Link Propagation: A Fast Semi-supervised Learning Algorithm for Link Prediction

Hisashi Kashima* Tsuyoshi Kato† Yoshihiro Yamanishi‡ Masashi Sugiyama§
Koji Tsuda¶

Abstract

We propose Link Propagation as a new semi-supervised learning method for link prediction problems, where the task is to predict unknown parts of the network structure by using auxiliary information such as node similarities. Since the proposed method can fill in missing parts of tensors, it is applicable to multi-relational domains, allowing us to handle multiple types of links simultaneously. We also give a novel efficient algorithm for Link Propagation based on an accelerated conjugate gradient method.

Keywords

Link Prediction, Supervised network inference, Semi-supervised learning, Biological networks, Social networks.

1 Introduction

Many phenomena in the world can be represented by sets of entities, and sets of static and dynamic relationships among the entities. Such relationships include friendships among people, actions such as someone clicking an on-line advertisement, and physical interactions among proteins. Collections of such relationships form *networks*.

The problem of predicting the structure of networks is called the *link prediction problem*, which is one of the important tasks of link mining [9] in the data mining community. A typical setting of the link prediction problem is to predict unknown parts of the structure of a network (or, the future structure of the network) from the known parts of the network. Link prediction has various applications in many fields such as social network analysis, marketing, and bioinformatics. For example, it can be used to predict relationships among participants such as friendships in social networks, or to predict users' future behaviors such as clicking advertisements for marketing. In the field of bioinformatics, predicting protein-protein interactions and regulatory relationships can provide guidance for the design of experiments for discovering new biological facts.

The link prediction problem can be seen as the problem of completing an adjacency matrix which represents the structure of a network. One of the standard approaches to the link prediction problem is to regard it as a binary classification problem of the elements of the adjacency matrix. In this paper, we add another dimension to the adjacency matrix, which indicates the extension of single-type link prediction to multiple-type link prediction. The extension results in introducing a third-order tensor which represents an adjacancy matrix with link types. Consequently, we consider a completion problem of the tensor as a generalization of the standard link prediction problem (Section 2). This is regarded as a binary classification problem of the elements of the tensor. The tensor representation of the network structure allows us to handle not only the existence of links but also the types of links, including temporal links. Furthermore, the tensor representation makes it possible to simultaneously predict multiple networks that have correlations with each other.

The link prediction problem can fall into two categories in accordance with the information used for prediction: (i) topological-information-based link prediction and (ii) node-information-based link prediction. The former type of link prediction uses only adjacency matrices. Based on observed parts of the adjacency matrices, the missing parts are predicted by using, for example, matrix factorization techniques [23, 28]. On the other hand, the latter type of link prediction exploits node information such as feature vectors of nodes or similarity values among nodes. One of the state-of-the-art approaches for this purpose is the pair-wise support vector machine (pair-wise SVM), which combines node-wise kernel matrices to construct a pair-wise kernel matrix [2, 3, 26]. In this paper, we discuss the latter type, i.e. link prediction based on node information.

In this paper, we propose using one of the well-known approaches of semi-supervised learning called label propagation [40, 41] for link prediction (Section 3.1). Label propagation was originally intended for use in node classification, but we apply the idea to pairs of nodes with multiple link types (i.e. $(\mathrm{node}, \mathrm{node}, \mathrm{type})$-triplets) and predict the relationships among the nodes. Since we need a triplet-wise similarity matrix to apply the label propagation idea to triplets, we propose to use the Kronecker product and the Kronecker sum of the element-wise similarity matrices (Section 3.2). To solve the resultant system of linear equations, we apply the conjugate gradient method (Section 4.1). Since naive

*IBM Research, Tokyo Research Laboratory
†Ochanomizu University, Center for Informational Biology
‡Mines ParisTech, Centre for Computational Biology
§Tokyo Institute of Technology, Department of Computer Science
¶Max Planck Institute for Biological Cybernetics

application of the conjugate gradient method causes serious scalability problems, we use an acceleration technique called "vec-trick" [22, 36] and its generalized versions for tensors, which significantly reduces the computation time and space requirements (Section 4.2). Moreover, we show interesting special cases that can be implemented very easily by using the functions of MATLAB® (Section 5).

Finally, we discuss the relationships between our approach and related approaches (Section 6), and demonstrate the performance of the proposed method by using several real-world network data sets (Section 7).

In summary, this work makes three main contributions:
(i) We develop a new semi-supervised link prediction method by applying the label propagation method [40, 41] to link prediction. The new method is the first method for tensor completion using auxiliary information. It allows us to handle not only strength of links among pairs of nodes, but also various types of links.
(ii) We propose using the Kronecker sum similarity as a novel similarity measure among node pairs, which is shown to outperform the existing Kronecker product similarity in many cases.
(iii) We propose an efficient learning algorithm based on the conjugate gradient method. It mitigates the scalability problem caused by naive application of the label propagation.

2 Link prediction problem

The link prediction problem is usually described as a task to predict how likely a link exists between an arbitrary *pair* of nodes. In this paper, we consider a more general problem of predicting multiple *types* of links among the pairs of nodes[1].

Let us denote two sets of nodes by $X := \{x_1, x_2, \ldots, x_M\}$ and $Y := \{y_1, y_2, \ldots, y_N\}$, and the types of link by $Z := \{z_1, z_2, \ldots, z_T\}$. Some or all of X and Y may be identical in accordance with applications. Note that $M := |X|$, $N := |Y|$ and $T := |Z|$. Taking an on-line store as an example, X, Y, and Z are sets of users, items, and possible actions by a user to an item, respectively. The actions include "click", "buy", and "evaluation". So, a type-z_k link between two nodes x_i and y_j indicates that an user x_i takes an action z_k to an item y_j (Figure 1(left)). Let us consider another example. If we want to predict the relationships among the members of a community, Both X and Y are the members, and Z is a set of relationship types among the members, for example, $Z := \{\text{friendship}, \text{working relationship}\}$. Note that we assume $X = Y$ in this case.

Since a type-z_k link for a node pair (x_i, y_j) can be considered as a (single-type) link for a triplet (x_i, y_j, z_k), multiple-type link prediction for node pairs is equivalent to single-type link prediction for node triplets. Therefore, we denote a set of single-type relationships among triplets by an $M \times N \times T$ third-order tensor,

$$[\mathcal{F}]_{i,j,k} := f_{ijk}.$$

The variable f_{ijk} indicates how likely a link exists for the triplet $(x_i, y_j, z_k) \in X \times Y \times Z$, which we refer to as *link strength* (Figure 1(right)). A large value of link strength indicates high confidence of the existence of a link, and a small value indicates high confidence of the absence of a link.

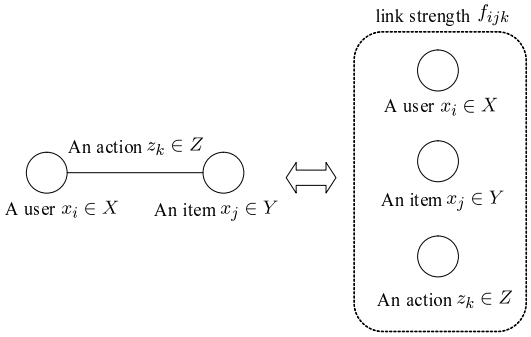


Figure 1: An example of a link. The fact that a user $x_i \in X$ gives an action (e.g. "evaluation") to an item $y_j \in Y$ is represented as a (single-type) link for a triplet (x_i, y_j, z_k). The link strength f_{ijk} indicates confidence of the existence of the link.

Now, we define another $M \times N \times T$ third-order tensor $\mathcal{F}^*$ which represents the observed parts of the network. $\mathcal{F}^*$ plays the role of the target values given in a training data set in supervised learning. Let E be the set of indices for triplets whose link strength is known. That is, E is the set of indices of the labeled instances. Each element of $\mathcal{F}^*$ is defined as

$$[\mathcal{F}^*]_{i,j,k} := \begin{cases} f^*_{ijk} & \text{if } (i,j,k) \in E, \\ 0 & \text{otherwise}, \end{cases}$$

where f^*_{ijk} is set to some positive value if a link exists for (x_i, y_j, z_k), and to some negative value if no link exists for (x_i, y_j, z_k). For $(i,j,k) \notin E$, $[\mathcal{F}^*]_{i,j,k}$ is filled with zero for convenience, since we do not use them. Particularly, we recommend to set f^*_{ijk} as

$$f^*_{ijk} := \begin{cases} |E|/|E^+| & \text{if a link exists for } (x_i, y_j, z_k), \\ -|E|/|E^-| & \text{if no link exists for } (x_i, y_j, z_k), \end{cases}$$

where $|E^+|$ and $|E^-|$ are the numbers of triplets with links and those without links, respectively. Note that

[1] The models and techniques used in this paper are easily extended to handling links among more than triplets, but we will limit ourselves on triplets since we want to keep the description simple and triplets can accommodate many important cases.

$(|E^+| + |E^-| = |E| \leq MNT)$. This way of setting the target values corresponds to the Fisher discriminant if we use the squared loss function [4].

Since we consider node-information-based link prediction, we are also given similarity matrices W_X, W_Y, and W_Z among the elements of X, Y, and Z, respectively[2]. Those matrices are non-negative and symmetric. In the previous example of $(\text{user}, \text{item}, \text{action})$-link prediction, W_X represents similarities among the users, where its (i, ℓ)-th element $[\mathsf{W}_X]_{i,\ell}$ indicates the similarity between the i-th user x_i and the ℓ-th user x_ℓ. Similarly, W_Y and W_Z are for the items and the link types, respectively.

In summary, the link prediction problem discussed in this paper is defined as follows.

INPUT:
· Three symmetric and nonnegative matrices W_X, W_Y, and W_Z for three entity sets X, Y, and Z.
· A third-order tensor $\mathcal{F}^*$ representing the known parts of the network.

OUTPUT: A third-order tensor $\mathcal{F}$ representing link strength for all triplets.

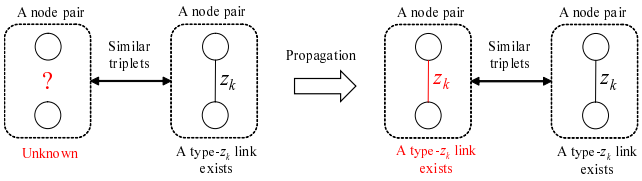
3 Link Propagation: A new semi-supervised link prediction method

In this section, we introduce our new approach to the link prediction problem. We refer to our semi-supervised link prediction method as "*Link Propagation*", which has the objective function (3.2) and either of the triplet-wise similarity matrices (3.4) and (3.6).

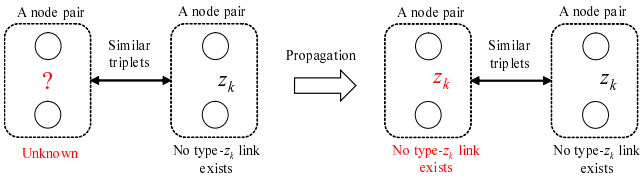
3.1 Formulation. Since the link prediction problem described in the previous section is a semi-supervised learning problem (more precisely, a transductive learning problem since we have the test dataset in the training phase), we use *label propagation* [40, 41], which is one of the state-of-the-art semi-supervised learning methods. The label propagation method was originally used for predicting the labels of unlabeled nodes by using the *label propagation principle*, that is, "Two nodes that are similar to each other are likely to have the same label". The idea of label propagation can be generalized to link prediction, since the link prediction problem can be regarded as a task of predicting labels for $(\text{node}, \text{node}, \text{type})$-triplets. Applying the label propagation method to triplets we can predict link strength as the labels for the triplets.

Modifying the label propagation principle, we can state the triplet version of the inference principle as "Two similar $(\text{node}, \text{node}, \text{type})$-triplets are likely to have the same link strength". In accordance with this "*link propagation principle*", we define the objective function to minimize as

[2]Even if they are not given, there is a possibility of constructing them from the observed links, for example, by similarities among the fibers of $\mathcal{F}^*$. However, we do not discuss this further in this paper.



(a) The link propagation principle for existence of a link



(b) The link propagation principle for absence of a link

Figure 2: The idea of the link propagation principle for (a) existence of a link and (b) absence of a link. The figures depict that if two triplets are similar to each other, their existence/absence of links is likely to be identical.

(3.1)

$$J(\{f_{ijk}\}) := \frac{\sigma}{2} \sum_{i,j,k,\ell,m,n} w_{ijk,\ell mn}(f_{ijk} - f_{\ell mn})^2 + \frac{1}{2} \sum_{(i,j,k)\in E} (f_{ijk} - f^*_{ijk})^2 + \frac{\mu}{2} \sum_{(i,j,k)\notin E} f^2_{ijk},$$

where $w_{ijk,\ell mn}$ is the symmetric triplet-wise similarity between two triplets (x_i, y_j, z_k) and (x_ℓ, y_m, z_n) (which will be defined later). The first term of Eq. (3.1) indicates that the two link strength values f_{ijk} and $f_{\ell mn}$ for the two triplets should be close to each other if the similarity $w_{ijk,\ell mn}$ between the two triplets is large. The second term is the loss function that fits the predictions to their target values for the triplets in E. The last term is a regularization term to prevent the predictions from being too far from zero, and also for numerical stability. $\sigma > 0$ and $\mu > 0$ are regularization parameters which balance the three terms in Eq. (3.1).

Now, we rewrite Eq. (3.1) using tensors. For that, we define an $M \times N \times T$ tensor $\mathcal{G}$ as

$$[\mathcal{G}]_{i,j,k} = \begin{cases} 1 & \text{if } (i,j,k) \in E, \\ \sqrt{\mu} & \text{otherwise,} \end{cases}$$

and let L be an $MNT \times MNT$ matrix called *Laplacian*

matrix defined as

$$\mathsf{L} := \mathsf{D} - \mathsf{W},$$

where D is a diagonal matrix whose diagonal elements are

$$[\mathsf{D}]_{i,i} := \sum_j [\mathsf{W}]_{i,j},$$

and W is a triplet-wise similarity matrix whose elements are defined as

$$[\mathsf{W}]_{MN(k-1)+M(j-1)+i,MN(n-1)+M(m-1)+\ell} := w_{ijk,\ell mn}.$$

Using $\mathcal{G}$ and L, Eq. (3.1) is rewritten as

$$\begin{aligned} J(\mathcal{F}) &= \frac{\sigma}{2}\mathbf{vec}\left(\mathcal{F}\right)^\top \mathsf{L}\mathbf{vec}\left(\mathcal{F}\right) \\ &\quad + \frac{1}{2} \parallel \mathbf{vec}\left(\mathcal{F} * \mathcal{G}\right) - \mathbf{vec}\left(\mathcal{F}^*\right) \parallel_2^2, \end{aligned} \tag{3.2}$$

where $*$ is the Hadamard product (i.e. the element-wise product of two tensors), and $\mathbf{vec}\left(\mathcal{A}\right)$ is the vector constructed by stacking the mode-1 fibers (i.e. column) of the tensor $\mathcal{A}$, defined as $\mathbf{vec}\left(\mathcal{A}\right)_{(k-1)NT+(j-1)N+i} := [A]_{i,j,k}$. Note that, when $X = Y$ and there is no link direction, the frontal slices of $\mathcal{F}^*$ are symmetric, then the solution $\mathcal{F}^*$ becomes symmetric.

To obtain $\mathcal{F}$ that minimizes Eq. (3.2), we differentiate Eq. (3.2) with respect to $\mathbf{vec}\left(\mathcal{F}\right)$, which results in

$$\frac{\partial J(\mathcal{F})}{\partial \mathbf{vec}\left(\mathcal{F}\right)} = \sigma \mathsf{L}\mathbf{vec}\left(\mathcal{F}\right) + \mathbf{vec}\left(\mathcal{F} * \mathcal{G}\right) - \mathbf{vec}\left(\mathcal{F}^*\right).$$

Setting this to $\mathbf{0}$ for obtaining the stationary point, we obtain the following linear equation,

$$\left(\sigma \mathsf{L} + \mathsf{diag}\left(\mathbf{vec}\left(\mathcal{G}\right)\right)\right)\mathbf{vec}\left(\mathcal{F}\right) = \mathbf{vec}\left(\mathcal{F}^*\right), \tag{3.3}$$

where the operator diag produces a diagonal matrix whose diagonal elements are given by its argument vector.

3.2 Designing the triplet-wise similarity matrix. Since it is not realistic to give all of the $M^2N^2T^2$ elements of the triplet-wise similarity matrix W, we consider systematic construction of W using the element-wise similarity matrices W_X, W_Y, and W_Z. For addressing the scalability issues discussed in the next section, we restrict the class of W, and consider two ways for constructing W.

The first one is the *Kronecker product similarity*, which is based on the idea that two triplets are similar to each other if each of the three cross-triplet pairs of nodes are similar to each other (Fig. 3(a)). The Kronecker product similarity matrix is defined as

$$\mathsf{W} := \mathsf{W}_Z \otimes \mathsf{W}_Y \otimes \mathsf{W}_X, \tag{3.4}$$

where $\otimes$ indicates the Kronecker product. This is equivalently expressed in an element-wise manner as

$$w_{ijk,\ell mn} := [\mathsf{W}_X]_{i,\ell}\,[\mathsf{W}_Y]_{j,m}\,[\mathsf{W}_Z]_{k,n}.$$

Thus, the Kronecker product similarity between two triplets is designed as the product of the similarities in each set. This is an extension of the pair-wise similarity used in kernel methods [2, 3, 26] to triplets. The Kronecker product similarity corresponds to the inner product in the product space of the three feature spaces, if W_X, W_Y, and W_Z are kernel matrices defined as the inner products in the feature spaces of W_X, W_Y, and W_Z, respectively. Using the Kronecker product similarity, we can express the Laplacian matrix in Eq. (3.3) as

$$\mathsf{L} = \mathsf{D}_Z \otimes \mathsf{D}_Y \otimes \mathsf{D}_X - \mathsf{W}_Z \otimes \mathsf{W}_Y \otimes \mathsf{W}_X, \tag{3.5}$$

where D_X is a diagonal matrix whose diagonal elements are defined as $[\mathsf{D}_X]_{i,i} := \sum_j [\mathsf{W}_X]_{i,j}$; D_Y and D_Z are defined similarly.

Since the product space of the Kronecker product similarity sometimes becomes too complex and is of overly high dimensions, we also consider another similarity with a more restricted feature space (if it is a kernel function), which we call the *Kronecker sum similarity*. The Kronecker sum similarity is based on the idea that two triplets are similar to each other if two of the three cross-triplet pairs of nodes are identical, and the other cross-triplet pair is similar to each other (Fig. 3(b)). We define the Kronecker sum similarity as

$$\begin{aligned} \mathsf{W} &:= \mathsf{W}_Z \oplus \mathsf{W}_Y \oplus \mathsf{W}_X \\ &= (\mathsf{W}_Z \otimes \mathsf{I}_N \otimes \mathsf{I}_M) + (\mathsf{I}_T \otimes \mathsf{W}_Y \otimes \mathsf{I}_M) \\ &\quad + (\mathsf{I}_T \otimes \mathsf{I}_N \otimes \mathsf{W}_X), \end{aligned} \tag{3.6}$$

where $\oplus$ indicates the Kronecker sum defined by $\mathsf{W}_Z \oplus \mathsf{W}_Y := \mathsf{W}_Z \otimes \mathsf{I}_N + \mathsf{I}_T \otimes \mathsf{W}_Y$, and I_M is an identity matrix of size $M \times M$. This is equivalently expressed in an element-wise manner as

$$\begin{aligned} w_{ijk,\ell mn} &:= [\mathsf{W}_X]_{i,\ell}\,\delta(j=m)\,\delta(k=n) \\ &\quad + \delta(i=\ell)\,[\mathsf{W}_Y]_{j,m}\,\delta(k=n) \\ &\quad + \delta(i=\ell)\,\delta(j=m)\,[\mathsf{W}_Z]_{k,n}, \end{aligned}$$

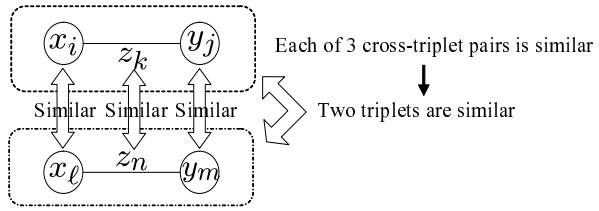
where δ is a function which returns 1 if the argument is true, and 0 otherwise. Using the Kronecker sum similarity, we can express the Laplacian matrix in Eq. (3.3) as

$$\mathsf{L} = \mathsf{L}_Z \oplus \mathsf{L}_Y \oplus \mathsf{L}_X, \tag{3.7}$$

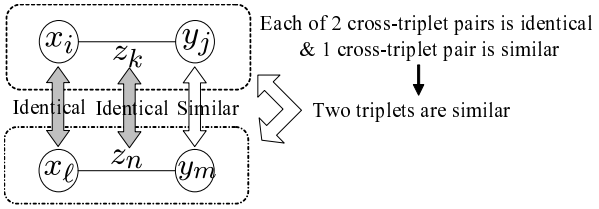
where L_X is the Laplacian matrix defined as $\mathsf{L}_X := \mathsf{D}_X - \mathsf{W}_X$. L_Y and L_Z are defined similarly.

As is clear from the definitions, the Kronecker product can give an arbitrary pair of triplets a similarity score greater than zero, while the Kronecker sum can give a positive score only to the pairs which share at least two elements of the triplets.

At first sight, since the Kronecker sum similarity has a fewer number of pairs with positive similarity values than



(a) Kronecker product similarity



(b) Kronecker sum similarity

Figure 3: Intuitive examples of (a) the Kronecker product similarity and (b) the Kronecker sum similarity.

the Kronecker product similarity, it seemingly can not fully exploit node similarity information. But as we will see in the experiments (Section 7), the Kronecker sum similarity is compatible with the Link Propagation method, since pairs with zero similarity values can utilize link information of each other through the other pairs with positive similarity values using the label propagation mechanism.

Another intuition behind the Kronecker sum similarity is that similar nodes tend to form triangle link structure, which is one of the generative processes of small world networks [10]. It is known that a variety of real-world networks including biological networks and social networks are small world networks.

The two definitions of the triplet-wise similarity can be seen as constructing a product graph over the triplets if we consider the element-wise similarity matrices as weighted graphs. The Kronecker product and the Kronecker sum correspond to the tensor product graph and the Cartesian product graph of the weighted graphs [14], respectively.

Finally, we mention the scalability problem occurred in using the Kronecker product/sum similarity matrix. As mentioned earlier, even if the element-wise similarity matrices are small, their Kronecker product becomes huge ($MNT \times MNT$), so it is not reasonable to store them explicitly in the memory. The matrix is rather sparse for the Kronecker sum similarity, but still needs much space. Since the kernel methods use the same similarity matrix we use as kernel matrices [2, 3, 26]. they also suffer from the severe scalability problem. In the next section, we overcome this problem by the conjugate gradient method using the "vec-tricks".

4 A fast algorithm for Link Propagation method

In this section, we propose a conjugate gradient method accelerated by using the technique called "vec-tricks". The resultant algorithm needs $O(M^2N^2T^2(M+N+T))$ time and $O(M^2+N^2+T^2+MNT)$ space.

4.1 Conjugate gradient method for Link Propagation.

The conjugate gradient method is a standard approach to solving a system of linear equations [11]. The algorithm of the conjugate gradient method for $\mathsf{A}\mathbf{f} = \mathbf{f}^*$ is shown in Algorithm 1. We modify it to solve our system of linear equations (3.3) for Link Propagation.

First, we replace A, $\mathbf{f}$, and $\mathbf{f}^*$ by using the correspondences, $\mathsf{A} = \sigma\mathsf{L} + \mathsf{diag}(\mathbf{vec}(\mathcal{G}))$, $\mathbf{f} = \mathbf{vec}(\mathcal{F})$, and $\mathbf{f}^* = \mathbf{vec}(\mathcal{F}^*)$. We also replace the other vectors $\mathbf{f}(t)$, $\mathbf{p}(t)$, $\mathbf{q}(t)$, and $\mathbf{r}(t)$ by tensors $\mathcal{F}(t)$, $\mathcal{P}(t)$, $\mathcal{Q}(t)$, and $\mathcal{R}(t)$, respectively. Then, we obtain the conjugate gradient algorithm for our system of linear equations (3.3) as detailed in Algorithm 2. Note that the algorithm is described using tensor notation in contrast to the standard conjugate gradient algorithm (Algorithm 1) being described in terms of vectors.

Most of the steps in Algorithm 2 are easily obtained by simple substitutions, but Line 2 and Line 4 need some derivation. Here, we derive only Line 2. Line 4 can be derived in a similar manner. First, we define the following two operators $\mathcal{L}^{\text{PROD}}$ and $\mathcal{L}^{\text{SUM}}$ for the Kronecker product and for the Kronecker sum, respectively, as

$$\mathcal{L}^{\text{PROD}}(\mathcal{B}) := (\mathsf{D}_Z \otimes \mathsf{D}_Y \otimes \mathsf{D}_X - \mathsf{W}_Z \otimes \mathsf{W}_Y \otimes \mathsf{W}_X)\,\mathbf{vec}(\mathcal{B}), \tag{4.8}$$

$$\mathcal{L}^{\text{SUM}}(\mathcal{B}) := (\mathsf{L}_Z \otimes \mathsf{L}_Y \otimes \mathsf{L}_X)\,\mathbf{vec}(\mathcal{B}), \tag{4.9}$$

where $\mathcal{B}$ is an $M \times N \times T$ tensor. Bearing the above correspondences in mind , $\mathbf{r}(0) := \mathbf{f}^* - \mathsf{A}\mathbf{f}(0)$ is rewritten as

$$\begin{aligned}\mathbf{vec}(\mathcal{R}(0)) &:= \mathbf{vec}(\mathcal{F}^*) - (\sigma\mathsf{L} + \mathsf{diag}(\mathbf{vec}(\mathcal{G})))\,\mathbf{vec}(\mathcal{F}(0)) \\ &= -\sigma\mathcal{L}^{\{\text{PROD}|\text{SUM}\}}\mathbf{vec}(\mathsf{F}(0)),\end{aligned}$$

where we used $\mathcal{F}^* = \mathcal{F}(0) = \mathsf{diag}(\mathbf{vec}(\mathcal{G}))\,\mathbf{vec}(\mathcal{F}(0))$, and Eqs. (3.5) and (3.7). In Algorithm 2, the operator $\mathcal{L}^{\{\text{PROD}|\text{SUM}\}}$ is replaced with $\mathcal{L}^{\text{PROD}}$ when we use the Kronecker product similarity, or with $\mathcal{L}^{\text{SUM}}$ when we use the Kronecker product similarity.

However, evaluation of Eqs. (4.8) and (4.9) is still a computational bottleneck of Algorithm 2, since the triplet-wise similarity matrix is huge.

4.2 The "vec-tricks".

We here show that computation of Eqs. (4.8) and (4.9) can be made significantly efficient

Algorithm 1 Conjugate Gradient $(\mathsf{A}, \mathbf{f}^*, \epsilon)$.

$\mathbf{f}(0) := \mathbf{f}^*$
$\mathbf{r}(0) := \mathbf{f}^* - \mathsf{A}\mathbf{f}(0)$, and $\mathbf{p}(0) := \mathbf{r}(0)$
for $t = 0, 1, 2, \ldots$ **do**
  $\mathbf{q}(t) := \mathsf{A}\mathbf{p}(t)$
  $\alpha(t) := \frac{\left\langle \mathbf{r}(t), \mathbf{p}(t) \right\rangle}{\left\langle \mathbf{p}(t), \mathbf{q}(t) \right\rangle}$
  $\mathbf{f}(t+1) := \mathbf{f}(t) + \alpha(t)\mathbf{p}(t)$
  $\mathbf{r}(t+1) := \mathbf{r}(t) - \alpha(t)\mathbf{q}(t)$
  $\beta(t) := \frac{\|\mathbf{r}(t+1)\|_2^2}{\|\mathbf{r}(t)\|_2^2}$
  if $\frac{\|\mathbf{r}(t+1)\|_2^2}{\|\mathbf{r}(0)\|_2^2} < \epsilon^2$, **return** $\mathbf{f}(t+1)$
  $\mathbf{p}(t+1) := \mathbf{r}(t+1) + \beta(t)\mathbf{p}(t)$
end for

Algorithm 2 Link Propagation $(\mathcal{F}^*, \mathcal{G}, \mathsf{W}_X, \mathsf{W}_Y, \mathsf{W}_Z, \sigma, \epsilon)$; $\mathcal{L}^{\{\text{PROD}|\text{SUM}\}}$ is replaced with $\mathcal{L}^{\text{PROD}}$ (Eq. (4.16)) for the Kronecker product similarity, or with $\mathcal{L}^{\text{SUM}}$ (Eq. (4.17)) for the Kronecker sum similarity.

$\mathcal{F}(0) := \mathcal{F}^*$
$\mathcal{R}(0) := -\sigma\mathcal{L}^{\{\text{PROD}|\text{SUM}\}}(\mathcal{F}(0))$, and $\mathcal{P}(0) := \mathcal{R}(0)$
for $t = 0, 1, 2, \ldots$ **do**
  $\mathcal{Q}(t) := \sigma\mathcal{L}^{\{\text{PROD}|\text{SUM}\}}(\mathcal{P}(t)) + \mathcal{G} * \mathcal{P}(t)$
  $\alpha(t) := \frac{\left\langle \mathcal{R}(t), \mathcal{P}(t) \right\rangle}{\left\langle \mathcal{P}(t), \mathcal{Q}(t) \right\rangle}$
  $\mathcal{F}(t+1) := \mathcal{F}(t) + \alpha(t)\mathcal{P}(t)$
  $\mathcal{R}(t+1) := \mathcal{R}(t) - \alpha(t)\mathcal{Q}(t)$
  $\beta(t) := \frac{\|\mathcal{R}(t+1)\|_2^2}{\|\mathcal{R}(t)\|_2^2}$
  if $\frac{\|\mathcal{R}(t+1)\|_2}{\|\mathcal{R}(0)\|_2} < \epsilon$, **return** $\mathcal{F}(t+1)$
  $\mathcal{P}(t+1) := \mathcal{R}(t+1) + \beta(t)\mathcal{P}(t)$
end for

by using the "vec-tricks" [22, 36], which accelerates the multiplication of matrix Kronecker products and a vectorized matrix/tensor.

Let A_X, A_Y, and A_Z be $M \times M$, $N \times N$, and $T \times T$ symmetric matrices, respectively. Let B be an $M \times N$ matrix, and $\mathcal{B}$ be an $M \times N \times T$ tensor. The basic idea of the "vec-tricks" lies in the following equation [22]:

$$(\mathsf{A}_Y \otimes \mathsf{A}_X)\mathbf{vec}(\mathsf{B}) = \mathbf{vec}(\mathsf{A}_X\mathsf{B}\mathsf{A}_Y). \tag{4.10}$$

The left-hand side of Eq. (4.10) needs $O(M^2N^2)$ computation time and space, while the right-hand side needs only $O(MN(M+N))$ time and $O(MN)$ space. Vishwanathan et al. [36] used this formula for accelerating the computation of the graph kernels.

Now, we generalize Eq. (4.10) to obtain its tensor version. Bearing in mind that matrices are second-order tensors, we rewrite Eq. (4.10) using mode-n multiplication of tensors [20] as

$$(\mathsf{A}_Y \otimes \mathsf{A}_X)\mathbf{vec}(\mathsf{B}) = \mathbf{vec}(\mathsf{B} \times_1 \mathsf{A}_X \times_2 \mathsf{A}_Y), \tag{4.11}$$

where mode-n multiplication is an operation that multiplies the mode-n fibers of a tensor by a matrix. For third-order tensors, these are defined as

$$[\mathcal{B} \times_1 \mathsf{A}_X]_{i,j,k} := \sum_{\ell=1}^{M} [\mathcal{B}]_{\ell,j,k}[\mathsf{A}_X]_{i,\ell},$$
$$[\mathcal{B} \times_2 \mathsf{A}_Y]_{i,j,k} := \sum_{\ell=1}^{N} [\mathcal{B}]_{i,\ell,k}[\mathsf{A}_Y]_{j,\ell},$$
$$[\mathcal{B} \times_3 \mathsf{A}_Z]_{i,j,k} := \sum_{\ell=1}^{T} [\mathcal{B}]_{i,j,\ell}[\mathsf{A}_Z]_{k,\ell},$$

and each of them returns an $M \times N \times T$ tensor. For more on general tensor calculation, see [20], for example. The form of Eq. (4.11) is naturally extendable to third-order (or higher-order) tensors, and we obtain the following equation.

$$(\mathsf{A}_Z \otimes \mathsf{A}_Y \otimes \mathsf{A}_X)\mathbf{vec}(\mathcal{B}) = \mathbf{vec}(\mathcal{B} \times_1 \mathsf{A}_X \times_2 \mathsf{A}_Y \times_3 \mathsf{A}_Z). \tag{4.12}$$

While the naive computation in the left-hand side needs $O(M^2N^2T^2)$ computation time and space, we can reduce it to $O(MNT(M+N+T))$ computation time and $O(MNT)$ space in the right-hand side.

Next, we consider the case with the Kronecker sum. Similar to the case of the Kronecker product, we have

$$(\mathsf{A}_Y \oplus \mathsf{A}_X)\mathbf{vec}(\mathsf{B}) = \mathbf{vec}(\mathsf{B}\mathsf{A}_Y + \mathsf{A}_X\mathsf{B}) \tag{4.13}$$
$$= \mathbf{vec}(\mathsf{B} \times_1 \mathsf{A}_X + \mathsf{B} \times_2 \mathsf{A}_Y). \tag{4.14}$$

Again, Eq. (4.14) is generalized to tensors as

$$(\mathsf{A}_Z \oplus \mathsf{A}_Y \oplus \mathsf{A}_X)\mathbf{vec}(\mathcal{B}) = \mathbf{vec}(\mathcal{B} \times_1 \mathsf{A}_X + \mathcal{B} \times_2 \mathsf{A}_Y + \mathcal{B} \times_3 \mathsf{A}_Z). \tag{4.15}$$

When $\mathsf{A}_X = \mathsf{A}_Y$, the number of multiplications can be reduced, since $\mathcal{B} \times_1 \mathsf{A}_X$ and $\mathcal{B} \times_2 \mathsf{A}_Y$ are essentially the same.

By using Eqs. (4.11) and (4.15), the computation of Eqs. (4.8) and (4.9) can be significantly simplified as

$$\mathcal{L}^{\text{PROD}}(\mathcal{B}) = \mathcal{B} \times_1 \mathsf{D}_X \times_2 \mathsf{D}_Y \times_3 \mathsf{D}_Z - \mathcal{B} \times_1 \mathsf{W}_X \times_2 \mathsf{W}_Y \times_3 \mathsf{W}_Z, \tag{4.16}$$
$$\mathcal{L}^{\text{SUM}}(\mathcal{B}) = \mathcal{B} \times_1 \mathsf{L}_X + \mathcal{B} \times_2 \mathsf{L}_Y + \mathcal{B} \times_3 \mathsf{L}_Z. \tag{4.17}$$

4.3 Discussion on efficiency. A great advantage of our Link Propagation algorithm is in its memory efficiency. It requires only $O(MNT + M^2 + N^2 + T^2)$ memory thanks to the "vec-tricks". In terms of the computational complexity, it requires $O(M^2N^2T^2(M+N+T))$ time theoretically, because each iteration of the conjugate gradient algorithm can be executed in $O(MNT(M+N+T))$ time, and $O(MNT)$ iterations are required to solve the linear

equations completely. In practice, the number of iterations required for convergence is much smaller than $O(MNT)$, and we observed that the predictive performance did not change after only several iterations in our experiments.

The improvement by using the "vec-tricks" is significant, because it is not clear so far how to apply the "vec-tricks" to kernel methods. If we imagine the triplet-wise extension of the pair-wise SVM using the same similarity matrix without the "vec-tricks", the space complexity is $O(M^2N^2T^2)$ and the time complexity is $O(M^3N^3T^3)$. The time complexity comes from the fact that the quadratic programming problem needs cubic time complexity with respect to the number of parameters (in the case of kernel mathods, it is the same as the number of training examples). As many fast optimization methods have been developed for SVM, its practical speed is not too slow in general. Nevertheless, as it is difficult to keep the whole kernel matrix in the memory, we cannot always use the fastest software packages in our problems.

5 Easily implementable special cases

In this section, we show special cases of pair-wise link prediction (i.e. when $T = 1$) with $\mu := 1$, where we can easily implement the proposed method by using the built-in functions of existing numerical computing environments such as MATLAB®.

Since setting $\mu := 1$ implies $[\mathsf{G}]_{i,j} = 1$ for all (i, j), it holds that $\mathsf{diag}(\mathbf{vec}(\mathsf{G})) = I_{MN}$. Therefore, Eq. (3.3) becomes

$$(\sigma\mathsf{L} + \mathsf{I}_{MN})\,\mathbf{vec}(\mathsf{F}) = \mathbf{vec}(\mathsf{F}^*)\,. \tag{5.18}$$

Note that $\mathcal{F} = \mathsf{F}$, $\mathcal{F}^* = \mathsf{F}^*$, and $\mathcal{G} = \mathsf{G}$ when $T = 1$.

When W is the Kronecker product similarity, Eq. (5.18) becomes

$$(\sigma\mathsf{D}_Y \otimes \mathsf{D}_X - \sigma\mathsf{W}_Y \otimes \mathsf{W}_X + \mathsf{I}_{MN})\,\mathbf{vec}(\mathsf{F}) = \mathbf{vec}(\mathsf{F}^*)\,.$$

By using Eq. (4.10), we obtain

$$\sigma\mathsf{D}_X\mathsf{F}\mathsf{D}_Y - \sigma\mathsf{W}_X\mathsf{F}\mathsf{W}_Y + \mathsf{F} = \mathsf{F}^*.$$

This equation is called the generalized Sylvester equation [22]. Vishwanathan et al. [36] proposed using S and T that satisfy $\mathsf{D}_Y \otimes \mathsf{D}_X + \mathsf{W}_Y \otimes \mathsf{W}_X \approx \mathsf{S} \otimes \mathsf{T}$ for approximating the equation as $\mathsf{TFS} + \mathsf{F} = \mathsf{F}^*$, and solving it by using the `dlyap` function in MATLAB®.

When W is the Kronecker sum similarity, Eq. (5.18) becomes

$$(\sigma\mathsf{L}_Y \oplus \mathsf{L}_X + \mathsf{I}_{MN})\,\mathbf{vec}(\mathsf{F}) = \mathbf{vec}(\mathsf{F}^*)\,. \tag{5.19}$$

We can derive the relation

$$\sigma\mathsf{L}_Y \oplus \mathsf{L}_X + \mathsf{I}_{MN} = \left(\sigma\mathsf{L}_Y + \frac{1}{2}\mathsf{I}_N\right) \oplus \left(\sigma\mathsf{L}_X + \frac{1}{2}\mathsf{I}_M\right),$$

which is substituted into Eq. (5.19) to obtain

$$\left(\left(\sigma\mathsf{L}_Y + \frac{1}{2}\mathsf{I}_N\right) \oplus \left(\sigma\mathsf{L}_X + \frac{1}{2}\mathsf{I}_M\right)\right)\mathbf{vec}(\mathsf{F}) = \mathbf{vec}(\mathsf{F}^*)\,.$$

By using Eq. (4.13), this can be rewritten as

$$\mathsf{F}\left(\sigma\mathsf{L}_Y + \frac{1}{2}\mathsf{I}_N\right) + \left(\sigma\mathsf{L}_X + \frac{1}{2}\mathsf{I}_M\right)\mathsf{F} = \mathsf{F}^*.$$

This equation is called the Sylvester equation [22], and can be solved by using the `lyap` function in MATLAB®. Unlike the case with the Kronecker product, no approximation is involved, and therefore we can obtain the exact solution.

6 Related work

The link prediction problem has been studied in the context of predicting biological networks such as protein-protein interaction networks and gene regulatory networks in the bioinformatics area, and also in the context of link mining [9] in the data mining community.

In bioinformatics, several node-information-based approaches were proposed, such as an EM-based approach [19] and metric-learning-based approaches [35, 38]. The pair-wise kernel which we will compare our method with in our experiments (Section 7) was proposed for predicting protein-protein interactions [3]. Interestingly, the same kernel was also proposed for entity resolution [26], and collaborative filtering [2], independently.

In the data mining community, the link prediction problem is studied as one of the fundamental tasks of link mining. There are several methods that utilize only structural information such as link metrics (e.g. [24]). Matrix factorization approaches [23, 28] are also grouped into topological-information-based methods.

There are also supervised learning methods using node information as well as topological information, for example, [13, 25]. There have also been several works (e.g. [27, 31]) that apply the framework of statistical relational learning to link prediction. A similar model is called the exponential random graph model in social network analysis [1]. Recently, sophisticated generative models of networks from Bayesian perspective have been proposed [5, 39].

Recently, there have been proposed several approaches to extending the existing network analysis methods to modeling the temporal dynamics of network structure. For example, Fu et al. [7] extended the exponential random graph model [1, 31] to temporal modeling. Some attempts (e.g. [29, 30]) use tensor analysis techniques [20] for temporal relation data as generalization of matrix analysis for pair-wise relations. Note that they do not exploit node information such as the node similarity matrices used in this paper.

The basic idea of label propagation was proposed by Zhou et al. [40] and Zhu et al. [41]. The scalability problems

are often discussed [8, 42], but the technique we used in this paper is totally different from theirs. To the best of our knowledge, we are the first to use auxiliary information in semi-supervised link prediction.

The matrix "vec-trick" (Eq. (4.10)) was used by Vishwanathan et al. [36] for accelerating the computation of the graph kernels.

7 Experiments

In this section, we show some experimental results for single-type link prediction (matrix completion) and multiple-type link prediction (third-order tensor completion) based on node information. Section 7.1 describes the results of single-type link prediction problems. We demonstrate that our semi-supervised link prediction performs better than the pair-wise kernel method, where both approaches are based on combined node information. Section 7.2 describes the results of multiple-type link prediction problems, where our task is simultaneous prediction of multiple networks related to each other. We demonstrate that predicting multiple networks simultaneously achieves better predictive performance than predicting each network separately.

Throughout all of the experiments, we set $\sigma = 0.001$ and $\mu = 1$ for Link Propagation. Note that for pair-wise link prediction, we take $T = |Z| = 1$, and $\mathcal{F}$ and $\mathcal{F}^*$ are the second-order tensors (i.e. matrices) F and F^*, respectively.

7.1 Pair-wise link prediction ($T = 1$). We compared our method with the pair-wise kernel method [2, 3, 26], which is one of the state-of-the-art link prediction methods using node information. In the pair-wise kernel method, link strength between a node pair (x_i, y_j) is modeled by

$$f(i,j) := \sum_{(\ell,m)} \alpha_{\ell,m} \kappa^{\text{PAIR}}((i,j),(\ell,m)).$$

The kernel $\kappa^{\text{PAIR}}((i,j),(\ell,m))$ represents similarity between two node pairs (x_i, y_j) and (x_ℓ, y_m), and the αs are the model parameters. In its original definition [2, 3, 26], the pair-wise kernel is defined by using the node-wise kernel κ as

(7.20)
$$\kappa^{\text{PAIR}}((i,j),(\ell,m)) := \kappa(i,\ell)\kappa(j,m) + \kappa(i,m)\kappa(j,\ell),$$

which corresponds to the Kronecker product similarity. Note that the above kernel is symmetrized.

Alternatively, we can use another pair-wise kernel corresponding to the Kronecker sum similarity as follows.

(7.21)
$$\kappa^{\text{PAIR}}((i,j),(\ell,m)) := \delta(i=\ell)\kappa(j,m) + \kappa(i,\ell)\delta(j=m) + \delta(i=m)\kappa(j,\ell) + \kappa(i,m)\delta(j=\ell).$$

Since the size of the pair-wise kernel matrices were too huge to construct explicitly in the memory, it was not reasonable to apply standard SVM implementations such as SVM^{light} [16]. Therefore, we used an on-line learning algorithm which processes one training example at each training step, so it is computationally and spatially efficient. In our experiments, we employed the passive-aggressive algorithm [6], which is an efficient on-line large-margin learning algorithm. We used the 1-norm version (PA-I) of the algorithm with $C = 1$. All of the kernels were normalized as $\kappa^{\text{PAIR}}((i,j),(\ell,m))/\sqrt{\kappa^{\text{PAIR}}((i,j),(i,j))\kappa^{\text{PAIR}}((\ell,m),(\ell,m))}$. All of the training data was processed three times in the training phase for better convergence and prediction.

We used three data sets for pair-wise link prediction. The first data set [38] contains the metabolic pathways of the yeast *S. Cerevisiae* in the KEGG/PATHWAY database [17]. Proteins are represented as nodes, and a link indicates that the two proteins are enzymes that catalyze successive reactions. The number of nodes in the network is 618, and the number of links is 2,782. In this data set, three kernel matrices based on gene expressions, localization sites, and phylogenetic profiles are given. We used them as the kernel matrices or the similarity matrices[3].

The second data set is a protein-protein interaction network data set constructed by von Mering et al. [37]. We followed Tsuda and Noble [32], and used the medium confidence network, containing $2,617$ nodes and $11,855$ links. In this data set, each protein is given a 76-dimensional binary vector, each of whose dimensions indicates whether or not the protein is related to a particular function. We used the inner product values between the vectors as the kernel matrix or the similarity matrix[4].

The third data set is a social network representing the co-authorships in the NIPS conferences, containing $2,865$ nodes and $4,733$ links. Authors correspond to nodes, and a link between two nodes means that there is at least one co-authored paper by the corresponding authors. In this data set, each author is given a feature vector, each of whose dimensions corresponds to occurrences of a particular word in the author's papers. We used the inner product of the vectors as the kernel matrix or the similarity matrix[5].

We randomly selected 10% of all the pairs ($|E|/(MNT) \approx 0.10$) as training data, and evaluated AUC on the remaining pairs; this procedure was repeated 10 times.

Figure 4 shows the averaged AUCs and their standard deviations for the metabolic network data. "Pair-wise Kernel (prod)" and "Pair-wise Kernel (sum)" denote the pair-wise kernel method using the passive-aggressive algorithm with the Kronecker product kernel (7.20) and the Kronecker sum

[3] Available at http://web.kuicr.kyoto-u.ac.jp/supp/yoshi/ismb05/. Although a kernel matrix based on chemical information is also given in this data set, we did not use it since it includes negative entries.

[4] Available at http://noble.gs.washington.edu/proj/maxent/.

[5] Available at http://ai.stanford.edu/~gal/data.html.

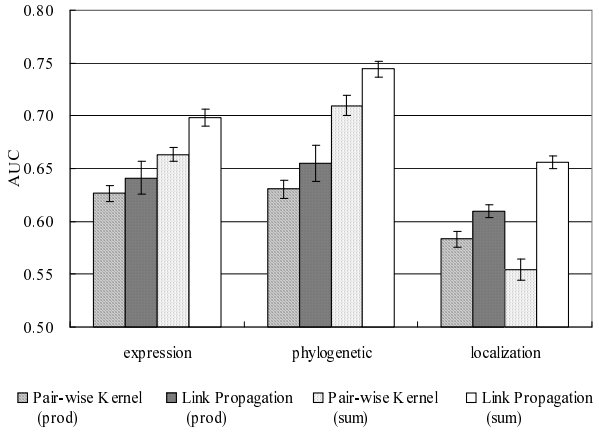


Figure 4: Summary of results for the KEGG metabolic network.

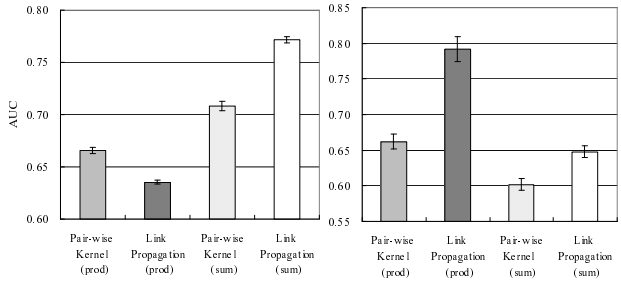


Figure 5: Summary of results for the protein-protein interaction network (left) and the social network (right).

kernel (7.21), respectively. "Link Propagation (prod)" and "Link Propagation (sum)" denote the proposed method with the Kronecker product similarity and the Kronecker sum similarity, respectively. Three results are shown for each of the information sources, gene expression (expression), phylogenetic profile (phylogenetic), and localization sites (localization). Figure 5 shows the results for the protein-protein interaction network data (left) and the social network data (right), respectively. In most of the cases, Link Propagation outperforms the pair-wise kernel method. Interestingly, despite its restricted feature space, the Kronecker sum performs better than the Kronecker product in many cases.

Next, we compare the computation time by each method. Figure 6 shows the average computation time in log scale spent on each data set in the training and test phases. Note that the passive-aggressive learner with the pair-wise kernels was trained with only one scan of the training data (which degrades the predictive performance though). All of

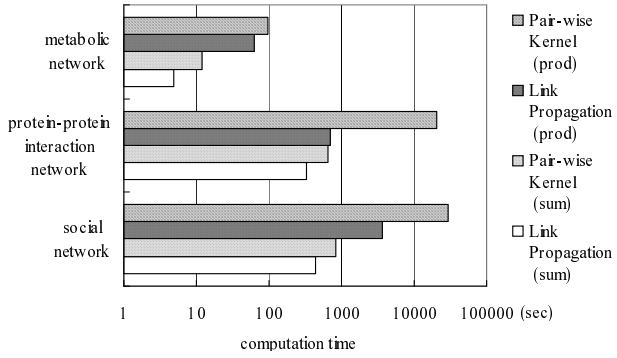


Figure 6: Comparison of computation time by each method.

the algorithms were implemented in R for Microsoft® Windows XP® on an IBM® IntelliStation® ZPro 6221 with an Intel® Xeon® 3.06-GHz CPU and 1.5-GB RAM.

The results show the efficiency of Link Propagation. We can see that Link Propagation is much faster than the pair-wise kernel method, and the improvement is significant when we use the Kronecker product similarity. Also, the Kronecker sum is consistently faster than the Kronecker product. This is because the number of pairs with positive Kronecker sum similarity is smaller than that for the Kronecker product similarity in the case of the pair-wise kernel method, and because the number of iterations needed for convergence by the Kronecker sum similarity is smaller than that the Kronecker product similarity in the case of Link Propagation.

7.2 Triplet-wise link prediction ($T > 1$). Using the proposed method for triplet-wise link prediction, we can predict two networks simultaneously. Alternatively, we can predict the two networks separately by using the proposed method for pair-wise link prediction. We compared the two approaches in order to investigate whether or not the simultaneous network prediction improves the predictive performance.

We used two biological network data sets for triplet-wise link prediction. Each data set contains two related networks. Therefore, tensors with $T = 2$ arise, when we consider two networks simultaneously. The first data set is two protein-protein interaction networks from different labs [15, 34]. In this data set, we collected two sets of protein-protein interactions detected from the yeast-two-hybrid system in two different labs, one of which forms a network (Ito) with $1,422$ nodes and 744 links, and the other (Uetz) forms a network with the same nodes as those of the Ito network and 888 links. They share 123 links in common.

The second data set is a pair of a physical protein-protein interaction network and a genetic protein-protein network stored in the MIPS database [12]. In the physical network,

two proteins have a link if the interaction between the two proteins is experimentally confirmed. In the genetic network, two proteins have a link if the simultaneous mutations in the two corresponding genes cause a cell death. The physical network consists of $1,225$ nodes and $3,474$ links, while the genetic network consists of the same nodes as those of the physical network and $1,333$ links. The two networks share 198 links in common.

In both of the two data sets, the kernel matrices and the similarity matrices are constructed using gene expressions, phylogenetic profiles, and localization sites by following the same procedure as Yamanishi et al. [38]. Also, we set the similarity between the two networks to one.

We randomly selected 50% of all the triplets ($|E|/(MNT) \approx 0.50$) as training data, and evaluated AUC for the remaining pairs; this procedure was repeated 10 times. We used a higher proportion of the data as training data than those we used for pair-wise link prediction, since we need the two networks to overlap to some degree.

Figure 7 shows the averaged AUCs and their standard deviations for the two protein-protein interaction networks with the Kronecker product similarity and the Kronecker sum similarity. "Ito (each)" and "Ito (simultaneous)" indicate the results for the Ito network by network-by-network prediction and simultaneous prediction, respectively. Similarly, "Uetz (each)" and "Uetz (simultaneous)" are for the Uetz network. Three results are shown for each of the information sources. We find that predicting the two networks simultaneously improved the predictive performances in many cases.

Figure 8 shows the results for the genetic network and the physical network with the Kronecker product similarity and the Kronecker sum similarity. "genetic (each)" and "genetic (simultaneous)" indicate the results for the genetic network by network-by-network prediction and simultaneous prediction, respectively. Similarly, "physical (each)" and "physical (sum)" are for the physical network. Three results are shown for each of the information sources. Although the improvement is not so significant as the experiment with the two protein networks, simultaneous prediction improves the performance especially when using the Kronecker sum similarity. Again, in both of the data sets, the Kronecker sum similarity consistently outperforms the Kronecker product similarity.

Finally, we show an experimental result for three networks. We constructed three networks consisting of 223 common proteins in the previous three networks, the genetic network, the Ito network, and the Uetz network. Each of them has 70–140 links, and they share 5–30 links in common. Figure 9 shows the results with the Kronecker product similarity and the Kronecker sum similarity. Since the number of links is small, the variance of the AUC values tends to be high. Even so, we can still see the trend that the results improve as the number of networks used in simultaneous prediction increases.

8 Concluding remarks

We proposed a new semi-supervised link prediction method by applying the label propagation technique to link prediction. This allows us to handle not only strength of the links among pairs of nodes, but also the type of links. We used the Kronecker sum similarity as the similarity matrices as well as the Kronecker product similarity. Moreover, we proposed an efficient learning algorithm based on the conjugate gradient method. Use of the tensor "vec-tricks" mitigated the scalability problem caused by naive application of label propagation. The experimental results showed that the proposed approach is quite promising.

Finally, we conclude this paper by mentioning some future work. First, we will consider *compressed representation of the solution*. Even if the similarity matrices and $\mathcal{F}^*$ are sparse, the solution $\mathcal{F}$ is usually dense, so it is hard even to store $\mathcal{F}$ in the main memory for large-scale problems. One possible approach might be to use compact tensor representations [20] for storing $\mathcal{F}$. *Use of topological information* is also promising. It is possible to construct similarity matrices from visible parts of $\mathcal{F}^*$. It would be interesting to compare our method using those similarity matrices with the other methods using only topological information such as matrix factorization [23, 28] and tensor decomposition [20]. *Information integration* is crucial, since we often have multiple similarity matrices obtained from various data sources. We will consider incorporating methods that adjust the weight of each similarity matrix automatically [18, 21, 33]. Our future work might also include *out of sample prediction* using approximated inference without solving entire systems, and *prediction with only positive links*.

References

[1] C. J. Anderson, S. Wasserman, and B. Crouch. A p^* primer: logit models for social networks. *Social Networks*, 21:37–66, 1999.

[2] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

[3] A. Ben-Hur and W. S. Noble. Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21(Suppl. 1):i38–i46, 2005.

[4] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] W. Chu, V. Sindhwani, Z. Ghahramani, and S. Keerthi. Relational learning with Gaussian processes. In *Advances in Neural Information Processing Systems 19*, 2007.

[6] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

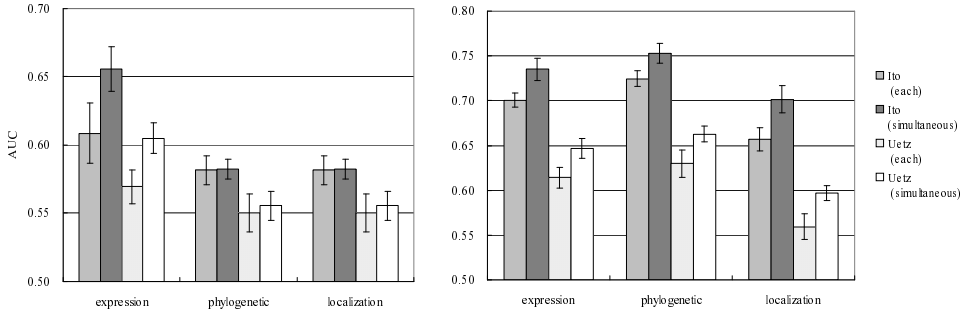


Figure 7: Summary of results for the two protein-protein interaction networks with the Kronecker product similarity (left) and the Kronecker sum similarity (right), respectively.

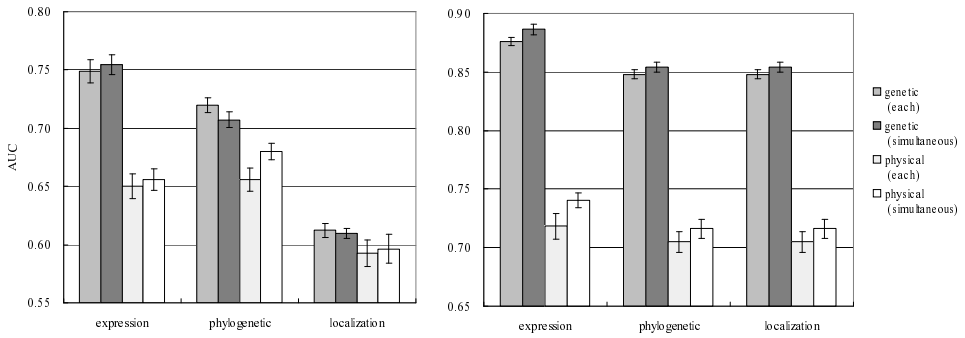


Figure 8: Summary of results for the genetic network and the physical network with the Kronecker product similarity (left) and the Kronecker sum similarity (right).

[7] W. Fu, F. Guo, S. Hanneke, and E. Xing. Recovering temporally rewiring networks: A model-based approach. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2007.

[8] J. Garcke and M. Griebel. Semi-supervised learning with sparse grids. In *Proceedings of the 22nd ICML Workshop on Learning with Partially Classified Training Data*, 2005.

[9] L. Getoor and C. P. Diehl. Link mining: a survey. *SIGKDD Explorations*, 7(2):3–12, 2005.

[10] D. Goldberg and F. Roth. Assessing experimentally derived interaction in a small world. *Proceedings of the National Academy of Sciences of the United States of America*, 100:4372–4376, 2003.

[11] G. H. Golub and C. F. V. Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996.

[12] U. Güldener, M. Münsterkötter, M. Oesterheld, P. Pagel, A. Ruepp, H.-W. Mewes, and V. Stümpflen. Mpact: the mips protein interaction resource on yeast. *Nucleic Acids Research*, 34:D436–D441, 2006.

[13] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *Workshop on Link Discovery: Issues, Approaches and Applications (LinkKDD)*, 2005.

[14] W. Imrich and S. Klavzar. *Product Graphs: Structure and Recognition*. Wiley, 2000.

[15] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interacome. *Proceedings of the National Academy of Sciences*, 98(8):4569–4574, 2001.

[16] T. Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers, 2003.

[17] M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori. The KEGG resources for deciphering the genome. *Nucleic Acids Research*, 32:D277–D280, 2004.

[18] T. Kato, H. Kashima, and M. Sugiyama. Integration of multiple networks for robust label propagation. In *Proceedings of the 8th SIAM International Conference on Data Mining (SDM)*, 2008.

[19] T. Kato, K. Tsuda, and K. Asai. Selective integration of multiple biological data for supervised network inference. *Bioinformatics*, 21(10):2488–2495, 2005.

[20] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. Technical Report SAND2007-6702, Sandia National Laboratories, 2007.

[21] G. R. G. Lanckriet, M. Deng, N. Cristianini, M. I. Jordan, and

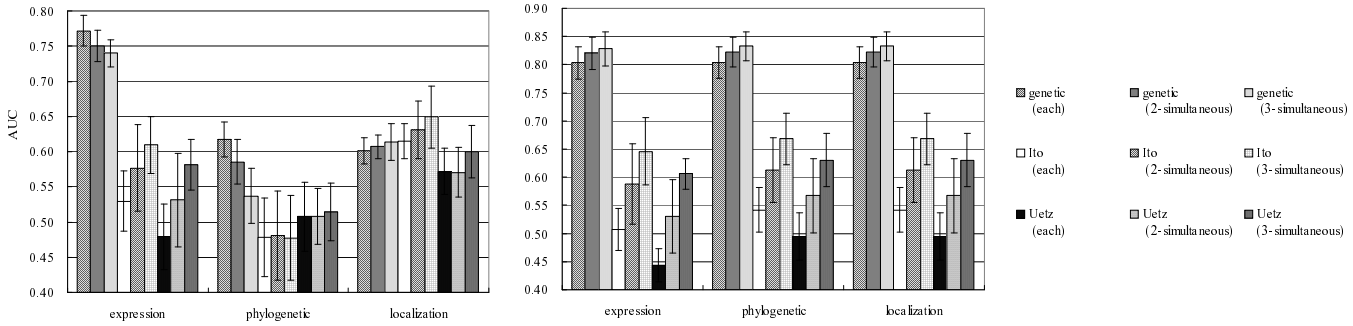


Figure 9: Summary of results for the genetic network, the Ito network, and the Uetz network with the Kronecker product similarity (left) and the Kronecker sum similarity (right).

W. S. Noble. Kernel-based data fusion and its application to protein function prediction in yeast. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, 2004.

[22] A. J. Laub. *Matrix Analysis for Scientists and Engineers*. Society for Industrial and Applied Mathematics, 2005.

[23] D. Lee and H. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*, 2001.

[24] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM)*, pages 556–559, 2004.

[25] J. O'Madadhain, J. Hutchins, and P. Smyth. Prediction and ranking algorithms for event-based network data. *SIGKDD Explorations*, 7(2):23–30, 2005.

[26] S. Oyama and C. D. Manning. Using feature conjunctions across examples for learning pairwise classifiers. In *Proceedings of the 15th European Conference on Machine Learning (ECML)*, pages 322–333, 2004.

[27] A. Popescul and L. H. Ungar. Statistical relational learning for link prediction. In *IJCAI Workshop on Learning Statistical Models from Relational Data*, 2003.

[28] N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, 2005.

[29] J. Sun, S. Papadimitriou, and P. Yu. Window-based tensor analysis on high-dimensional and multi-aspect streams. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM)*, pages 1076–1080, 2006.

[30] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *Proceedings of the 12th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 374–383, 2006.

[31] B. Taskar, M. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Advances in Neural Information Processing Systems 16*, 2004.

[32] K. Tsuda and W. S. Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 20(Suppl. 1):i326–i333, 2004.

[33] K. Tsuda, H. Shin, and B. Schölkopf. Fast protein classification with multiple networks. *Bioinformatics*, 21 Suppl. 2, 2005.

[34] P. Uetz, L. Giot, G. Cagney, T. Mansfield, R. Judson, J. Knight, D. Lockshon, V. Narayan, and et al. A comprehensive analysis of protein-protein interactions in saccharomyces cerevisiae. *Nature*, 403(6770):623–627, 2000.

[35] J.-P. Vert and Y. Yamanishi. Supervised graph inference. In *Advances in Neural Information Processing Systems 15*, 2005.

[36] S. V. N. Vishwanathan, K. Borgwardt, and N. Schraudolph. Fast computation of graph kernels. In *Advances in Neural Information Processing Systems 19*, 2007.

[37] C. von Mering, R. Krause, B. Snel, M. Cornell, S. Olivier, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417:399–403, 2002.

[38] Y. Yamanishi, J.-P. Vert, and M. Kanehisa. Supervised enzyme network inference from the integration of genomic data and chemical information. *Bioinformatics*, 21:i468–i477, 2005.

[39] K. Yu, W. Chu, S. Yu, V. Tresp, and Z. Xu. Stochastic relational models for discriminative link prediction. In *Advances in Neural Information Processing Systems 19*, 2007.

[40] D. Zhou, O. Bousquet, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, 2004.

[41] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 2003.

[42] X. Zhu and J. Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, 2005.