Multi-Domain Manifold Learning for Drug-Target Interaction Prediction

Ruichu Cai * Zhenjie Zhang † Srinivasan Parthasarathy ‡ Anthony K. H. Tung §
Zhifeng Hao ¶ Wen Zhang ‖

Abstract

Drug-target interaction (DTI) provides novel insights about the genomic drug discovery, and is a critical technique to drug discovery. Recently, researchers try to incorporate different information about drugs and targets for prediction. However, the heterogeneous and high-dimensional data poses huge challenge to existing machine learning methods. In the last few years, extensive research efforts have been devoted to the utilization of manifold property on high dimensional data, e.g. dimension reduction methods preserving local structures of the manifolds. Motivated by the successes of these studies, we propose a general framework incorporating both manifold structures and known interaction/non-interaction information to predict the drug-target interactions. To overcome the challenges of domain scaling and information inconsistency, we formulate the problem with *Semidefinite Programming*(SDP), including new constraints to improve the robustness of the learning procedure. A variety of optimization techniques are also designed to enhance the scalability of the problem solver. Effectiveness of the method is evaluated by experiments on the benchmark dataset. Compared with state-of-the-art methods, the proposed methods generate much more accurate drug-target interaction prediction.

1 Background and Motivation

The identification of drugs and target interactions (DTIs) plays an important role in drug discovery. The high-throughput experiments over the genome, transcriptome and proteome, are helpful to the understanding of the genomic spaces populated by common protein classes, and the identification of potentially useful compounds. However, the wet experiments are time-consuming and costly. With the increasing drug-related information collected in the wet experiments, the computational methods become a promising approach for the DTIs.

The early DTI methods analysis the drug structures or utilize quantitative structure activity relationship (QSAR), e.g. Keiser's work [8], Humberto's multi-target QSAR model [12], Li's docking-based method [10]. However, those methods do not scale for large datasets. In recent years, machine learning methods are introduced to the DTI prediction. In 2008, Yamanishi et al. [26] formulate the drugCtarget interaction inference as a supervised learning problem for a bipartite graph. In 2009, Yamanishi et al.[1] use bipartite local models to predict targets of a given drug, and to predict drugs targeting a given target. In 2010, Xia [23] adopt the semi-supervised method by incorporating heterogeneous biological spaces. More recently, handful machine learning methods are also applied to this problem, such as network inference [24, 25], random walk [2] , information integration [27, 5], and restricted Boltzmann machine [20].

For all the exiting methods, the heterogeneous and high-dimensional data is huge challenge to be addressed. Interestingly, we find that the data distributions in these high dimensional spaces usually only span on some low dimensional manifolds, where manifold is a topological space that resembles local structure near each point. Thus, manifold learning is a promising approach of incorporating high-dimensional and heterogeneous data. In traditional manifold learning algorithms, such as ISOMAP [17], LLE [13] and SDE [21], they focus on effective dimensionality reduction over a single domain. Though the manifold alignment techniques [19, 4, 3, 16] try to discover some one-to-one correspondence between objects on two different manifolds based on some labeled pairs of identical objects, the assumption on the existence of the *one-to-one* mapping between two manifolds does not hold in drug-target prediction problem, in which some drugs in design do not interact with any existing target, depending on some complex functionality across genomic domain for targets and chemical

*Faculty of Computer, Guangdong University of Technology
†Advanced Digital Sciences Center, Illinois at Singapore Pte.
‡Department of Computer Science and Engineering, The Ohio State University
§School of Computing, National University of Singapore
¶Faculty of Computer, Guangdong University of Technology
‖School of Computer, Wuhan University

domain for drugs.

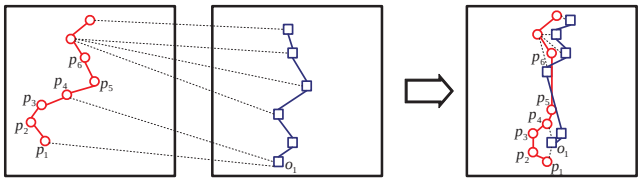


Figure 1: Example of mapping from two domains to an interaction space

Motivated by such observations, we believe that it is meaningful and beneficial to formulate the drug-target interaction prediction problem as a general class of new manifold learning problems, with better extensibility to handle manifolds in multiple domains with totally different properties and distance measures. To be more precise, given the data sets in a variety of domains, as well as some known interactive and non-interactive pairs across domains, mappings are constructed to transform the manifolds into a virtual interaction space. Objects with strong interactions are expected to be close to each other in the interaction space, while non-interactive objects are far away from each other.

Unfortunately, previous solutions to single-domain manifold learning, manifold alignment and structured embedding [14, 15, 3] cannot be applied directly to our multi-domain manifold learning problem, due to some new challenges. Firstly, the inconsistencies between manifold structure and interaction knowledge may lead to difficulties in resolving information conflicts. To better preserve the interactive and non-interactive pairs in the target space of the mappings, it is necessary to sacrifice parts of the manifold structures, by partitioning the manifolds into pieces and scaling each piece with different factor. Secondly, the preservation of local manifold structures has proven its usefulness in structural learning [13], but sometimes could cause inconsistency problem with the interaction/non-interaction information.

In Figure 1, we present an example to illustrate the hardness of our problem. In the original two domains on the left, the manifolds are marked with solid lines and the interaction pairs are linked with dashed lines. After mapping the objects to the new space on the right, the manifolds of the circles are divided into two groups, partially discarding the neighborhood information on the intersection between the bottom group and top group. Based on the manifold structure, the square object o_1 is likely to interact with the two circle objects p_2 and p_3, since their manifold neighbors p_1 and p_4 are both linked to o_1. To balance the effects of manifold structure and interaction knowledge, the learning algorithm must be flexible enough to take consistent relaxations on both sides. Following the kernel learning framework used in [21, 7], we formalize the problem of multi-domain manifold learning with *Semidefinite Programming* (SDP), with well-designed constraints and scalability optimizations, to handle the problems of local scaling and information inconsistency. Based on our SDP-based formulation, we also present two optimization techniques to further enhance the scalability of our method, i.e. *Split-And-Merge* and *Pairwise Querying*. These optimization techniques greatly improve the usefulness of our method on large-domain problems and large-scale datasets used in existing studies, upon existing studies, e.g. [11].

The main contributions of the paper are listed as follows.

1. We present a new general framework for drug-target interaction prediction by mapping data from different domains into a virtual interaction space.

2. We show an elegant Semidefinite Programming solution to the mapping learning problem, utilizing manifold properties of the domains and including new constraints to tackle the problems of domain inconsistency and local scaling.

3. We propose two optimization strategies to improve the scalability of the multi-domain manifold learning framework, supporting real application domains with thousands of samples.

2 Methods and Technical Solutions

2.1 Preliminaries In this section, we formalize the drug-target interaction prediction problem definition and the general algorithm framework. Given an object set $O = \{o_1, o_2, \ldots, o_n\}$ and domain set $\mathbb{D} = \{\mathbb{D}_1, \mathbb{D}_2, \ldots, \mathbb{D}_l\}$, each object $o_i \in O$ belongs to one and only one domain in $\mathbb{D}$. For each domain $\mathbb{D}_j$, d_j denotes the dissimilarity function on objects in $\mathbb{D}_j$, i.e. $d_j(o_i, o_h)$ estimates the difference between o_i and o_h in $\mathbb{D}_j$. There exist two types of object pairs, namely *interactive pair* and *non-interactive pair*. An interactive (resp. non-interactive) pair consists of two objects o_i and o_h, indicating that o_i and o_h are strongly (resp. weakly) related. We use $\mathcal{I}$ and $\mathcal{N}$ to denote the sets of all interactive and non-interactive pairs respectively. Note that, we do not make any distinction between the different types of interaction (for example, activate or deactivate a protein) between a drug and a target.

PROBLEM 1. ***Interaction Prediction Problem*** *Given the object sets O, interactive set $\mathcal{I}$ and non-interactive set $\mathcal{N}$, predict the interaction of any unresolved pair (o_i, o_h).*

While different solutions have been derived on these concrete problems in their own areas, we formulate Problem 1 as an embedding learning problem, utilizing the fact that most of these domains follow manifold property. In the rest of the paper, we use $\mathbb{S}$ to denote a virtual interaction space employing Euclidean distance as the underlying dissimilarity function, i.e. $d(x, y) = \|x - y\|_2$ for any $x, y \in \mathbb{S}$.

PROBLEM 2. ***Multi-Domain Manifold Learning*** *Given the object set O, interactive pairs $\mathcal{I}$, non-interactive pairs $\mathcal{N}$ and two threshold parameters $\{\theta, \gamma\}$, construct mappings $\phi_j : \mathbb{D}_j \mapsto \mathbb{S}$ for each domain $\mathbb{D}_j$, such that:1) the mappings preserve the manifold structures in original domains;2) for any two objects $o_i \in \mathbb{D}_j$ and $o_h \in \mathbb{D}_t$, we have $d(\phi_j(o_i), \phi_t(o_h)) \leq \theta$ if $(x, y) \in \mathcal{I}$, and $d(\phi_j(o_i), \phi_t(o_h)) \geq \gamma$ if $(x, y) \in \mathcal{N}$*

In other words, the mapping transformations $\{\phi_j\}$ are expected to group the interactive pairs within distance θ, and partition non-interactive pairs with margin no smaller than γ. Moreover, the mappings are also supposed to satisfy some desirable properties to maintain the local structure information of the original manifolds.

In this paper, we present a general solution to Problem 2, following the kernel learning framework for manifold problem [21, 7]. To simplify the notation, we use o'_i to denote the location of o_i in $\mathbb{S}$ after applying the mapping, i.e. $o'_i = \phi_j(o_i)$ if $o_i \in \mathbb{D}_j$. The matrix R includes the vectors of all the objects after the mappings, i.e. $R = (o'_1, o'_2, \ldots, o'_n)^T$.

Then, the kernel matrix, $K_{n\times n} = \{k_{ih}\}$, consists of the inner product between any pair of rows in R, i.e. $K = RR^T$ or $k_{ih} = (o'_i)^T o'_h$ for any $1 \leq i, h \leq n$. Given the kernel matrix K, the squared Euclidean distance between any pair of objects, o'_i and o'_h in $\mathbb{S}$, is simplified as $k_{ii} + k_{hh} - 2k_{ih}$. Therefore, given the kernel matrix K, we predict the existence of another interactive pair between o_i and o_h, if $k_{ii} + k_{hh} - 2k_{ih} \leq \theta^2$. And it is sufficient to construct the kernel matrix K, instead of explicitly computing the exact vector representations of the objects in $\mathbb{S}$.

3 Kernel Learning with SDP

In this section, we introduce the details on the construction of SDP to discover the optimal kernel matrix K preserving both manifold structure and interaction knowledge. As is mentioned in the introduction section, the major challenges of multi-domain manifold learning stem from domain scaling and information inconsistency problems. To overcome these difficulties, we design constraints on the ratios, of the distances to the neighbor objects on the manifold, instead of absolute distance values. If o_r and o_h are both neighbors to o_i on the original manifold in domain $\mathbb{D}_j$, our SDP program preserves the local structure of the manifold by the following constraint.

$$\frac{d_j(o_i, o_r)}{d_j(o_i, o_h)}\beta \leq \frac{k_{ii} + k_{rr} - 2k_{ir}}{k_{ii} + k_{hh} - 2k_{ih}} \leq \beta^{-1}\frac{d_j(o_i, o_r)}{d_j(o_i, o_h)} \tag{3.1}$$

Here, β $(0 < \beta \leq 1)$ is the relaxation parameter specified by the user. A β smaller than 1 intuitively allows the manifolds to gradually change its scaling from point to point, while the orders of the distances in the neighborhoods are approximately sustained. Recall the example in Figure 1. Without the β-relaxation, it is impossible to find an embedding grouping p_1 and p_4 together while preserving the distances $d_1(p_4, p_5)$ and $d_1(p_5, p_6)$ at the same time.

The following constraints are set up for the interaction information contained in interactive pair and non-interactive pair. In particular, if (o_i, o_h) is a known interactive pair, we expect the kernel matrix to follow

$$\begin{aligned} d(o'_i, o'_h) = k_{ii} + k_{hh} - 2k_{ih} \leq \theta^2 + \psi_{ih} \\ \text{s.t. } \forall i, h : \psi_{ih} \geq 0 \end{aligned} \tag{3.2}$$

Here, ψ_{ih} is the relaxation factor of the interaction relationship. The existence of ψ_{ih} improves the robustness of our solution, since not all interactive pairs are strictly within distance of θ. Similarly, each non-interactive pair (o_i, o_h) also poses a constraint as

$$\begin{aligned} d(o'_i, o'_h) = k_{ii} + k_{hh} - 2k_{ih} \geq \gamma^2 - \psi_{ih} \\ \text{s.t. } \forall i, h : \psi_{ih} \geq 0 \end{aligned} \tag{3.3}$$

In SDP, it is important to guarantee the uniformity of the optimal solution. Otherwise, SDP package using interior point method may not converge. The following two constraints ensure the uniformity condition of our program. Firstly, the requirement on *centering property* [21] is posed to avoid any possible shifting of the optimal solution. This constraint is simply written as

$$\sum_{ih} k_{ih} = 0 \tag{3.4}$$

Secondly, the global scaling effect leads to multiple optimal solutions. To understand the reason for this phenomenon, refer to example in Figure 2. Given the embedding on the left satisfying all constraints mentioned before, it is easy to construct another embedding on the right, which does not violate any constraint but with larger variance.

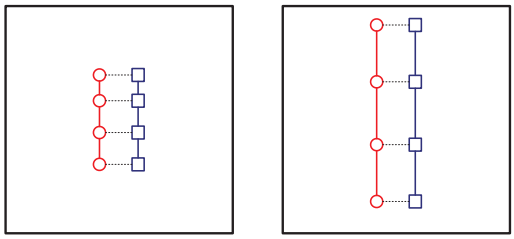


Figure 2: The embedding in left space and the embedding in right space incur the same penalty cost, but with different energies (trace of the kernel matrix)

To prevent our program from infinite growing, one parameter C is specified to bound the energy, i.e. the trace of the kernel matrix K, as

$$\sum_{1 \le i \le n} k_{ii} = C \tag{3.5}$$

Finally, the objective function over the kernel matrix K is designed to minimize the inconsistencies between the manifolds and the interactive and non-interactive labels, i.e. the sum on the relaxations over the labeled pairs

$$\text{Minimize:} \sum_{(o_i, o_h) \in \mathcal{I} \cup \mathcal{N}} \psi_{ih} \tag{3.6}$$

To see why the program above is a semidefinite program, we construct a new matrix K^* as follows, which organizes the relaxation factors $\{\psi_{ih}\}$ on the diagonal of the sub-matrix in the right-bottom corner. It is straightforward to verify that K^* is a semidefinite matrix, if and only if K is semidefinite.

$$K^* = \begin{pmatrix} K & 0 \\ 0 & \{\psi_{ij}\} \end{pmatrix}$$

As a summary, our SDP formulation of multi-domain manifold learning problem takes Eq. (3.6) as the objective function, subject to the constraints in Eq. (3.1)-(3.5). The optimal mapping is thus trained with standard cross validation on the training data set, by using any existing SDP solver package.

The training of the SDP is unfortunately very expensive, due to the large number of variables and constraints posed on our program. If we use k-nearest neighbor to construct the manifold around each object in the original domain, the target matrix K^* is of $O((n+|\mathcal{I}|+|\mathcal{N}|)^2)$ entries, along with $O(k^2 n+|\mathcal{I}|+|\mathcal{N}|)$ constraints. To enhance the performance, it is also important to find the optimal parameters on θ, β and C. In practice, we cut down the number of parameters for training, which is covered in Section 4. Even by parameter reduction, it is still necessary to search in a large parameter space. Therefore, the training of the program incurs expensive costs on both memory and computation time, even with the most optimized SDP solver package. In Section 3.1, we discuss some optimization techniques to address the scalability issue, rendering some more practical solutions for multi-domain manifold learning.

3.1 Optimization Techniques In this section, we present two optimization techniques to overcome the difficulty on scalability, including *Split And Merge* and *Pairwise Querying*, which exploits the usefulness of partial kernel solution and prediction on individual pairs respectively.

3.1.1 Split And Merge Scalability is an important issue in manifold learning, because of the expensive cost on the computation of the optimal kernel matrix. In [22], for example, a sampling based method was proposed to generate anchor points. Every point in the dataset is thus represented by linear combination of the anchor points. The complexity of the semidefinite program is greatly reduced without much sacrifice on result quality, if the sample set is large enough to bolster the complete manifold. Unfortunately, this method is inapplicable in our problem, since most of the interaction label information is lost if a single sample set is retrieved.

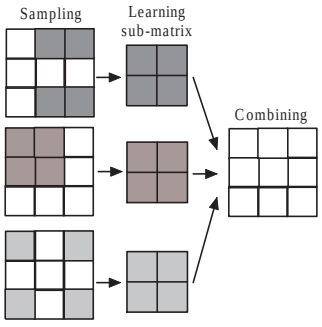


Figure 3: The framework of Split and Merge

To overcome the difficulties, we present a new split-and-merge framework in this section, as shown in Figure 3. In the rest of the section, we use *basic SDP* to denote the standard learning method introduced in last section. Generally speaking, the framework randomly samples on the original problem with input $(O, \mathcal{I}, \mathcal{N})$ and create t sub-problems with new in-

puts $\{(O_1, \mathcal{I}_1, \mathcal{N}_1), \ldots, (O_t, \mathcal{I}_t, \mathcal{N}_t)\}$. Each sub-problem $(O_j, \mathcal{I}_j, \mathcal{N}_j)$ consists of a subset of objects $O_j \subseteq O$. All the interactive pairs related to O_j are included, i.e. $\mathcal{I}_j = \{(o_i, o_h) \in \mathcal{I} \mid o_i \in O_j \ \& \ o_h \in O_j\}$. Similarly, the corresponding non-interactive pair set $\mathcal{N}_j = \{(o_i, o_h) \in \mathcal{N} \mid o_i \in O_j \ \& \ o_h \in O_j\}$ is also retrieved. Given the small sub-problem $(O_j, \mathcal{I}_j, \mathcal{N}_j)$, the basic SDP is run to find the solution to the kernel learning problem, generating the optimal kernel matrix K'_j for the sub-problem.

After the generation of the kernel matrices for all sub-problems, i.e. $K' = \{K'_1, \ldots, K'_t\}$, the results are aggregated with the following iteration-based scheme. At the initial stage of the aggregation process, a random semidefinite matrix, $K^0 = \{k^0_{ij}\}$, is generated with a diagonal matrix with only positive values in diagonal entries. For each global kernel matrix K^s after s iterations, the matrix K^s_j of size $(|O_j|)^2$ denotes the sub-matrix of K^s with only entries corresponding to objects in O_j. With the distance function $\Delta(\cdot, \cdot)$ between two semidefinite matrices of same size, each iteration tries to improve the quality of the global kernel matrix K^s using *Newton's Optimization Method*, with the following quality measure,

$$Q(K^s, K') = \sum_{1 \le j \le t} \Delta(K^s_j, K'_j).$$

We employ *Von-Neumann Entropy* as the distance function between two semidefinite matrices [9], i.e.

$$V(M_1, M_2) = tr(M_1 \log M_1 - M_1 \log M_2 - M_1 + M_2).$$

Since each sub-problem is optimized with centering property with constraint in Eq. (3.4), the measurement does not make sense when the sub-matrix K^s_j is not centered at the origin of the space. To tackle this problem, one more step of transformation is conducted to align K'_j with K^s_j before applying the distance function. Specifically, we have

$$\Delta(K^s_j, K'_j) = V(K^s_j \ , \ K'_j + c \cdot \mathbf{1} \cdot \mathbf{1}^T).$$

The vector $\mathbf{1}$ is the identity vector of size $1 \times |O_j|$, and c is a scalar value calculated by $c = \frac{\mathbf{1}^T (K^s_j - K'_j)\mathbf{1}}{|O_j|^2}$. The operation, $K'_j + c \cdot \mathbf{1} \cdot \mathbf{1}^T$, ensures that the new matrix has the same center as K^s_j and that the pair-wise distance between two points in O_j remains unchanged, thus eliminating the problem of shifting centering.

To apply Newton's Method, the matrix derivative is first derived as

$$\frac{\partial \Delta(K^s_j, K'_j)}{\partial K^s_j} = \left(\log\left(K^s_j\right)^{-1} - \log\left(K'_j + c \cdot \mathbf{1} \cdot \mathbf{1}^T\right)^{-1}\right).$$

Algorithm 1 Split-And-Merge $(O, \mathcal{I}, \mathcal{N})$

1: Construct t random sub-problems, $\{(O_j, \mathcal{I}_j, \mathcal{N}_j)\}$
2: Compute the optimal kernel matrix K'_j for each sub-problem
3: Initial K^0 with a random positive diagonal matrix
4: Set $s = 0$
5: **while** K^s does not converge **do**
6: Compute the derivative $\frac{\partial \Delta(K^s_j, K'_j)}{\partial K^s_j} = \left(\log\left(K^s_j\right)^{-1} - \log\left(K'_j + c \cdot \mathbf{1} \cdot \mathbf{1}^T\right)^{-1}\right)$ for each $1 \le j \le t$
7: Compute the new kernel matrix $K^{s+1} = K^s + \left(\sum_{1 \le j \le t} R(K^s_j)\right)\left(\sum_{1 \le j \le t} R\left(\frac{\partial \Delta(K^s_j, K'_j)}{\partial K^s_j}\right)\right)$
8: **end while**
9: Return the kernel matrix K^s

Since K^s_j is a sub-matrix of K^s, the derivative $\frac{\partial \Delta(K^s, K'_j)}{\partial K^s}$ is constructed by reversely mapping entries in K^s_j back to the derivative matrix on K^s and filling all other entries with 0. We use $\frac{\partial \Delta(K^s, K'_j)}{\partial K^s} = R\left(\frac{\partial \Delta(K^s_j, K'_j)}{\partial K^s_j}\right)$ to denote the reverse mapping process. By linearity property of derivative, the matrix derivative on the original kernel matrix K^s is written as

$$\frac{\partial Q(K^s, K')}{\partial K^s} = \sum_{1 \le j \le t} R\left(\frac{\partial \Delta(K^s_j, K'_j)}{\partial K^s_j}\right).$$

Thus, the new kernel matrix K^{s+1}_j is updated according to Newton's method:

$$K^{s+1} = K^s + \left(\sum_{1 \le j \le t} R(K^s_j)\right)\left(\sum_{1 \le j \le t} R\left(\frac{\partial \Delta(K^s_j, K'_j)}{\partial K^s_j}\right)\right).$$

Note that the new kernel matrix K^{s+1}_j may not be positive semidefinite. To maintain the positive semidefinite property, the algorithm trims K^{s+1}_j by removing any negative eigenvalues and corresponding eigenvectors from the kernel matrix. In Algorithm 1, we summarize the split-and-merge framework.

3.1.2 Pairwise Querying By employing *Split-And-Merge* strategy, when there are a large number of variables in the sub-problems, the computational cost on optimizing the kernel matrix for every individual sub-problem remains high. In particular, every iteration in Algorithm 1 involves the calculation of inverse matrix, incurring cubic complexity in terms of the number of variables in the matrix. In this part of the section, we

Algorithm 2 Pairwise-Querying (O, $\mathcal{I}$, $\mathcal{N}$, θ, γ, o_i, o_r)

1: Construct an empty object set $\hat{O}$
2: Add o_i and o_r into $\hat{O}$
3: Add o_i's neighbors and o_r's neighbors into $\hat{O}$
4: Construct the connection graph G_c by connecting the edge between two objects, if either 1) they are neighbors in its own domain; or 2) they form an interaction pair
5: Find the shortest path between o_i and o_r on the connection graph G_c
6: **if** the shortest path exists **then**
7: **for** each o_h on the shortest path **do**
8: Add each object o_h and its neighborhood into $\hat{O}$
9: **end for**
10: **else**
11: Predict null interaction between o_i and o_r
12: **end if**
13: Construct a new problem with inputs $\hat{O}$, θ, γ, $\hat{I}$ and $\hat{N}$, which only involve objects in $\hat{O}$
14: Report the prediction between o_i and o_r using results of the problem above

present *Pairwise Querying* strategy, which improves the scalability of the solver, by sacrificing the accuracy of the distance estimation in the virtual interaction space.

In *Pairwise Querying*, our method intends to estimate the distance between a particular pair of objects in the virtual interaction space, instead of rendering the distances for every pair of objects. Generally speaking, given a target pair of objects o_i and o_r, our new *pairwise querying* strategy constructs a local kernel matrix K' with respect to a group of local objects $O' \subseteq O$. The local kernel matrix K' is of size $n' \times n'$, in which n' is the cardinality of the local object group, i.e. $n' = |O'|$.

In our standard formulation proposed in Section 3, the distance between o_i and o_r is estimated based on the kernel matrix K. In *pairwise querying*, our method directly estimates the lower bound and upper bound on $d(\phi_j(o_i), \phi_t(o_r))$ in the optimization program. To accomplish this goal, we first select a group of objects in O related to o_i and o_r, and insert them into O'. Two new optimization problems over O' are formulated by adding the following constraint and updating the objective functions as follows. Specifically, the constraint below estimates the total penalty incurred on all pairs of the objects in O'.

$$\sum_{(o_l,o_h)\in\mathcal{I}\cup\mathcal{N}, o_l,o_h\in O'} \psi_{lh} = \Psi \tag{3.7}$$

To estimate the lower bound of the distance between o_i and o_r, we employ the following objective function:

$$\text{Minimize: } k_{ii} + k_{rr} - 2k_{ir} \tag{3.8}$$

Similarly, we maximize the same objective to estimate the upper bound of the distance:

$$\text{Maximize: } k_{ii} + k_{rr} - 2k_{ir} \tag{3.9}$$

As a summary, the lower bound (resp. upper bound) estimations employ Eq. (3.8) (resp. Eq. (3.9)) as objective function respectively, subject to constraints in Eq. (1)-(5),(7). When Ψ equals to the actual penalty incurred in the global SDP with all objects, our optimizations return exact lower bound and upper bound on the distance estimation. This is because the constraints used in *pairwise querying* is an absolute subset of the global SDP. Thus, the feasible solutions for full kernel K must render a feasible solution to K', by selecting appropriate rows and columns in K. To ensure accurate lower bound and upper bound estimation, we take different values on Ψ and return the minimal lower bound and maximal upper bound. In practice, the lower bound is usually loose, since overestimated penalty leads to underestimate on the distances close to 0. Therefore, we only use the upper bound estimation in our experiments.

Regarding the selection of the objects for O', our current implementation simply picks up the objects in the following order. Before starting the selection procedure, our algorithm first constructs a connection graph G_c, in which each node is an object in the data set and an edge between two objects are added if they are neighbors or an interaction/non-interaction between them is observable. In the first step of the selection, o_i and o_r are added into O'. Second, we insert the neighbors of o_i and o_r in their respective domains into O'. Finally, for each pair of objects residing in O', we add all objects on the shortest path between them in G_c into O'. The details of the Pairwise-Querying algorithm is given in Algorithm 2.

4 Empirical Evaluation

4.1 Experimental Setup On drug-target interaction problem, we test the algorithms on four pharmaceutically useful drug classes, including "Enzyme", "Ion channel", "GPCR" and "Nuclear receptor". These data sets were previously used in [24]. In drug-target interaction problem, if the protein is a known target of the drug, the protein and the drug are acceptabed as an interaction with each other. Otherwise, they are non-interaction. Thus, the non-interactive pairs include,

Drug-Target	# of target	# of drugs	# of inter. pairs	# of non-inter. pairs
Enzyme	664	445	2926	292554
Ion channel	204	210	1476	41364
GPCR	95	223	635	20550
Nuclear receptor	26	54	90	1314

Table 1: Statistics on the experimental data sets

those experimentally tested as non-interactive and those without experimental conclusions.

The details of these data sets and the similarity functions can be found in [24]. Since the original data use similarity scores to measure the difference between two proteins (or drugs), we have to convert the data to dissimilarity measures as distances. To transfer similarity values to distances, we employ the transformation function $d(o_i, o_h) = -\log(v_{ih})$, given that v_{ih} is the original similarity measure. In Table 1, the statistical information of the datasets is summarized.

There are several tunable parameters on our program, including the ratio relaxation parameter β, the trace parameter C, the threshold parameters θ and γ, as well as the manifold neighborhood size k. However, we fix all of the parameters except C in our experiments. The threshold parameters θ and γ are fixed, because C can scale the space to fit the existing thresholds. The neighborhood size k is fixed, because of the high computation complexity of large neighborhood. In particular, parameters are fixed as $\beta = 2/3$, $\theta = 1 = \gamma = 1$, and $k = 2$. The trace parameter C is tuned, with 3-fold cross validation on the training set, in the range $[0.5n, 4.5n]$ with step $0.5n$, where n is the size of the data set.

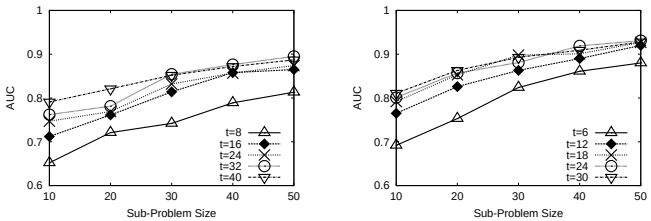
When implementing the split-and-merge strategy, there are two more parameters, including the number of objects in the sub-problem $|O_j|$, and the number of sub-problems t. The default value of $|O_j|$ is 50, and t is set at $\frac{2|O|}{|O_j|}$ which forces each object to appear in 2 sub-problems in expectation. In pairwise querying, the default setting on the neighborhood size k is 3, with a varying range from 2 to 10.

For the experiments on protein-drug testing, we use 10-fold cross-validation for parameter tuning. The overall performance is the average over the all 10 rounds of the trainings. Our experiment simply conducts the parameter tuning on the training set and evaluates the results on the testing set. The algorithm is implemented in Matlab 2009a environment. The package SDPT3 is used as the basic solver for the semidefinite programming [18]. All the experiments are conducted on a PC with Intel Core2 Duo E7500 2.93GHz CPU, 2GB RAM.

4.2 Results on Drug-Target Interaction Prediction On the protein-drug testing task, we compare the performance of three different methods, including our Multi-Domain Manifold Learning method without pairwise querying (MDML), our Multi-Domain Manifold Learning method with pairwise querying (PQ), Simple Mapping method (SM) [24],Bipartite Graph method (BG) [24],BLM [1],LapRLS [23] and KBMF2K [6].

Following the notation in [24], we use TP, TN, FP, FN to denote the cardinalities of true positive, true negative, false positive and false negative respectively. The following performance measures are recorded in our experiments, including AUC (integral on ROC curve), Sensitivity (TP/(TP+FN) when $\theta' = 1$), Specificity (TN/(TN+FP) when $\theta' = 1$), and PPV (TP/(TP+FP) when $\theta' = 1$).

The results in Table 2 show that MDML and PQ perform better results than benchmark methods. More importantly, the benchmark methods usually require a variety of information about drugs or proteins for prediction, while our proposed methods just make use of the pairwise similarity. In particular, MDML achieves highest AUC on GPCR and Nuclear receptor, while PQ is the best on Enzyme. These results show that our framework better preserves the interaction information in the virtual space. It is also interesting to see that MDML and PQ are much better on sensitivity, implying that our method is more capable of discovering true interactions. This property is extremely desirable for drug design industry.



(a) On *Ion channel* data (b) On *GPCR* data

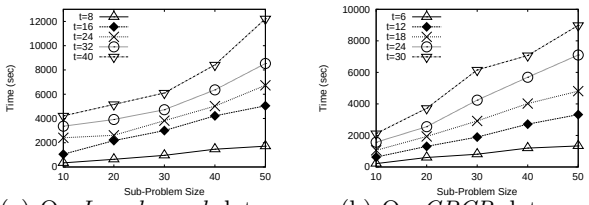
Figure 4: Impact of sub-problem size on the accuracy of MDML

In Figure 4, we test the impact of sub-problem size on the prediction quality of MDML. The results verify the intuition that the prediction quality is improved by increasing the sub-problem size or equivalently reducing the number of sub-problems. It also shows that the number of sub-problem is not significant when the

Dataset	Method	AUC	Sens.	Spec.	PPV
Enzyme	SM	0.904	0.574	0.995	0.57
	BG	0.852	n.a.	n.a.	n.a.
	BLM	0.843	n.a.	n.a.	n.a.
	LapRLS	0.914	n.a.	n.a.	n.a.
	KBMF2K	0.889	n.a.	n.a.	n.a.
	MDML	0.903	0.716	0.994	0.526
	PQ	**0.923**	0.735	0.980	0.591
Ion Channel	SM	0.851	0.271	0.999	0.936
	BG	0.864	n.a.	n.a.	n.a.
	BLM	0.881	n.a.	n.a.	n.a.
	LapRLS	0.920	n.a.	n.a.	n.a.
	KBMF2K	**0.924**	n.a.	n.a.	n.a.
	MDML	0.865	0.450	0.997	0.84
	PQ	0.881	0.472	0.982	0.892
GPCR	SM	0.899	0.234	0.996	0.681
	BG	0.904	n.a.	n.a.	n.a.
	BLM	0.627	n.a.	n.a.	n.a.
	LapRLS	0.788	n.a.	n.a.	n.a.
	KBMF2K	0.837	n.a.	n.a.	n.a.
	MDML	**0.920**	0.347	0.998	0.810
	PQ	0.901	0.314	0.971	0.713
Nuclear Receptor	SM	0.843	0.148	0.999	0.954
	BG	0.815	n.a.	n.a.	n.a.
	BLM	0.458	n.a.	n.a.	n.a.
	LapRLS	0.563	n.a.	n.a.	n.a.
	KBMF2K	0.756	n.a.	n.a.	n.a.
	MDML	**0.950**	0.500	0.990	0.667
	PQ	0.903	0.341	0.961	0.601

Table 2: Experimental results on drug-target prediction

each individual sub-problem is large enough, while the prediction can be further enhanced if there are more objects appearing in single sub-problems.



(a) On *Ion channel* data (b) On *GPCR* data

Figure 5: Impact of sub-problem size on the efficiency of MDML

The results on the tests of efficiency with varying sub-problem size for MDML are shown in Figure 5. The computation time of our multi-domain manifold learning algorithm increases slowly with respect the sub-problem size. Considering the limited improvement of prediction quality in Figure 4, we believe our split-and-merge strategy performs well, even when the original learning problem are partitioned into very small sub-problems. Note that the other data sets are not included in these tests, because the algorithm is capable of returning results only when the sub-problem size is sufficiently small.

Our next group of experiments investigate the computational complexity of PQ, which only depends on the neighborhood size. Recall that Algorithm 2 includes more objects in the training, when the neighbors of the two querying objects grow. The computation time of PQ with different neighborhood sizes are reported in Figure 6. The numbers are average of 100 runs over randomly picked object pairs. It shows that PQ on a single pair of objects is very efficient. It could return the results within 10 seconds, even when we include 10 objects in the neighborhoods. It proves the value of the strategy of *pairwise querying* in real applications, in which other techniques could generate candidates for input of interaction prediction, i.e. protein filtering in drug industry.

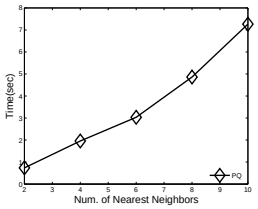


Figure 6: Impact of number of nearest neighbors of PQ

Finally, the ROC curves of MDML method on the 4 data sets are presented in Figure 7, which also confirms the superiority of our proposal.

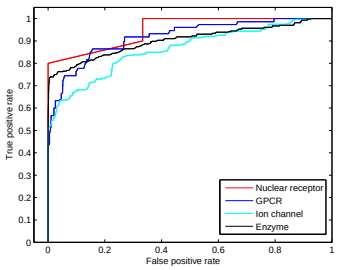


Figure 7: ROC curves on the data sets for protein-drug testing tasks

5 Significance and Impact

In this paper, we propose an extension of manifold learning to multiple domains for the drug-target interaction prediction problem. A general solution based on semidefinite programming is presented to exploit the information in manifold as well as known interaction. To overcome the limitation on the scalability issue, we derive two optimization techniques, i.e. split-and-merge and pairwise querying strategies. The experiments on the benchmark datasets demonstrate that the effectiveness of the proposed methods.

Our proposals not only provide a solution for a wide

class of interaction prediction applications, such as image annotation, web page tagging and drug-drug interaction, but also have opened new research opportunities on a couple of different directions, for example designing more scalable and efficient methods to solve the multi-domain manifold learning problem, verifying our model and algorithm on larger data.

6 Acknowledgements

This research is supported in part by NSFC-Guangdong Joint Found(U1501254, U1401251), Natural Science Foundation of China (61472089, 61572143, 61103126, 61572368), Natural Science Foundation of Guangdong province (2014A030306004, 2014A030308008), Key Technology Research and Development Programs of Guangdong Province (2013B051000076, 2015B010108006, 2015B010131015), Science and Technology Plan Project of Guangzhou City(2014Y2-00027). Z. Zhang is supported by the Human-Centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center from Singapores A*STAR.

References

[1] K. Bleakley and Y. Yamanishi. Supervised prediction of drug-target interactions using bipartite local models. *Bioinformatics*, 25(18):2397–2403, 2009.

[2] X. Chen, M. Liu, and G. Yan. Drug-target interaction prediction by random walk on the heterogeneous network. *Molecular BioSystems*, 8(7):1970–8, 2012.

[3] Z. Cui, H. Chang, S. Shan, and X. Chen. Generalized unsupervised manifold alignment. In *Advances in Neural Information Processing Systems*, pages 2429–2437, 2014.

[4] E. Elhamifar and R. Vidal. Sparse manifold clustering and embedding. In *NIPS*, pages 55–63, 2011.

[5] D. Emig, A. Ivliev, O. Pustovalova, L. Lancashire, S. Bureeva, Y. Nikolsky, and M. Bessarabova. Drug target prediction and repositioning using an integrated network-based approach. *PLoS One*, 8(4):e60618, 2013.

[6] M. Gönen. Predicting drug–target interactions from chemical and genomic kernels using bayesian matrix factorization. *Bioinformatics*, 28(18):2304–2310, 2012.

[7] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *ICML*, 2004.

[8] M. J. Keiser, B. L. Roth, B. N. Armbruster, P. Ernsberger, J. J. Irwin, and B. K. Shoichet. Relating protein pharmacology by ligand chemistry. *Nature biotechnology*, 25(2):197–206, 2007.

[9] B. Kulis, M. Sustik, and I. Dhillon. Learning low-rank kernel matrices. In *ICML*, pages 505–512, 2006.

[10] H. Li, Z. Gao, L. Kang, H. Zhang, K. Yang, K. Yu, X. Luo, W. Zhu, K. Chen, J. Shen, et al. Tarfisdock: a web server for identifying drug targets with docking approach. *Nucleic acids research*, 34(suppl 2):W219–W224, 2006.

[11] V. Mahadevan, C. W. Wong, J. C. Pereira, T. Liu, N. Vasconcelos, and L. K. Saul. Maximum covariance unfolding : Manifold learning for bimodal data. In *NIPS*, pages 918–926, 2011.

[12] F. J. Prado-Prado, E. Uriarte, F. Borges, and H. González-Díaz. Multi-target spectral moments for qsar and complex networks study of antibacterial drugs. *European journal of medicinal chemistry*, 44(11):4516–4521, 2009.

[13] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[14] B. Shaw and T. Jebara. Structure preserving embedding. In *ICML*, page 118, 2009.

[15] L. Song, J. Huang, A. J. Smola, and K. Fukumizu. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *ICML*, page 121, 2009.

[16] S. Tan, Z. Guan, D. Cai, X. Qin, J. Bu, and C. Chen. Mapping users across networks by manifold alignment on hypergraph. In *AAAI*, 2014.

[17] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[18] K.-C. Toh, M. J. Todd, and R. H. Tütüncü. Sdpt3a matlab software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11(1-4):545–581, 1999.

[19] C. Wang and S. Mahadevan. Manifold alignment using procrustes analysis. In *ICML*, pages 1120–1127, 2008.

[20] Y. Wang and J. Zeng. Predicting drug-target interactions using restricted boltzmann machines. *Bioinformatics*, 29(13):i126–i134, 2013.

[21] K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *ICML*, page 106. ACM, 2004.

[22] K. Q. Weinberger and G. Tesauro. Metric learning for kernel regression. In *AISTATS*, pages 608–615, 2007.

[23] Z. Xia, X. Zhou, Y. Sun, and L. Wu. Semi-supervised drug-protein interaction prediction from heterogeneous spaces. In *The Third International Symposium on Optimization and Systems Biology*, volume 11, pages 123–131. Citeseer, 2009.

[24] Y. Yamanishi. Supervised bipartite graph inference. In *NIPS*, pages 1841–1848, 2008.

[25] Y. Yamanishi. Chemogenomic approaches to infer drug–target interaction networks. In *Data Mining for Systems Biology*, pages 97–113. Springer, 2013.

[26] Y. Yamanishi, M. Araki, A. Gutteridge, W. Honda, and M. Kanehisa. Prediction of drug–target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, 24(13):i232–i240, 2008.

[27] Y. Yamanishi, M. Kotera, M. Kanehisa, and S. Goto. Drug-target interaction prediction from chemical, genomic and pharmacological data in an integrated framework. *Bioinformatics*, 26(12):i246–i254, 2010.