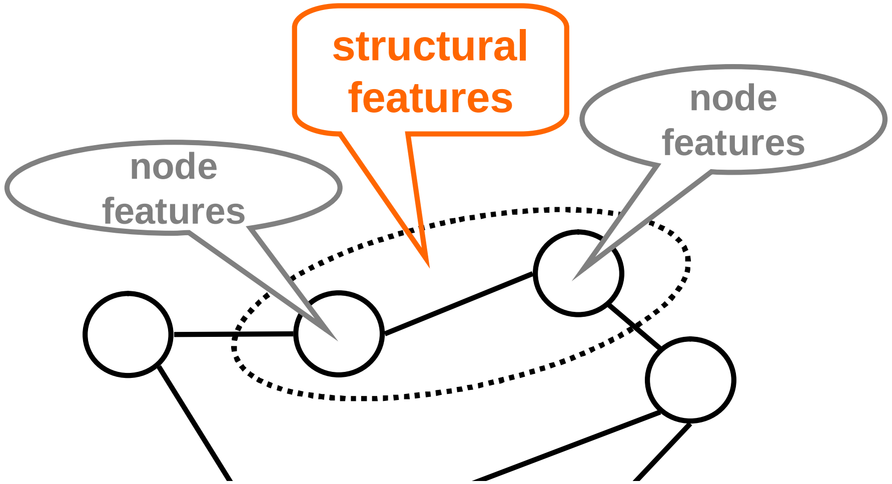
- **Yamanishi et al. propose more efficient way using kernel canonical correlation analysis (KCCA) by using only *O*(#nodes) parameters**
- **Kato et al. incorporate integration of multiple data sources**

Yamanishi et al.: *Protein Network Inference from Multiple Genomic Data: A Supervised Approach*, ISMB 2004

Vert & Yamanishi: *Supervised Graph Inference*, NIPS 2004

Kato, Tsuda and Asai: *Selective Integration of Multiple Biological Data for Supervised Network Inference*, Bioinformatics, Vol. 21, 2005

Link prediction based on structural information



Link metrics define how likely a link exists between two nodes

- **Link metrics := Degrees of likelihood of a link existing between two nodes based on the structural information around them**
  - Usually, inherently defined for node pairs
  - Traditionally proposed in the context of social network analysis in sociometrics
  - Also proposed in information retrieval
  - Example:
    - “the friends of your friends are probably in your friends”

    $\Rightarrow$ “Two nodes with many common neighbor nodes probably has a link”

- **How to use the link metrics ?**
  - Prediction only from positive examples : To predict links in descending order of any link metric
  - Supervised learning: To include as a part of feature vectors for node pairs

Several existing link metrics

- **Several link metrics are used in studies of social networks and scale-free networks**
  - **Common neighbors:** Likelihood of link existence is proportional to the number of common neighbors

$$\text{common neighbors} := |\Gamma(i) \cap \Gamma(j)|$$

Γ(i) is the set of neighbor nodes of node *i*

  - **Weighted common neighbors:**

$$\text{Adamic/Adar} := \sum_{k \in \Gamma(i) \cap \Gamma(j)} \frac{1}{\log |\Gamma(k)|}$$

$$\text{Jaccard's coefficient} := \frac{|\Gamma(i) \cap \Gamma(j)|}{|\Gamma(i) \cup \Gamma(j)|}$$

(used in information retrieval)

  - **Long-distance common neighbors:**

$$\text{Katz}_\beta := \sum_{l=1}^{\infty} \beta^l |\text{paths}_{i,j}^{(l)}|$$

$\text{path}_{i,j}^{(l)}$ is the set of paths of length *l* from node *i* to *j*

    - almost identical to the diffusion kernel
  - **Preferential attachment:** Nodes with many neighbors will get more neighbors

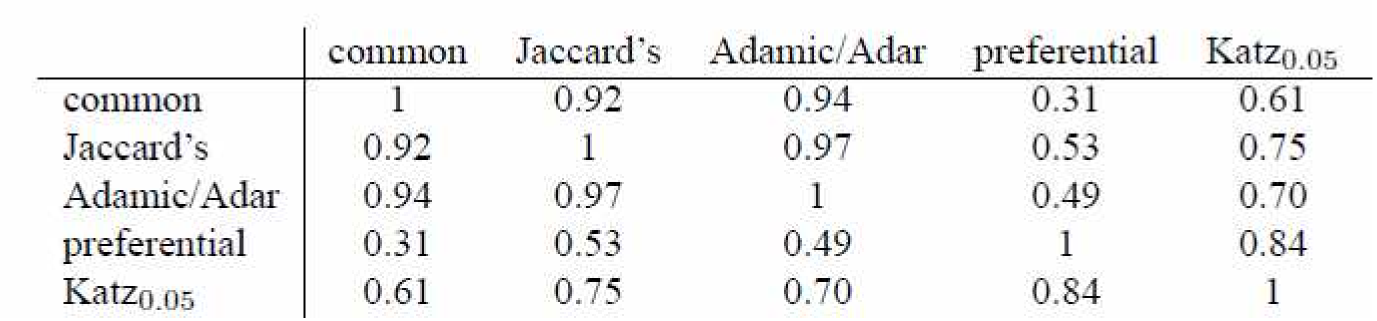
$$\text{preferential attachment} := |\Gamma(i)| \cdot |\Gamma(j)|$$

- **Most of them are inspired by corresponding evolution models of network structure**

Liben-Nowelly & Kleinberg: *The Link Prediction Problem for Social Networks*, CIKM 2004

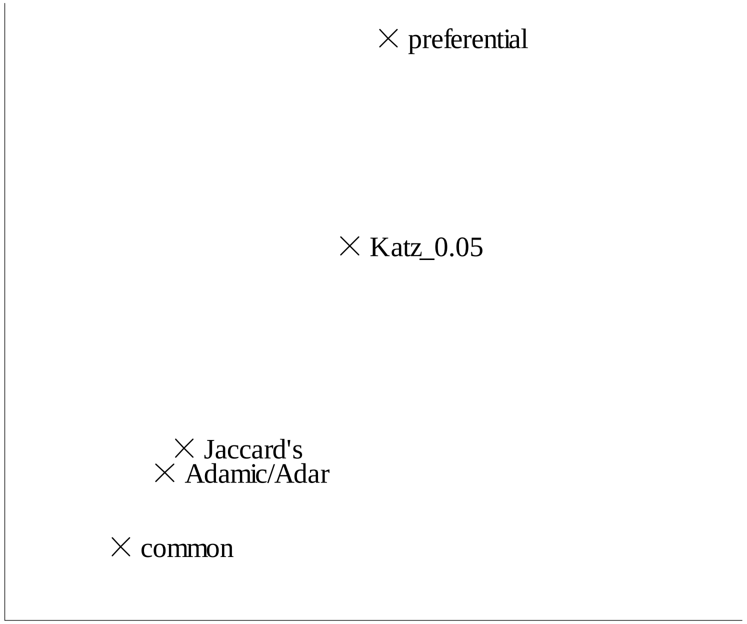
Appendix: Relations among link metrics

- **Spearman correlations (= correlations of orders) for metabolic network data)**
  - Large coefficients indicate that the corresponding metrics are similar



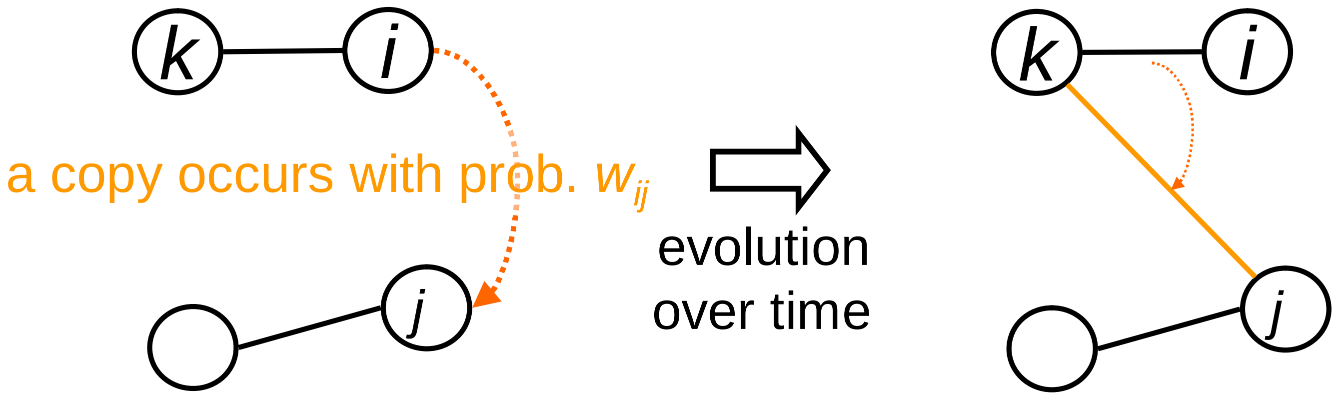
	common	Jaccard's	Adamic/Adar	preferential	$\mathrm{Katz}_{0.05}$
common	1	0.92	0.94	0.31	0.61
Jaccard's	0.92	1	0.97	0.53	0.75
Adamic/Adar	0.94	0.97	1	0.49	0.70
preferential	0.31	0.53	0.49	1	0.84
$\mathrm{Katz}_{0.05}$	0.61	0.75	0.70	0.84	1

  - Visualization by MDS
    - preferential attachment and common neighbors are most apart



"How high can we go only with structural information ?" We parameterized link metrics to improve prediction performance

- **Motivation: Will parameterized versions of link metrics improve predictive performance ?**
- **We propose a systematic way of deriving parameterized link metrics from parameterized network evolution models**
- **As an instance, we parameterized the "copy-and-paste" model**
  - Originally proposed for modeling evolution of WWW structure by Kleinberg et al.
  - Also makes sense as an evolution model of biological networks

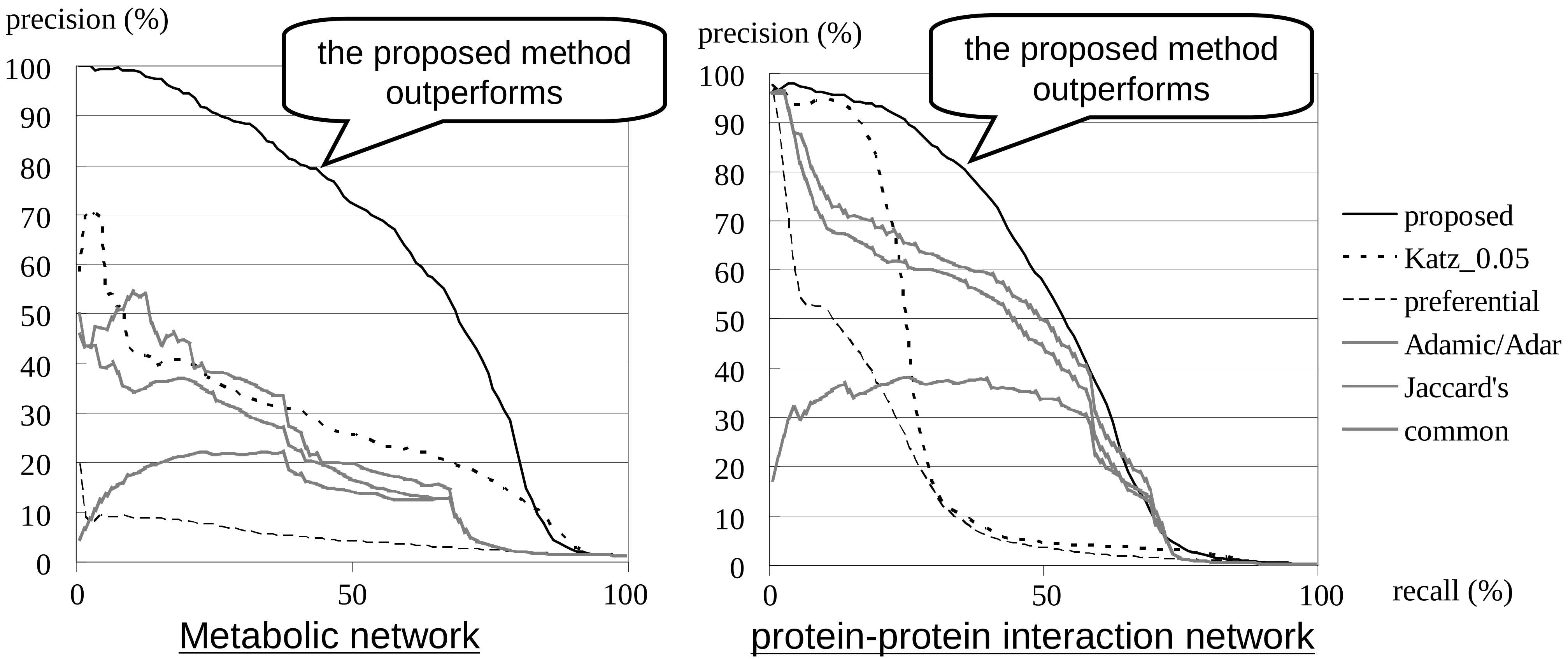


H. Kashima and N. Abe: A Parameterized Probabilistic Model of Network Evolution for Supervised Link Prediction, *IEEE International Conference on Data Mining* (ICDM), 2006 (To appear).

Experimental results: Parameterization works well

- **Task: Structure prediction of two biological networks**
- **The proposed method outperformed over various link metrics**

(parameterization of the other metrics ... future work)



Metabolic network

#nodes：700

protein-protein interaction network

#nodes： 3000

The proposed framework enables to derive parameterized link metrics from parameterized network evolution models

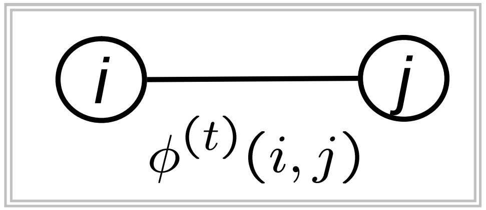
- **Our framework offers a systematic way of deriving link metrics**
  - from the parameterized network evolution models

- **In many cases, enough evolution history of network structures is not available, so parameter estimation is not possible**

- **Our solution**

1. **Derive the stationary expected state of the network structure**
2. **Fit the stationary expected state to the known part of the network**

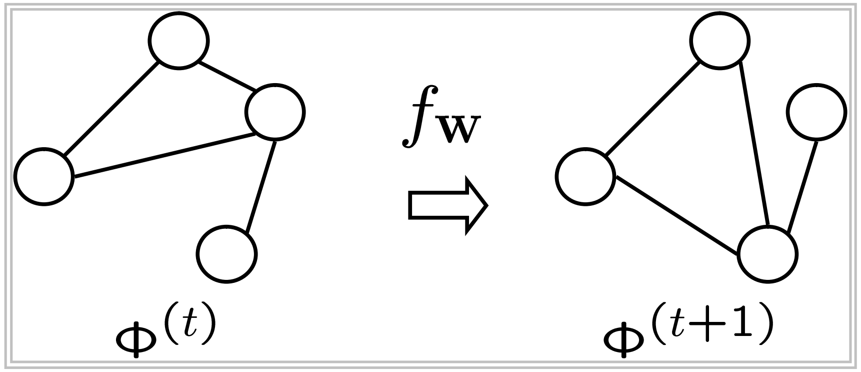
Step 1. Derive the expected stationary state of the network structure



- **Definition:** $\Phi^{(t)} :=$ **network structure at time** ***t***
  - link label $\phi^{(t)}(i,j) \in \{0,1\}$:= 1 if a link exists between (*i*, *j*)
    := 0 if not exists
- **Underlying Markov model of network evolution**

$$\Phi^{(t+1)} = f_{\mathbf{w}}(\Phi^{(t)})$$

  - $f_{\mathbf{w}}$ is the transition function with parameter **w**



- **Problem: w can not be identified since the evolution history** $\Phi^{(1)}, \Phi^{(2)}, \Phi^{(3)}, \ldots$ **is not available**
  - We only know “the partial structure of the current network”
) We need a constraint ...
- **Solution: Consider the “stationary expected state”** $E[\Phi^{(\infty)}]$ **as the “representative state”**

$$E[\Phi^{(\infty)}] = \lim_{t\to\infty} E[f_{\mathbf{w}}(\Phi^{(t)})]$$

Step 2: Fit the stationary expected state to the known part of the network

- **We hypothesize the stationary expected state $E[\Phi^{(\infty)}]$ represents the current network structure**
- **Maximize the following objective function with respect to the parameter w**

$$\underset{\mathbf{w}}{\text{maximize}} \sum_{\{i,j\}\in E} \log E[\phi^{(\infty)}(i,j)] - \sum_{\{i,j\}\in \bar{E}} \log E[\phi^{(\infty)}(i,j)]$$

i.e. Find the most appropriate parameter w* that reproduces the observed part of the network

- $E[\phi^{(\infty)}(i,j)] = 1$ for existing links
- $E[\phi^{(\infty)}(i,j)] = 0$ for non-existing links

- $E[\phi^{(\infty)}(i,j)]$ **for unknown part** $(i,j) \notin E \cup \bar{E}$ **are the predicted link labels**

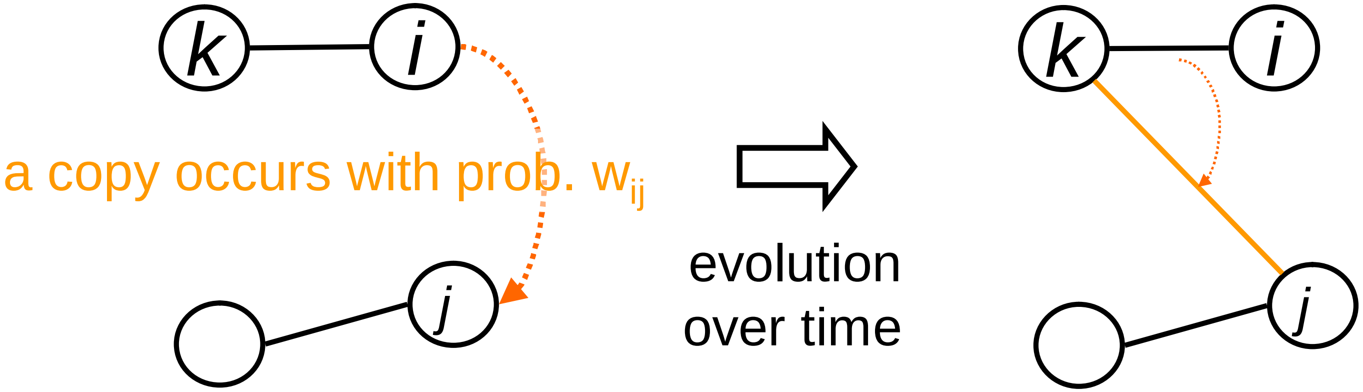
An example:
Our network evolution Markov model f_w is the "copy-and-paste" model

- **At each step, a link label is copied somewhere in the network**
  - from node *i* to node *j* with probability $w_{i,j}$ $\sum_{i,j} w_{i,j} = 1,\ w_{i,j} \geq 0$
  - Procedure

1. Select a pair of nodes (i,j) with probability $w_{i,j}$
2. Select node *k* uniformly at random (other than *j*)
3. Link label $\phi^{(t)}(i,k)$ is copied to (j,k)

$$\phi^{(t+1)}(j,k) := \phi^{(t)}(i,k)$$

- **$w_{i,j}$ is interpreted as node *i* 's "influence" on node *j***
  - *j*-san's association is affected by *i*-san's association

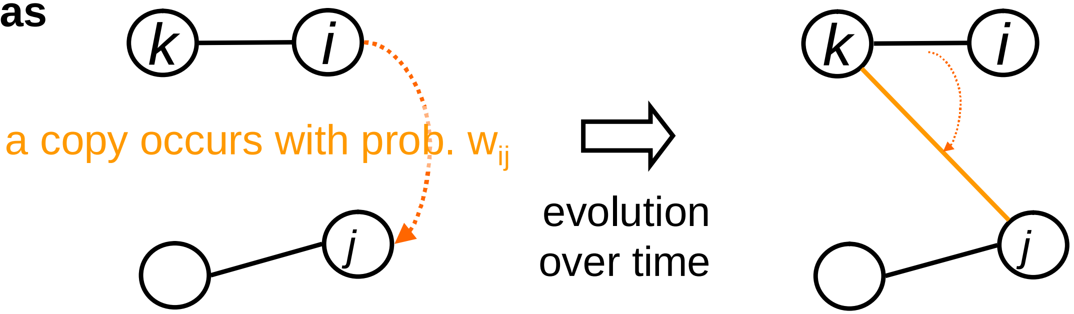


In the "copy-and-paste" model, a link label at the next time step is determined by one of two possibilities

- **Link label $\phi^{(t+1)}(i,j)$ at time *t+1* is determined by one of the following two possibilities**

1. The link label was copied from somewhere at time *t*
2. A copy occurred somewhere else at time *t* , so the link label remained the same

$$\phi^{(t+1)}(i,j) = \underbrace{\frac{1}{|V|-1}\left(\sum_{k\neq i,j} \underbrace{w_{kj}\phi^{(t)}(k,i)}_{\phi^{(t)}(k,i)\ \textbf{was copied from k to j}} + \underbrace{w_{ki}\phi^{(t)}(k,j)}_{\phi^{(t)}(k,j)\ \textbf{was copied from k to i}}\right)}_{\textbf{1. The link label was copied at time } t} + \underbrace{\left(1-\frac{1}{|V|-1}\sum_{k\neq i,j} w_{kj}+w_{ki}\right)\phi^{(t)}(i,j)}_{\textbf{2. The copy at time } t \textbf{ was occurred somewhere else}}$$



The stationary expected state of the “copy-and-paste” model is fed into the optimization problem

- **The original “copy-and-paste” network evolution model is**

$$\phi^{(t+1)}(i,j) = \frac{1}{|V|-1}\left(\sum_{k\neq i,j} w_{kj}\phi^{(t)}(k,i) + w_{ki}\phi^{(t)}(k,j)\right) + \left(1 - \frac{1}{|V|-1}\sum_{k\neq i,j} w_{kj} + w_{ki}\right)\phi^{(t)}(i,j)$$

$\Downarrow$ **take the expectation and $t \to \infty$**

- **The derived stationary expected state is**

$$E[\phi^{(\infty)}(i,j)] = \frac{\sum_{k\neq i,j} w_{kj} E[\phi^{(\infty)}(k,i)] + w_{ki} E[\phi^{(\infty)}(k,j)]}{\sum_{k\neq i,j} w_{kj} + w_{ki}}$$

- **The optimization problem is**

$\Downarrow$ **substitute into the optimization problem**

$$\underset{\mathbf{w}}{\text{maximize}} \sum_{\{i,j\}\in E} \log E[\phi^{(\infty)}(i,j)] - \sum_{\{i,j\}\in \bar{E}} \log E[\phi^{(\infty)}(i,j)]$$

The resulted optimization problem is solved by using EM algorithm

- **The stationary expected state has unobserved variables (= link labels to be predicted)**

some unobserved

$$E[\phi^{(\infty)}(i,j)] = \frac{\sum_{k\neq i,j} w_{kj} E[\phi^{(\infty)}(k,i)] + w_{ki} E[\phi^{(\infty)}(k,j)]}{\sum_{k\neq i,j} w_{kj} + w_{ki}}$$

- **Our solution: EM algorithm**
  - 【M-Step】 Fix the unobserved link labels, estimate the parameter
    - *Exponentiated gradient descent* solves this efficiently

$$\mathbf{w}^* = \underset{\mathbf{w}}{\mathrm{argmax}} \sum_{\{i,j\}\in E} \log E[\phi^{(\infty)}(i,j)] - \sum_{\{i,j\}\in \bar{E}} \log E[\phi^{(\infty)}(i,j)]$$

  - 【E-Step】 Fix the parameter, and evaluate the expected value of the unobserved link labels
    - by solving a system of simultaneous linear equations

- **In practice, we can do this in a sequential manner, not in a batch manner**

In summary, we proposed a new *parametric* model for link prediction

We introduced

- **A link prediction method based on evolution models of network structure**
- **A parameterized evolution model based on the node-copy model**
- **An efficient estimation algorithm**

Methods for link prediction

- **Network-structured data**
- **Link prediction problem**
- **Link prediction methods based on node information**
- **Link prediction methods based on structural information**
    - [Our contribution] A parameterized model for link prediction