International Journal of Knowledge Discovery in Bioinformatics,
vol.1, no.1, pp.66-80, 2010.

A Transfer Learning Approach and Selective Integration of Multiple Types of Assays for Biological Network Inference

Tsuyoshi Kato[1,2,*], Kinya Okada[3], Hisashi Kashima[4], and Masashi Sugiyama[5]

[1] AIST Computational Biology Research Center,
2-42 Aomi, Koto-ku, Tokyo 135.0064, Japan.
[2] Center for Informational Biology, Ochanomizu University,
2-1-1 Ohtsuka, Bunkyo-ku, Tokyo 112.8610, Japan.
[3] KO Institute for Medical Bioinformatics,
Yokohama, Kanagawa 227.0033, Japan.
[4] IBM Research, Tokyo Research Laboratory,
1623-14 Shimo-tsuruma, Yamato, Kanagawa, 242-8502 Japan.
[5] Tokyo Institute of Technology, Department of Computer Science,
2-12-1, O-okayama, Meguro-ku, Tokyo 152-8552 Japan.

*Corresponding author

Abstract

Inferring the relationship among proteins is a central issue of computational biology and a diversity of biological assays are utilized to predict the relationship. However, as experiments are usually expensive to perform, automatic data selection is employed to reduce the data collection cost. Although data useful for link prediction are different in each local sub-network, existing methods cannot select different data for different processes.

This paper presents a new algorithm for inferring biological networks from multiple types of assays. The proposed algorithm is based on transfer learning and can exploit local information effectively. Each assay is automatically weighted through learning and the weights can be adaptively different in each local part.

Our algorithm was favorably examined on two kinds of biological networks: a metabolic network and a protein interaction network. A statistical test confirmed that the weight that our algorithm assigned to each assay was meaningful.

Keywords: Supervised network inference, Support vector machine, Local model, Transfer learning. Multiple kernel learning.

Background

A thorough understanding of cellular processes is the central goal of molecular biology. For this purpose, it is required to understand not only individual functions but also relationships among their components. Nowadays, a huge amount of data on molecular relationships can be generated through high throughput assays and analyzed as molecular networks. Although each of the assays contains useful information, molecular networks reconstructed from a single assay often have too much noise. To cope with this problem, several groups have tried to integrate data obtained from multiple types of assays for reliable network reconstruction (Pavlidis et al., 2002; Kato et al., 2005; Yamanishi et al., 2005).

How can we integrate data from multiple types of assays? A primitive method is to average data from the multiple types of assays and reconstruct a network from the averaged data (Yamanishi et al., 2005; Vert & Yamanishi, 2005). This method in a sense treats all types of assays equivalently, even if they may not be equivalent; that is, the assays may differ from each other in quality and resolution. To take non-equivalence among multiple types of assays into account in data integration, several groups have proposed methods to optimize the weight assigned to each type of assay (Kato et al., 2005; Lanckriet et al., 2004).

Although the existing methods allow each type of assay to be weighted, the weights are *global* in the entire network. Namely, every edge is predicted using weights common to the whole data. Since the mechanisms underlying cellular processes are complicated and heterogeneous, a type of assay may help shed light on some local cellular processes even if a low weight is assigned to the type of assay. Consequently, due to the heterogeneity of the relationship between assays and cellular mechanisms, the *global weighting* is too coarse, and finer modeling is desired.

Bleakley et al. (2007) have proposed a method that uses *local models* to cope with the heterogeneity issue. Their method builds a local model for each node (they call it a *target node*) and trains it with only its local (i.e., neighboring) information, resulting in scoring functions that are not corrupted by the irrelevant effects of distant parts of its molecular network. One shortcoming of their method is that the amount of local information is often limited, because most of the nodes have a few edges due to the fact that the node connectivity in molecular networks follows a power law distribution (Caldarelli, 2007). Insufficient information for training adversely affects the generalization ability of the method and restricts our ability to automatically acquire the genuine weights of the assays.

We present herein a novel algorithm for edge prediction. Our algorithm is based on Bleakley et al.'s approach, but our algorithm is extended to enjoy two remarkable features. First, in order to address the loss of generalization ability due to the scarcity of local information, we apply transfer learning to building local models. Transfer learning is an approach to machine learning that learns a task together with other related tasks simultaneously. This often leads to a better model for the target task, because it allows the learner to share appropriate information across the tasks. The use of this approach is

motivated by the observation that node A is likely to be linked with node B when they share common neighbors in molecular networks (Chua et al., 2006; Samanta & Liang, 2003; Okada et al., 2005). According to this observation, the task of building the local model of a target node is likely to be related to that of its neighboring nodes. Transfer learning builds the local model of a target node with the help of its neighboring nodes. Furthermore, transfer learning optimizes weights assigned to the types of assays when building local models, offering selected assays to each local part of a molecular network. In this study, we reconstruct molecular networks using this method and validate the efficiency of our method.

The rest of this section summarizes abbreviations which are used frequently in this paper. SVM is the abbreviation of the support vector machine (Schölkopf & Smola, 2002). RBF is the radial basis function (Schölkopf & Smola, 2002). BBVK, TPPK, and MLPK are the BBV kernel (Bleakley et al., 2007), the tensor product pairwise kernel (Ben-Hur & Noble, 2005), and the metric learning pairwise kernel (Vert et al., 2007), as detailed in the section of Methods. PIF is the protein interaction profile defined in the section of Discussion. GO is the gene ontology. We will show the full name again at the first occurrence except this paragraph.

Results

A number of large-scale biological networks have become available through high-throughput experiments, even though they are rough and incomplete. In order to refine them, more detailed surveys/studies need to be carried out. Even if some edges are already known for a given node, it is necessary to verify whether the node has other edges or not in the biological network. For such cases, we consider a network reconstruction task that involves building a scoring function to predict the existence of edges of a specified node (which is the one called target node in the previous section) with input nodes, in addition to its known edges. Let us cite an example. The adjacency matrix of the undirected graph in Figure 1(a) is shown in the bottom of Figure 1(b). Basically, a feature vector is given to each node as in the top of (b). We consider the case where a part of the adjacency matrix is missing, such as shown in Figure 1(c). We pose a binary classification problem which is to predict whether an input node is linked with the target node using the given feature vectors. This is same as what BBV algorithm attempts to do (See the section of Methods). For example, we assume that the target node is node 1. The first row of (b) implies that the training set includes only node 3, and the other nodes 2, 4, 5, 6, 7, and 8 are unknown samples in the classification task that we denote by $\mathcal{T}_1$. We also consider another classification task whose target node is node 3, and denote the task by $\mathcal{T}_3$. With the help of the fact that the node 3 is already known to be a neighbor of node 1, we regard $\mathcal{T}_3$ as a related task of $\mathcal{T}_1$, and our algorithm performs transfer learning of $\mathcal{T}_1$ with $\mathcal{T}_3$ to obtain a better solution of $\mathcal{T}_1$.

Our method is tested on two kinds of biological networks in *Saccharomyces cerevisiae*. One is the metabolic network in which enzymes involved in successive metabolic reactions are linked. This network, which contains 769 nodes (mainly enzymes) and

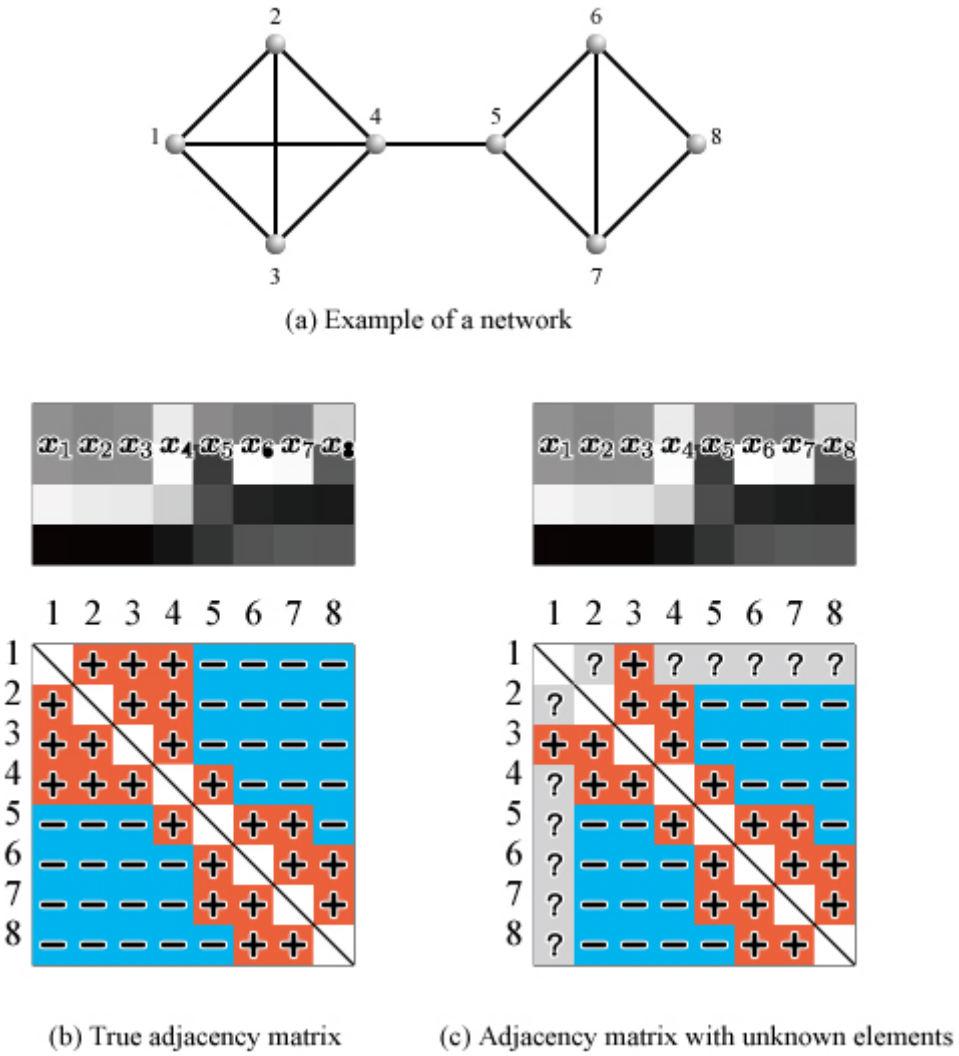


Figure 1: This paper deals with an undirected graph. Figure (a) shows an example of the undirected graph. In this example, each node has a 4-dimensional feature vector, $\mathbf{x}_1, \ldots, \mathbf{x}_8$, as shown in the top row of (b). The gray levels indicate the value of each element. The bottom of (b) is its adjacency matrix. Task $\mathcal{T}_1$ predicts the existence of edges in pairs (1, 2), (1, 4), (1, 5), (1, 6), (1, 7), and (1, 8). BBV algorithm (see the section Methods) cannot work well due to the extremely small training set $\mathcal{V}_{1,+} = \{3\}$, $\mathcal{V}_{1,-} = \varnothing$. We regard $\mathcal{T}_3$ as the related task to perform transfer learning, where node 3 is the known neighbor of node 1.

3,702 undirected edges (functional links), is identical to the one used by Yamanishi et al. (2005). The other is the protein interaction network of von Mering et al. (2002). Each edge in the network indicates the physical interaction of two proteins and is rated with a confidence level: high, middle or low. We use only high confidence edges as in

previous studies (Kato et al., 2005; Bleakley et al., 2007), because they are supported by multiple experiments. By removing proteins without any edges, we obtain a network of 984 proteins and 2,438 edges. Actually, the two networks are quite different: they share only 188 proteins and 41 edges.

In order to predict the edges, we use a diverse collection of protein data, and convert them into kernel matrices. Three of the kernel matrices are from sequence similarities. Sequence similarities are computed using three biological sequence alignment tools: BLAST (Altschul et al., 1990), ALN (Gotoh, 1982), and Pfam (Finn et al., 2008). The matrix formed with similarity scores is not guaranteed to be positive semi-definite, although the kernel matrices must be positive semi-definite for most of kernel methods (Schölkopf & Smola, 2002). To generate positive semi-definite kernel matrices, feature vectors are made from scores of each protein against all the proteins and the inner product among the feature vectors is taken. The scores for Pfam are the expected values derived from probabilistic models. The next dataset for network inference contains microarray gene expressions. Gene expressions are measured under various experimental conditions. A feature vector consists of the gene expressions. We use an RBF kernel function to generate a kernel matrix from the feature vectors. The resultant kernel matrices are available from the following website:

http://noble.gs.washington.edu/proj/sdp-svm/.

Following the website, we denote the five kernel matrices by ‘blast,’ ‘expr,’ ‘fft,’ ‘pfam_hmm,’ and ‘sw,’ respectively. Each kernel can be used for the protein interaction network and the metabolic network.

We use three pairwise kernels that are computed from node kernels: BBVK (Bleakley et al.,2007), TPPK (Ben-Hur & Noble, 2005), and MLPK (Vert et al., 2007). Then, we combine transfer learning with each of the three pairwise kernels. We determine the hyper-parameters of the algorithms by cross-validation over the training set, and take 300 nodes with the highest degree-of-nodes as the target gene to pose 300 tasks. Then, we randomly select 50% of proteins for the training set of each task.

Figure 2 compares the averaged AUC (Area under the curve) of ROC (Receiver Operating Characteristic) to predict metabolic gene network and protein interactions. The plotted AUC value is the average over 300 tasks. Transfer learning improves the prediction performance substantially for all the cases compared to conventional learning (BBV algorithm), and the differences are more than 0.1. AUCs among the three pairwise kernels do not differ very much.

To demonstrate the effect of data integration, we apply the learning algorithms to each data source and evaluate the ROC scores. The averages are also plotted in Figure 2. We observe improvements due to data integration especially in the results of conventional learning. For the metabolic network, transfer learning performance is improved by data integration. However, almost the same ROC scores are obtained for the protein interaction network. That may be because every data source we used is sufficiently informative to predict protein interactions well. For further investigation, we find the

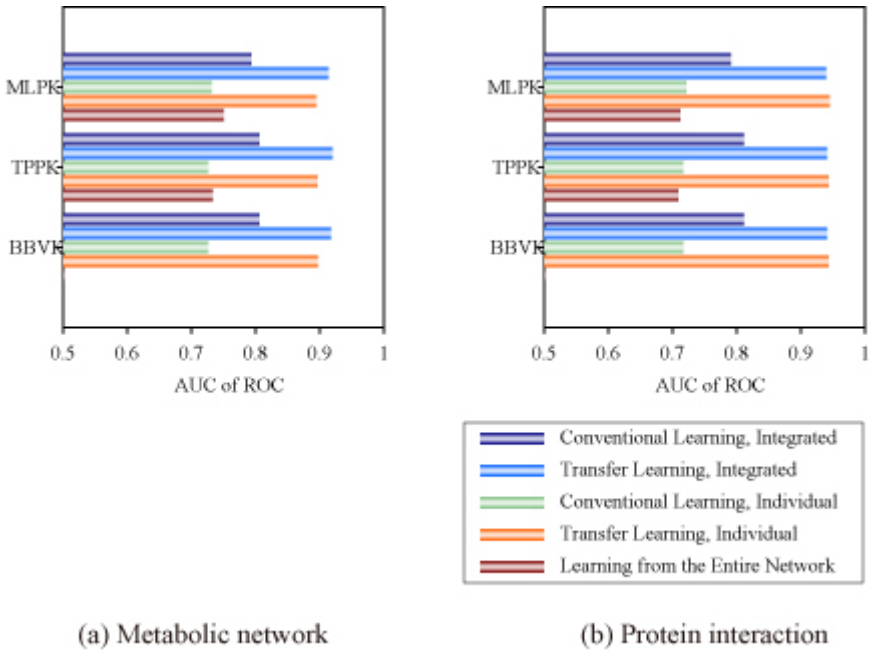
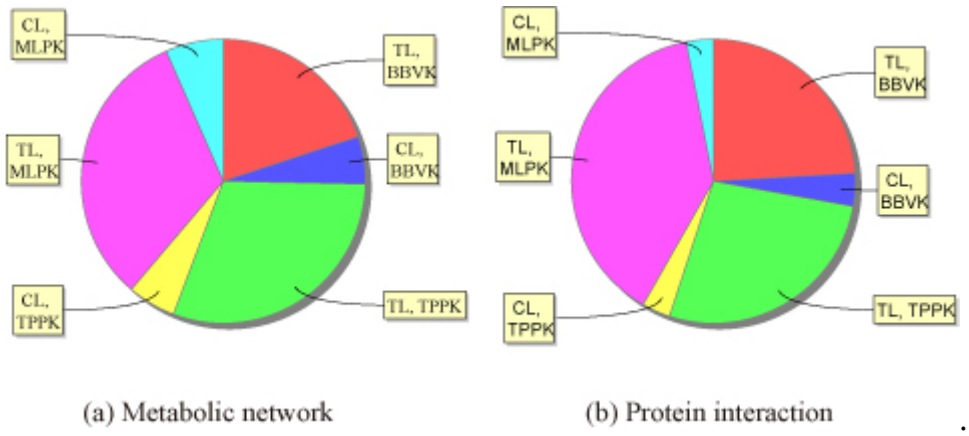


Figure 2: Averaged AUC of ROC. 'Transfer Learning, Integrated' indicates our algorithm that integrates multiple data sources and predicts the links of networks with transfer learning. 'Conventional Learning, Integrated' is the case without transfer learning, i.e. BBV algorithm. 'Transfer Learning, Individual'means the transfer learning from an individual data source, and the average is plotted. 'Conventional Learning, Individual' is the averaged ROC score of conventional learning from each data source. 'Learning from the Entire Network, Integrated' is the approach to pick training samples from the entire network. Namely, that is the global model.

algorithm that achieves the best performance among the six combinations for each task. Figure 3 shows how many times each algorithm achieves the best prediction performance. For both networks, transfer learning yields the best AUC much more frequently than non-transfer learning. The differences among pairwise kernels are not so large.

Our problem setting is different from that of the global models used by Ben-Hur & Noble (2005) and Vert et al. (2007). Global models attempt to infer the existence/absence of unknown dges in the whole network, whereas our algorithm predicts edges that link a specified target node to the other nodes. Despite differences in problem setting, it is still possible to use global models in our setting. In the training stage, global models use all the known edges as positive samples, and they randomly pick an equal number of absent edges as negative samples. From our experiences in problem setting, we find that improvements can be achieved by slightly modifying the way to pick negative samples, namely, to choose all the absent edges of the target node and to pick the other absent edges randomly so that the number of negative training samples is equal to the number of positive ones. One disadvantage is that their



(a) Metabolic network (b) Protein interaction

Figure 3: The number of times each algorithm achieves the best prediction performance among the six methods. In the problem we tackle, we can assign each node to a prediction task. For each task, we perform the six methods: 'TL, BBVK', 'CL, BBVK', 'TL, TPPK', 'CL, TPPK', 'TL, MLPK', and 'CL, MLPK' where TL and CL are the abbreviations for transfer learning and conventional learning, respectively. BBVK, TPPK, and MLPK are the types of pairwise kernels. For most of the tasks, transfer learning achieves a best prediction performance.

approaches need much more computational time for training compared to our method, because the number of training samples is much larger. Due to the heavy computation, we abandon the cross-validation within the training set to determine the value of the regularization parameter of SVM for comparison of global models with ours in terms of prediction performance. Instead, we try several values of the regularization parameter and report the best performance. Although the way of evaluation is advantageous to their approaches, their approaches could not obtain performance superior to our transfer learning (See Figure 2).

Discussion

In building a local model for each node through our algorithm, the weight of each type of assay data is optimized. To find the extent to which each assay data is effective in the entire local model building process, we count the number of resulting local models that have high weight of each data. Because the results do not remarkably vary among any of the kernels (MLPK, TPPK or BBVK) and/or any of cutoff thresholds of weight (0.5, 0.6, 0.7 or 0.8) (Tables 2–7 in Supplemental Information http://www.net-machine.net/~kato/pdf/t-kato-ijkdb2009a-suppl.pdf), hereafter we describe the results of using BBVK and 0.8 as the cutoff threshold.

Regarding the local models for the protein interaction network, expression data and Pfam data are highly weighted compared to other data (Table 1 and Figure 4). Although

Table 1: Number of local models that have a high weight of each assay data. blast: sequence similarity data calculated by BLAST, expr: expression data, fft: hydropathy profile, pfam hmm: Pfam data. sw: sequence similarity data calculated by ALN. Local models that have high weight (≥ 0.8) in the case of using BBVK were counted.

	Metabolic Network	Protein Interactions
blast	8	5
expr	0	40
fft	0	0
pfam_hmm	149	15
sw	0	0

the multiple selection as shown in this case – some local models prefer the expression data while others prefer the Pfam data – may indicate that the protein interaction network is a superimposition of various interactions induced by distinct mechanisms, it may be simply derived from biologically meaningless artifacts, e.g., experimental properties, including sampling bias, protocol, and experimental principle. Therefore, we next describe the pattern of local models with polarized preference for those assay data.

In the case of weighting highly the expression data in the local model of node *u*, it is expected that not only *u* links with a node or nodes (i.e., neighbors of *u*) whose protein interaction profile (PIF) is similar to that of *u* but also the PIF of the neighbor(s) is highly correlated with its/their expression profile. To verify this, we count the gene ontology (GO) terms of proteins whose local models have high weight of the expression data. As a result, GO terms, 'ribosome biogenesis' (GO: 0042254) and 'RNA metabolic process' (GO: 0016070) of the GO biological process terms, are significantly over-represented (*p*-values are 2.3E-19 and 1.2E-06, respectively), indicating that the proteins are involved in ribosome function. In addition to the fact that ribosome-related proteins are densely connected in the protein interaction network, as described previously (Han et al., 2004), they are highly correlated with the expression profile (Jansen et al., 2002).. Those findings well explain the assignment of a high weight to gene expression data for those local models through the transfer learning framework.

Similarly, we analyze proteins whose local models have high weights of Pfam data, and find that the GO term, 'protein catabolic process' (GO: 0030163), is significantly over-represented (*p*-value: 3.3E-10). As most of them (12/15) indeed comprise proteasome complexes well known as protein catabolizers (i.e., they are subunits of proteasome complexes), they are likely to be densely connected in the protein interaction network according to the basic idea that protein complexes are detected as densely connected subnetworks. In addition, since most of the proteasome subunits share common domains (alpha/beta subunit domains), their Pfam profiles are likely to be similar to each other. Those observations justify the weight generated through data selection in our algorithm.

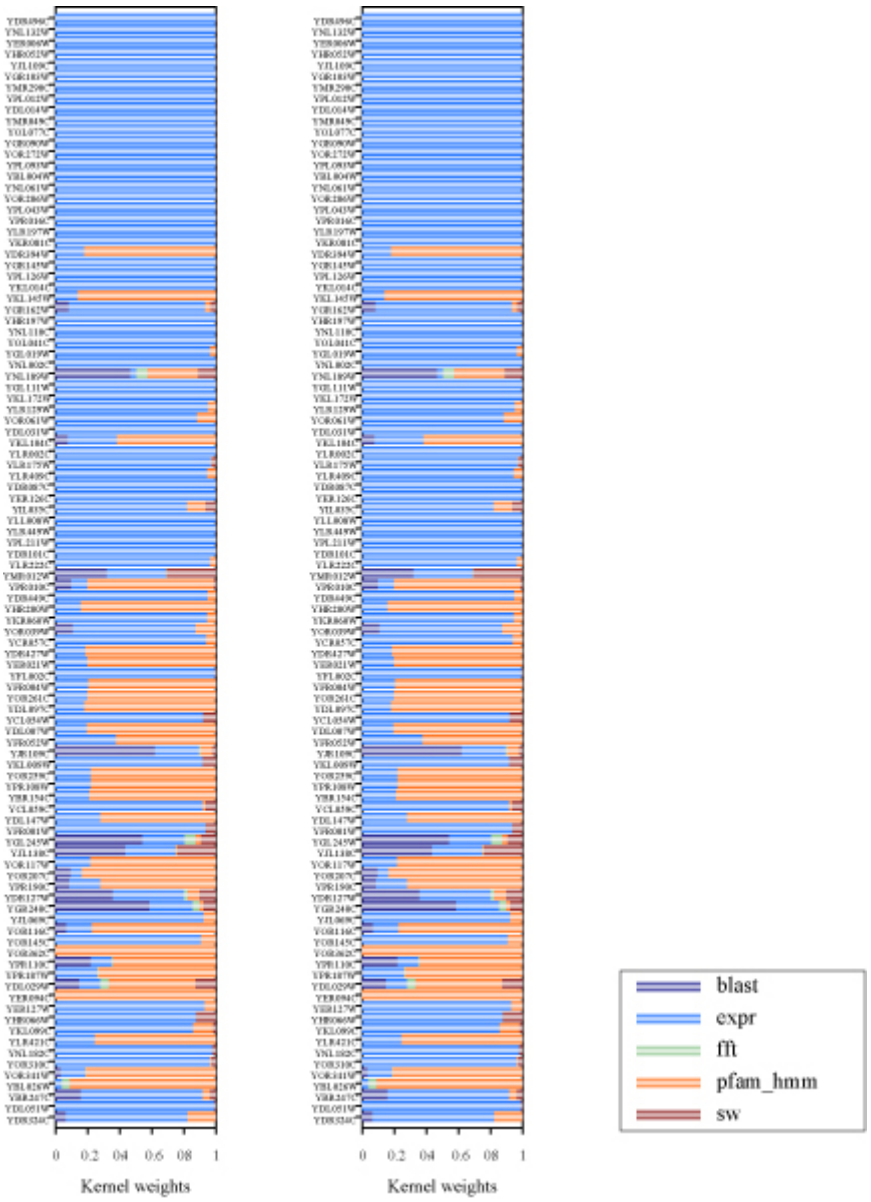


Figure 4: Weights of multiple types of assays for each target gene as determined by transfer learning with BBVK.

On the other hand, data indicating sequence similarity, particularly Pfam data, are highly weighted in the entire local model building process for the metabolic network (Table 1). Here we conduct pathway category analysis, instead of GO analysis, of proteins whose local models assign high weights to Pfam data. As a result, we find that categories indicating signal transduction, ‘cell cycle - yeast,’ ‘MAPK signaling pathway - yeast,’ and ‘phosphatidylinositol signaling system,’ are significantly over-represented (*p*-values are 7.0E-03, 9.6E-03, and 1.1E-02, respectively). Following the basic idea that signaling/interaction targets are usually recognized by domains, the preference for Pfam data at least in these pathways seems to be biologically reasonable.

Conclusions

In this paper, we discussed the effect of transfer learning in network inference. Our algorithm utilizes the small-world property of related learning tasks and we experimentally showed that the proposed algorithm substantially improves prediction accuracy. We also illustrated that multiple kernel learning works well and yields biologically plausible data selection.

Our algorithm uses only partial (i.e., local) information of assay data when building each local model. As described in the previous section, the optimized weight reflects biologically meaningful properties, at least in part, of protein interaction and metabolic networks. In future studies, we will further investigate biological interpretation of networks. The current data are noisy so they may not be believed absolutely, but we have to resort to using the current data to evaluate the prediction performance. We will investigate the performance again when the more reliable data become available.

Methods

This section presents our algorithm that infers a biological network from multiple assays. The description shown in this section uses the following notations. Vectors are denoted by boldface lower-case letters and matrices, by boldface upper-case letters. Elements of vectors and matrices are not printed in boldface. The transposition of matrix $\mathbf{A}$ is denoted by $\mathbf{A}^\top$. The $n \times n$ identity matrix is denoted by $\mathbf{I}_n$. The n-dimensional column vector all of whose elements are one is denoted by $\mathbf{1}_n$. We use $\mathbb{R}$ to denote the set of real numbers, $\mathbb{R}^n$ to denote the set of n-dimensional real column vectors, and $\mathbb{R}^{m \times n}$ to denote the set of $m \times n$ real matrices. The set of real nonnegative numbers is denoted by $\mathbb{R}_+$, and the set of n-dimensional real nonnegative vectors is denoted by $\mathbb{R}_+^n$. We use $\mathbb{S}^n$ to denote the set of $n \times n$ symmetric matrices, $\mathbb{S}_+^n$ to denote the set of $n \times n$ symmetric *positive semi-definite* matrices. $\mathbb{N}$ is the set of natural numbers. $\mathbb{N}_n$ is a subset of $\mathbb{N}$, and is defined by $\mathbb{N}_n \equiv \{i \in \mathbb{N} | i \leq n\}$. Symbols $\leq$ and $\geq$ are used to denote not only the standard inequalities between scalars, but also the componentwise inequalities between vectors.

BBV algorithm.

Our algorithm is based on the approach using local models proposed by Bleakley et al. (2007). We refer to their approach as BBV, taking the first letters of the three authors' names. BBV builds a classifier for a given node with index $u \in \mathbb{N}_n$ where n is the number of nodes. Node u is what we have referred to as the *target node*. The purpose of the classifier is to predict the existence or the absence of links with the given target node u. BBV assumes that a feature vector is given to every node. The feature vectors are used for SVM learning. Additionally, we are given two exclusive subsets of nodes $\mathcal{V}_{u,+}$ and $\mathcal{V}_{u,-}$; that is, $\mathcal{V}_{u,+} \cap \mathcal{V}_{u,-}$ is the null set. Every node in $\mathcal{V}_{u,+}$ is linked

with the target node *u*, and every node in $\mathcal{V}_{u,-}$ is not linked with *u*. Members in $\mathcal{V}_{u,+}$ are termed *neighbors* in graph theory. We call the subset $\mathcal{V}_u \equiv \mathcal{V}_{u,+} \cup \mathcal{V}_{u,-}$ the *training set*. The set of the other nodes $\mathbb{N}_n \setminus \mathcal{V}_u$ is called the *test set*. In summary, the BBV algorithm predicts the existence or the absence of the link between the target node and the node in the test set by using feature vectors. Note that we cannot use any class labels of the test set for SVM learning. In the case of Figure 1(c), the training set of $\mathcal{T}_1$ is $\mathcal{V}_1 = \{3\}$. The set of positives $\mathcal{V}_{1,+}$ in the training set consists only of node 3, and the set of negatives $\mathcal{V}_{u,-}$ is empty. Therefore, we can not train SVM because SVM learning requires at least one sample for each class.

The BBV algorithm consists of the learning stage and the prediction stage. In the learning stage, the classifier learns the classification rule of the class labels from the feature vectors in the training set $\mathcal{V}_u$. In the prediction stage, the classifier predicts the labels for the nodes in the test set $\mathbb{N}_n \setminus \mathcal{V}_u$, which corresponds to predicting the existence or absence of links to the target node.

Outline of our algorithm.

We introduce the idea of transfer learning into the BBV algorithm. Transfer learning is a framework to improve performance of a task using related learning tasks. This idea is useful in the current context since we can find the task relations in the network inference problem, as mentioned in Background. We will formulate the transfer learning approach. Let us denote the local model of a target node *u* by $\mathcal{V}_u$. We assume that the learning task $\mathcal{T}_u$ is related to the tasks associated with the neighbors of *u*, say $\mathcal{T}_v, \forall v \in \mathcal{V}_{u,+}$. Thus, we propose to employ the transfer learning of $\mathcal{T}_u$ with the help of $\mathcal{T}_v, \forall v \in \mathcal{V}_{u,+}$.

Support vector machine.

BBV employs the support vector machine (SVM) as a classifier. Basically, SVM finds a hyper-plane that distinguishes positive training examples from negative ones in a feature space. For that purpose, SVM formulates a scalar-valued linear score function. The parameters of the score function consist of the normal vector of the hyper-plane and the bias term that shifts the hyper-plane. If the norm of the normal vector is minimized subject to the constraints that the minimum score among positive examples is $+1$ and the maximum score among negative ones is -1, the *margin* between the positive set and the negative set is maximized. However, a hyper-plane satisfying those constraints does not always exist. To cope with this problem, the so-called *soft margin* SVM relaxes the constraints so that some examples can lie in the margin, but are given penalties according to the Hinge loss. Hereafter, we refer to the soft margin SVM as simply SVM.

Transfer learning with feature vectors.

To describe our algorithm, we begin with the learning algorithm of BBV using

numerical formulas, and introduce the transfer learning framework to the algorithm. Given *d*-dimensional feature vectors $\mathbf{x}_v \in \mathbb{R}^d$ for all nodes $v = 1, \ldots, n$, the BBV learning algorithm for a task $\mathcal{T}_u$ is expressed as

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}_u\|^2 + C_\alpha \sum_{v \in \mathcal{V}_u} \xi_{u,v}, \\ \text{wrt} \quad & \mathbf{w}_u \in \mathbb{R}^d, \quad \forall v \in \mathcal{V}_u : \ \xi_{u,v} \in \mathbb{R}_+, \\ \text{subj to} \quad & \forall v \in \mathcal{V}_{u,+} : \mathbf{w}_u^\top \mathbf{x}_v + b \geq +1 - \xi_{u,v}, \\ & \forall v \in \mathcal{V}_{u,-} : \mathbf{w}_u^\top \mathbf{x}_v + b \leq -1 - \xi_{u,v}, \end{aligned}$$

where C_α is a constant to be determined in advance. In the context of multi-task learning, Evgeniou et al. (2005) suggested to append to the objective function a term that forces $\mathbf{w}_v$ of every related task to be close to $\mathbf{w}_u$. We borrow this idea and formulate our transfer learning algorithm as follows:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}_u\|^2 + C_\alpha \sum_{v \in \mathcal{V}_u} \xi_{u,v} + \frac{1}{2}\sum_{v \in \mathcal{V}_u} \left(\|w_v\|^2 + C_\rho \|w_v - w_u\|^2 \right) \\ \text{wrt} \quad & \forall v_1 \in \mathcal{V}_u \cup \{u\}, \forall v_2 \in \mathcal{V}_{v_1} : \quad \mathbf{w}_{v_1} \in \mathbb{R}^d, \ \xi_{v_1,v_2} \in \mathbb{R}_+, \\ \text{subj to} \quad & \forall v_1 \in \mathcal{V}_u \cup \{u\}, \forall v_2 \in \mathcal{V}_{v_1,+} : \quad \mathbf{w}_{v_1}^\top \mathbf{x}_{v_2} + b \geq +1 - \xi_{v_1,v_2}, \\ & \forall v_1 \in \mathcal{V}_u \cup \{u\}, \forall v_2 \in \mathcal{V}_{v_1,-} : \quad \mathbf{w}_{v_1}^\top \mathbf{x}_{v_2} + b \leq -1 + \xi_{v_1,v_2}, \end{aligned} \tag{1}$$

where C_ρ is a constant, trading off the strength of transferring for regularization and loss.

Learning with kernels.

We now extend our learning algorithm to a kernel method. What are kernel methods? Kernel methods work on kernel matrices instead of feature vectors. Kernel matrices are alternative representation to feature vectors, and are positive semidefinite similarity matrices. The simplest kernel matrix consists of the inner product between feature vectors. Kernel methods are designed so that when the inner product between feature vectors is used to generate a kernel matrix, the score function learned from the kernel matrix yields exactly the same function as the one obtained directly from the feature vectors.

The BBV algorithm is readily kernelized because it directly uses SVM, which is a typical kernel method. However, since we incorporate transfer learning in our algorithm, kernelization of our algorithm may not be straightforward. In order to feed a kernel matrix to our algorithm, we need to re-formulate the original learning algorithm (1). We will present the kernelized version of our learning algorithm and show that the kernelized learning problem can be reduced to an optimization problem that can be solved efficiently using conventional techniques. Finally, we will further extend the algorithm so that it works on multiple kernel matrices.

Transfer learning from a single kernel matrix.

We now present the kernelized version of our algorithm. Let us denote the original

kernel matrix among n nodes by $\mathbf{K}^{\text{node}}$. Our kernelized learning algorithm requires the submatrices of the original kernel matrix; for $\forall a, \forall b \in \mathcal{V}_u \cup \{u\}$, we denote by $\mathbf{K}^{\text{node}}_{(a,b)}$ the submatrix in which the rows and columns correspond to $\mathcal{V}_a$ and $\mathcal{V}_b$, respectively. Each element of the submatrix $\mathbf{K}^{\text{node}}_{(a,b)}$ is given by

$$\forall i \in \mathbb{N}_{\ell_a}, \forall j \in \mathbb{N}_{\ell_b} : \left[\mathbf{K}^{\text{node}}_{(a,b)}\right]_{ij} = K^{\text{node}}_{v_{a,i}, v_{b,j}}$$

where $\mathcal{V}_a \equiv \{v_{a,1}, \ldots, v_{a,\ell_a}\}$ and $\mathcal{V}_b \equiv \{v_{b,1}, \ldots, v_{b,\ell_b}\}$. We assume that the elements are arranged in ascending order, i.e., $v_{a,1} < \cdots < v_{a,\ell_a}$ and $v_{b,1} < \cdots < v_{b,\ell_b}$. For each task $\mathcal{T}_a$ with the training set $\mathcal{V}_a$, we define the label vector $\mathbf{y}_a \in \mathbb{R}^{\ell_a}$ by

$$\left[y_a\right]_i \equiv \begin{cases} +1 & \text{if } v_{a,i} \in \mathcal{V}_{a,+}, \\ -1 & \text{if } v_{a,i} \in \mathcal{V}_{a,-}, \end{cases}$$

for $i = 1, \ldots, \ell_a$. Let the positive training set of task $\mathcal{T}_u$ be $\mathcal{V}_{u,+} = \{v_1, \ldots, v_M\}$ where $M = \left|\mathcal{V}_{u,+}\right|$. Using the Lagrangian multiplier technique (Boyd & Vandenberghe, 2004), we can derive the kernel version of our algorithm as a minimization problem with the following objective function:

$$\begin{aligned} J(\bar{\alpha}) \equiv{}& \frac{1}{2} F_0 \boldsymbol{\alpha}_u^\top \mathrm{diag}(\mathbf{y}_u) \mathbf{K}^{\text{node}}_{(u,u)} \mathrm{diag}(\mathbf{y}_u) \boldsymbol{\alpha}_u + \sum_{i=1}^{M} F_1 \boldsymbol{\alpha}_u^\top \mathrm{diag}(\mathbf{y}_u) \mathbf{K}^{\text{node}}_{(u,v_i)} \mathrm{diag}(\mathbf{y}_{v_i}) \boldsymbol{\alpha}_{v_i} \\ &+ \frac{1}{2} \sum_{i=1}^{M} F_2 \boldsymbol{\alpha}_{v_i}^\top \mathrm{diag}(\mathbf{y}_{v_i}) \mathbf{K}^{\text{node}}_{(v_i,v_i)} \mathrm{diag}(\mathbf{y}_{v_i}) \boldsymbol{\alpha}_{v_i} + \sum_{i=1}^{M} \sum_{j=i+1}^{M} F_3 \boldsymbol{\alpha}_{v_i}^\top \mathrm{diag}(\mathbf{y}_{v_i}) \mathbf{K}^{\text{node}}_{(v_i,v_j)} \mathrm{diag}(\mathbf{y}_{v_j}) \boldsymbol{\alpha}_{v_j} \\ &- \mathbf{1}_{\ell_u}^\top \boldsymbol{\alpha}_u - \sum_{i=1}^{M} \mathbf{1}_{\ell_{v_i}}^\top \boldsymbol{\alpha}_{v_i} \end{aligned}$$

where

$$F_0 \equiv \frac{1 + 2C_\rho}{1 + 2(1+M)C_\rho}, \quad F_1 \equiv \frac{2C_\rho}{1 + 2(1+M)C_\rho},$$

$$F_2 \equiv \frac{1}{1 + 2C_\rho} + \frac{4C_\rho^2}{(1 + 2(1+M)C_\rho)(1 + 2C_\rho)},$$

$$F_3 \equiv \frac{1}{1 + 2C_\rho}.$$

The derivation is in the supplemental document. The variable for this optimization problem is

$$\bar{\boldsymbol{\alpha}} \equiv \left[\boldsymbol{\alpha}_{v_1}^\top, \ldots, \boldsymbol{\alpha}_{v_M}^\top, \boldsymbol{\alpha}_u^\top\right]^\top,$$

and the constraints are

$$\mathbf{0} \le \bar{\boldsymbol{\alpha}} \le C_\alpha \mathbf{1}, \qquad \left\langle \bar{\mathbf{y}}, \bar{\boldsymbol{\alpha}} \right\rangle = 0,$$

where we concatenate the label vectors as

$$\bar{\mathbf{y}} \equiv \left[\mathbf{y}_{v_1}^\top, \ldots, \mathbf{y}_{v_M}^\top, \mathbf{y}_u^\top\right]^\top.$$

Pairwise kernels.

As the BBV algorithm deals with feature vectors of nodes directly, the kernelized BBV algorithm can handle only kernels among nodes (*node kernels*). In the context of predicting the existence of an edge in a node pair, however, it is natural and would be more powerful to use kernels among node pairs (*pairwise kernels*). In BBV, one of the elements in every node pair should always be the target node. Since various node pairs need to be dealt with in the context of transfer learning, the use of the pairwise kernels is not allowed in BBV. On the other hand, our kernelized algorithm can handle the pairwise kernels, which is an advantage over the original BBV algorithm. To our knowledge, there are two useful pairwise kernels proposed so far: the tensor product pairwise kernel (TPPK) (Ben-Hur & Noble, 2005) and the metric learning pairwise kernel (MLPK) (Vert et al., 2007). For $\mathcal{V}_a = \{v_{a,1}, \ldots, v_{a,\ell_a}\}$ and $\mathcal{V}_b = \{v_{b,1}, \ldots, v_{b,\ell_b}\}$ the elements of TPPK and MLPK are defined by

$$\forall i \in \mathbb{N}_{\ell_a}, \forall j \in \mathbb{N}_{\ell_b}:$$

$$\left[\mathbf{K}^{\mathrm{tppk}}_{(a,b)}\right]_{ij} \equiv K^{\mathrm{node}}_{a,b} K^{\mathrm{node}}_{v_{a,i},v_{b,j}} + K^{\mathrm{node}}_{a,v_{b,j}} K^{\mathrm{node}}_{b,v_{a,i}},$$

$$\left[\mathbf{K}^{\mathrm{mlpk}}_{(a,b)}\right]_{ij} \equiv \left(K^{\mathrm{node}}_{a,b} - K^{\mathrm{node}}_{a,v_{b,j}} - K^{\mathrm{node}}_{v_{a,i},b} + K^{\mathrm{node}}_{b,v_{b,j}}\right)^2,$$

respectively. TPPK is based on a tensorization of the original feature space, whereas MLPK is derived from convex optimization of a distance metric learning problem. In order to distinguish the above pairwise kernels from the kernel used by the original BBV algorithm, we refer to the original one as the BBV kernel (BBVK), which is expressed as

$$\mathbf{K}^{\mathrm{bbvk}}_{(a,b)} \equiv \mathbf{K}^{\mathrm{node}}_{(a,b)}.$$

Hereafter, we use the superscript ‘pw’ to denote a pairwise kernel.

Connection to SVM.

An attractive advantage of our algorithm is that through slight modification of the original kernel matrices, the kernel version of our algorithm is reduced to an optimization problem that shares the same form as the original SVM. Since there are a number of studies for efficient SVM training, we can directly exploit these techniques or the software for our algorithm. Next, we show this fact. We modify the kernel matrices as

$$\begin{aligned}
&\forall i \in \mathbb{N}_M, && \mathbf{G}_{(v_i,v_i)} \equiv F_2 \mathbf{K}^{\mathrm{pw}}_{(v_i,v_i)},\\
&\forall i \in \mathbb{N}_M, \forall j \in \mathbb{N}_M \setminus \{i\}, && \mathbf{G}_{(v_i,v_j)} \equiv F_3 \mathbf{K}^{\mathrm{pw}}_{(v_i,v_j)},\\
&\forall i \in \mathbb{N}_M, && \mathbf{G}_{(v_i,u)} \equiv F_1 \mathbf{K}^{\mathrm{pw}}_{(v_i,u)},\\
& && \mathbf{G}_{(u,v_j)} \equiv F_1 \mathbf{K}^{\mathrm{pw}}_{(u,v_i)},\\
& && \mathbf{G}_{(u,u)} \equiv F_0 \mathbf{K}^{\mathrm{pw}}_{(u,u)},
\end{aligned}$$

for $\mathcal{V}_{u,+} = \{v_1, \ldots, v_M\}$. We arrange these matrices as follows:

$$\mathbf{G}_{\mathrm{fea}} \equiv \begin{bmatrix} \mathbf{G}_{(v_1,v_1)} & \cdots & \mathbf{G}_{(v_1,v_M)} & \mathbf{G}_{(v_1,u)} \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{G}_{(v_M,v_1)} & \cdots & \mathbf{G}_{(v_M,v_M)} & \mathbf{G}_{(v_M,u)} \\ \mathbf{G}_{(u,v_1)} & \cdots & \mathbf{G}_{(u,v_M)} & \mathbf{G}_{(u,u)} \end{bmatrix}.$$

Note that if we let $\ell \equiv \ell_{v_1} + \ldots + \ell_{v_M} + \ell_u$, the size of $\mathbf{G}_{\mathrm{fea}}$ is $\ell \times \ell$ and the length of $\mathbf{y}$ is ℓ. Then, the objective function can be rearranged as

$$J(\overline{\boldsymbol{\alpha}}) \equiv \frac{1}{2}\overline{\boldsymbol{\alpha}}^{\top}\mathrm{diag}(\overline{\mathbf{y}})\mathbf{G}_{\mathrm{fea}}\mathrm{diag}(\overline{\mathbf{y}})\overline{\boldsymbol{\alpha}} - \mathbf{1}_{\ell}^{\top}\overline{\boldsymbol{\alpha}}.$$

This optimization algorithm turns out to be a special case of the one appearing in Evgeniou et al. (2005). If we regard $\overline{\mathbf{y}}$ and $\mathbf{G}_{\mathrm{fea}}$ as a class label vector and a kernel matrix, respectively, in a standard binary classification problem, the objective function of our algorithm will exactly coincide with the objective function of SVM. We can actually prove that $\mathbf{G}_{\mathrm{fea}}$ is positive semi-definite. The constraints of $\overline{\boldsymbol{\alpha}}$ are also the same as those of SVM. Hence, by a slight modification of kernel matrices as described above, standard SVM software can be used to obtain the solution to our transfer learning problem.

Transfer learning from multiple kernel matrices.

Let us consider the situation where we have multiple information sources. Suppose the number of information sources is n_{kern}, and the data from each information source is represented by a kernel matrix. Let us denote the n_{kern} pairwise kernel matrices by $K^1, \ldots, K^{n_{\mathrm{kern}}}$. The algorithm we develop in this paper allows us to optimize the coefficients in a linear combination of the pairwise kernel matrices. There is a sophisticated algorithm called SDP/SVM (Lanckriet et al., 2004), which chooses the coefficients so that useful data are emphasized in SVM classification. Here we show that some rearrangement allows us to directly use SDP/SVM in the current context. Letting $\boldsymbol{\mu} \in \mathbb{R}^{n_{\mathrm{kern}}}$ be the coefficient vector, we can express the linear combination of the kernel matrices as

$$\mathbf{K}^{\mathrm{int}} = \sum_{k=1}^{n_{\mathrm{kern}}} \mu_k \mathbf{K}^k.$$

We modify the integrated kernel matrix $\mathbf{K}^{\mathrm{int}} \in \mathbb{S}_+^n$ to obtain $\mathbf{G}^{\mathrm{int}} \in \mathbb{S}_+^n$ as

$$\begin{aligned}
&\forall i \in \mathbb{N}_M, && \mathbf{G}^{\mathrm{int}}_{(v_i,v_i)} \equiv F_2 \mathbf{K}^{\mathrm{int}}_{(v_i,v_i)}, \\
&\forall i \in \mathbb{N}_M, \forall j \in \mathbb{N}_M \setminus \{i\}, && \mathbf{G}^{\mathrm{int}}_{(v_i,v_j)} \equiv F_3 \mathbf{K}^{\mathrm{int}}_{(v_i,v_j)}, \\
&\forall i \in \mathbb{N}_M, && \mathbf{G}^{\mathrm{int}}_{(v_i,u)} \equiv F_1 \mathbf{K}^{\mathrm{int}}_{(v_i,u)}, \\
& && \mathbf{G}^{\mathrm{int}}_{(u,v_j)} \equiv F_1 \mathbf{K}^{\mathrm{int}}_{(u,v_i)}, \\
& && \mathbf{G}^{\mathrm{int}}_{(u,u)} \equiv F_0 \mathbf{K}^{\mathrm{int}}_{(u,u)},
\end{aligned}$$

for $\mathcal{V}_{u,+} = \{v_1, \ldots, v_M\}$. Once the optimal values of the coefficients $\boldsymbol{\mu}$ are found, the composite kernel matrix is just fed into our learning algorithm. We can also consider the

modified kernel matrix $\mathbf{G}^k \in \mathbb{S}_+^n$ obtained from each kernel matrix $\mathbf{K}^k \in \mathbb{S}_+^n$ in a similar way. It can easily be shown that the matrix $\mathbf{G}^{\text{int}}$ associated with the integrated kernel matrix $\mathbf{K}^{\text{int}}$ is expressed as a linear combination of the modified kernel matrices:

$$\mathbf{G}^{\text{int}} = \sum_{k=1}^{n_{\text{kern}}} \mu_k \mathbf{G}^k.$$

Hence, we can straightforwardly employ $\left\{\mathbf{G}^k\right\}_{k=1}^{n_{\text{kern}}}$ as inputs of the SDP/SVM algorithm. The learning problem of SDP/SVM results in a quadratically constrained quadratic program (QCQP) that becomes intractable as the number of training samples increases. Recent studies (Bach et al., 2004; Sonnenburg et al., 2006; Rakotomamonjy et al., 2008) have addressed this issue and these new algorithms allow us to tackle large-scale problems such as our learning problem.

Gene ontology and Kegg pathway category analysis

The GO terms (Gene Ontology Consortium, 2004) assigned to yeast gene products were downloaded from the website of the Gene Ontology Consortium (http://www.geneontology.org/). After mapping the GO terms into the more general parent GO Slim terms (Generic GO Slim; obtained from the same website), we assigned the GO Slim terms to each yeast gene product. On the other hand, the Kegg pathway category assigned to yeast gene products was downloaded from the website of the Kegg pathway database (http://www.genome.jp/kegg/pathway.html). P-values for over-represented GO Slim terms/Kegg pathway categories were calculated based on hypergeometric distribution by our Perl script.

Acknowledgement

We thank W. Noble for making the data public. TK would like to thank Dr Goro Terai for fruitful discussion.

References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *J. Mol. Biol.*, 215 (3), 403-410.

Bach, F. R., Lanckriet, G. R. G., & Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the 21st international conference on machine learning*. New York.

Ben-Hur, A., & Noble, W. S. (2005). Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21 (Suppl. 1), i38-i46.

Bleakley, K., Biau, G., & Vert, J.-P. (2007). Supervised reconstruction of biological networks with local models. *Bioinformatics*, 23(13), i57-i65.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University

Press.

Caldarelli, G. (2007). *Scale-free networks: Complex webs in nature and technology*. Oxford Univ Press.

Chua, H. N., Sung, W. K., & Wong, L. (2006). Exploiting indirect neighbours and topological weight to predict protein function from protein-protein interactions. *Bioinformatics*, 22 (13), 1623-1630.

Gene Ontology Consortium (2004). The gene ontology (GO) database and informatics resource. *Nucleic Acids Research*, 32 (Database issue), D258-D261.

Evgeniou, T., Micchelli, C. A., & Pontil, M. (2005). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6, 615-637.

Finn, R. D., Tate, J., Mistry, J., Coggill, P. C., Sammut, S. J., Hotz, H. R., Ceric, G., Forslund, K., Eddy, S. R., Sonnhammer, E. L., & Bateman, A. (2008). The Pfam protein families database. *Nucleic Acids Res.*, 36, D281-D288.

Gotoh, O. (1982). An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162 (3), 705-708.

Han, J.-D. J., Bertin, N., Hao, T., Goldberg, D. S., Berriz, G. F., Zhang, L. V., Dupuy, D., Walhout, A. J., Cusick, M. E., Roth, F. P., & Vidal, M. (2004). Evidence for dynamically organized modularity in the yeast protein-protein interaction network. *Nature*, 430(6995), 88-93.

Jansen, R., Greenbaum, D., & Gerstein, M. (2002). Relating whole-genome expression data with protein-protein interactions. *Genome Research*, 12, 37-46.

Kato, T., Tsuda, K., & Asai, K. (2005). Selective integration of multiple biological data for supervised network inference. *Bioinformatics*, 21(10), 2488-2495.

Lanckriet, G. R. G., Bie, T. D., Cristianini, N., Jordan, M., & Noble, W. (2004). A statistical framework for genomic data fusion. *Bioinformatics*, 20(16), 2626-2635.

von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S. G., Fields, S., & Bork, P. (2002). Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417(6887), 399-403.

Okada, K., Kanaya, S., & Asai, K. (2005). Accurate extraction of functional associations between proteins based on common interaction partners and common domains. *Bioinformatics*, 21 (9), 2043-2048.

Pavlidis, P., Weston, J., Cai, J., & Noble, W. S. (2002). Learning gene functional classifications from multiple data types. *Journal of Computational Biology*, 9(2), 401-411.

Rakotomamonjy, A., Bach, F., Canu, S., & Grandvalet, Y. (2008). SimpleMKL. *Journal of Machine Learning Research*, 9, 2491-2521.

Samanta, M. P., & Liang, S. (2003). Predicting protein functions from redundancies in largescale protein interaction networks. *Proc Natl Acad Sci USA*, 100 (22), 12579-12583.

Schölkopf, B. & Smola, A. J. (2002). *Learning with kernels*. MIT Press, Cambridge MA.

Sonnenburg, S., Rätsch, G., Schäfer, C., & Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7, 1531-1565.

Vert, J. P., Qiu, J., & Noble, W. S. (2007). A new pairwise kernel for biological network inference with support vector machines. *BMC Bioinformatics*, 10 (Suppl 10), S8.

Vert, J.-P., & Yamanishi, Y. (2005). Supervised graph inference. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems 17* (pp. 1433-1440). Cambridge, MA: MIT Press.

Yamanishi, Y., Vert, J.-P., & Kanehisa, M. (2005). Supervised enzyme network inference from the integration of genomic data and chemical information. *Bioinformatics*, 21 (Suppl 1), i468-i477.

Biography

Tsuyoshi Kato received the B.E., M.E., and PhD degree from Tohoku University, Sendai, Japan, in 1998, 2000, and 2003. From 2003 to 2005, he was with the National Institute of Advanced Industrial Science Technology (AIST) as a postdoctoral fellow in the Computational Biology Research Center (CBRC) at Tokyo. From 2005 to 2008, he was an assistant professor at the Graduate School of Frontier Sciences, University of Tokyo. Since 2008, he has been an associate professor at the Center for Informational Biology, Ochanomizu University. Currently, he is also a guest researcher at CBRC. His current scientific interests include bioinformatics and statistical pattern recognition.

Kinya Okada received the BS degree in Agriculture from the University of Tohoku, Sendai, Japan in 2003, and the MS degree in Information Science from Nara Institute of Science and Technology, Nara, Japan in 2004, and the PhD degree in Science from the University of Tokyo, Kashiwa, Japan in 2008. Currently, he is in KO Institute for Medical Bioinformatics, Kanagawa, Japan. His research interest is bioinformatics.

Hisashi Kashima received the M.E. and Ph.D. degrees in informatics from Kyoto University, Kyoto, Japan, in 1999 and 2007, respectively. He has been a Researcher at the Tokyo Research Laboratory, IBM Research, Tokyo, Japan, since 1999. His research interests include machine learning and its applications to bioinformatics, autonomic

computing, and business intelligence.

Masashi Sugiyama received the B.E., M.E., and Ph.D. degrees in computer science from Tokyo Institute of Technology, Tokyo, Japan, in 1997, 1999, and 2001, respectively. In 2001, he was appointed as a Research Associate at the Tokyo Institute of Technology, where since 2003, he has been an Associate Professor. His research interests include machine learning and signal/image processing.

Supplemental Information for A Transfer Learning Approach and Selective Integration of Multiple Assays for Biological Network Inference

Tsuyoshi Kato[1,2], Kinya Okada[3], Hisashi Kashima[4] and Masashi Sugiyama[5]

June 27, 2009

[1] AIST Computational Biology Research Center,
2-42 Aomi, Koto-ku, Tokyo 135–0064, Japan.
[2] Center for Informational Biology, Ochanomizu University,
2-1-1 Ohtsuka, Bunkyo-ku, Tokyo 112–8610, Japan.
[3] KO Institute for Medical Bioinformatics,
Yokohama, Kanagawa 227–0033, Japan.
[4] IBM Research, Tokyo Research Laboratory,
1623-14 Shimo-tsuruma, Yamato, Kanagawa, 242-8502 Japan.
[5] Tokyo Institute of Technology, Department of Computer Science,
2-12-1, O-okayama, Meguro-ku, Tokyo 152-8552 Japan.

1 Derivation of the Dual Problem

As written in the main text, the primal form of our algorithm is given by

$$\begin{aligned}
\min \quad & \frac{1}{2}\|\boldsymbol{w}_u\|^2 + C_\alpha \sum_{v_1 \in \mathcal{V}_u \cup \{u\}} \sum_{v_2 \in \mathcal{V}_{v_1,+}} \xi_{v_1,v_2} \\
& + \frac{1}{2} \sum_{v \in \mathcal{V}_v} \left(\|\boldsymbol{w}_v\|^2 + C_\rho \|\boldsymbol{w}_u - \boldsymbol{w}_v\|^2 \right), \\
\text{wrt} \quad & \forall v_1 \in \mathcal{V}_u \cup \{u\}, \forall v_2 \in \mathcal{V}_{v_1,+} : \\
& \boldsymbol{w}_{v_1} \in \mathbb{R}^d,\ \xi_{v_1,v_2} \in \mathbb{R}, \\
\text{subj to} \quad & \forall v \in \mathcal{V}_{u,+} : \boldsymbol{w}_u^\top \boldsymbol{x}_v + b \geq +1 - \xi_{u,v}, \\
& \forall v \in \mathcal{V}_{u,-} : \boldsymbol{w}_u^\top \boldsymbol{x}_v + b \leq -1 + \xi_{u,v},
\end{aligned}$$

The main text also shows the kernelized algorithm. We now show its derivation from the primal form to the kernelized version. Letting

$$\boldsymbol{W} = [\boldsymbol{w}_{v_1}, \ldots, \boldsymbol{w}_{v_M}, \boldsymbol{w}_u],$$

the objective function can be re-expressed as

$$\frac{1}{2}\mathrm{vec}(\boldsymbol{W})^\top \left(\boldsymbol{Q}^{\mathrm{task}} \otimes \boldsymbol{I}_d \right) \mathrm{vec}(\boldsymbol{W}) + C_\alpha \sum_{v_1 \in \mathcal{V}_u \cup \{u\}} \sum_{v_2 \in \mathcal{V}_{v_1,+}} \xi_{v_1,v_2}$$

where we have defined

$$\boldsymbol{Q}^{\mathrm{task}} = \left(\boldsymbol{I}_{M+1} + C_\rho \begin{bmatrix} \boldsymbol{I}_M & -\boldsymbol{1}_M^\top \\ -\boldsymbol{1}_M & M \end{bmatrix} \right).$$

Evgeniou et al. (2005) show the dual problem

$$\begin{aligned} \min \quad & \frac{1}{2} \bar{\boldsymbol{\alpha}}^\top \operatorname{diag}(\bar{\boldsymbol{y}}) \left(\left(\boldsymbol{Z} \left(\boldsymbol{Q}^{\mathrm{task}} \right)^{-1} \boldsymbol{Z}^\top \right) \circ \boldsymbol{K}^{\mathrm{node}} \right) \operatorname{diag}(\bar{\boldsymbol{y}}) \bar{\boldsymbol{\alpha}} - \boldsymbol{1}_{\ell_u}^\top \boldsymbol{\alpha}_u - \sum_{i=1}^{M} \boldsymbol{1}_{\ell_{v_i}}^\top \boldsymbol{\alpha}_{v_i}. \\ \text{wrt} \quad & \boldsymbol{\alpha} \in \mathbb{R}^\ell \\ \text{subj to} \quad & \boldsymbol{0}_\ell \leq \boldsymbol{\alpha} \leq C_\alpha \boldsymbol{1}_\ell, \qquad \langle \bar{\boldsymbol{y}}, \bar{\boldsymbol{\alpha}} \rangle = 0 \end{aligned}$$

where we let $\boldsymbol{Z} \in \mathbb{N}^{M \times \ell}$ be the indicator of a task and a sample such that

$$\boldsymbol{Z} \equiv \begin{bmatrix} \boldsymbol{1}_{\ell_{v_1}} & \boldsymbol{0}_{\ell_{v_1}} & \cdots & \boldsymbol{0}_{\ell_{v_1}} & \boldsymbol{0}_{\ell_{v_1}} \\ \boldsymbol{0}_{\ell_{v_2}} & \boldsymbol{1}_{\ell_{v_2}} & \cdots & \boldsymbol{0}_{\ell_{v_2}} & \boldsymbol{0}_{\ell_{v_2}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0}_{\ell_{v_M}} & \boldsymbol{0}_{\ell_{v_M}} & \cdots & \boldsymbol{1}_{\ell_{v_M}} & \boldsymbol{0}_{\ell_{v_M}} \\ \boldsymbol{0}_{\ell_u} & \boldsymbol{0}_{\ell_u} & \cdots & \boldsymbol{0}_{\ell_u} & \boldsymbol{1}_{\ell_u} \end{bmatrix}. \tag{1}$$

The inverse of the matrix $\boldsymbol{Q}$ is given by

$$\left(\boldsymbol{Q}^{\mathrm{task}} \right)^{-1} = \begin{bmatrix} F_2 \boldsymbol{1}_M \boldsymbol{1}_M^\top + (F_3 - F_2) \boldsymbol{I}_M & F_1 \boldsymbol{1}_M \\ F_1 \boldsymbol{1}_M^\top & F_0 \end{bmatrix} \tag{2}$$

where F_0, F_1, F_2, and F_3 are

$$\begin{aligned} F_0 &\equiv \frac{1 + 2C_\rho}{1 + 2(1+M)C_\rho}, \\ F_1 &\equiv \frac{2C_\rho}{1 + 2(1+M)C_\rho}, \\ F_2 &\equiv \frac{1}{1 + 2C_\rho} + \frac{4C_\rho^2}{(1 + 2(1+M)C_\rho)(1 + 2C_\rho)}, \\ F_3 &\equiv \frac{1}{1 + 2C_\rho} \end{aligned}$$

which are same as the definitions in the main text. Hence, if we substitute (2) into the objective function of the problem (1), we obtain

$$\begin{aligned} J(\bar{\boldsymbol{\alpha}}) \equiv & \frac{1}{2} F_0 \boldsymbol{\alpha}_u^\top \operatorname{diag}(\boldsymbol{y}_u) \boldsymbol{K}_{(u,u)}^{\mathrm{node}} \operatorname{diag}(\boldsymbol{y}_u) \boldsymbol{\alpha}_u \\ & + \sum_{i=1}^{M} F_1 \boldsymbol{\alpha}_u^\top \operatorname{diag}(\boldsymbol{y}_u) \boldsymbol{K}_{(u,v_i)}^{\mathrm{node}} \operatorname{diag}(\boldsymbol{y}_{v_i}) \boldsymbol{\alpha}_{v_i} \\ & + \frac{1}{2} \sum_{i=1}^{M} F_2 \boldsymbol{\alpha}_{v_i}^\top \operatorname{diag}(\boldsymbol{y}_{v_i}) \boldsymbol{K}_{(v_i,v_i)}^{\mathrm{node}} \operatorname{diag}(\boldsymbol{y}_{v_i}) \boldsymbol{\alpha}_{v_i} \\ & + \sum_{i=1}^{M} \sum_{j=i+1}^{M} F_3 \boldsymbol{\alpha}_{v_i}^\top \operatorname{diag}(\boldsymbol{y}_{v_i}) \boldsymbol{K}_{(v_i,v_j)}^{\mathrm{node}} \operatorname{diag}(\boldsymbol{y}_{v_j}) \boldsymbol{\alpha}_{v_j} \\ & - \boldsymbol{1}_{\ell_u}^\top \boldsymbol{\alpha}_u - \sum_{i=1}^{M} \boldsymbol{1}_{\ell_{v_i}}^\top \boldsymbol{\alpha}_{v_i}. \end{aligned}$$

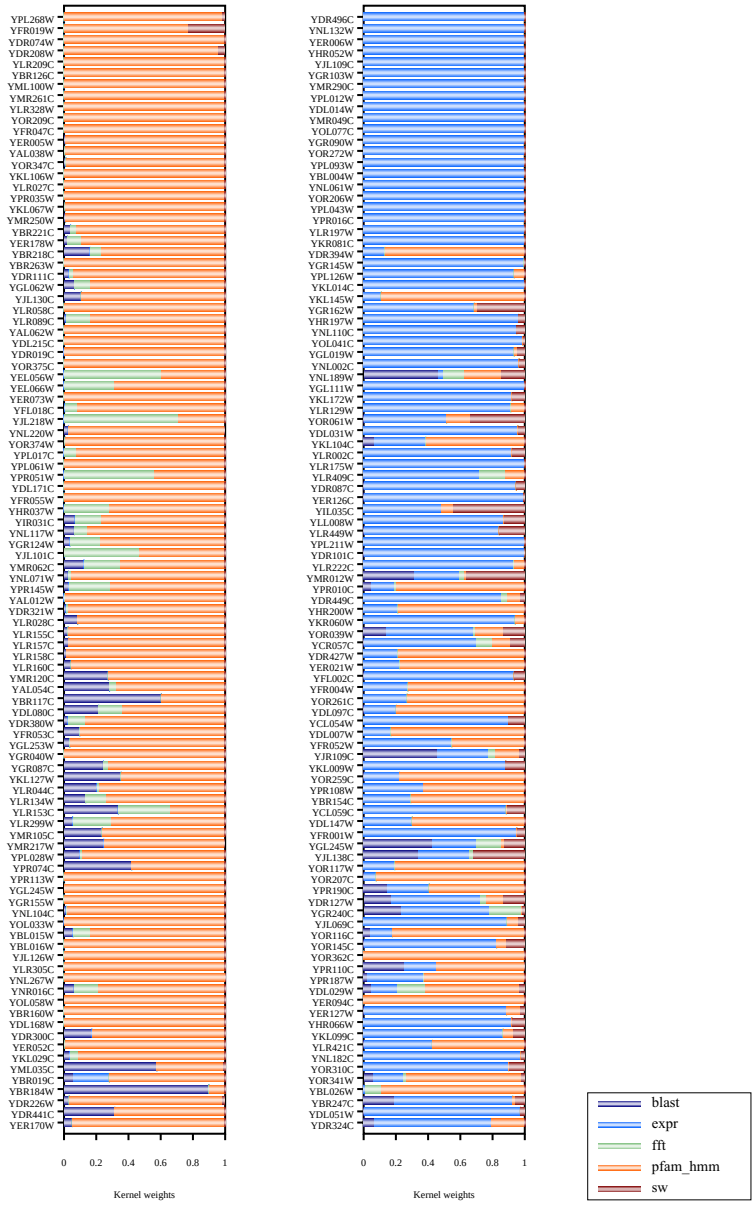
Table 2: The number of local models that have high weight of each assay data for the protein interaction network in the case of using MLPK.

Cutoff	fft	expr	blast	sw	pfam_hmm
1	0	0	0	0	0
0.9	0	0	0	0	0
0.8	9	54	9	0	17
0.7	16	76	34	6	33
0.6	21	86	47	8	44
0.5	25	99	57	11	50

Table 3: The number of local models that have high weight of each assay data for the metabolic network in the case of using MLPK.

Cutoff	fft	expr	blast	sw	pfam_hmm
1	0	0	0	0	0
0.9	0	0	0	0	0
0.8	3	0	6	0	128
0.7	8	0	15	1	194
0.6	11	0	21	1	207
0.5	11	0	31	2	220

2 Supplemental Data



(a) Metabolic network (b) Protein Interactions

Figure 4: The weights of multiple assays for each target gene that are determined by transfer learning with TPPK.

(a) Metabolic network (b) Protein Interactions

Figure 5: The weights of multiple assays for each target gene that are determined by transfer learning with MLPK.

Table 4: The number of local models that have high weight of each assay data for the protein interaction network in the case of using TPPK.

Cutoff	fft	expr	blast	sw	pfam_hmm
1	0	0	0	0	0
0.9	0	0	0	0	0
0.8	0	30	5	0	15
0.7	1	74	27	1	30
0.6	3	84	44	1	43
0.5	4	94	54	3	57

Table 5: The number of local models that have high weight of each assay data for the metabolic network in the case of using TPPK.

Cutoff	fft	expr	blast	sw	pfam_hmm
1	0	0	0	0	0
0.9	0	0	0	0	0
0.8	0	0	8	0	148
0.7	0	0	10	0	209
0.6	0	0	12	0	231
0.5	1	0	18	0	256

Table 6: The number of local models that have high weight of each assay data for the protein interaction network in the case of using BBVK.

Cutoff	fft	expr	blast	sw	pfam_hmm
1	0	0	0	0	0
0.9	0	0	0	0	0
0.8	0	40	5	0	15
0.7	1	83	28	1	28
0.6	1	92	40	1	48
0.5	3	101	54	4	59

Table 7: The number of local models that have high weight of each assay data for the metabolic network in the case of using BBVK.

Cutoff	fft	expr	blast	sw	pfam_hmm
1	0	0	0	0	0
0.9	0	0	0	0	0
0.8	0	0	8	0	149
0.7	0	0	10	0	222
0.6	0	0	11	0	243
0.5	2	0	17	0	257