Latent Feature Kernels for Link Prediction on Sparse Graphs

Canh Hao Nguyen and Hiroshi Mamitsuka

***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 networks with similarity or relatedness semantics. In this paper, we use network structures for link prediction with a more general network type with latent feature models. The problem with these models is the computational cost to train the 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 the kernel framework: optimality, efficiency, and nonlinearity. On sparse graphs, we show that our proposed kernels are close enough to the ideal kernels defined directly on latent features. We apply our method to real data of protein–protein interaction and gene regulatory networks to show the merits of our method.**

***Index Terms*— Latent feature kernels, latent feature models, link prediction, sparse graphs.**

I. Introduction

LINK prediction is a major problem in relational data learning [1]. 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 is common practice, modeling relationships is usually more difficult. For link prediction, these relationships 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 aim of this paper to deal with a more general type of network structures to build link models and to train them on large-sized real data.

Manuscript received June 8, 2012; revised August 16, 2012; accepted August 19, 2012. Date of publication October 2, 2012; date of current version October 15, 2012. The work of C. H. Nguyen and H. Mamitsuka was supported in part by the Institute for Bioinformatics Research and Development (BIRD) of the Japan Science and Technology (JST) agency and KAKENHI Grants-in-Aid for Scientific Research under Grant 24300054 of MEXT, Japan. The work of C. H. Nguyen was supported in part by the Japan Society for the Promotion of Science.

The authors are with the Bioinformatics Center, Institute of Chemical Rerearch, Kyoto University, Kyoto 611-0011, Japan (e-mail: canhhao@kuicr.kyoto-u.ac.jp; mami@kuicr.kyoto-u.ac.jp).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2012.2215337

In principle, there are two different kinds of information used to learn a link model for link prediction. One is the information on entities of networks such as nodes. The methods falling in this category usually use the information of a pair of nodes to induce the label with or without a link [2]–[4]. 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 the entities of networks, 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 structures of networks themselves have patterns, such as social networks [5], [6] and biological networks [7], [8]. There are different models for networks such as class-based [9], [10], feature-based [11], [12], or space-based [13], [14] models that have been proven to be useful for networks from a variety of domains of application. It is an objective of this paper to show that network structures themselves contain information for the task.

The second kind of information used to predict links is the structures (topologies) of the networks themselves, with or without node information. Similarity networks, because of their analogy to kernels [15]–[20], 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 [21]. Relational graphical models [10], [22], [23] are more general, allowing the incorporation of node information with topological templates. These models are difficult to adapt to specific network models as well as computationally expensive. One special type of network structure, which is of interest recently, is models with latent features [8], [12], [24]. For this type of networks, the available models are usually the plain latent feature models [6], [8] or matrix-based models [13], [14]. Training these models usually requires the generation of latent features explicitly. This is a very time-consuming process, and usually does not obtain globally optimal solutions. It is difficult to scale them to networks with thousands of nodes. More details of the network structure models can be seen in [25]. In this paper, we deal with latent feature models of network structures.

The problem we are faced with is that networks are known to be generated from latent features. It is the case in biological networks where a link is a physical attachment of proteins. These networks are nonsimilarity networks in the sense that a link between two nodes does not mean that the nodes have some features in common [26]. Methods for social networks usually use similarity semantics and, therefore, are not suitable for our problem. The features that generate the links in the networks are usually not known for all the nodes in the

networks, and therefore the methods using information on network entities are not applicable. We need to use patterns of network structures to model the networks themselves.

On our targeted applications, one key point is that these networks are reasonably sparse. We also show some statistics in the application sections on real data. The good news is that, in sparse networks, if their structures can be learned statistically, the connectivity pattern for each node must be governed by a small number of factors. By having a small number of different factors for a node, there is a high chance that the links from the node share the same factor, i.e., have the same generative pattern. In essence, the connectivity patterns can give some information about the global structure of the networks, making it possible to predict new links in the networks using these patterns (see Section IV for a formal analysis).

We provide a kernel method to the link prediction problem using network structures that are modeled with latent features. The overall description of the method is depicted in Fig. 1. We design kernels to encode the structures implicitly, without the need for 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 analytically how suitable the kernel is to sparse networks. That is, given that the networks are sparse, our defined kernels will be close enough to the ideal kernels, which are the kernels that capture the semantics of the model but not computable from data. By using latent feature implicitly together with support vector machine (SVM) classifiers [27] on top, our proposed method has the following advantages.

1) Our method is significantly faster than other methods that infer latent features explicitly.
2) It gives globally optimal solutions unlike other methods that may converge to local minima.
3) It can inherit seamlessly other advantages of SVMs such as nonlinearity.

Because of these advantages, our method gives a very high predictive performance for link prediction problems and is applicable to real datasets of large sizes. We also show that the use of network structures gives higher performance than when using basic node information because the nodes tend to contain redundant information for the link prediction task.

The rest of this paper is organized as follows. We describe the generative model of network structures with latent features in Section II. We then develop our kernels for this model and make a simulation to demonstrate the idea in Section III. We show how suitable the kernels are for sparse networks in Section IV. We show our application in networks with latent features of protein–protein interaction (PPI) in Section V. We also show an application to a nonsimilarity network type of gene regulatory network in Section VI. We then sum up and conclude this paper in Section VII.

II. Latent Feature Models of Graphs

A. Biological Motivation

A protein (node) can be regarded as a collection of domains (features). A PPI is caused by a physical interaction between two domains from the two proteins [28]. However, the knowledge of domains may be incomplete, and domain–domain interactions are not well understood. Therefore, we wish to incorporate the domain–domain interaction knowledge in an implicit way. Given enough links, we want to infer the features that are shared by interacting protein pairs and play the role of domains, where some pairs of features are likely to interact to generate PPIs.

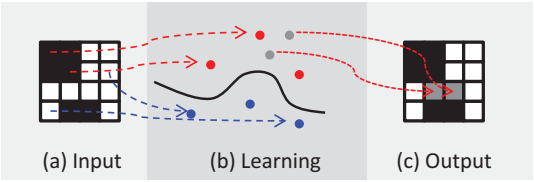

Fig. 1. Overall description of the method. Given an adjacency matrix as input, the method embeds cells (links or nonlinks) into a feature space and uses classification to infer new links.

A key difference between these networks and social networks is in the pair of features (each feature from each node) that cause a link. In PPI networks, the pair of features are the two domains with 3-D shapes that complement each other. They are usually different features. On the other hand, in social networks, the pair of features are usually the same, such as the same hobby of friends. This makes our PPI networks nonsimilarity networks, as opposed to similarity networks in social networks.

B. Latent Feature Models of Graphs

We describe a latent feature model of graphs, as also appearing in [6]. In this model, a link (edge) in the graph is generated by the relation between the latent features of the adjacent nodes. Let $A \in \mathbb{R}^{n\times n}$ denote 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. Let $F \in \mathbb{R}^{n\times d}$ denote a binary matrix where each row is a d-dimensional vector of latent features. Let $W \in \mathbb{R}^{d\times d}$, named feature interaction matrix, denote a real matrix encoding strengths of feature interactions. That is, W_{ij} encodes the strength of interaction between the ith feature and the jth feature. We say that the ith feature interacts with the jth feature if and only if $W_{ij} \neq 0$. We say that (F, W) is a latent feature model for A if

$$A = FWF^T. \tag{1}$$

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 (1) is replaced by the element-wise operator $\text{round}(\min(x, 1))$, where x is the entry on the right-hand side of (1). It can be rewritten as

$$A = \text{round}(\min(FWF^T, 1)). \tag{2}$$

1) Simulation Example: The idea of the latent feature model is depicted in Fig. 2. In this example, the set of nodes have three latent features, with the first two nodes having two features and the rest having one feature each. The feature

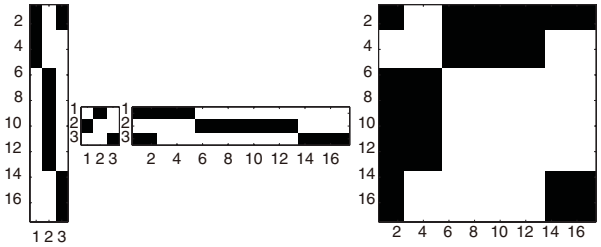

Fig. 2. $FWF^T \rightarrow A$. The adjacency of the graph (rightmost) is generated by the three matrices: the latent feature matrix (F, leftmost), the feature interacting matrix (W), and the transpose of the latent feature matrix (F^T). A black cell indicates a positive entry, and a white cell indicates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

interaction matrix indicates that the first and the second features interact, and also the third feature interacts with itself.

For the benefit of using latent feature models, we refer to [6] for details. The following properties are observed.

1) 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.
2) If W is diagonal, then the generated graph can be decomposed as a set of cliques. This is a model of similarity graphs.
3) If W is symmetric, then the generated graph is symmetric. It is usually used to encode undirected graphs.
4) If F has exactly one nonzero entry in each row (such as the ones generated by Chinese restaurant processes), then generating F is equivalent to clustering of nodes.
5) 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.

III. Latent Feature Kernels for Link Prediction

We describe our kernel method to link prediction given the latent feature model of the graph structures. The idea is to use the model to derive a kernel between all pairs of the nodes (named link kernel). We define the following terms: a link is considered as a positive pair of nodes, whereas the nonexistence of a link (nonlink) is encoded by a negative pair of nodes.

The link prediction problem is then formulated as a binary classification problem. In the end, we classify the two classes, the link class and the nonlink class, using SVMs. The overall strategy is shown in Algorithm 1.

The kernel between pairs is based on that between nodes themselves (named node kernel). We first describe the node kernel (K_n) that encodes the latent feature model, and then the link kernel.

A. Node Kernels With Latent Features

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 [5]. However, latent features are not observable, and, therefore, we must estimate the similarity, in the 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 ratio of common neighbors of the nodes, as follows:

$$K_n = \mathbb{D}^{-\frac{1}{2}} A A^T \mathbb{D}^{-\frac{1}{2}} \tag{3}$$

where $\mathbb{D}$ is a diagonal degree matrix: $\mathbb{D} \in R^{n \times n}$, and

$$\mathbb{D}_{ii} = \sum_{j=1}^{n} A_{ij}. \tag{4}$$

A special case of this kernel is when all the nodes have exactly one feature (such as generated by Chinese restaurant processes). The resulting $K_n(a, b)$ takes either 1 or 0 for any node a and b, depending on whether they have a feature in common or not. An implication is that, in networks in which 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 transformations 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 power we take on K_n, the looser the idea can be kept. However, note that the most general version of spectral transform [20] is not guaranteed to conserve the latent feature assumption.

Note that this node kernel occurs as a term in the series expansion of many path/walk-based kernels [5], [15], [16], [29], [30] designed for similarity graphs. However, we find that our proposed kernel in particular encodes the latent feature model that is suitable for our problem.

Algorithm 1 Network-Structure-Based Link Prediction

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 an SVM on K.

Output: New adjacency matrix based on the SVM.

B. Latent Features Kernels for Links

We call the link kernels built on the node kernels K_n (with latent feature assumption) latent feature kernels. 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. In other words, the similarity between a pair of nodes $\{a, b\}$ and another one $\{c, d\}$ is based on the similarity of: 1) a to c and b to d and 2) the similarity of a to d and b to c. This leads to two issues.

The first issue is how to define a combined similarity of two pairs of nodes. We break this down to a fixed order of nodes,

such as using the similarity in 1) only. This is basically the problem of finding a map between input a and b to c and d, respectively. Therefore, this is a structured output learning problem [31], [32] with the special property that the input and output are from the same space. In the context of kernel methods, we can use the existing joint kernels [33] to define an ordered kernel between the two pair of nodes as follows:

$$K^{\times}((a,b),(c,d)) = K_n(a,c) \cdot K_n(b,d) \tag{5}$$

or

$$K^{\times}((a,b),(c,d)) = K_n(a,c) + K_n(b,d) \tag{6}$$

or any other variations of joint kernels.

The second issue is how to account for the unordered nature of the links in general. That is, the similarity is supposed to be defined over both orders of the two pairs of nodes 1 and 2 at the same time. We follow the solution in [23] and [2][1] to break the asymmetry of the ordering by summing them up together

$$K((a,b),(c,d)) = K^{\times}((a,b),(c,d)) + K^{\times}((a,b),(d,c)). \tag{7}$$

It is noteworthy that the pairwise kernel as defined in [2] is a special case with the joint kernel defined in (5). We use this pairwise kernel in our later experiments. The interpretation of the kernel is as follows.

Let a_i and b_j, respectively, be the ith and jth features of a and b in the kernel-induced feature space of K_n. Then the feature space of K consists of the following features for a pair (a,b):

$$a_i b_j + a_j b_i. \tag{8}$$

If the node kernel K_n indeed captures the likelihood of two nodes having common features, the link kernel indicates the chance of having two pairs of nodes with common features. For example, when the pair $\{a,c\}$ shares common features, the likelihood of the pair $\{b,d\}$, $K((a,b),(c,d))$ sharing the same features is high.

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

C. Bipartite Graphs

We describe a special case of latent feature kernels for bipartite graphs. We show that in this case, our kernel framework is equivalent to structured output learning. Suppose that the graph is a bipartite graph; so for any link $\{a,b\}$, a is in the first part and b is in the second part of the graph. The similarity of two links $\{a,b\}$ and $\{c,d\}$, in a similar manner as before, would be the similarity between the corresponding nodes.

However, the key difference in bipartite graphs is that the pair of nodes in any link is ordered. It only makes sense to define the similarity between links by a combined similarity between nodes of the same parts. Therefore, in the same manner, we define a kernel between the two links as

$$K((a,b),(c,d)) = K^{\times}((a,b),(c,d)). \tag{9}$$

This is exactly the case of joint kernel map representation of structured output learning with kernels [32]. One difference is in the objective function for learning. While structured output learning estimates the density of the training data, our method samples negative examples and classifies the two classes instead.

[1]The same solution is also used in [34]. However, we do not use this formulation because of some of its drawbacks, a discussion of which is beyond the scope of this paper.

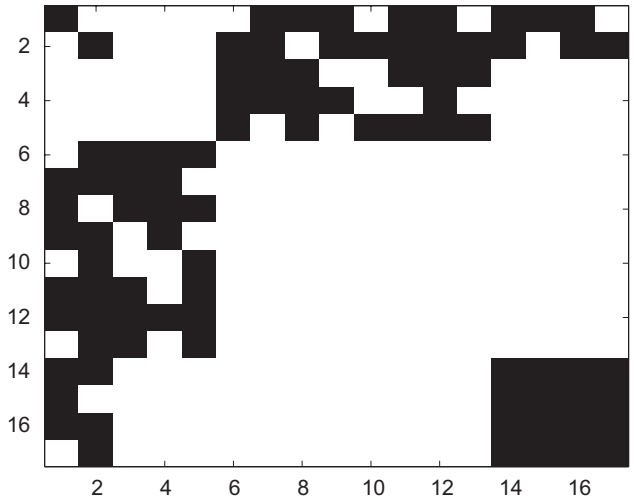

Fig. 3. Adjacency matrix of an incomplete graph from the simulation example.

D. 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 Fig. 3 (only 80% of the links are known). We show step by step the idea of node kernels and latent features.

1) Node Kernel: We visualize the nodes using the node kernel defined in (3). We use kernel principal component analysis and plot the nodes using the first three principal components in Fig. 4. The nodes in the figure are the nodes of the graphs. The cyan edges are the observed links. The red dashed edges are the missing links according to the model in the simulation example. First, we can observe that the cyan edges follow certain patterns of direction and endpoints. Another point that can be observed is that the dashed red edges follow the same patterns (endpoints lie in clusters and edges connect the same cluster pairs). This is exactly what we want in learning—testing data having the same distribution as training data. What is left to be done is to keep these patterns in some spatial representations of the edges.

2) 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 Fig. 5. The ellipses 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 lie somewhere in between the nodes with only one of the two features.

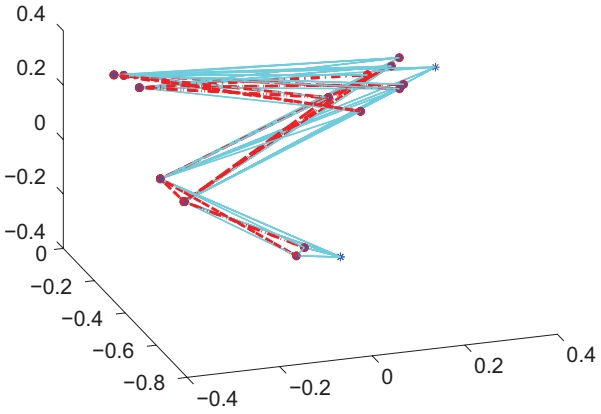

Fig. 4. Visualizing the graph with the node kernel. The cyan edges are the observed links of the graph, while the dashed red edges are the missing links according to the model in the simulation. We can observe that the red links follow the same patterns as the cyan ones.

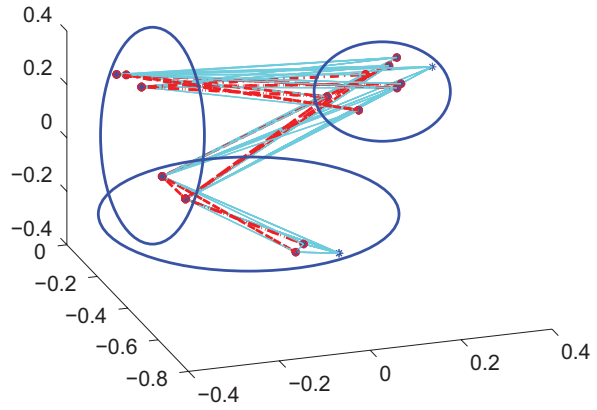

Fig. 5. Positive class of links. The nodes are grouped by their common latent features. This shows that nodes sharing more common features tend to group together.

3) Negative Links: As shown in Fig. 4, the observed edges and missing edges have similar patterns. However, since we use SVM for classification, we also wish the negative class (nonlink) not to follow the same patterns. Therefore, we show all the edges that belong to the nonlink class in Fig. 6. We can observe that they do not follow the patterns of the positive class, having the same endpoints but connecting cluster pairs different from the link classes.

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 other methods such as those using Chinese restaurant processes or Indian buffet processes (IBP) [6], [9], [12]. These methods group these nodes together explicitly, whereas our method uses a soft version of putting them close in a space. We believe that this is the key to robustness, allowing the inference step to reach globally optimal solutions and be scalable to large networks.

IV. Node Kernels on Sparse Graphs

Our objective is to model networks with latent feature models. The problem is that, in reality, intrinsic features are not possible to obtain, and inferring them explicitly is often intractable. Therefore, we use network structures (topology) to model the networks themselves. The key observation is that these networks are sparse, as they are generated by a small number of features, producing certain patterns on network structures. Hence, in this section, we show that, for sparse networks, the information we extract from network structures is close to the information of the features that are not available. In other words, the models we construct for the networks are close to the true generative models of the networks themselves.

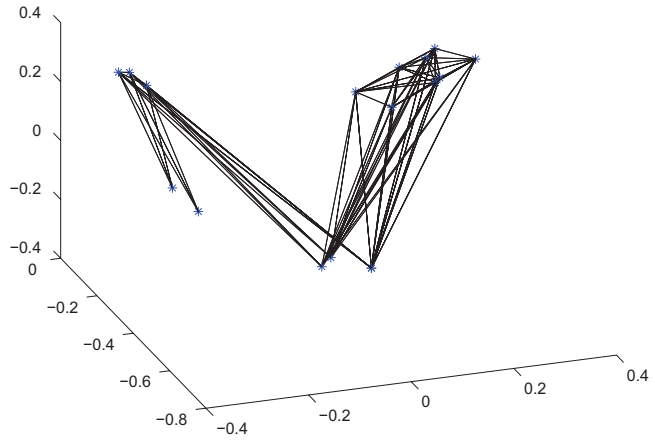

Fig. 6. Set of nonlink class. Compared to the link class, the nonlink class has different positions and orientations.

Toward this end, we define kernels that encode the similarity between nodes if the features are fully observed, named ideal kernels. The ideal kernels, which are what ideally the node kernels should be but this is not possible to obtain, are the benchmarks. As our method uses kernels with SVMs, we formally show how close the node kernels built on network structures could be to the ideal kernels. Given that the ideal kernels contain information of latent features, if the node kernels that we obtain are the ideal kernels, then the success of our link kernels rests solely on the ability of the link kernels to represent the links given the node kernels. Given the way the link kernels are defined on the basis of the node kernels, we expect that the link kernels that we obtain are also close to those built on ideal kernels in some sense. The main result is contained in the four propositions. We also show here an idea of how our defined kernels reflect the latent features of the graphs.

A. Ideal Kernels

Consider a graph generated by a latent feature model of 1. If the model (F, W) were observable, we would be able to design an ideal node kernel encoding the similarity of nodes in this graph. The semantic similarity of two nodes under this model is the number of latent features they share, being weighted by the importance of the features. Therefore, we define the ideal kernels as follows.

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

$$K^*(D) = FDF^T. \tag{10}$$

A kernel value $K^*(D)_{ij} = F_i D (F_j)^T$ is the weighted sum of the number of common features between the ith and jth nodes. However, since latent feature models are obtained with difficulty, the ideal kernels are not available. Any kernel defined from data with latent feature models should be close to some ideal kernels.

Below, we will show that the node kernel we define is close to ideal kernels for sparse networks, which are the target of our method.

B. Relationship Between Node Kernels and Ideal Kernels on Sparse Graphs

When the graphs are sparse (as 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 in the sense defined therein. Let S_i denote 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 the feature causes both of the nodes i and j to have links to another node, then $(AA^T)_{ij} > 0$, and 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 $(K_n)_{ij} > 0$. ■

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) in which K_n is not sparser than any ideal kernel.

Proposition 2 (Monotonicity): Suppose that the pairs of nodes have the same number of neighbors in the sense that $\mathbb{D}_{ii}\mathbb{D}_{jj} = \mathbb{D}_{kk}\mathbb{D}_{ll}$ [$\mathbb{D}$ is defined as in (4)]. 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 numbers of neighbors for the two pairs of nodes. The number of neighbors of a pair of nodes is defined to be the product of the numbers of neighbors of the two 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 in which 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 common neighborhood of the nodes k and l. Therefore, $(AA^T)_{ij} \geq (AA^T)_{kl}$.

$$(K_n)_{ij} = \frac{(AA^T)_{ij}}{\sqrt{\mathbb{D}_{ii}\mathbb{D}_{jj}}} \geq \frac{(AA^T)_{kl}}{\sqrt{\mathbb{D}_{ii}\mathbb{D}_{jj}}} = (K_n)_{kl} \tag{11}$$

from the definition of the node kernel in (3). ■

Lemma 1 (One Feature Interaction): For any latent feature model (F, W), there exists another latent feature model $(F^\times, W^\times)$ that gives: 1) the same adjacency matrix; 2) the same set of ideal kernels; and 3) the feature interaction matrix with at most one nonzero entry in any row and column.

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

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

$$A = FWF^T = \sum_{ij} W_{ij} F_{\cdot i} (F_{\cdot j})^T \tag{12}$$

where $F_{\cdot i}$ is the ith column of F.

Let $I = \{(i, j)\}$ that $W_{ij} \neq 0$. Then

$$A = \sum_{(i,j)\in I} W_{ij} F_{\cdot i} (F_{\cdot j})^T. \tag{13}$$

We then construct the $F^\times$ and $W^\times$ by sequentially appending the feature columns and feature interaction matrix values in (13) as follows.

1) Mark all the indices in I as available.
2) Initialize $F^\times$ and $W^\times$ to empty matrices.
3) Go to the next available index in I, pick the pair (i, j), and then continue as follows.
   a) If $i = j$, meaning that w_{ii} indicates a self-interacting feature, then append the feature vector $F_{\cdot i}$ at the end of the already constructed $F^\times$. Append a new row and column of $W^\times$ with the only nonzero element W_{ij} on the diagonal of $W^\times$. Mark the index of W_{ij} in I unavailable. Repeat the process from step 3.
   b) If W is symmetric (A is symmetric), then $W_{ij} = w_{ji}$. Append the feature vectors $F_{\cdot i}$ and $F_{\cdot j}$ to the end of $F^\times$. Suppose that the size of $W^\times$ is k, then append two new rows and columns of $W^\times$, setting the values of $W^\times_{k+1,k}$ and $W^\times_{k,k+1}$ to be W_{ij}. Mark the indices of W_{ij} and W_{ji} in I unavailable. Repeat the process from step 3.
   c) If W is not symmetric, then append the features as the symmetric case to $F^\times$. For $W^\times$, only one nonzero element is added to $W^\times_{k,k+1}$. Mark the index of W_{ij} in I unavailable. Repeat the process from step 3.

After all indices in I are marked unavailable, we have

$$F^\times W^\times (F^\times)^T = \sum_{(i,j)\in I} W_{ij} F_{\cdot i} (F_{\cdot j})^T = FWF^T = A. \tag{14}$$

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

Hence, we have constructed a new model with a feature which either interacts or is interacted with at most one feature. This is to say that there is a mathematically equivalent model in which 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 the row vectors of WF^T are pairwise orthogonal.

Proof: When the row vectors of WF^T are pairwise orthogonal, then $WF^T \times (WF^T)^T$ is diagonal. Denote $D = WF^T \times FW^T$. Then

$$K_n(A) = FWF^T \times FW^T F^T = FDF^T. \quad (15)$$

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

Corollary 1: If W is the inverse of a mixing matrix of an independent component analysis model [35] for F^T and WF^T. $\vec{1} = \vec{0}$, then $K_n \in \mathbb{K}$.

$\vec{1}$ denotes the vector of all 1 and $\vec{0}$ denotes the vector of all 0. The corollary comes from the fact that independent components are uncorrelated and, in this case, orthogonal.

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

The corollary comes from the fact that WF^T has only one nonzero entry in each column, and 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. Let $E = F^T F$. Then $E_{lk} = (F_{\cdot l})^T F_{\cdot k}$ is the number of nodes having both features l and k. When $l \neq k$, we expect E_{lk} to be small in comparison with E_{ll} and E_{kk} for sparse models. In other words, E is diagonally dominant. We show the relationship between the diagonal dominance of D and the sparsity of F as well as how close K_n is to ideal kernels.

Given Lemma 1, we assume that W_{il} and W_{jk} are the only nonzero entries in rows i and j. Recall that $K_n = FDF^T = FWEW^T F^T$. 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, the off-diagonal elements of E are 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 number of nodes having the two features l and k (should also be small for sparse models, $l \neq k$ implying $i \neq j$). $\| \cdot \|_p$ 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

$$\| D - \hat{D} \|_p = \left(\sum_{i\neq j} |D_{ij}|^p \right)^{\frac{1}{p}} = \left(\sum_{i\neq j} |W_{il}W_{jk}E_{lk}|^p \right)^{\frac{1}{p}} \leq \delta. \quad (16)$$

This shows that the sparser the model, the closer D is to ideal kernels. ■

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

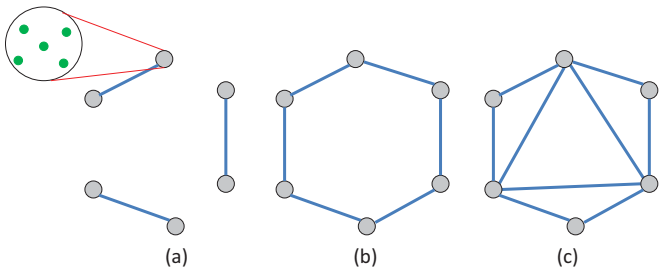

Fig. 7. Simulation example networks with latent features. A circle represents a group of five nodes with the same latent feature sets. An edge between circles is a set of links from all nodes in one circle to all in the other one. (a)–(c) Networks become denser.

TABLE I
LINK PREDICTION RESULTS ON SIMULATION NETWORKS WITH DIFFERENT SPARSITY LEVELS

Network	Mean AUC score
(a)	0.979 ± 0.020
(b)	0.903 ± 0.026
(c)	0.843 ± 0.044

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 applications section.

C. Simulation

We conduct a simulation to show that, when the network is sparse in the sense that a node has a small number of latent features and each feature only interacts with one another, then our kernels can capture well the similarity in the shared latent features. By this we mean that our defined kernels are close enough to ideal kernels, which are supposed to give perfect performance. The simulation networks are generated according to the topology in Fig. 7.

The experiment was set up as follows. We sampled 50% of the links for training and used the rest for testing. We used our method, and we report the average area under the curve (AUC) scores of 50 runs in Table I.

We can observe that, when the network was generated by the simple model in (a), our method gave perfect AUC scores. As the models became denser, our method gave poorer performance. This might be due to the fact that our proposed kernels become further from the ideal kernels. Our simulation suggests that our kernels are good for very sparse models of networks (that generate sparse networks).

V. APPLICATION: 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 main target is non-similarity networks such as PPI networks. An experiment in the following subsection verifies that PPI networks are indeed nonsimilarity networks. There are other networks studied in bioinformatics but they are too sparse and not large enough to study structures statistically.

TABLE II
NUMBER OF NODES AND LINKS FOR DIFFERENT SUBNETWORKS WITH VARYING MINIMUM DEGREE m

m	Yeast		Fruit-fly	
	Nodes	Links	Nodes	Links
1	4762	21 836	6644	21501
2	3775	20 849	4783	19640
3	2891	19 139	3397	16944
4	2260	17 315	2539	14452
5	1811	15 582	1923	12072
6	1507	14 112	1403	9568
7	1245	12 607	1098	7796
8	1046	11 255	713	5243
9	890	10 064		
10	756	8924		

We used the PPI networks of yeast and the fruit fly from the DIP database [36] because of their largest hand-curated networks available. For statistical study, we focused on only the largest connected component of the m-core of the network (the largest connected subnetwork with all the nodes having the degrees of m or more). In our experiments, we show the results for all the values of m from 1 to 10 (8 for the fruit fly). We show in Table II the sizes of the data. We will show later on that the other methods based on latent feature inference do not scale to these sizes.

Real PPI networks are sparse: less than 20% of the nodes in yeast and none in the fruit fly have degrees of 10 or more. This is the reason for sparsity analysis in Section IV.

For each subnetwork, we sampled the nonlink class data from arbitrary pairs of nodes that are not known to have any link between them. The number of nonlinks is the same as the number of links to make the data balanced. This process was repeated five times. We showed the average AUC score of 20 random train/test splits with the ratio 90 : 10. We used SVM (soft margin parameter $C = 0.001$, but the results were the same for a range of C from 0.0001 to 0.1) as the classifier for our link kernels. We first show the appropriateness of the assumption of latent features as opposed to the usual assumption of similarity. We then show the time required to build one model to clarify the difference in real computation time between our method and IBP [6] (with parameter $\alpha = 3$, which we found to be a good tradeoff to be able to train on our smallest datasets and induce a model with a sufficient number of latent features). We then show the performance of link kernels in link prediction. We also compare the performance of link kernels to that of sequence-based link prediction to show the advantage using network structures.

A. Latent Feature Versus Similarity

Fig. 8 shows the fitness of the latent feature model on yeast PPI networks as opposed to the similarity model. To compare the models, i.e., the assumptions of similarity and latent features, we use the simple nearest neighbor classifiers, resulting in the two following methods.

1) SimNN for similarity assumption: Using the direct method [7]. The more walks of length two between the nodes, the more likely they have a link between them. Using the normalized adjacency matrix, this method happens to be equivalent to using the node kernels and predicting links based on the similarity (kernel value) of the two nodes.
2) LK-NN for latent feature assumption: The latent feature kernel is used as the similarity measure of pairs of nodes. A pair of nodes takes the label of its closest pair of nodes with a known label (link or nonlink).

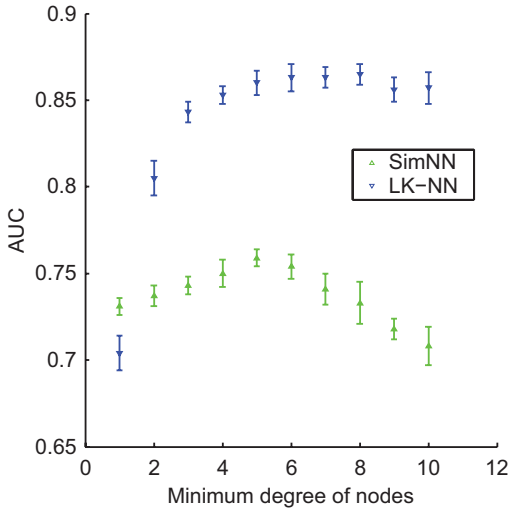

Fig. 8. Latent feature versus similarity assumption. AUC of the direct method for link prediction (vertical) at different minimum degree of the network (horizontal axis).

Fig. 8 shows significantly different AUC scores of the two methods on the yeast PPI networks. LK-NN was usually much higher (around 0.1 and more). It shows that the latent feature model was more suitable to the PPI networks than the conventional similarity one. The exception was at the subnetwork with the minimum degree of nodes of 1. Since we were using only network structures, the nodes with degree 1 may not contribute to the network structure in these models. Therefore, results for the subnetworks with nodes with larger degrees demonstrated the reasonability of the assumptions used. For this reason, the latent feature assumption was more reasonable than the similarity assumption here, and the methods for similarity networks were not recommended to be used.

B. Execution Time

We show execution time required to build one model using our link kernels in Fig. 9. To compare with IBP, we also show the execution time in the process of building one model before it burns in in Fig. 10. Since IBP requires a long computation time, 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.

We can see that the link kernels took less than 90 s for the smallest subnetwork, which increased to 480 s for the largest subnetwork of 4762 nodes. On the other hand, IBP took many hours for the smallest subnetwork of 756 nodes and did not

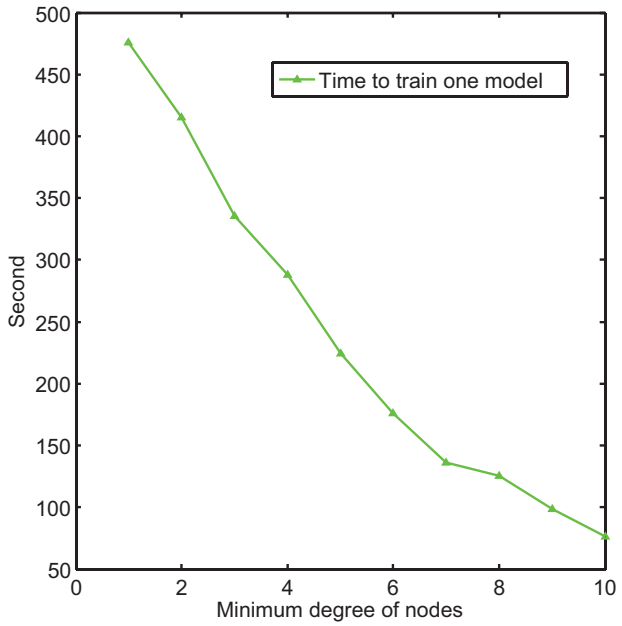

Fig. 9. Time required to build one model using link kernels (in seconds) at subnetworks with different minimum degrees. Note that, for the smallest subnetwork, the execution time is less than 90 s.

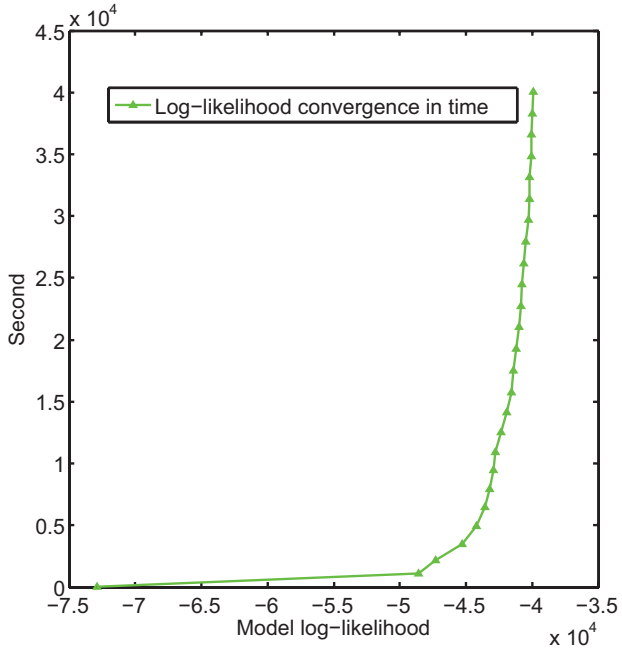

Fig. 10. Time to train an IBP model of the smallest subnetwork with $m = 10$. The horizontal axis is the log-likelihood obtained from the model during the training process as a function of time (vertical axis).

burn in on larger ones within one day. The reason is that IBP uses sampling methods that are usually very expensive computationally. We conclude that our method can save by many orders of magnitude the time to train one model, making training on networks with thousands of nodes possible. This is a key advantage of our kernel method.

C. Link Prediction Results

We compare the prediction ability of our method using link kernel to the baseline of using IBP as in [6, Fig. 11].

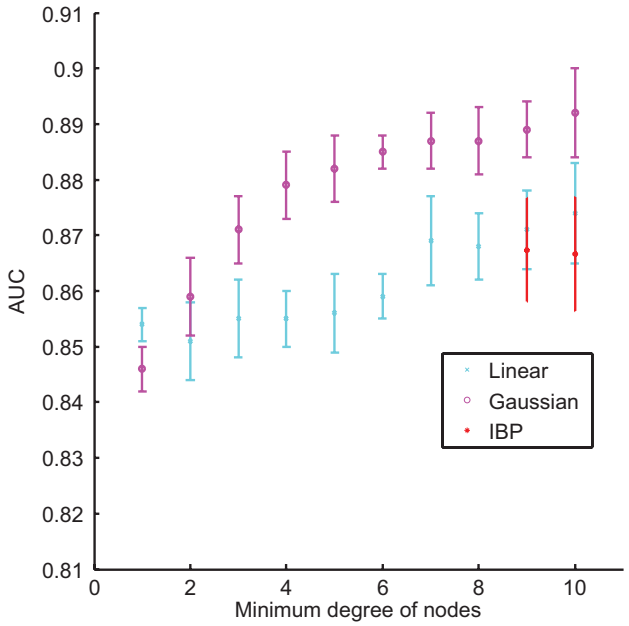

Fig. 11. Link prediction results on the yeast PPI network. AUC of link prediction with different methods (vertical) at different minimum node degrees of the network (horizontal axis). Note that IBP does not scale with larger datasets, and hence the results of IBP are unavailable.

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 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 ($K(x, z) = \exp(-\gamma |x - z|^2)$ with $\gamma = 2$). The incomplete results of IBP were due to the fact that the experiments took too much time (more than one day) to train one model.

We can read from yeast's results in Fig. 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 provide a significantly higher AUC scores than IBP. One surprising result was that even using network topology only, we achieved 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 shows that there are patterns for the topology of the PPI networks. This result also shows that our method is effective in encoding these patterns.

Similarly, we can see from the results on the fruit fly PPI network in Fig. 12 that our method based on kernels outperforms IBP. The AUC scores on fruit fly PPI networks are much lower than on yeast because of the fact that fruit fly networks are sparser, involving more proteins and there are no proteins with degrees of 10 or more.

D. Comparison to Sequence-Based Prediction

As opposed to using network structures, traditional methods use node information such as protein sequences. Therefore, we

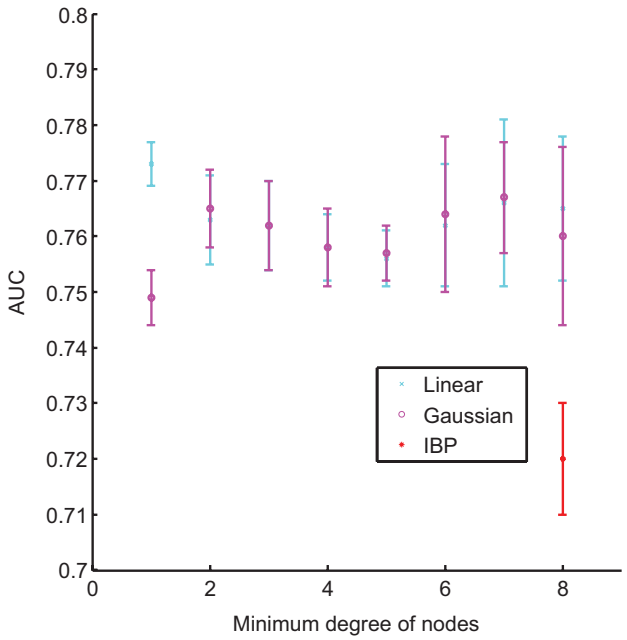

Fig. 12. Link prediction results on the fruit fly PPI network, showing the AUC of link prediction with different methods (vertical) at different minimum degrees of the network (horizontal axis). IBP results are available only for the 8-core.

also compared our method to spectrum kernels on sequences to predict links. The results are not shown in Figs. 11 and 12 because their ranges are different. The highest AUC score was 0.71 ± 0.008 for yeast and 0.65 ± 0.016 for fruit fly. We observe that network-based methods gave higher AUC scores, being statistically significant (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 because only small parts may determine the interaction ability of proteins to others.

VI. Application: Other Nonsimilarity Networks

We extend our application domain to another type of non-similarity network: the gene regulatory network of E. coli [37]. This network has the largest number of links among all gene regulatory networks and is well studied. Regulatory networks are not well known to have latent features that generate the network structures. They do not fit into any other categories of similarity or bipartite networks. The regulatory network of E. Coli is not too sparse, and it is a scale-free network.

We carried out experiments in the same manner as in the previous section for the PPI network. We extracted the largest connected components with the minimum degrees m. We show the sizes of our data in Table III. We then used SVM with default parameters as in the previous section to train the classifiers for out method.

A. Latent Feature Versus Similarity

Given that gene regulatory networks are not known to be any kind of networks that are usually used for link prediction, we wish to see how appropriate and general our model is compared to the most common similarity network type. We use the simple direct method as before to highlight the merit of similarity measures. We show the average AUC scores in Fig. 13. We can observe that the similarity network assumption gives very low AUC scores, almost close to random. Our method, on the other hand, gives very high AUC scores. We conclude that, while the latent feature model trained with our method is flexible for this network, it is not the same for similarity networks. In fact, gene regulatory networks are closer to bipartite networks in which there is a set of common regulators that regulates most other genes, including themselves. Therefore, they have some bipartite structures in them, among other links, that make them not bipartite networks. This explains why our network model is flexible enough to capture this type of structure whereas similarity networks totally fail to do so.

TABLE III
Number of Nodes and Links for Different Subnetworks of E. Coli Gene Regulatory Network With Varying Minimum Node Degrees m

m	Nodes	Links
1	1437	3901
2	1064	3528
3	678	2763
4	446	2091
5	239	1302

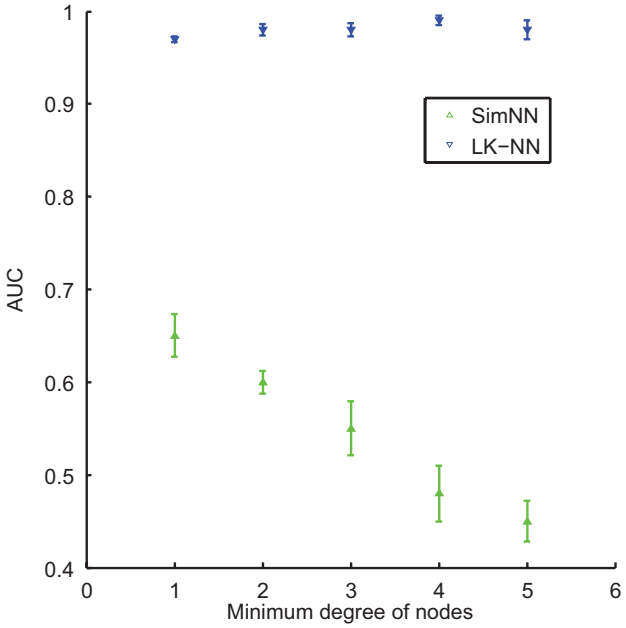

Fig. 13. Latent feature versus similarity assumption on gene regulatory networks. AUC of the direct method for link prediction (vertical) is shown at different minimum degrees of the network (horizontal axis).

B. Execution Time

We also compared the execution times of our method to the method that explicitly infers the latent features of IBP.

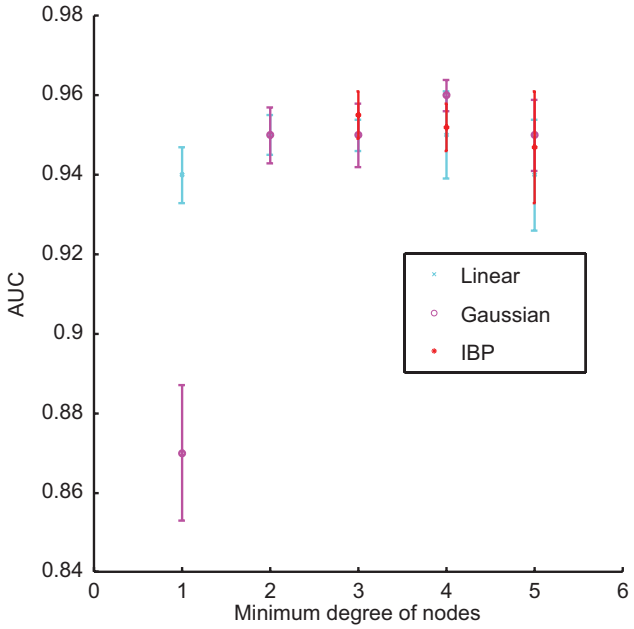

Fig. 14. Link prediction results on E. Coli gene regulatory network. AUC of link prediction with different methods (vertical) is shown at different minimum degrees of the network (horizontal axis). IBP results are missing due to their large time consumption.

Our method took less than 2 s on average, whereas IBP cost 5097 ± 1021 s on the smallest subnetwork with the minimum degree of nodes of 5. Our method is, therefore, significantly faster. The same argument can be applied, and IBP uses sampling methods that are computationally very expensive. Our method, however, only uses convex quadratic programming, and therefore is very efficient.

C. Link Prediction Results

We also compare the link prediction ability of our method with the link kernel to the baseline of using IBP in Fig. 14. We follow the same settings as in the previous section. We also compare the linear and Gaussian kernels to IBP. We reach similar conclusions as in the networks with known latent features in the previous section. The performance of our method becomes higher in subnetworks with a higher minimum degree (m). It is noteworthy that our method takes only a very small fraction of time compared to IBP.

It is surprising that methods using the latent features implicitly or explicitly gave very high AUC scores on these subnetworks (0.9 or more). This means that latent feature models are general enough to capture the structure of this network. That is, the gene regulatory network is likely to have latent features as well.

VII. Conclusion

We have 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 cast the problem as a supervised learning one, inheriting the optimality, efficiency, and nonlinearity of the kernel framework. We showed the suitability of the kernels on sparse networks. We applied our methods to two types of networks: the networks of PPIs where latent features are expected and the networks for gene regulations, which are known not to belong to similarity or bipartite networks. The results showed that our kernel-based method gave a higher performance than the direct latent feature inference method of 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 give higher predictive performance than node information. We conclude that for sparse networks with latent feature models, our method is able to utilize the relevant information in network structures to give significantly faster execution time, more scalable solutions, and higher performances.

Acknowledgment

The authors would like to thank the editors, the anonymous reviewers, and the laboratory members for their helpful comments.

References

[1] L. Getoor and B. Taskar, *Introduction to Statistical Relational Learning* (Adaptive Computation and Machine Learning). Cambridge, MA: MIT Press, Nov. 2007.

[2] A. Ben-Hur and W. S. Noble, "Kernel methods for predicting protein-protein interactions," *Bioinformatics*, vol. 21, no. 1, pp. 38–46, 2005.

[3] T. Kato, K. Tsuda, and K. Asai, "Selective integration of multiple biological data for supervised network inference," *Bioinformatics*, vol. 21, no. 10, pp. 2488–2495, 2005.

[4] K. Bleakley, G. Biau, and J.-P. Vert, "Supervised reconstruction of biological networks with local models," *Bioinformatics*, vol. 23, no. 13, pp. 57–65, 2007.

[5] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 58, pp. 1019–1031, May 2007.

[6] K. Miller, T. Griffiths, and M. Jordan, "Nonparametric latent feature models for link prediction," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Cambridge, MA: MIT Press, 2009, pp. 1276–1284.

[7] J.-P. Vert and Y. Yamanishi, "Supervised graph inference," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2005, pp. 1433–1440.

[8] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *J. Mach. Learn. Res.*, vol. 9, pp. 1981–2014, Jun. 2008.

[9] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda, "Learning systems of concepts with an infinite relational model," in *Proc. 21st Nat. Conf. Artif. Intell.*, vol. 1. 2006, pp. 381–388.

[10] Z. Xu, V. Tresp, K. Yu, and H.-P. Kriegel, "Infinite hidden relational models," in *Proc. 22nd Conf. Uncertain. Artif. Intell.*, 2006, pp. 1–8.

[11] T. Griffiths and Z. Ghahramani, "Infinite latent feature models and the Indian buffet process," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2005.

[12] D. J. Navarro and T. L. Griffiths, "A nonparametric Bayesian method for inferring features from similarity judgments," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2006, pp. 1033–1040.

[13] N. Srebro, J. D. M. Rennie, and T. S. Jaakola, "Maximum-margin matrix factorization," in *Advances in Neural Information Processing Systems*, vol. 17. Cambridge, MA: MIT Press, Jul. 2005, pp. 1329–1336.

[14] C. Ding, "Orthogonal nonnegative matrix tri-factorizations for clustering," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2006, pp. 126–135.

[15] R. I. Kondor and J. D. Lafferty, "Diffusion kernels on graphs and other discrete input spaces," in *Proc. Int. Conf. Mach. Learn.*, 2002, pp. 315–322.

[16] J. S. Kandola, J. Shawe-Taylor, and N. Cristianini, "Learning semantic similarity," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2002, pp. 657–664.
[17] A. J. Smola and R. I. Kondor, "Kernels and regularization on graphs," in *Proc. COLT*, 2003, pp. 144–158.
[18] T. Kato, H. Kashima, M. Sugiyama, and K. Asai, "Conic programming for multitask learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 1, pp. 957–968, Jul. 2010.
[19] M. Shimbo, T. Ito, D. Mochihashi, and Y. Matsumoto, "On the properties of von Neumann kernels for link analysis," *J. Mach. Learn.*, vol. 75, no. 1, pp. 37–67, 2009.
[20] J. Kunegis and A. Lommatzsch, "Learning spectral graph transformations for link prediction," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 561–568.
[21] Y. Yamanishi, "Supervised bipartite graph inference," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2008, pp. 1841–1848.
[22] B. Taskar, M. F. Wong, P. Abbeel, and D. Koller, "Link prediction in relational data," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2003.
[23] J. Basilico and T. Hofmann, "Unifying collaborative and content-based filtering," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 9.
[24] Z. Xu, V. Tresp, K. Yu, S. Yu, and H.-P. Kriegel, "Dirichlet enhanced relational learning," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 1004–1011.
[25] M. A. Hasan and M. J. Zaki, "A survey of link prediction in social networks," in *Social Network Data Analytics*, C. Aggarwal, Ed. New York: Springer-Verlag, 2011, ch. 9, pp. 243–275.
[26] C. H. Nguyen and H. Mamitsuka, "Kernels for link prediction with latent feature models," in *Proc. Eur. Conf. Mach. Learn. Knowl. Disc. Databases*, vol. 2. 2011, pp. 517–532.
[27] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory*, 1992, pp. 144–152.
[28] M. Deng, S. Mehta, F. Sun, and T. Chen, "Inferring domain-domain interactions from protein-protein interactions," in *Proc. 6th Annu. Int. Conf. Comput. Biol.*, 2002, pp. 117–126.
[29] M. E. J. Newman, "Clustering and preferential attachment in growing networks," *Phys. Rev. E*, vol. 64, no. 2, pp. 102–115, Jul. 2001.
[30] F. Fouss, A. Pirotte, J. M. Renders, and M. Saerens, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 355–369, May 2007.
[31] G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan, *Predicting Structured Data* (Neural Information Processing). Cambridge, MA: MIT Press, 2007.
[32] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 104.
[33] T. Hofmann, I. Tsochantaridis, and Y. Altun, "Learning over discrete output spaces via joint kernel functions," in *Proc. Adv. Neural Inf. Process. Syst., Workshop Kernel Methods*, 2002.
[34] H. Kashima, S. Oyama, Y. Yamanishi, and K. Tsuda, "On pairwise kernels: An efficient alternative and generalization analysis," in *Proc. 13th Pacific-Asia Conf. Adv. Knowl. Disc. Data Mining*, 2009, pp. 1030–1037.
[35] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York: Wiley, 2001.
[36] L. Salwínski, C. S. Miller, A. J. Smith, F. K. Pettit, J. U. Bowie, and D. Eisenberg, "The database of interacting proteins: 2004 update," *Nucl. Acids Res.*, vol. 32, no. 1, pp. 449–451, 2004.
[37] S. Gama-Castro, H. Salgado, M. Peralta-Gil, A. Santos-Zavaleta, L. Muntildeiz-Rascado, H. Solano-Lira, V. Jimenez-Jacinto, V. Weiss, J. S. Garciacutea-Sotelo, A. Loacutepez-Fuentes, L. Porroacuten-Sotelo, S. Alquicira-Hernaacutendez, A. Medina-Rivera, I. Martiacutenez-Flores, K. Alquicira-Hernaacutendez, R. Martiacutenez-Adame, C. Bonavides-Martiacutenez, J. Miranda-Riacuteos, A. M. Huerta, A. Mendoza-Vargas, L. Collado-Torres, B. Taboada, L. Vega-Alvarado, M. Olvera, L. Olvera, R. Grande, E. Morett, and J. Collado-Vides, "RegulonDB version 7.0: Transcriptional regulation of escherichia coli k-12 integrated within genetic sensory response units (gensor units)," *Nucl. Acids Res.*, vol. 39, no. 1, pp. 98–105, 2011.

Canh Hao Nguyen received the B.S. degree from the University of New South Wales, Sydney, Australia, in 2002, and the M.S. and Ph.D. degrees from the Japan Advanced Institute of Science and Technology, Ishikawa, Japan, in 2006 and 2009, respectively.

He is currently a Post-Doctoral Fellow. His current research interests include machine learning methods with applications in network and biological data.

Hiroshi Mamitsuka received the B.S. degree in biophysics and biochemistry, the M.E. degree in information engineering, and the Ph.D. degree in information sciences from the University of Tokyo, Tokyo, Japan, in 1988, 1991, and 1999, respectively.

He has done extensive work in machine learning, data mining and bioinformatics. His current research interests include mining from graphs and networks in biology and chemistry.