

# Kernels for Link Prediction with Latent Feature Models

Canh Hao Nguyen and Hiroshi Mamitsuka

Bioinformatics Center, ICR, Kyoto University,  
Gokasho, Uji, Kyoto, 611-0011, Japan  
{canhhao,mami}@kuicr.kyoto-u.ac.jp

**Abstract.** Predicting new links in a network is a problem of interest in many application domains. Most of the prediction methods utilize information on the network's entities such as nodes to build a model of links. Network structures are usually not used except for the networks with similarity or relatedness semantics. In this work, we use network structures for link prediction with a more general network type with latent feature models. The problem is the difficulty to train these models directly for large data. We propose a method to solve this problem using kernels and cast the link prediction problem into a binary classification problem. The key idea is not to infer latent features explicitly, but to represent these features implicitly in the kernels, making the method scalable to large networks. In contrast to the other methods for latent feature models, our method inherits all the advantages of kernel framework: optimality, efficiency and nonlinearity. We apply our method to real data of protein-protein interactions to show the merits of our method.

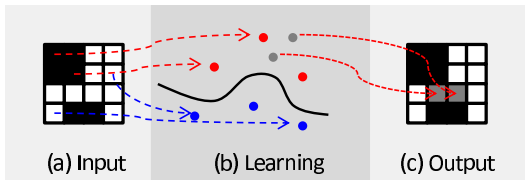
## 1 Introduction

Link prediction is a major problem in relational data learning [6]. In Social Networks and Collaborative Filtering, one needs to suggest links for entities for recommendation. In Bioinformatics and Chemoinformatics, potentially valid links such as interactions are required to speed up experimental processes. In order to predict links in relational data, one needs to provide a common model for both entities and relationships (such as links) in the data. As these two types of information are of different natures, models are difficult to design and learn. While modeling entities are of common practice, modeling relationships is usually of more difficulty. For link prediction, these relations are interpreted differently and reflect different semantics. While in social networks, links are usually of similarity or relatedness nature, it is not the case elsewhere. It is the target of this work to deal with a more general type of network structures, to build link models and to train them on large-sized real data.

In principle, there are two different kinds of information used to learn a link model for link prediction. One is the information of network entities such as nodes of the networks. The methods falling in this category usually use the information of a pair of nodes to induce the label of having or not having a link

[2,8,3]. Typical examples are sequences or profile information of genes, which are used to predict links (edges) in their networks. By using only the information on networks' entities, the models of the networks assume an independent and identical distribution of links. In other words, the structures of the links themselves are completely ignored. This is an unrealistic assumption in many domains where networks' structures themselves have patterns, such as social networks [12,13] and biological networks [18,1]. It is an objective of this work to show that network structures themselves contain information for the task.

The second kind of information used to predict links is structures (topologies) of the networks themselves. Similarity networks, due to its analogy to kernels [10,7,17,16,11], can be modeled with kernels and therefore, there exist scalable methods. Bipartite networks can also rely on kernels for each of their parts together with aligning the parts [19]. However, it is more difficult to deal with networks without similarity or relatedness semantics. A common network type is modeled with latent feature models [14,1]. This network model includes similarity networks and bipartite networks as special cases. For this type of networks, the available models are usually the plain latent feature models or matrix based models [5,1,13]. Training these models usually requires to generate latent features explicitly. This is a very time-consuming process, usually does not applicable to medium-sized networks. Another problem is that approximation in the training process leads to suboptimal solutions. Without approximation, these models do not scale to the sizes of real data, even for medium-sized networks. It is our motivation to be able to learn a model of this general type of networks efficiently and to avoid suboptimal solutions.



**Fig. 1.** Given an adjacency matrix as input, the method embeds cells (links or nonlinks) into a feature space and using classification to infer new links

We provide a kernel method to link prediction problem using network structures for the network structures modeled with latent features. The overall description of the method is depicted in Figure 1. We design kernels to encode the structures implicitly, without the need of generating the latent features themselves. The idea is to give high kernel values to the pairs of links that potentially share latent features. We show how suitable the kernel is to sparse networks. By not inducing the latent feature directly, our method is much faster compared to the long execution times faced by the methods that explicitly induce latent features. Our method also gives a globally optimal solution as opposed to the other methods. Nonlinearity can be incorporated into the model seamlessly. For

these advantages, our method gives a very high predictive performance for link prediction problems and applicable to real datasets of large sizes.

The paper is organized as follows. We describe the generative model of networks' links with latent features in Section 2. We then develop our kernels for this model in Section 3. We visualize the idea of the kernels for this model in Section 4. We show our application in protein-protein interaction networks in Section 5 and conclude the paper.

## 2 Latent Feature Models of Graphs

### 2.1 Biological Motivation

An example is that a protein (node) is a collection of domains (features). A protein-protein interaction (PPI) is caused by an interaction between two domains from the two proteins [4]. However, the knowledge of domain may be incomplete, and domain-domain interaction is far less understood. Therefore, we wish to incorporate the domain-domain interaction knowledge in an implicit way. Given enough links, we want to infer from many pairs of interacting proteins' common features that play the role of domains and some pairs of features are likely to interact for PPI task.

### 2.2 Latent Feature Models of Graphs

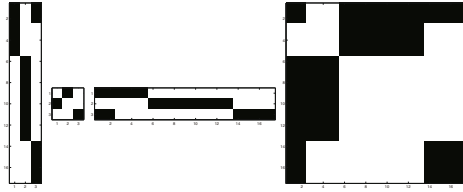
We describe a latent feature model of graphs as also appeared in [13]. In this model, a link (edge) in the graph is generated by the relation between the latent features of the adjacent nodes. Denote  $A \in \mathbb{R}^{n \times n}$  as the adjacency matrix of the graph. In general,  $A$  can be any real matrix. For our special purpose of modeling undirected networks, we assume that  $A$  is a binary symmetric matrix. Denote  $F \in \mathbb{R}^{n \times d}$  as a binary matrix where each row is a  $d$ -dimensional vector of latent features. Denote  $W \in \mathbb{R}^{d \times d}$  as a real matrix encoding strength of feature interactions. That is,  $W_{ij}$  encodes the strength of interaction between the  $i^{th}$  feature and the  $j^{th}$  feature. We define that  $(F, W)$  is a latent feature model for  $A$  when

$$A = FWF^T. \quad (1)$$

It is possible that there are several causes for a link but we only record one link (existence of a link rather than multiplicity of the link). Therefore, the equality in the equation (1) is replaced by the element-wise operator  $\min(x, 1)$  where  $x$  is the entry of the right hand side of the equation (1). It can be rewritten as follows

$$A = \min(FWF^T, 1). \quad (2)$$

**Simulation Example:** The idea of latent feature model is depicted in Figure 2. In this example, the set of nodes have three latent features, with the first two nodes has two features. As for the feature interaction matrix, it indicates that the first and the second features interact, also the third feature interact with itself.



**Fig. 2.**  $FWF^T \rightarrow A$ : the adjacency of the graph (rightmost) is generated by the three matrices: the latent feature matrix ( $F$  - leftmost), the feature interacting matrix ( $W$ ) and the transpose of the latent feature matrix ( $F^T$ ). A black cell indicates a positive entry and a white cell indicates a 0 entry. In our latent feature model, product of the first three matrices generates the last one.

For the benefits of using latent feature models instead of latent class models, we refer to [13] for details. The following properties are observed.

- Each entry of  $W$  generates a subgraph given  $F$ . The overall generated graph is a superimposition of these subgraphs. This makes the model an additive one.
- If  $W$  is diagonal, then the generated graph can be decomposed as a set of cliques. This is a model of similarity graph.
- If  $W$  is symmetric, then the generated graph is symmetric. It is usually used to encode undirected graphs.
- If  $F$  has exactly one nonzero entry in each row (such as the ones generated by Chinese Restaurant Processes (CRP)), then generating  $F$  is equivalent to clustering of nodes.
- If the nodes can be divided into two groups, each with a separate set of features, and  $W$  has only interactions of features from the two different sets, then the generated graph is bipartite.

These properties make latent feature models general generative models for graphs.

### 2.3 Ideal Kernels

Given a latent feature model as a generative model of graphs, one wish to define a kernel that encodes the similarity of the nodes using this model. The semantic similarity of two nodes under this model is the amount of latent features they share, weighted by the importance of each feature. Therefore, we define the *ideal kernels* as follows.

**Definition (Ideal kernels).** Define the set of ideal kernels for a latent feature model of nodes  $F$  to be  $\mathbb{K} = \{K^*(D)\}_D$ , with  $D \in \mathbb{R}^{d \times d}$  is any diagonal matrix with positive entries, and

$$K^*(D) = FDF^T. \quad (3)$$

A kernel value  $K^*(D)_{ij} = F_i D(F_j)^T$  is the weighted sum of the number of common feature between the  $i$ -th and  $j$ -th nodes. However, since latent feature

models are hardly obtained, the ideal kernels are not available. Any kernel defined from data with latent feature models should be close to some ideal kernels in some senses.

### 3 Link Kernels with Latent Features

We describe our kernel method to link prediction given the latent feature model assumption of the graph structures. The idea is to use the model to derive a kernel between all pairs of the nodes (called link kernel). We define the following terms: a link is considered as a *positive* pair of nodes while a *negative* pair of nodes encodes the non-existence of a link (nonlink).

---

**Input:** Adjacency matrix  $A$ .

1. Construct a node kernel  $K_n$  following latent feature models.
2. Construct a link kernel  $K$  based on  $K_n$ .
3. Learn a SVM on  $K$ .

**Output:** New adjacency matrix based on the SVM.

---

**Fig. 3.** Network Structure based Link Prediction

The link prediction problem is then formulated as a binary classification problem. In the end, we classify the two classes, link class and nonlink class using Support Vector Machines. The overall strategy is in Figure 3.

The kernel between pairs is based on the kernel between nodes themselves (called node kernel). We first describe the node kernel ( $K_n$ ) that encodes the latent feature model, its relations to ideal kernels in sparse graphs, and then the link kernel.

#### 3.1 Node Kernels with Latent Features

As latent feature models are the generative models of graphs, we wish to define the similarity of two nodes as the likelihood of having common latent features. This is inherently different from similarity models where similarity means the likelihood of reaching the other node through random walks [12]. However, latent features are implicit, we must estimate the similarity, in form of kernels, between nodes empirically. Knowing that nodes with common features tend to link to common neighbors in latent feature models, we define the basic node kernels as the amount of common neighbors of the nodes, as follows.

$$K_n = \text{norm}(AA^T), \quad (4)$$

where *norm* indicates the normalization operator to make the diagonal elements all 1. Specifically,

$$(K_n)_{ij} = \frac{(AA^T)_{ij}}{\sqrt{(AA^T)_{ii}(AA^T)_{jj}}}. \quad (5)$$

A special case of this kernel is when all the nodes have exactly one feature (such as generated by CRP), then  $K_n(a, b)$ , for any node  $a$  and  $b$ , is either 1 or 0, indicating if they have the feature in common or not. An implication is that in networks where latent features are expected to be sparse,  $K_n$  behaves similarly to this extreme case, showing a good indication of common latent features.

Given the basic node kernel, additional tricks can be applied on top of this kernel to make new families of kernels. Diffusion kernels and exponential kernels on top of  $K_n$  still conserve the idea of latent features. Of course, the higher the exponential we take on  $K_n$  the looser the idea can be kept. However, please note that the most general version of spectral transform [11] is not guaranteed to conserve the latent feature assumption.

Note that this node kernel can be one of many terms in other kernels for similarity graphs such as path-based [15,10,7,12]. However, we find that our proposed kernel in particular encodes the latent feature model that is suitable for our problem.

### 3.2 Relation to Ideal Kernels on Sparse Graphs

When the graphs are sparse (in our applications), sparse models are required to model them. The following propositions show the relationship between the node kernel  $K_n$  and the ideal kernels when the models are sparse. Denote  $S_i$  as the set of latent features of node  $i$ .

**Proposition 1 (Positivity).** *If  $S_i \cap S_j \neq \emptyset$  (two nodes  $i$  and  $j$  share at least one latent feature) and that feature interacts with another feature, then  $(AA^T)_{ij} > 0$ , therefore  $(K_n)_{ij} > 0$ .*

*Proof.* The proof can be easily seen because when they share a feature, they have a common nonempty neighborhood and therefore,  $(K_n)_{ij} > 0$ .  $\square$

This shows that if the values of ideal kernels on two nodes are positive, the value of  $K_n$  is positive as well. This means that whenever the two nodes are similar (positive kernel value) in the model, the kernel  $K_n$  can recognize that. This is useful for sparse graphs (causing sparse kernels),  $K_n$  is not sparser than any ideal kernel.

**Proposition 2 (Monotonicity).** *Suppose that the edges have the same amount of neighbors in the sense that  $(AA^T)_{ii}(AA^T)_{jj} = (AA^T)_{kk}(AA^T)_{ll}$ . If  $(S_i \cap S_j) \supseteq (S_k \cap S_l)$  then  $(K_n)_{ij} \geq (K_n)_{kl}$ .*

The first assumption is about the equal amounts of neighbors for the two pairs of nodes. The amount of neighbors of a pair of nodes is defined to be the product

of numbers of its adjacent nodes. The conclusion is that, the more latent features they share, the higher the kernel value is. This is a property of the ideal kernels by the way they are defined, showing an analogy of  $K_n$  to ideal kernels.

*Proof.* As  $(S_i \cap S_j) \supseteq (S_k \cap S_l)$ , the common neighborhood of the nodes  $i$  and  $j$  is a superset of the the common neighborhood of the nodes  $k$  and  $l$ . Therefore,  $(AA^T)_{ij} \geq (AA^T)_{kl}$ .

$$(K_n)_{ij} = \frac{(AA^T)_{ij}}{\sqrt{(AA^T)_{ii}(AA^T)_{jj}}} \geq \frac{(AA^T)_{kl}}{\sqrt{(AA^T)_{ii}(AA^T)_{jj}}} = (K_n)_{kl} \tag{6}$$

from the definition of the node kernel in (4). □

**Lemma 1 (One feature interaction).** *For any latent feature model  $(F, W)$ , there exists another latent feature model  $(F', W')$  that gives (i) the same adjacency matrix, (ii) the same set of ideal kernels and (iii) the feature interaction matrix is nonzero on at most one of its entries in any row and column.*

In other words, there exists another mathematically equivalent model giving the same ideal kernel sets and adjacency matrix in which each feature interacts with only one another feature.

*Proof.* The idea of the proof is to place a nonzero  $w_{ij}$  in one new row and column of  $W'$  and duplicate the columns of  $F$  when necessary to make  $F'$ , keeping the adjacency matrix unchanged. Supposed that the (unnormalized) adjacency matrix  $A$  is computed by

$$A = FWF^T = \sum_{ij} w_{ij} F_{.i} (F_{.j})^T, \tag{7}$$

where  $F_{.i}$  is the  $i$ -th column of  $F$ .

Denote  $I = \{(i, j)\}$  that  $w_{ij} \neq 0$  then

$$A = \sum_{(i,j) \in I} w_{ij} F_{.i} (F_{.j})^T. \tag{8}$$

We then construct the  $F'$  and  $W'$  by sequentially appending feature columns and feature interaction matrix values in the formula 8 as follows.

1. Mark all the indices in  $I$  as available.
2. Initialize  $F'$  and  $W'$  to empty matrices.
3. Go the the next available index in  $I$ , pick the pair  $w_{ij}$ , then
  - If  $i = j$ , meaning that  $w_{ii}$  indicates a self interacting feature, then append the feature vector  $F_{.i}$  at the end of the already constructed  $F'$ . Append a new row and column of  $W'$  with the only nonzero element  $w_{ij}$  on the diagonal of  $W'$ . Mark the index of  $w_{ij}$  in  $I$  unavailable. Repeat the process from step 3.

- If  $W$  is symmetric ( $A$  is symmetric) then  $w_{ij} = w_{ji}$ . Append the feature vectors  $F_{.i}$  and  $F_{.j}$  to the end of  $F'$ . Suppose the size of  $W'$  is  $k$ , then append two new rows and columns of  $W'$  with the only nonzero elements are  $w_{ij}$  at  $W'_{k+1,k}$  and  $W'_{k,k+1}$ . Mark the indices of  $w_{ij}$  and  $w_{ji}$  in  $I$  unavailable. Repeat the process from step 3.
- If  $W$  is not symmetric then append the features as the symmetric case to  $F'$ . For  $W'$ , only one nonzero element is added to  $W'_{k,k+1}$ . Mark the index of  $w_{ij}$  in  $I$  unavailable. Repeat the process from step 3.

By doing this, then

$$F'W'(F')^T = \sum_{(i,j) \in I} w_{ij}F_{.i}(F_{.j})^T = FWF^T = A. \tag{9}$$

Also, each row or column of  $W'$  has at most one nonzero entry by the way  $W'$  is constructed. Since the set of columns of  $F'$  is the same as  $F$ 's, the set of ideal kernels are the same (only different by feature weighting).  $\square$

Hence, we have constructed a new model with a feature either interacts or is interacted with at most one another feature. This is to say that there is a mathematically equivalent model that each feature only interacts with one another feature. We use this fact to facilitate our sparsity reasoning as follows.

**Proposition 3 (Ideal condition).**  $K_n$  is an ideal kernel ( $K_n \in \mathbb{K}$ ) if  $WF^T$  has all row vectors uncorrelated.

*Proof.* When  $WF^T$  has row vectors uncorrelated, then:  $WF^T \times (WF^T)^T$  is diagonal. Denote  $D = WF^T \times FW^T$ , then:

$$K_n(A) = FWF^T \times FW^T F^T = FDF^T. \tag{10}$$

Since  $D$  is diagonal with nonnegative entries,  $K_n \in \mathbb{K}$ .  $\square$

**Corollary 1.** If  $W$  is an unmixing matrix of an Independent Component Analysis model for  $F^T$ , then  $K_n \in \mathbb{K}$ .

**Corollary 2.** If each node has only one latent feature, such as  $F$  is generated by a Chinese Restaurant Process or any class-based model in a model that  $W$  has only one nonzero entry in each row or column (guaranteed by Lemma 1), then  $K_n \in \mathbb{K}$ .

This is from the fact that  $WF^T$  has only one nonzero entry in each column, therefore any pair of row vectors would have no nonzero entries in common. This makes row vectors of  $WF^T$  uncorrelated.

When the model is sparse,  $K_n$  is close to an ideal kernel as follows. Denote  $E = F^T F$ , therefore  $E_{lk} = (F_{.l})^T F_{.k}$  is number of nodes having both features  $l$  and  $k$ . When  $l \neq k$ , we expect  $E_{lk}$  to be small for sparse models. Given the Lemma 1, we assume that  $W_{il}$  and  $W_{jk}$  are the only nonzero entries in rows  $i$



and  $j$ . Recall that  $K_n = FDF^T = FEWE^T F^T$ . Since the entries of  $W$  are scalars,  $D_{ij} = W_{il}W_{jk}E_{lk}$  means that  $D_{ij}^2/(D_{ii}D_{jj}) = E_{lk}^2/(E_{ll}E_{kk})$ . When  $F$  is sparse, off-diagonal elements of  $E$  is much smaller than diagonal ones (*diagonally dominant*), the same thing can be said for  $D$ . This means that  $D$  is as diagonally dominant as  $E$ . We show quantitatively how close  $D$  is to an ideal kernel.

**Proposition 4 (Approximation).** *When the model is sparse in the sense that  $(\sum_{i \neq j} |W_{il}W_{jk}E_{lk}|^p)^{\frac{1}{p}} \leq \delta$  for some small  $\delta \in \mathbb{R}$ ,  $p \geq 0$ , there exists an ideal kernel  $F\hat{D}F^T$  that is close to  $K_n$  in the sense that  $\|D - \hat{D}\|_p \leq \delta$ .*

In the condition  $(\sum_{i \neq j} |W_{il}W_{jk}E_{lk}|^p)^{\frac{1}{p}}$ , an entry  $W_{il}W_{jk}E_{lk}$  is the weighted the number of nodes having the two features  $l$  and  $k$  (should also be small for sparse models,  $l \neq k$  implies  $i \neq j$ ).  $\|\cdot\|_p$  is the  $p$ -norm of a matrix.

*Proof.* We construct  $\hat{D}$  as a diagonal matrix with  $\hat{D}_{ii} = D_{ii} = W_{il}^2 E_{ll}$ . Given that  $D_{ij} = W_{il}W_{jk}E_{lk}$ , then

$$\|D - \hat{D}\|_p = \left(\sum_{i \neq j} |D_{ij}|^p\right)^{\frac{1}{p}} = \left(\sum_{i \neq j} |W_{il}W_{jk}E_{lk}|^p\right)^{\frac{1}{p}} \leq \delta. \tag{11}$$

This shows that the more sparse the model, the closer  $D$  is to ideal kernels.  $\square$

When a model is *sparse* in the sense that each node has very few latent features, each latent feature interacts with one another features (by the Lemma 1),  $K_n$  should be close to the ideal kernel  $F\hat{D}F^T$  since  $F\hat{D}F^T$  is linear in  $\hat{D}$ .

All these properties make the kernel  $K_n$  a good approximation of ideal kernels in sparse models. Sparse models are suitable for our applications in sparse networks (nodes with small degrees). We elaborate more in the application part.

### 3.3 Link Kernels with Latent Features

Given the node kernel  $K_n$ , a kernel between two links is usually defined to be the combined similarity between the pairs of nodes of the links. We choose the widely used and experimentally justified tensor product pairwise kernel [2] to be the kernel for pairs as follows.

$$K((a, b), (c, d)) = K_n(a, c) \cdot K_n(b, d) + K_n(a, d) \cdot K_n(b, c). \tag{12}$$

Here is the kernel for the two pairs of nodes  $(a, b)$  and  $(c, d)$ . Denote  $a_i, b_j$  as the features of  $a$  and  $b$  respectively in the feature space of  $K_n$ , then the feature space of  $K$  consists of the following features for a pair  $(a, b)$ :

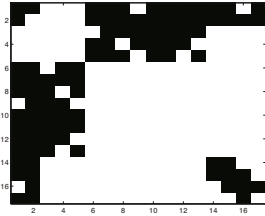
$$a_i b_j + a_j b_i, \tag{13}$$

as in [2]. Given that the node kernel  $K_n$  is supposed to be the likelihood of having common latent features, link kernel indicates the chance of having two pairs of nodes with common features. For example, when the pair  $\{a, c\}$  share common features and so do the pair  $\{b, d\}$ ,  $K((a, b), (c, d))$  is high.

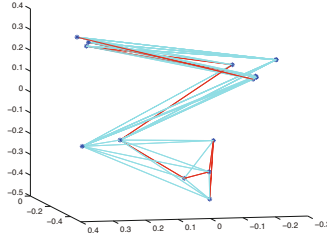
Nonlinearity can be incorporated into this kernel, such as Gaussian kernels on top of it.

## 4 Demonstration

We show the idea of our kernel using the simulation example in the previous section. Suppose that we observe an incomplete graph of the example as in the adjacency matrix in Figure 4 (80% of the links are observed). We show step by step the idea of node kernels and latent features.



**Fig. 4.** Adjacency matrix of an incomplete graph from the simulation example

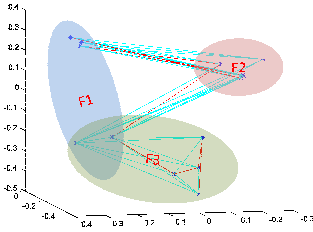


**Fig. 5.** Visualizing the graph with the node kernel. The cyan edges are the observed links of the graph while the red edges are the missing ones according to the model. We observed that the red ones follow the same patterns as the cyan ones.

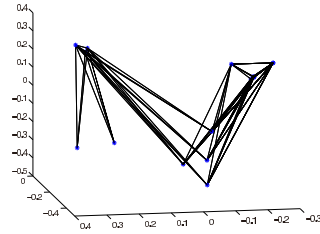
**Node Kernel:** We visualize the nodes using the node kernel defined in (4). We use kernel PCA and plot the nodes using the first three components in Figure 5. The nodes in the figure are the nodes of the graphs. The cyan edges are the observed links. The red edges are the missing links according to the model. First, we can observe that the cyan edges follow certain patterns of direction and endpoints. Another point can be observed is that the red edges follows the same patterns (endpoints lie in clusters, edges connect the same cluster pairs). This is exactly what we want in learning, testing data having the same distribution as training data. What left to be done is to keep these patterns in some spatial representations of the edges.

**Latent Features:** We look into the model to show the distribution of nodes with features. We label the nodes with the same feature in shaded ellipses in Figure 6. The labels of the eclipses,  $F1$ ,  $F2$  and  $F3$  correspond to the three latent features in the example. We can observe that the node kernel makes these nodes close to each other. The nodes with two features ( $F1$  and  $F3$ ) lie somewhere in between the nodes with only one of the two features ( $F1$  or  $F3$ ).

**Negative Links:** As shown in Figure 5 that the observed edges and missing edges have similar patterns. However, we use SVM to classify, we also wish the negative class (nonlink) not to follow the patterns. Therefore, we show all the edges that belong to the nonlink class in Figure 7. We can observe that they do not follow that patterns of the positive class (the same endpoints but connecting different cluster pairs from positive links').



**Fig. 6.** The positive class of links. The nodes are grouped by their common latent features. It shows that nodes sharing more common features tend to group together.



**Fig. 7.** The set of nonlink class. Compared to the link class, the nonlink class has totally different positions and orientations.

The demonstration shows that our designed node kernel successfully discovers the patterns of the link class as opposed to the nonlink class. It tends to group the nodes with common features close to each other as we designed. We wish to clarify the difference of our method from the other methods such as using Chinese Restaurant Processes or Indian Buffet Processes [14,9,13]. These methods group these nodes together explicitly while our method use a *soft* version of putting them close in a space. We believe that this is the key to be robust, allowing the inference step to be globally optimal and computationally efficient.

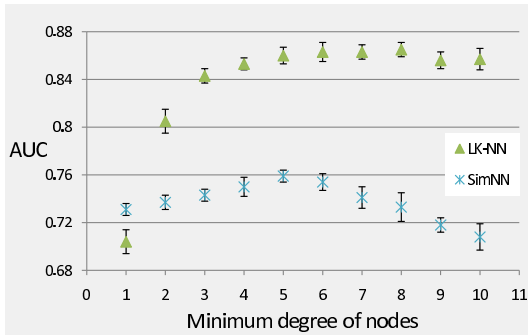
## 5 Application on Non-similarity Networks with Latent Features

Our targeted application is to model network structures with the latent feature models. Even though the model includes similarity networks as special cases, our target is to model more difficult ones, which are not similarity ones. Social networks are usually similarity ones, therefore not the prime target of our method. The PPI networks are the typical examples as it is non-similarity (we also show the experiments for this in the following subsection). There are other networks studied in Bioinformatics but they are too sparse and not large enough to study structures statistically.

We used the PPI networks of yeast and fruit-fly from DIP database for their largest hand-curated networks available. For statistical study, we attracted only the largest connected component of the network in which the degree of each node is not less than  $m$  (minimum degree of nodes). In our experiments, we show all the values of  $m$  from 1 to 10 (8 for fruit-fly). For yeast, the subnetwork with  $m = 1$  has 4762 nodes and 21836 links, and the subnetwork with  $m = 10$  has 756 nodes and 8924 links. For fruit-fly, the subnetwork with  $m = 1$  has 6644 nodes and 21501 links, and the subnetwork with  $m = 8$  has 713 nodes and 5243 links. This is the size of data that we expect and we will show latter on that the other methods based on latent feature inference do not scale to this size.

Real PPI networks are sparse: less than 20% of the nodes in yeast and none in fruit-fly have degrees of 10 or more. This is the reason for sparsity analysis in Section 3.2. The sparse networks require the models to be sparse as well.

For each subnetwork, we used SVM ( $C = 0.001$ , but the results are the same for a range of  $C$  from 0.0001 to 0.1) as the classifier for our link kernels. We showed the average AUC score of different train/test splits with the ratio 90/10. We first showed the appropriateness of the assumption of latent features as opposed to the usual assumption of similarity. We then showed the time required to build one model to reflect the difference in execution times or our method compared to Indian Buffet Process (IBP) as described in [13] (with parameter  $\alpha = 3$ , which we found to be a good trade-off to be able to train on our smallest datasets and induce a model with enough number of latent features). We then showed the performance of link kernels in link prediction. We also compared to sequence based link prediction to show the advantage using network structures.



**Fig. 8.** Latent feature versus similarity assumption: AUC of the direct method for link prediction (vertical) at different minimum degree of the network (horizontal axis)

### 5.1 Latent Feature versus Similarity

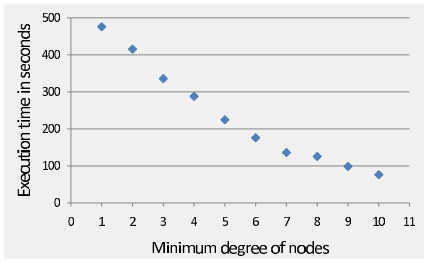
We show the fitness of the latent feature model on yeast PPI networks as opposed to the similarity model in Figure 8. We used the direct method [18] to predict links based on the two models. The idea of the direct method is simply nearest neighbor classifiers. For similarity assumption, probability of having link between two nodes is proportional to how similar they are in their connectivity patterns (called SimNN). On the other hand, with latent feature model, probability of a link between two nodes is determined by the class of its nearest neighbor in the *link kernel* (called LK-NN), not the kernel between nodes.

The figure showed a significantly different AUC scores of the two methods on yeast's networks. LK-NN was usually much higher (around 0.1 and more). It showed that latent feature model was more suitable to PPI networks than the conventional similarity one. The exception was at the subnetwork with the minimum degree of nodes of one. Since we were using only network structures, the nodes with degree one may not contribute to the network structure in these

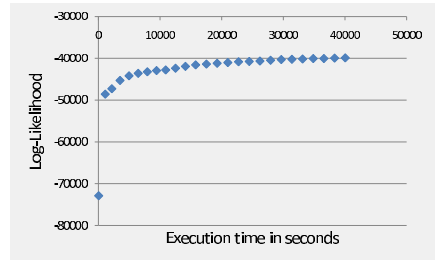
models. Therefore, results for the subnetworks with larger nodes' degrees demonstrated the reasonability of the assumptions used. For this reason, the latent feature assumption was more reasonable than similarity assumption here, and the methods for similarity networks were not recommended to use.

## 5.2 Execution Time

We show execution time required to build one model using our link kernels in Figure 9. To compare to IBP, we also show the execution time in the process of building one model before it *burns in* in Figure 10. For the long times required by IBP, we only show the execution time for the smallest subnetwork (minimum degree of 10 with only 756 nodes), which is supposed to require the least time among all the subnetworks.



**Fig. 9.** The time required to build one model using link kernels (in seconds) at subnetworks with different minimum degrees. Note that for the smallest subnetwork, the execution time is less than 90 seconds.

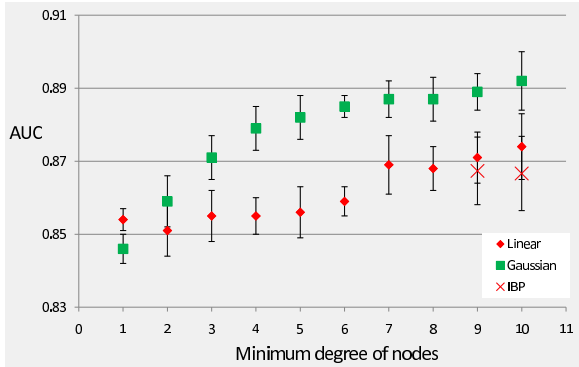


**Fig. 10.** The time to train an IBP model of the smallest subnetwork with  $m = 10$ . The vertical axis is the log-likelihood obtained from the model during the training process as a function of time in the horizontal axis.

We can see that the link kernels required less than 90 seconds for the smallest subnetwork, and increased to less than 500 seconds for the largest subnetwork of 4762 nodes. On the other hand, IBP required many hours for the smallest subnetwork of 756 nodes and did not burn-in on larger ones within one day. We conclude that our method using kernel saved many orders of magnitude the time to train one model, making training on medium-sized networks possible. This is a key advantage of our kernel method.

## 5.3 Link Prediction Results

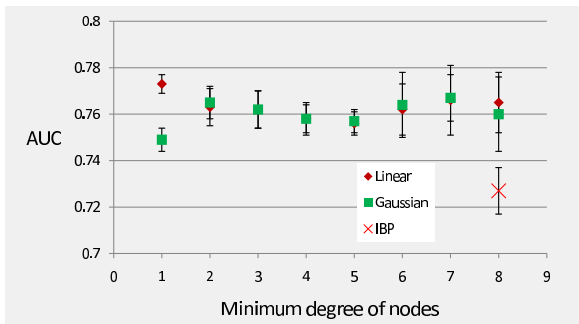
We compared the prediction ability of our method using link kernel to the baseline of using IBP as in [13] in Figure 11. The results are for different subnetworks with different minimum degrees. For small minimum degrees, the subnetworks have higher coverage on the whole network while the large minimum degrees



**Fig. 11.** Link prediction results on yeast PPI network: AUC of link prediction with different methods (vertical) at different minimum degrees of the network (horizontal axis). Note that IBP does not scale with larger datasets, its results are not available.

will extract denser parts of the network, making statistical inference on this part more reliable. We show two versions of our link kernels: linear kernels and Gaussian kernels ( $\gamma = 2$ ). The incomplete results of IBP was due to the fact that experiments took too much time (more than one day) to train one model.

We can read from yeast’s results in Figure 11 that linear kernels had similar AUC scores with IBP. However, when using the nonlinear version of Gaussian kernels, AUC scores were significantly higher. We conclude that our kernel based method provided a significantly higher AUC scores than that of IBP. One surprise was that even using only network topology, we archived high AUC scores close to 0.9. These scores were much higher than random prediction. Given that the PPI networks are known to be noisy and incomplete, this experiment showed that there are patterns of the topology of the PPI networks. It also showed that our method was effective to encode these patterns.



**Fig. 12.** Link prediction results on fruit-fly PPI network: AUC of link prediction with different methods (vertical) at different minimum degrees of the network (horizontal axis). IBP results are missing due to their time consumption.

Similarly, we can see in the fruit-fly’s results in Figure 12 that our method based on kernels outperformed IBP. The results on fruit-fly’s networks were much lower than on yeast due to the fact that fruit-fly’s ones are sparser, involving more proteins and there are no proteins with degrees of 10 or more.

#### 5.4 Comparison to Sequence-Based Prediction

As opposed to using network structures, traditional methods use nodes’ information such as protein sequences. Therefore, we also compared to spectrum kernels on sequences to predict links in the same manner. The results were not shown in the Figure 11 and 12 because they ranged differently. The highest AUC score on any subset for yeast was  $0.71 \pm 0.008$  and  $0.65 \pm 0.016$  for fruit-fly. We observe that network structures gave statistically significantly higher AUC scores (with *t*-tests at 0.01 level). This might be due to the fact that kernels based on sequences contain too much redundant information, since sequence-based kernels use all *k*-mers across the protein sequences. Sequence based kernels are redundant as only small parts determine its interaction ability to others.

## 6 Conclusion

We studied the problem of predicting new links using network structures in a more general type of networks. Specifically, we studied the networks that can be modeled by generative processes with latent features. This is a more general model of networks than the usually assumed similarity networks in most of the applications. In order to model real networks of medium or large size, we used kernels and casted the problem as a supervised learning one, inheriting optimality, efficiency and nonlinearity of the kernel framework. We showed the suitability of the kernels on sparse networks. We applied to the non-similarity networks of protein-protein interactions. The results showed that our kernel-based method gave a much higher performance than direct latent feature inference by IBP. Our method was also many orders of magnitude faster, and scaled to the sizes of real networks, unlike IBP. It was also shown that network structures gave higher results than information of the nodes. We conclude that for sparse networks with latent feature models, our method is able to utilize the relevant information in network structures to give faster, more scalable solutions, and significantly higher performances.

**Acknowledgments.** C.H.N. is supported by JSPS. H.M. has been partially supported by BIRD, JST (Japan Science and Technology Agency). We acknowledge anonymous reviewers for helpful comments.

## References

1. Airoldi, E.M., Blei, D.M., Fienberg, S.E., Xing, E.P.: Mixed membership stochastic blockmodels. *Journal of Machine Learning Research* 9, 1981–2014 (2008)
2. Ben-Hur, A., Noble, W.S.: Kernel methods for predicting protein-protein interactions. In: *ISMB (Supplement of Bioinformatics)*, pp. 38–46 (2005)

3. Bleakley, K., Biau, G., Vert, J.-P.: Supervised reconstruction of biological networks with local models. In: ISMB/ECCB (Supplement of Bioinformatics), pp. 57–65 (2007)
4. Deng, M., Mehta, S., Sun, F., Chen, T.: Inferring domain-domain interactions from protein-protein interactions. In: Proceedings of the Sixth Annual International Conference on Computational Biology, pp. 117–126. ACM Press, New York (2002)
5. Ding, C.: Orthogonal nonnegative matrix tri-factorizations for clustering. In: SIGKDD, pp. 126–135 (2006)
6. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press, Cambridge (2007)
7. Kandola, J.S., Shawe-Taylor, J., Cristianini, N.: Learning semantic similarity. In: NIPS, pp. 657–664 (2002)
8. Kato, T., Tsuda, K., Asai, K.: Selective integration of multiple biological data for supervised network inference. *Bioinformatics* 21(10), 2488–2495 (2005)
9. Kemp, C., Tenenbaum, J.B., Griffiths, T.L., Yamada, T., Ueda, N.: Learning systems of concepts with an infinite relational model. In: Proceedings of the 21st National Conference on Artificial Intelligence, vol. 1, pp. 381–388. AAAI Press, Menlo Park (2006)
10. Kondor, R.I., Lafferty, J.D.: Diffusion kernels on graphs and other discrete input spaces. In: ICML, pp. 315–322 (2002)
11. Kunegis, J., Lommatzsch, A.: Learning spectral graph transformations for link prediction. In: ICML 2009: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 561–568. ACM, New York (2009)
12. Liben-Nowell, D., Kleinberg, J.M.: The link prediction problem for social networks. In: CIKM, pp. 556–559 (2003)
13. Miller, K., Griffiths, T., Jordan, M.: Nonparametric latent feature models for link prediction. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C.K.I., Culotta, A. (eds.) *Advances in Neural Information Processing Systems*, vol. 22, pp. 1276–1284 (2009)
14. Navarro, D.J., Griffiths, T.L.: A nonparametric bayesian method for inferring features from similarity judgments. In: NIPS, pp. 1033–1040 (2006)
15. Newman, M.E.J.: Clustering and preferential attachment in growing networks. *Physical Review E* 64(2), 025102+ (2001)
16. Shimbo, M., Ito, T., Mochihashi, D., Matsumoto, Y.: On the properties of von neumann kernels for link analysis. *Machine Learning Journal* 75(1), 37–67 (2009)
17. Smola, A.J., Kondor, R.: Kernels and regularization on graphs. In: Schölkopf, B., Warmuth, M.K. (eds.) *COLT/Kernel 2003*. LNCS (LNAI), vol. 2777, pp. 144–158. Springer, Heidelberg (2003)
18. Vert, J.-P., Yamanishi, Y.: Supervised graph inference. In: NIPS (2004)
19. Yamanishi, Y.: Supervised bipartite graph inference. In: NIPS, pp. 1841–1848 (2008)