

Research article

Open Access

Adaptive diffusion kernel learning from biological networks for protein function prediction

Liang Sun^{1,2}, Shuiwang Ji^{1,2} and Jieping Ye^{*1,2}

Address: ¹Center for Evolutionary Functional Genomics, The Biodesign Institute, Arizona State University, Tempe, AZ, USA and ²Department of Computer Science and Engineering, Arizona State University, Tempe, AZ, USA

Email: Liang Sun - sun.liang@asu.edu; Shuiwang Ji - shuiwang.ji@asu.edu; Jieping Ye* - jieping.ye@asu.edu

* Corresponding author

Published: 25 March 2008

Received: 30 May 2007

BMC Bioinformatics 2008, 9:162 doi:10.1186/1471-2105-9-162

Accepted: 25 March 2008

This article is available from: <http://www.biomedcentral.com/1471-2105/9/162>

© 2008 Sun et al; licensee BioMed Central Ltd.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/2.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

Background: Machine-learning tools have gained considerable attention during the last few years for analyzing biological networks for protein function prediction. Kernel methods are suitable for learning from graph-based data such as biological networks, as they only require the abstraction of the similarities between objects into the kernel matrix. One key issue in kernel methods is the selection of a good kernel function. Diffusion kernels, the discretization of the familiar Gaussian kernel of Euclidean space, are commonly used for graph-based data.

Results: In this paper, we address the issue of learning an optimal diffusion kernel, in the form of a convex combination of a set of pre-specified kernels constructed from biological networks, for protein function prediction. Most prior work on this kernel learning task focus on variants of the loss function based on Support Vector Machines (SVM). Their extensions to other loss functions such as the one based on Kullback-Leibler (KL) divergence, which is more suitable for mining biological networks, lead to expensive optimization problems. By exploiting the special structure of the diffusion kernel, we show that this KL divergence based kernel learning problem can be formulated as a simple optimization problem, which can then be solved efficiently. It is further extended to the multi-task case where we predict multiple functions of a protein simultaneously. We evaluate the efficiency and effectiveness of the proposed algorithms using two benchmark data sets.

Conclusion: Results show that the performance of linearly combined diffusion kernel is better than every single candidate diffusion kernel. When the number of tasks is large, the algorithms based on multiple tasks are favored due to their competitive recognition performance and small computational costs.

Background

Many types of genomic data can be represented as a graph (network), where the nodes represent genes or proteins, and edges may represent similarities between protein sequences, edges in a metabolic pathway, and physical interactions between proteins [1]. Machine learning tools

have been commonly used to analyze biological networks for knowledge discovery and pattern analysis [2]. In this paper, we focus on learning from biological networks for protein function prediction. This problem has been studied extensively by using computational approaches recently [1]. Neighborhood-based methods [3,4] assign

functions to proteins based on the most frequent functions within a neighborhood of the protein and they differ mainly in how the "neighborhood" of a protein is defined. More sophisticated prediction functions have been exploited in [5,6]. Methods based on network diffusion [7,8] view the protein network as a flow network and functions of proteins are diffused from annotated proteins to their neighbors in various ways. Other approaches for protein function annotation from biological networks include the graph-cut-based approaches [9,10] and those derived from the kernel methods [11-13].

Kernel methods are versatile tools for learning from graph-based data, as they only require the characterization of similarities between objects by the use of kernel trick [2,14]. Diffusion kernels [15], which can be considered as the discretization of the well-known Gaussian kernel of Euclidean space, are commonly used for graph-based data. In kernel methods, the information on the data is conveyed only in the kernel function, which uniquely determines the mapping of the original inputs onto a feature space. Thus, one of the central issues in kernel methods is the selection of a good kernel function for a specific problem at hand. A recent trend in kernel learning (selection) is to formulate it as convex programs, which lead to a globally optimal solution [16]. The idea of learning a linear combination of pre-specified kernels for Support Vector Machines (SVM) was originally proposed in [17] where this problem was formulated as semidefinite programs (SDP) and Quadratically Constrained Quadratic Programs (QCQP). In general, approaches based on learning a convex combination of kernels offer the additional advantage of facilitating heterogeneous data integration from different sources [18].

The objective functions for kernel learning used in [17] are performance measures for hard margin SVM, 1-norm soft margin SVM, and 2-norm soft margin SVM. An alternative criterion for kernel matrix learning is the Kullback-Leibler (KL) divergence [19] between the two zero-mean Gaussian distributions defined by the input and output kernel matrices [20]. One particularly appealing feature of the KL divergence criterion is that unlabeled (test) data can be integrated naturally into the training process, thereby improving generalizations. The formulations in [17] also use unlabeled data, but in a weak form by enforcing the trace magnitude of the kernel matrix including both training and test data in the constraint. Direct incorporation of unlabeled data by the formulations in [17] through the KL divergence criterion involves a matrix determinant term. The resulting formulation is a so-called *maximum-determinant problem* [21], which is a general framework that contains semidefinite programming (SDP) [16] as a special case. Although its theoretical soundness, experiences with semidefinite programming

indicate that it is computationally expensive and thus can not be scaled to large-scale problems. The maximum-determinant problem is a more general framework than SDP and the path-following algorithms used to solve it is more expensive.

Diffusion kernels [15] capture the long-range relationships between vertices of graphs and are state-of-the-art for building kernels for graphs. In this paper, we focus on learning diffusion kernels constructed from biological networks, using the KL divergence criterion. In particular, we show that when the KL divergence criterion is used to optimize a convex combination of diffusion kernels with different parameters, the resulting optimization problem does not involve the matrix determinant term and thus can be solved by gradient descent methods. Previous studies [22,23] have shown that the removal of the matrix-determinant term in the KL divergence criterion has limited effect on its performance. When this modified criterion is used to learn a linear combination of diffusion kernels, the resulting optimization problem is convex and thus solutions by gradient descent methods are guaranteed to be globally optimal. A protein typically performs multiple functions. Most existing approaches formulate a separate task for each of the functions and they are learned independently. They decouple the functions of proteins and potentially compromise the performance as the functions of proteins are usually related. We show that our single-task kernel learning formulation based on the KL divergence criterion can be extended to the multi-task case by enforcing all tasks to share a common kernel. The resulting formulation leads to a single optimization problem, which learns multiple functions of proteins simultaneously. Experimental results show that this multiple-task kernel learning in a joint optimization framework keeps competitive prediction performance, while its computational cost is similar to that for a single task, thus dramatically reducing the time complexity.

Methods

We study the problem of protein function prediction from biological networks, which are represented as graphs. For a graph \mathcal{G} , the vertices represent proteins and edges characterize the relationship between proteins. In the following discussion, the vertex and edge sets are denoted as V and E , respectively. The total number of proteins in the network is $n = |V|$. The adjacency matrix A is used to denote the similarity between vertices where $A_{i,j}$ describes the similarity between vertices v_i and v_j . The functions of some proteins in the network are already known and the goal of protein function prediction is to infer the functions of unannotated proteins based on the functions of annotated proteins and the network topology. In particu-

lar, for a graph $\mathcal{G} = (V, E)$, the vertices in V can be partitioned into a training set and a test set. The functions of proteins in the training set are already known while those of proteins in the test set are unknown. Each edge in E reflects the local similarities between its ending vertices. The learning problem is to predict the functions of proteins in the test set based on the label information of training set and the topology of the graph.

Background and Related Work

Kernel methods are particularly suitable for learning from graph-based data, as they only require the similarities between proteins to be encoded in the kernel matrix. In kernel methods, a symmetric function $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow R$, where \mathcal{X} denotes the input space, is called a kernel function if it satisfies the Mercer's condition [14]. When used for a finite number of samples in practice, this condition can be stated as follows: for any $x_1, \dots, x_n \in \mathcal{X}$ the Gram matrix $K \in \mathbb{R}^{n \times n}$, defined by $K_{ij} = \kappa(x_i, x_j)$ is positive semidefinite. Any kernel function κ implicitly maps the input set \mathcal{X} to a high-dimensional (possibly infinite) Hilbert space \mathcal{H}_κ equipped with the inner product $(\cdot, \cdot)_{\mathcal{H}_\kappa}$ through a mapping $\phi_\kappa : \mathcal{X} \rightarrow \mathcal{H}_\kappa$

$$\kappa(x, z) = (\phi_\kappa(x), \phi_\kappa(z))_{\mathcal{H}_\kappa}. \tag{1}$$

The adjacency matrix A can't be directly used as a kernel matrix. First, the adjacency matrix contains the local similarity information only, which may not be effective for function prediction. Secondly, the adjacency matrix may not even be positive semidefinite. To derive a kernel matrix from the adjacency matrix, the idea of random walk and network diffusion has been used. The basic idea is to compute the global similarity between vertices v_i and v_j as the probability of reaching v_j at some time point T when the random walker starts from v_i . This idea is justified at least to some extent by observing that the random walker tends to meander around its origin as there is a larger number of paths of length $|T|$ to its neighbors than to remote vertices [2].

To avoid some potential problems such as the choice of value for T and assurance of positive semidefiniteness for the kernel matrix, a random walk with an infinite number of infinitesimally small steps is used instead. It can be formally described as:

$$K = \lim_{s \rightarrow \infty} \left(I + \frac{\beta L}{s} \right)^s = e^{\beta L}, \tag{2}$$

where β is a parameter for controlling the extent of diffusion and $L \in \mathbb{R}^{n \times n}$ is the graph Laplacian matrix defined as

$$L = \text{diag}(Ae) - A, \tag{3}$$

where A is the adjacency matrix, e is the vector of all ones, and $\text{diag}(Ae)$ is a diagonal matrix with the diagonal entries being the corresponding row summation of the matrix A . It turns out that for any symmetric matrix L , $e^{\beta L}$ is always positive definite and thus can be used as a kernel matrix. The diffusion effect of such kernel can be explicitly seen when it is expanded as [2]:

$$e^{\beta L} = I + \beta L + \frac{\beta^2}{2} L^2 + \frac{\beta^3}{6} L^3 + \dots, \tag{4}$$

where the local information encoded in L is continuously diffused by repeated multiplications. The parameter β in the diffusion kernel controls the extent of diffusion and it has a similar effect as the scaling parameter in Gaussian kernels. If the β is too small, the local information can not be diffused effectively, resulting in a kernel matrix that only captures local similarities. On the other hand, if it is too large, the neighborhood information will be lost. Furthermore, the optimal value for β is problem and data-dependent. Thus it is highly desirable to tune the β value adaptively from the data.

We approach the kernel tuning problem by learning an optimal kernel as a linear combination of pre-specified diffusion kernels constructed with different values of β . This is motivated from the work in [17] where the optimal kernel for SVM, in the form of a linear combination of pre-specified kernels, is learned based on the large margin criteria. In particular, the generalized performance measure based on 1-norm soft margin SVM used in [17] is

$$\omega_{S1}(K) = \max_{\alpha: C \geq \alpha \geq 0, \alpha^T \gamma = 0} \{2\alpha^T e - \alpha^T G(K)\alpha\}, \tag{5}$$

where $C > 0$ is the regularization parameter in SVM, e is the vector of all ones, $G(K)$ is defined by $G_{ij}(K) = k(x_i, x_j)y_i y_j$, and the i -th entry of γ denoted as y_i is the class label (1 or -1) of the i -th data point x_i . Lanckriet *et al.* [17] showed that when the optimal kernel is restricted to the linear combination of the given p kernels K_1, \dots, K_p , the kernel learning problem can be formulated as a semidefinite program. Furthermore, when the coefficients of the linear combination are constrained to be non-negative, the kernel learning problem can be formulated as a Quadratically Constrained Quadratic Program [16]. As was shown in [20], an alternative performance measure is the KL divergence between the two zero-mean Gaussian distributions associated with the input and output kernel matrices. We show that when this KL divergence criterion

is used to learn a linear combination of diffusion kernels constructed with different values of β , the resulting optimization problem can be solved efficiently. We further show that it can be extended to the multiple-task case. Such integration of multiple tasks into one optimization problem can potentially exploit the complementary information among different tasks.

Diffusion Kernel Learning: The Single-Task Case

We focus on learning an optimal kernel for a single task, which will then be extended to the multi-task case. The underlying idea is that the Laplacian matrix L , defined in Eq. (3), contains the connectivity information of all vertices in the graph. By adaptively tuning the kernel constructed from L on the training vertices, the entries corresponding to test vertices are expected to be tuned in some optimal way as well. To restrict the search space and improve the generalization ability, we focus on learning an optimal kernel as a linear combination of a set of diffusion kernels constructed with different values of β , indicating different extents of diffusion. In particular, we choose a sequence of values for β as β_1, \dots, β_p , and the corresponding diffusion kernels can be constructed as

$$K_i = e^{\beta_i L}, \quad i = 1, \dots, p. \tag{6}$$

We may assume that the kernels defined in Eq. (6) reflect our (weak) prior knowledge about the problem. The goal is to integrate the tuning of the coefficients into the learning process and the algorithm can adaptively select an optimal linear combination of the given kernels. Note that it is numerically favorable to normalize the kernels though this does not affect the results theoretically [14]. We normalize the kernels as follows:

$$\tilde{K}_i = \frac{e^{\beta_i L}}{\text{trace}(e^{\beta_i L})}, \tag{7}$$

and the optimal kernel can be represented as

$$K_{opt} = \sum_{i=1}^p \alpha_i \tilde{K}_i = \sum_{i=1}^p \alpha_i \frac{e^{\beta_i L}}{\text{trace}(e^{\beta_i L})}, \tag{8}$$

for a set of non-negative coefficients $\{\alpha_i\}_{i=1}^p$

Kullback-Leibler Divergence Formulation

Kernel matrices are positive semidefinite and thus can be used as the covariance matrices for Gaussian distributions. It was shown in [20] that the kernel matrix can be learned by minimizing the Kullback-Leibler (KL) divergence between the zero-mean Gaussian distributions associated with the input and output kernel matrices. In this paper, we focus on learning the optimal coefficients

α_i from the data automatically based on minimizing this KL divergence criterion. As described in [20], the KL divergence between the zero-mean Gaussian distributions defined by the input kernel K_x and output kernel K_y can be expressed as

$$\text{KL}(N_y | N_x) = \frac{1}{2} \text{trace}(K_y K_x^{-1}) + \frac{1}{2} \log |K_x| - \frac{1}{2} \log |K_y| - \frac{n}{2}, \tag{9}$$

where $|\cdot|$ denotes the matrix determinant, N_x and N_y denote the zero-mean Gaussian distributions associated with K_x and K_y , respectively, and n is the number of samples. When the output kernel K_y is defined as $K_y = \mathbf{y}\mathbf{y}^T$, the KL divergence in Eq. (9) can be expressed as

$$\text{KL}(N_y | N_x) = \frac{1}{2} \mathbf{y}^T K_x^{-1} \mathbf{y} + \frac{1}{2} \log |K_x| + \text{const}, \tag{10}$$

where "const" denotes terms that are independent of K_x , and K_x is the input kernel matrix, which is defined as a linear combination of the given p kernels as

$$K_x = \sum_{i=1}^p \alpha_i \tilde{K}_i + \lambda I = \sum_{i=1}^p \alpha_i \frac{e^{\beta_i L}}{\text{trace}(e^{\beta_i L})} + \lambda I. \tag{11}$$

Note that a regularization term, with λ as the regularization parameter, is added to Eq. (11) to deal with the singularity problem of kernel matrices as in [20], and we require $\sum_{i=1}^p \alpha_i = 1$ as in multiple kernel learning (MKL) [17]. The optimal coefficients $\alpha = [\alpha_1, \dots, \alpha_p]^T$ are computed by minimizing $\text{KL}(N_y | N_x)$. By substituting Eq. (11) into Eq. (10), and removing the constant term, we obtain the following optimization problem:

$$\begin{aligned} \min_{\alpha} & \left\{ a^T \left(\sum_{i=1}^p \alpha_i \tilde{K}_i + \lambda I \right)^{-1} a + \log \left| \sum_{i=1}^p \alpha_i \tilde{K}_i + \lambda I \right| \right\} \\ \text{s.t.} & \sum_{i=1}^p \alpha_i = 1, \\ & \alpha \geq 0, \end{aligned} \tag{12}$$

where $\alpha = (\alpha_1, \dots, \alpha_p)^T$, $\alpha \geq 0$ denotes that all components of α are non-negative, and the vector $a \in \mathbb{R}^n$ is the problem-specific target vector, corresponding to the general target in Eq. (9), defined as follows:

$$a_i = \begin{cases} 1 & \text{if } v_i \text{ is in the positive class,} \\ -1 & \text{if } v_i \text{ is in the negative class,} \\ 0 & \text{if } v_i \text{ is in the test set.} \end{cases} \quad (13)$$

Note that we assign the label 0 to vertices in the test set so that it will not bias towards either class. Similar idea has been used in [24] for semi-supervised learning. In multiple kernel learning [17], the sum-to-one constraint on the weights is enforced as in Eq. (12). We present results on both constrained and unconstrained formulations in the experiments. Results show that the constrained formulations achieved better performance than the unconstrained ones.

Recall that the graph Laplacian matrix L is symmetric, so its eigen-decomposition can be expressed as

$$L = PDP^T,$$

where

$$D = \text{diag}(d_1, \dots, d_n) \quad (14)$$

is the diagonal matrix of eigenvalues and $P \in \mathbb{R}^{n \times n}$ is the orthogonal matrix of corresponding eigenvectors. According to the definition of the function of matrices [25], we have

$$e^{\beta_i L} = PD_i P^T, \quad (15)$$

where

$$D_i = \text{diag}(e^{\beta_i d_1}, \dots, e^{\beta_i d_n}). \quad (16)$$

The main result is summarized in the following theorem:

Theorem 1. *Given a set of p diffusion kernels, as defined in Eq. (7), the problem of learning the optimal kernel matrix, in the form of a convex combination of the given p kernel matrices as in Eq. (12), can be formulated as the following optimization problem:*

$$\min_{\alpha} \sum_{j=1}^n \left(\frac{b_j^2}{g_j} + \log(g_j) \right) \quad (17)$$

$$\text{subject to } \sum_{i=1}^p \alpha_i = 1, \quad (18)$$

$$\alpha \geq 0, \quad (19)$$

where $b = (b_1, \dots, b_n) = P^T a$, g_j is the j -th diagonal entry of the diagonal matrix G , defined as

$$G = \sum_{i=1}^p \alpha_i \frac{D_i}{\text{trace}(D_i)} + \lambda I, \quad (20)$$

and D_i is the diagonal matrix defined in Eq. (16).

Proof. The first term in Eq. (12) can be written as:

$$\begin{aligned} a^T \left(\sum_{i=1}^p \alpha_i \tilde{K}_i + \lambda I \right)^{-1} a &= a^T \left(\sum_{i=1}^p \alpha_i \frac{e^{\beta_i L}}{\text{trace}(e^{\beta_i L})} + \lambda I \right)^{-1} a \\ &= a^T P \left(\sum_{i=1}^p \alpha_i \frac{D_i}{\text{trace}(e^{\beta_i L})} + \lambda I \right)^{-1} P^T a \\ &= b^T \left(\sum_{i=1}^p \alpha_i \frac{D_i}{\text{trace}(D_i)} + \lambda I \right)^{-1} b \\ &= b^T G^{-1} b = \sum_{j=1}^n \frac{b_j^2}{g_j}, \end{aligned} \quad (21)$$

where the third equality follows from the property of the trace, that is,

$$\text{trace}(e^{\beta_i L}) = \text{trace}(PD_i P^T) = \text{trace}(D_i).$$

Similarly, the second term in Eq. (12) can be written as:

$$\begin{aligned} \log \left| \sum_{i=1}^p \alpha_i \tilde{K}_i + \lambda I \right| &= \log \left| \sum_{i=1}^p \alpha_i \frac{e^{\beta_i L}}{\text{trace}(e^{\beta_i L})} + \lambda I \right| \\ &= \log \left| \sum_{i=1}^p \alpha_i \frac{e^{\beta_i D}}{\text{trace}(e^{\beta_i D})} + \lambda I \right| \\ &= \log |G| \\ &= \log \left(\prod_{j=1}^n g_j \right) \\ &= \sum_{j=1}^n \log(g_j). \end{aligned} \quad (22)$$

By combining the first term in Eq. (21) and the second term in Eq. (22), we prove the theorem.

The formulation in Theorem 1 is a nonlinear optimization problem. It involves a nonlinear objective function with p variables and linear equality and inequality con-

straints. Due to the presence of the log term in the objective, it is a non-convex problem and a globally optimal solution may not exist. However, our experimental results show that this formulation consistently produces superior performance.

Convex Formulation

The optimization problem in Theorem 1 is not convex. Previous studies [22,23] indicate that the removal of the log determinant term in the KL divergence criterion in Eq. (12) has a limited effect on the performance. This leads to the following optimization problem:

$$\min_{\alpha} a^T \left(\sum_{j=1}^n \alpha_j \tilde{K}_j + \lambda I \right)^{-1} a \quad (23)$$

$$\text{subject to } \sum_{i=1}^p \alpha_i = 1, \quad (24)$$

$$\alpha \geq 0. \quad (25)$$

Following Theorem 1, we can show that the optimization problem above can be simplified as

$$\min_{\alpha} \sum_{j=1}^n \frac{b_j^2}{g_j} \quad (26)$$

$$\text{subject to } \sum_{i=1}^p \alpha_i = 1,$$

$$\alpha \geq 0.$$

where g_j and b are defined as in Theorem 1.

The optimization problem in Eq. (26) is convex and thus a globally optimal solution exists. Numerical experiments indicate that the simple gradient descent algorithm converges very quickly to the optimal solution. Furthermore, the prediction performance of this convex formulation is comparable to that of the formulation proposed in Theorem 1. This convex formulation shares some similarities with the one in [26], where a set of Laplacian matrices derived from multiple networks is combined.

Diffusion Kernel Learning: The Multi-Task Case

It is known that proteins often perform multiple functions, which are typically related. Many existing function prediction approaches decouple multiple functions and formulate each function prediction problem as a separate binary-class classification problem. Such methods do not

consider the relationship among the multiple functions of a protein and potentially compromise the overall performance.

We propose to extend our formulation for the single-task case to deal with multiple tasks simultaneously. In particular, we formulate a single optimization problem for the simultaneous prediction of multiple functions for a protein. The joint learning of multiple functions can potentially exploit the relationship among functions and improve the performance. In terms of computational complexity, the proposed joint optimization problem is shown to be comparable to that of the single-task formulation.

A key observation is that when the pre-specified diffusion kernels are computed from the same biological network with different values of β , the graph Laplacian matrices are the same for all tasks. By enforcing all tasks to share a common linear combination of kernels, we obtain the following joint optimization problem:

$$\min_{\alpha} \sum_{k=1}^t (a^{(k)})^T \left(\sum_{i=1}^p \alpha_i \tilde{K}_i + \lambda I \right)^{-1} a^{(k)} + t \log \left| \sum_{i=1}^p \alpha_i \tilde{K}_i + \lambda I \right| \quad (27)$$

$$\text{subject to } \sum_{i=1}^p \alpha_i = 1, \quad (28)$$

$$\alpha \geq 0, \quad (29)$$

where $a^{(k)} \in \mathbb{R}^n$ for $i = 1, \dots, t$ is the vector of class labels for the k -th task as in Eq. (13), and t is the number of tasks. Note that all t tasks are related in this joint formulation by enforcing a common kernel matrix for all tasks. The objective function in Eq. (27) uses an equal weight for all tasks. If some tasks are known to be more important than others, a more general objective function with varying weights for different tasks may be used instead. Following Theorem 1, we can simplify the optimization problem in Eq. (27), as summarized in the following theorem:

Theorem 2. *Given a set of p diffusion kernels, as defined in Eq. (7), the problem of optimal multi-task kernel learning, in the form of a convex combination of the given p kernels, can be formulated as the following optimization problem:*

$$\min_{\alpha} \sum_{k=1}^t \sum_{j=1}^n \frac{b_k^2(j)}{g_j} + t \sum_{j=1}^n \log(g_j) \quad (30)$$

$$\text{subject to } \sum_{i=1}^p \alpha_i = 1, \tag{31}$$

$$\alpha \geq 0, \tag{32}$$

where g_j is defined as in Theorem 1, $b_k = P^T a^{(k)}$, $a^{(k)}$ is defined as in Eq. (13) for the k -th task, and t is the total number of tasks.

Proof. The first term in Eq. (27) can be rewritten as

$$\begin{aligned} \sum_{k=1}^t \left(a^{(k)T} \left(\sum_{i=1}^p \alpha_i \tilde{K}_i + \lambda I \right)^{-1} a^{(k)} \right) &= \sum_{k=1}^t \left(b_k^T \left(\sum_{i=1}^p \alpha_i \frac{D_i}{\text{trace}(D_i)} + \lambda I \right)^{-1} b_k \right) \\ &= \sum_{k=1}^t (b_k^T G^{-1} b_k) = \sum_{k=1}^t \sum_{j=1}^n \frac{b_k^2(j)}{g_j}. \end{aligned}$$

Similarly, the second term can be rewritten as

$$t \log \left| \sum_{i=1}^p \alpha_i \tilde{K}_i + \lambda I \right| = t \sum_{j=1}^n \log(g_j). \tag{33}$$

The detailed intermediate steps of derivation are the same as those in the proof of Theorem 1 and thus are omitted. By combining these two terms together, we prove the theorem.

The optimization problem in Theorem 2 is not convex. Similar to the single-task case, the log determinant term in Eq. (27) may be removed, which leads to the following convex optimization problem:

$$\min_{\alpha} \sum_{k=1}^t \sum_{j=1}^n \frac{b_k^2(j)}{g_j} \tag{34}$$

$$\text{subject to } \sum_{i=1}^p \alpha_i = 1, \tag{35}$$

$$\alpha \geq 0. \tag{36}$$

Experimental evidences show that this convex optimization problem is comparable to the formulation in Theorem 2 in prediction performance.

Results and Discussion

We evaluate the performance of the proposed formulations on two benchmark data sets, and compare them with relevant methods, including the Neighbor Counting approach [4] and the FS-Weighted Averaging approach

[5,6]. We construct 60 diffusion kernels from each data set using different values for β and the proposed formulations are applied to compute a linear combination of the pre-computed kernels. The performance of the obtained kernel is compared with that of the individual kernel. To see the relative performance of the objective functions, we also use the 1-norm soft margin SVM criterion, proposed in [17], to compute the linear combination of kernels and the results are presented. All of the formulations proposed in this paper are solved using the MATLAB [27] function *fmincon* which employs the sequential quadratic programming method [28]. The QCQP formulation for optimizing the 1-norm soft margin SVM criterion is solved using the MOSEK [29] software package. After the kernels are computed, they are fed into SVM for classification and the LIBSVM [30] software package is used in the experiments. All of the experiments are performed on a PC with Intel Pentium D 820 2.8G CPU and 2G RAM.

In the following experiments, a total of 60 diffusion kernels are pre-computed and the values of β used are $\beta_i = 0.1 \times i$, for $i = 1, \dots, 60$. In order to investigate the performance of each individual kernel, we use each kernel for the classification and compute the average Receiver Operating Characteristic (ROC) values over all of the tasks. The ROC value produced by the best averaged individual kernel is used as a baseline. It is called rBaseline as all tasks are *restricted* to use the same kernel. We further relax the requirement that all tasks use the same kernel and compute the sequence of ROC values achieved by the best individual kernel for each of the tasks. This is considered another baseline called uBaseline as the kernel used by each task is *unrestricted*. Note that the kernel matrices for both rBaseline and uBaseline represent the single best candidate kernel in the ideal case that the labels of test data are known, and their performance is not guaranteed in practice. In contrast, the kernel matrices computed by the proposed formulations are the optimal kernel matrices in the form of linear combination of the given candidate kernel matrices. In order to evaluate the effectiveness of the weights obtained by the proposed formulations, we assign each kernel the same weight and compute the performance of the combined kernel. It is called eBaseline as all kernel matrices have an *equal* weight.

For convenience of presentation, the formulations proposed in Theorem 1, Eq. (26), Theorem 2, and Eq. (34) are denoted as DKL_{KL} , DKL , $mDKL_{KL}$, and $mDKL$, respectively. For DKL_{KL} and $mDKL_{KL}$, we also propose to remove the constraints in their optimization problems and the resulting formulations are denoted as DKL_{KL}^u and $mDKL_{KL}^u$, respectively. (See the caption of Table 1 for detailed description.) The method based on optimizing 1-

Table 1: Summary of the proposed formulations. DKL_{KL}^u

Algorithm	Single task	Multiple tasks	Convexity	Constraint
DKL	✓		✓	✓
DKL_{KL}	✓			✓
DKL_{KL}^u	✓			
mDKL		✓	✓	✓
$mDKL_{KL}$		✓		✓
$mDKL_{KL}^u$		✓		

These formulations are categorized in terms of the number of tasks, convexity, and whether the weights on kernels are constrained.

DKL_{KL} , DKL, and DKL_{KL}^u denote formulations using the Kullback-Leibler (KL) divergence criterion, the KL divergence criterion with the log term removed and the unconstrained version for single task, respectively. A lower case m is added before each method to denote the corresponding formulations for multiple tasks.

norm soft margin SVM criterion by solving QCQP proposed in [17] is denoted as SM1. The six proposed formulations are summarized in Table 1.

Experiments on the Ligand Data Set

The Ligand data set was derived by Vert and Kanehisa [31] from the Ligand database of chemical reactions in biological pathways [32]. It contains a graph reflecting the interactions between proteins and the function information for them. The graph is a yeast biological network in which a path between vertices implies a possible series of reactions catalyzed by proteins along it. The numbers of vertices and edges in this graph are 753 and 7860, respectively. For the functions of proteins, the functional categories of the MIPS Comprehensive Yeast Genome Database (CYGD) [33] are considered as the gold standard. These categories are not mutually exclusive, and each protein may have multiple functions. There are 36 different functions considered for this data set.

Comparison of ROC Values

We use the ROC as the performance measure and the λ value is fixed to 10^{-6} in the experiments. Our experimental results show that the algorithms are not sensitive to the value of λ , as long as it is neither too large nor too small. Figure 1 plots the number of tasks with ROC value above a threshold for all methods. The average ROC values achieved by all methods are also summarized in Table 2. In order to test statistical significance, we also compute the p -values of Wilcoxon signed test and the results are reported in Table 3. We can observe that mDKL achieves the best performance among all methods. All the pro-

posed formulations except $mDKL_{KL}^u$ outperform the three baseline methods. This implies that the computed linear combination of kernels can potentially exploit the complementary information in different kernels and thus improve performance. The ROC value achieved by SM1 is lower than those of the three baseline methods, implying that the SVM criterion is less effective for such tasks. Note that the SM1 criterion also uses information from unlabeled data, but in a weak form. The $mDKL_{KL}^u$ formulation achieves a ROC value lower than the three baseline methods. This shows that the constraints have important normalizing effects and can not be removed. By comparing the relative performance of formulations with and without the log term, we can conclude that removing this term usually does not affect the performance. Another important observation is that mDKL and $mDKL_{KL}$ outperform DKL and DKL_{KL} , implying that constraining the multiple tasks to share a common kernel does not degrade the performance if the kernel used is a linear combination of kernels obtained by the proposed formulations. In contrast, if the kernel used is a single kernel, this restriction will degrade the performance, as illustrated by the relative performance of rBaseline and uBaseline. For the eBaseline method, it can be observed that, except for $mDKL_{KL}^u$, all of other proposed formulations outperform it. This illustrates that our formulations can compute an optimal kernel matrix by assigning different weights to the candidate kernel matrices. We can observe from Table 3 that the difference between the performance of the two baseline methods (rBaseline and eBaseline) and that of DKL and mDKL are statistically significant. All diffusion kernel based approaches are competitive with the Neighbor Counting approach [4] and the FS-Weighted Averaging approach [5,6]. Neighbor Counting and FS-Weighted Averaging use the local information, more specifically the level-1 neighborhood (Neighbor Counting) and both level-1 and level-2 neighborhoods (FS-Weighted Averaging), for the prediction. The experimental results show the effectiveness of capturing the long-range relationships (global information) between proteins in the network in diffusion kernels [15].

Figure 2 plots the average ROC values for the 60 kernels (the maximum mean ROC value is used in rBaseline) and Figure 3 plots the best ROC values for the 36 tasks. We can observe that for tasks 29 and 33, the best ROC values are small. This implies that all the kernels perform poorly for these two tasks. To illustrate the relative performance of the proposed formulations with that of the baseline

Table 2: Mean ROC values and execution time (in seconds) of various methods on the Ligand Data Set.

Algorithm	Mean ROC	Time
uBaseline	0.8346	----
rBaseline	0.8223	----
eBaseline	0.8267	----
DKL	0.8523	165.52
DKL _{KL}	0.8523	665.13
	0.8365	4406.64
DKL ^u _{KL}		
mDKL	0.8542	5.61
mDKL _{KL}	0.8538	12.33
	0.8213	62.80
mDKL ^u _{KL}		
Neighbor Counting	0.7010	----
FS-Weighted Averaging	0.7785	----
SM1	0.8162	739.69

rBaseline denotes the baseline method that requires all tasks share a common kernel, uBaseline corresponds to the baseline method without this requirement, and eBaseline denotes the baseline in which all of the candidate kernel matrices are assigned the same weight.

method graphically, we plot in Figure 4 the ROC values obtained by the proposed formulations with respect to uBaseline using scatter plots. We can observe that there are two points below the 45-degree line in each plot. Those two points correspond to tasks 29 and 33 and they are difficult to classify by all methods. As most points in the plots are above the 45-degree line, we can conclude that the proposed formulations outperform uBaseline on most tasks.

Table 3: p-values obtained from Wilcoxon signed test comparing DKL and mDKL with other formulations for the Ligand data set.

Algorithm	DKL	mDKL
uBaseline	1.456E-2	5.690E-3
rBaseline	8.188E-4	2.091E-4
eBaseline	7.524E-5	2.548E-5
DKL	1.000	1.249E-2
DKL _{KL}	3.217E-1	1.279E-2
	1.112E-4	3.852E-5
DKL ^u _{KL}		
mDKL	1.249E-2	1.000
mDKL _{KL}	1.404E-1	6.849E-2
	3.852E-5	7.013E-6
mDKL ^u _{KL}		

For the Ligand data set, each algorithm produces an ROC vector consisting of ROC values over each task. We use Wilcoxon signed test to test the difference of the paired data. The null hypothesis of this test is that the ROC vectors produced by the two compared algorithms have the common median. The numbers reported in this table are the p-values that represent the probabilities that the null hypothesis is true. Typically, the null hypothesis is rejected if $p < 0.05$.

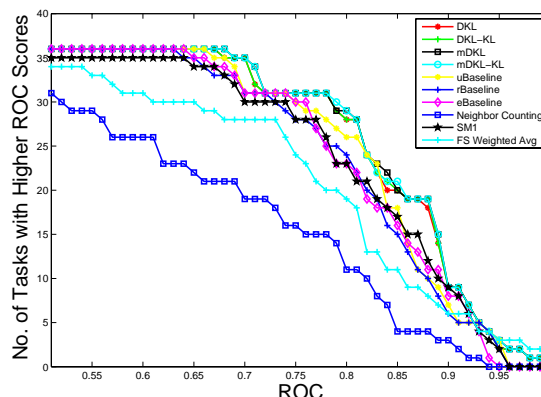


Figure 1 Comparison of ROC values for various algorithms on the Ligand Data Set. The horizontal axis represents the ROC values and the vertical axis is the number of tasks with ROC values above the corresponding horizontal axis value.

Comparison of Execution Time

In order to compare the efficiency of various kernel learning methods, we list in Table 2 the execution time of the compared methods. It can be observed that all methods based on multiple tasks are more efficient than their single-task counterparts. In particular, the execution time of mDKL is roughly 1/36 of that of DKL, which is consistent with our theoretical analysis. In general, convex formulations are more efficient than their non-convex original

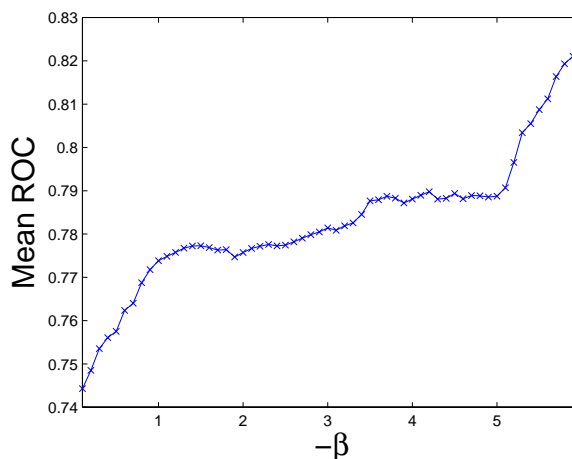


Figure 2 Mean ROC values over 36 tasks for each kernel on the Ligand Data Set (the kernel with the maximum mean ROC value is used in rBaseline). The horizontal axis denotes the $-\beta$ values used to build the corresponding kernel and the vertical axis is the mean ROC value.

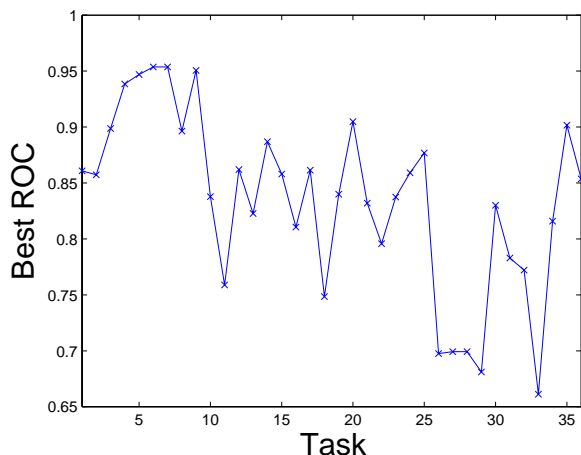


Figure 3
Best ROC values for tasks achieved by the best kernel (uBaseline) on the Ligand Data Set. The horizontal axis represents the tasks and the vertical axis is the corresponding best ROC value.

formulations and the optimization problems with the constraints removed take a longer time to converge. By taking the performance into account, the DKL and mDKL may be the best choices in practice.

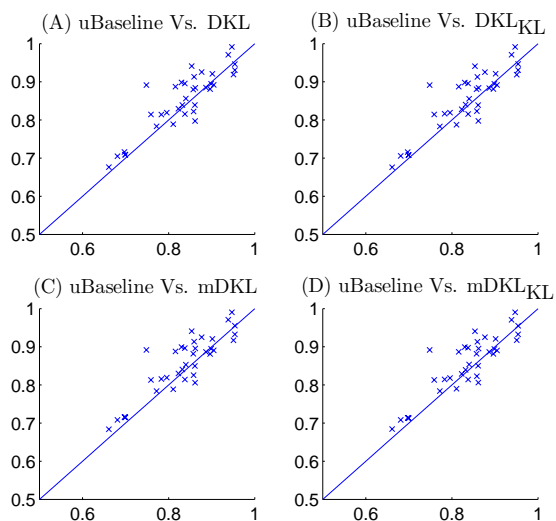


Figure 4
Comparison of the relative performance of the proposed formulations with that of uBaseline on the Ligand Data Set. The horizontal axis represents uBaseline and the vertical axis corresponds to DKL, DKL_{KL}, mDKL, mDKL_{KL}. Each point in the scatter plots corresponds to ROC values produced by the compared methods on the same task.

Stability Test

In order to obtain a robust performance estimate for the various methods, we randomly partition the data set into a training set and a test set ten times and the average ROC values and standard deviations across splittings are reported in Table 4. Compared with the results in Table 2, we can see that the relative performance of each method in these two tables is very similar. In particular, mDKL and mDKL_{KL} achieve the best overall performance. Except for the two unconstrained formulations DKL^u_{KL} and mDKL^u_{KL}, all of other proposed formulations achieve higher ROC values than the three baseline methods. It is worth noting that the performance of uBaseline and rBaseline is obtained by using the labels of both the training and test data and such performance is not guaranteed in practice when only the labels of the training data are used.

Experiments on the von Mering Data Set

The von Mering data set was created by von Mering *et al.* [34] from protein-protein interactions identified via six different methods. It contains a graph consisting of 2617 vertices (proteins) and 11855 edges. There are 76 different functions (tasks) associated with the proteins in the graph. The performance of different methods is reported in Figure 5. Two baseline methods, rBaseline and uBaseline, constructed exactly the same way as those for the Ligand data set are used and their performance is summarized in Figure 6 and Figure 7, respectively. The value for is again set to 10⁻⁶ in the experiments. Figure 8 compares the relative performance of the proposed formulations with that of the uBaseline graphically.

Table 4: Average ROC values and the corresponding standard deviations over 11 splittings on the Ligand Data Set. One of the splittings was specified by the contributor of the data and the remaining ten splittings are randomly generated.

Algorithm	Average ROC	Standard deviation
uBaseline	0.8326	0.0064
rBaseline	0.8181	0.0062
eBaseline	0.8193	0.0070
DKL	0.8493	0.0034
DKL _{KL}	0.8493	0.0032
	0.8318	0.0061
DKL ^u _{KL}		
MDKL	0.8507	0.0033
mDKL _{KL}	0.8507	0.0033
	0.8209	0.0031
mDKL ^u _{KL}		

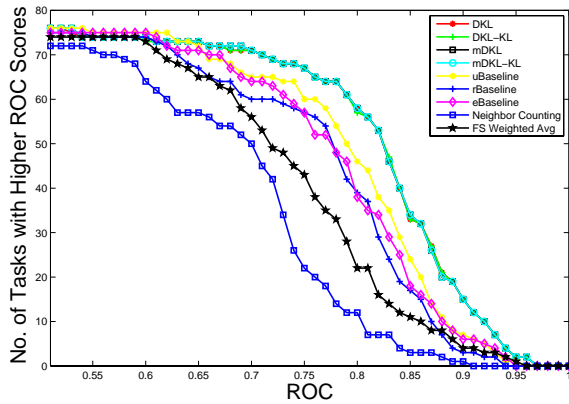


Figure 5
Comparison of ROC values for various algorithms on the von Mering Data Set. The horizontal axis represents the ROC values and the vertical axis is the number of tasks with ROC values above the corresponding horizontal axis value.

Comparison of ROC Values

We use the ROC values of each method to compare their relative performance. Similar to Figure 1 for the Ligand data set, Figure 5 plots the change of the number of tasks with ROC value above a certain threshold as the threshold varies for each of the compared method. For ease of comparison, Table 5 also lists the average ROC values achieved by the compared methods. Similarly, the *p*-values of Wilcoxon signed test for this data set are reported in Table 6. As the SM1 formulation requires excessive storage and

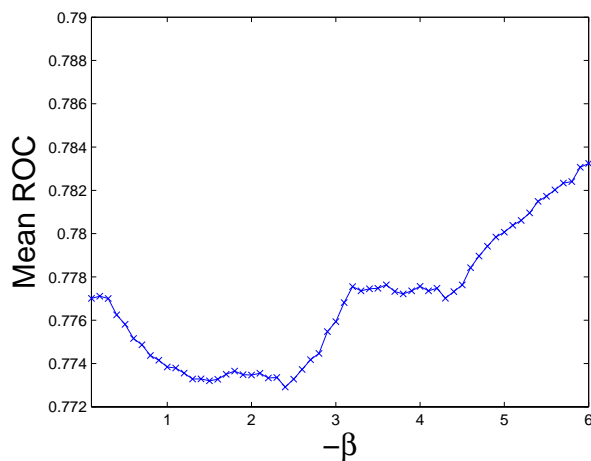


Figure 6
Mean ROC values over 76 tasks for each kernel on the von Mering Data Set. The horizontal axis denotes the $-\beta$ values used to build the corresponding kernel and the vertical axis is the mean ROC values.

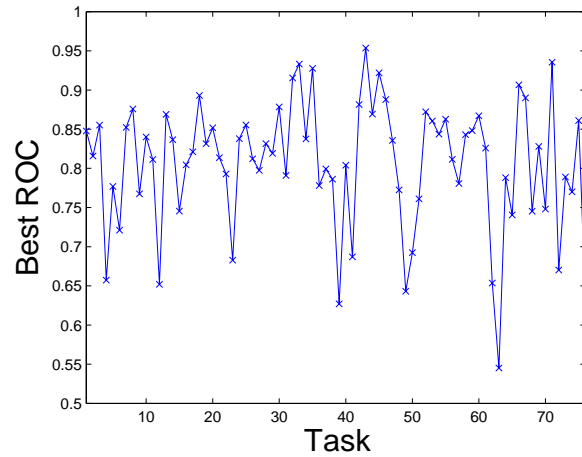


Figure 7
Best ROC values for different tasks achieved by different kernels on the von Mering Data Set. The horizontal axis represents the tasks and the vertical axis is the corresponding best ROC values.

computational time for this relatively large data set, we are not able to obtain its result in this experiment. From these results we can observe that mDKL and mDKL_{KL} achieve the best performance. In general, the performance of DKL,

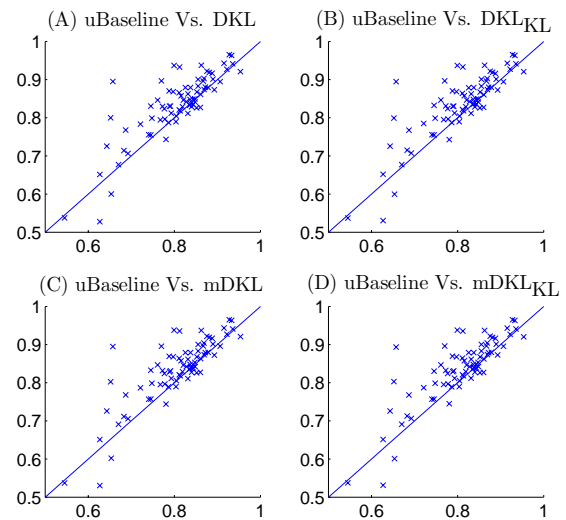


Figure 8
Comparison of the relative performance of the proposed formulations with that of uBaseline on the von Mering Data Set. The horizontal axis represents uBaseline and the vertical axis corresponds to DKL, DKL_{KL}, mDKL, mDKL_{KL}. Each point in the scatter plots corresponds to ROC values produced by the compared methods on the same task.

DKL_{KL} , $mDKL$, and $mDKL_{KL}$ is very close. All of the proposed formulations except DKL_{KL}^u perform better than the three baseline methods. The difference between DKL and DKL_{KL} as well as the difference between $mDKL$ and $mDKL_{KL}$ is very small, which further confirms that the removal of the log term does not affect the performance of algorithm much. For the formulations with constraints removed, i.e., DKL_{KL}^u and $mDKL_{KL}^u$, their performance is the lowest among the proposed formulations. Similar to the case for the Ligand data set, we conclude that constraining the multiple tasks to share a common kernel does not degrade the performance if the kernel used is a linear combination of kernels obtained by the proposed formulations. In contrast, if the kernel used is a single kernel, this restriction will degrade the performance, as illustrated by the relative performance of $rBaseline$ and $uBaseline$. In terms of the $eBaseline$, we can observe from Table 5 that all of our proposed formulations achieve higher ROC values than the $eBaseline$ method, in which all of the kernel matrices are assigned the same weight. We can observe from Table 6 that the difference between the performance of all of the three baselines and that of DKL and $mDKL$ is statistically significant. We can again observe that all diffusion kernel based approaches are competitive with the Neighbor Counting approach and the FS-Weighted Averaging approach.

Figure 8 presents the scatter plots of four proposed formulations with respect to $uBaseline$. It can be observed that most points are above the 45-degree line, which implies that the linear combination of kernels is better than the

Table 5: Mean ROC values and execution time (in seconds) of various methods on the von Mering Data Set.

Algorithm	Mean ROC	Time
$uBaseline$	0.8061	----
$rBaseline$	0.7832	----
$eBaseline$	0.7945	----
DKL	0.8339	3441.13
DKL_{KL}	0.8340	7445.67
	0.7968	35384.84
DKL_{KL}^u		
$mDKL$	0.8345	54.64
$mDKL_{KL}$	0.8345	67.52
	0.8129	151.49
$mDKL_{KL}^u$		
Neighbor Counting	0.7076	----
FS-Weighted Averaging	0.7544	----
SMI	----	----

Table 6: p-values obtained from Wilcoxon signed test comparing DKL and mDKL with other formulations for the von Mering data set.

Algorithm	DKL	mDKL
$uBaseline$	5.849E-7	2.966E-7
$rBaseline$	1.510E-10	1.038E-10
$eBaseline$	4.268E-10	2.863E-10
DKL	1.000	6.531E-2
DKL_{KL}	3.776E-1	1.669E-1
	6.870E-11	3.193E-11
DKL_{KL}^u		
$mDKL$	6.531E-2	1.000
$mDKL_{KL}$	5.971E-2	1.637E-1
	3.193E-11	8.252E-12
$mDKL_{KL}^u$		

For the von Mering data set, each algorithm produces an ROC vector consisting of ROC values over each task. We use Wilcoxon signed test to test the difference of the paired data. The null hypothesis of this test is that the ROC vectors produced by the two compared algorithms have the common median. The numbers reported in this table are the p-values that represent the probabilities that the null hypothesis is true. Typically, the null hypothesis is rejected if $p < 0.05$.

ideally best individual kernel. In general, the performance of DKL_{KL} , DKL , $mDKL_{KL}$, $mDKL$ is better than $uBaseline$. And this is also confirmed by the mean ROC values listed in Table 5.

Comparison of Execution Time

Table 5 also lists the execution time of various kernel learning methods. Similar conclusions can be drawn from this table as to the execution time on the Ligand data set. All methods based on multiple tasks are more efficient than their single-task counterparts. By comparing the results in Table 2 and Table 5 we can also observe that as the number of tasks increases, the time difference between methods based on multiple tasks and those based on single tasks increases too. Thus, the formulations based on multiple tasks are preferred when the number of tasks is large.

Stability Test

Similar to the Ligand data set, we generate ten random splittings of the data into training and test sets and report the average ROC values and standard deviations in Table 7. By comparing with results in Table 5, we can see that the relative performance of each method is similar in both tables. All of the proposed formulations outperform $eBaseline$.

Conclusion

In this paper, we address the issue of learning an optimal diffusion kernel based on KL divergence criterion for protein function prediction. By exploiting the special structure of the diffusion kernel, we show that this KL

Table 7: Average ROC values and the corresponding standard deviations over 11 splittings on the von Mering Data Set. One of the splittings was specified by the contributor of the data and the remaining ten splittings are randomly generated.

Algorithm	Average ROC	Standard deviation
uBaseline	0.8078	0.0045
rBaseline	0.7863	0.0064
eBaseline	0.7909	0.0078
DKL	0.8398	0.0042
DKL _{KL}	0.8398	0.0042
	0.7991	0.0059
DKL ^u _{KL}		
mDKL	0.8402	0.0042
mDKL _{KL}	0.8402	0.0042
	0.8194	0.0046
mDKL ^u _{KL}		

divergence based kernel learning problem can be formulated as a simple optimization problem, which can be solved efficiently. We also extend the formulation to the multi-task case where we predict multiple functions of a protein simultaneously.

We have conducted experiments on two benchmark data sets. Our results show that the performance of linearly combined diffusion kernel is better than every single candidate diffusion kernel. Results also show that the removal of the log term in the KL divergence criterion does not degrade its recognition performance, while it leads to a reduced computational cost. When the number of tasks is large, the algorithms based on multiple tasks are favored due to their competitive recognition performance and small computational costs. One possible extension is to incorporate the learning of the regularization parameter in the proposed formulations as in [17]. The difference between the proposed learning framework and those in [17] is that our formulations require that the eigenvectors of the candidate kernel matrices to be the same. Thus the proposed formulations may not be applied for heterogeneous data integration. We plan to apply the proposed algorithms for the analysis of other graph-based biological data.

Authors' contributions

LS designed the methodology, implemented programs, and participated in manuscript preparation. SJ derived the KL divergence formulation, and drafted the manuscript. JY originally conceived the project, guided the implementation, and drafted the manuscript. All authors have read and approved the final manuscript.

Acknowledgements

This research is sponsored in part by the Arizona State University and by the National Science Foundation under Grant No. IIS-0612069.

References

- Pandey G, Kumar V, Steinbach M: **Computational Approaches for Protein Function Prediction: A Survey.** In *Tech Rep TR 06-028, Department of Computer Science and Engineering University of Minnesota, Twin Cities, MN; 2006.*
- Schölkopf B, K T, JP V: *Kernel Methods in Computational Biology* Cambridge, MA: MIT Press; 2004.
- Hishigaki H, Nakai K, Ono T, Tanigami A, Takagi T: **Assessment of prediction accuracy of protein function from protein-protein interaction data.** *Yeast* 2001, **18**:523-531.
- Schwikowski B, Uetz P, Fields S: **A network of protein-protein interactions in yeast.** *Nature Biotechnology* 2000, **18**:1257-1261.
- Chua HN, Sung WK, Wong L: **Exploiting Indirect Neighbours and Topological Weight to Predict Protein Function from Protein-Protein Interactions.** *Bioinformatics* 2006, **22**:1623-1630.
- Chua HN, Sung WK, Wong L: **Using Indirect Protein Interactions for the Prediction of Gene Ontology Functions.** *BMC Bioinformatics* 2007, **8**:S8.
- Nabieva E, Jim K, Agarwal A, Chazelle B, Singh M: **Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps.** *Bioinformatics* 2005, **21**:302-310.
- Weston J, Elisseeff A, Zhou D, Leslie CS, Noble WS: **Protein ranking: From local to global structure in the protein similarity network.** *Proc Natl Acad Sci* 2004, **101**:6559-6563.
- Vazquez A, Flammini A, Maritan A: **Global protein function prediction from protein-protein interaction networks.** *Nature Biotechnology* 2003, **21**:697-700.
- Karaoz U, Murali TM, Letovsky S, Zheng Y, Ding C, Cantor CR, Kasif S: **Whole-genome annotation by using evidence integration in functional-linkage networks.** *Proc Natl Acad Sci* 2004, **101**:2888-2893.
- Ben-Hur A, Noble WS: **Kernel methods for predicting protein protein interactions.** *Bioinformatics* 2005, **21(Suppl 1)**:i38-i46.
- Roth V, Fischer B: **Improved functional prediction of proteins by learning kernel combinations in multilabel settings.** *BMC Bioinformatics* 2007, **8**:S12.
- Tsuda K, Noble WS: **Learning kernels from biological networks by maximizing entropy.** *Bioinformatics* 2004, **20**:326-333.
- Schölkopf B, Smola AJ: *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* Cambridge, MA: MIT Press; 2002.
- Kondor RI, Lafferty JD: **Diffusion Kernels on Graphs and Other Discrete Structures.** *ICML 2002*:315-322.
- Boyd S, Vandenberghe L: *Convex Optimization* Cambridge: Cambridge University Press; 2004.
- Lanckriet G, Cristianini N, Bartlett P, Ghaoui LE, Jordan MI: **Learning the Kernel Matrix with Semidefinite Programming.** *Journal of Machine Learning Research* 2004, **5**:27-72.
- Lanckriet G, Bie TD, Cristianini N, Jordan M, Noble W: **A statistical framework for genomic data fusion.** *Bioinformatics* 2004, **20**:2626-2635.
- Kullback S, Leibler RA: **On Information and Sufficiency.** *Annals of Mathematical Statistics* 1951, **22**:79-86.
- Lawrence ND, Sanguinetti G: **Matching kernels through Kullback-Leibler divergence minimisation.** In *Technical Report CS-04-12, Department of Computer Science The University of Sheffield; 2004.*
- Vandenberghe L, Boyd S, Wu S: **Determinant Maximization with Linear Matrix Inequality Constraints.** *SIAM Journal on Matrix Analysis and Applications* 1998, **19**:499-533.
- Smola AJ, Bartlett PL: **Sparse greedy Gaussian process regression.** *NIPS* 2001:619-625.
- Smola AJ, Schölkopf B: **Sparse greedy matrix approximation for machine learning.** *ICML 2000*:911-918.
- Zhou D, Bousquet O, Lal TN, Weston J, Schölkopf B: **Learning with Local and Global Consistency.** *NIPS* 2004:321-328.
- Golub GH, Van Loan CF: *Matrix Computations* 3rd edition. Baltimore, MD: The Johns Hopkins University Press; 1996.
- Tsuda K, Shin H, Schölkopf B: **Fast protein classification with multiple networks.** *Bioinformatics* 2005, **21**:59-65.
- The Matlab Package** [<http://www.mathworks.com>]
- Nocedal J, Wright S: *Numerical Optimization* 2nd edition. New York: Springer; 2006.
- The MOSEK Package** [<http://www.mosek.com>]
- Chang CC, Lin CJ: *LIBSVM: a library for support vector machines* 2001 [<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>].

31. Vert JP, Kanehisa M: **Graph-Driven Feature Extraction From Microarray Data Using Diffusion Kernels and Kernel CCA.** *NIPS* 2003:1425-1432.
32. **The Ligand data set** [<http://www.genome.ad.jp/ligand/>]
33. **The MIPS Comprehensive Yeast Genome Database** [<http://mips.gsf.de/genre/proj/yeast/>]
34. von Mering C, Krause R, Snel B, Cornell M, Oliver S, Fields S, Bork P: **Comparative assessment of large-scale data sets of protein-protein interactions.** *Nature* 2002, **417(6887)**:399-403.

Publish with **BioMed Central** and every scientist can read your work free of charge

"BioMed Central will be the most significant development for disseminating the results of biomedical research in our lifetime."

Sir Paul Nurse, Cancer Research UK

Your research papers will be:

- available free of charge to the entire biomedical community
- peer reviewed and published immediately upon acceptance
- cited in PubMed and archived on PubMed Central
- yours — you keep the copyright

Submit your manuscript here:
http://www.biomedcentral.com/info/publishing_adv.asp

