An *In Silico* Model for Interpreting Polypharmacology in Drug-Target Networks

Ichigaku Takigawa, Koji Tsuda and Hiroshi Mamitsuka

Summary

Recent analysis on polypharmacology leads to the idea that only small fragments of drugs and targets are a key to understanding their interactions forming polypharmacology. This idea motivates us to build an *in silico* approach of finding significant substructure patterns from drug-target (molecular graph-amino acid sequence) pairs. This article introduces an efficient *in silico* method for enumerating, from given drug-target pairs, all frequent subgraph-subsequence pairs, which can be then examined by hypothesis testing for statistical significance further. Unique features of the method are in scalability, computational efficiency and technical soundness in terms of computer science and statistics. The presented method was applied to 11,219 drug-target pairs in DrugBank to obtain significant substructure pairs, which can divide most of the original 11,219 pairs into eight highly exclusive clusters, implying that the obtained substructure pairs are indispensable components for interpreting polypharmacology.

Keywords: Frequent pattern mining, Graphs, Strings, Likelihood-ratio test, Polypharmacology, Drug-target networks

1 Introduction

Polypharmacology (or drug promiscuity) is a recently emerging concept in drug-target interactions, due to mainly the following three reasons: 1) multi-targeted drugs have been clinically successful, particularly as dual or multiplex kinase inhibitors [1]. 2) a lot of approved drugs are not necessarily so selective [2], where a typical example is cancer drugs such as Gleevec (imatinib) and Sutent (sunitinib) which can bind to multiple kinases [3]. 3) network science, particularly scale-freeness of drug-target networks imply the robustness of biological systems [4, 5], by which dysfunction of only a single protein can be in most cases compensated, indicating that inhibiting a single target would be therapeutically insufficient [6].

Recent analysis suggests that targets of promiscuous drugs cannot necessarily be similar to each other [2, 7], meaning that only a small part of each target might be connected to the principle behind polypharmacology. Furthermore recent research shows that smaller drugs in molecular weight are likely to be more promiscuous [5], suggesting that only small fragments in each ligand would be related to drug promiscuity. They have brought us a hypothesis that fragments in drug-target pairs, or paired fragments, must be important factors behind polypharmacology. Thus naturally an *in silico* approach for analyzing polypharmacology based on this hypothesis is to use molecular graphs for drugs (or chemical compounds) and amino acid sequences for targets (or proteins) and examine paired fragments (or substructures) in molecular graphs and amino acid sequences of drug-target pairs [8]. We introduce a data-driven approach for mining substructure pairs which significantly shared in currently available drug-target (graph-sequence) pairs. A unique feature of this approach is scalability and efficiency for covering all possible substructure (subgraph-subsequence) pairs which significantly co-occur in given drug-target pairs. Furthermore, in [8], obtained significant substructure pairs were used for clustering current drug-target interactions into eight classes, which are highly exclusive each other, implying that each cluster corresponds to one unique type of promiscuous drugs (or targets) forming polypharmacology.

2 Materials

The 'small molecules' dataset of DrugBank [9] (version 2.5 as of January 29, 2009), a standard database on drug information, contains 11,219 drug-target interactions, which are the input as interacting pairs, including 4,191 compounds which were linked to 4,362 targets. On the other hand, non-interacting pairs are all possible combinations from 4,191 compounds and 4,362 targets except the 11,219 drug-target pairs (See **Note 1**). In drug-target pairs, 1,447 (34.5%) out of 4,191 drugs were promiscuous drugs, i.e. each with at least two targets, and this ratio was consistent with 35% in [7]. These promiscuous drugs were involved with 8,475 interactions (75.5% of all 11,219 drug-target pairs) and 171,029 interaction pairs. Drugs are treated as molecular graphs (See **Note 2**) and targets are represented by amino acid sequences.

3 Methods

The main input of the method is drug-target (or compound-protein) pairs, which are turned into graph-sequence pairs in the method. The method tries to find subgraph-subsequence pairs (See **Note 3**), which significantly occur in given drug-target pairs, comparing with non-interacting pairs. This method has two components: (1) all subgraph-subsequence pairs frequently occurring in drug-target pairs are enumerated, and (2) the significance of a frequent subgraph-subsequence

pair is evaluated. We describe Steps (1) and (2) in Subsections 3.1 and 3.2, respectively. Note that Step (1) corresponds to the entire procedure of the method, and in Step (1), Step (2) is performed every time a frequent subgraph-subsequence pair is obtained. Note that Step (2) uses both drug-target pairs and non-interacting pairs, while Step (1) uses drug-target pairs only.

3.1 Mining frequent subgraph-subsequence pairs

3.1.1 Preliminaries

Given a dataset of graph-sequence pairs, we can count the number of graph-sequence pairs which contain a certain subgraph-subsequence pair. We call this number *support* of the corresponding subgraph-subsequence pair, following the literature of *frequent pattern mining* [10]. When the support of a subgraph-subsequence pair is larger than or equal to a given threshold value, which is called *minimum support*, this pair is called a *frequent* subgraph-subsequence pair. That is, the support of a frequent subgraph-subsequence pair must be larger than or equal to the minimum support. We can further define *marginal support* of a subgraph as the number of given graph-sequence pairs which have this subgraph. The marginal support can be defined for subsequences as well.

The first, key idea for enumerating all frequent subgraph-subsequence pairs efficiently is the following property, which is called *downward closure*:

Proposition 1 (Downward closure)**.** *A subgraph-subsequence pair is infrequent if this pair contains any smaller infrequent subgraph-subsequence pairs.*

This property is powerful, because if you find an infrequent subgraph-subsequence pair, you do not have to search pairs with larger subgraphs and subsequences which include this pair. Then this idea naturally leads to a so-called *pattern-growth* approach, in which we can start with smallest subgraph-subsequence pairs and extend them to larger pairs, and if we come across an infrequent pair, then we can stop going on to larger pairs due to the downward closure property.

Without loss of generality, we explain this approach more, focusing on sequences only (rather than graph-sequence pairs). The pattern growth procedure naturally generates a hierarchy, which can be represented in a rooted ordered tree, called an *enumeration tree*. The enumeration tree is a rooted ordered spanning tree over all frequent subsequences, roughly with the following two features: 1) the null sequence is on the root, and sequences with only one letter are the children of the root. 2) each node corresponds to a frequent subsequence in a one-to-one manner, where a subsequence with a larger number of letters are attached to nodes in a deeper level (See **Note 4**).

An important point of the enumeration tree is that we can enumerate all subsequences completely without any duplication by traversing an enumeration tree as a search space. This tree-

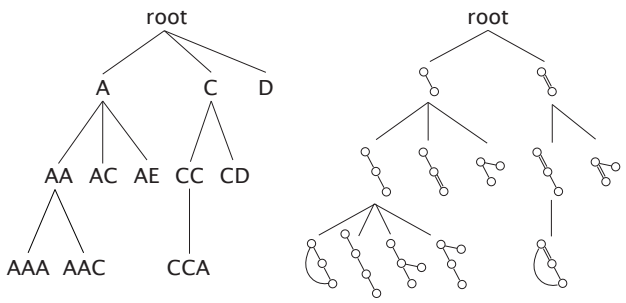


Figure 1: Samples of enumeration trees for (a) frequent subsequences and for (b) frequent subgraphs.

shaped search space thus ensures the uniqueness of each subsequence attached to a node and the completeness on searching all frequent subsequences.

In fact, an enumeration tree can be generated by considering the following three points: 1) one node has only one parent node but can have more than one child nodes. 2) a subsequence of a child must be a larger but the minimum. For example, AC in Fig. 1 (a) is a longer sequence but the minimum of longer sequences. 3) an order of sibling nodes is defined by using some criterion, by which for example, AC can be a child of A but cannot be a child of C in Fig. 1 (a). That is, A is prior to C, by which AC is generated from A, being faster than that AC is generated from C.

The enumeration tree can be generated for subgraphs in a similar manner, as shown in Fig. 1 (b), and these subgraphs are used in the next subsection (See **Note 5**).

We here define some notations which will be used in the next subsection. Let $\mathcal{Q}$ be given subgraph-subsequence pairs. Let $\mathcal{Q}_g$ and $\mathcal{Q}_s$ be subgraph-subsequence pairs containing subgraph g and subsequence s, respectively. Similarly let $\mathcal{Q}_{(g,s)}$ be subgraph-subsequence pairs containing both subgraph g and subsequence s. Let $\mathcal{T}_g$ and $\mathcal{T}_s$ be enumeration trees for subgraphs and subsequences, respectively. Here $\varnothing_g$ and $\varnothing_s$ indicate the root nodes of $\mathcal{T}_g$ and $\mathcal{T}_s$, respectively. Similarly, parent(g) represents a subgraph of the parent of a node to which g is assigned in $\mathcal{T}_g$, and parent(s) represents a subsequence of the parent of a node to which s is assigned in $\mathcal{T}_s$. The support of a pair of subgraph g and subsequence s is denoted by $\text{support}((g,s)|\mathcal{Q}) = |\mathcal{Q}_{(g,s)}|$. Similarly the marginal support of subsequence s is denoted by $\text{support}((*,s)|\mathcal{Q})$. Let σ be the minimum support.

3.1.2 Mining algorithm

For mining frequent subgraph-subsequence pairs, we combine two enumeration trees, one for frequent subgraphs and the other for frequent subsequences. That is, all combinations of frequent subgraphs and subsequences can cover all frequent subgraph-subsequence pairs, and the search space for all these combinations can be defined by a product graph of the two enumeration

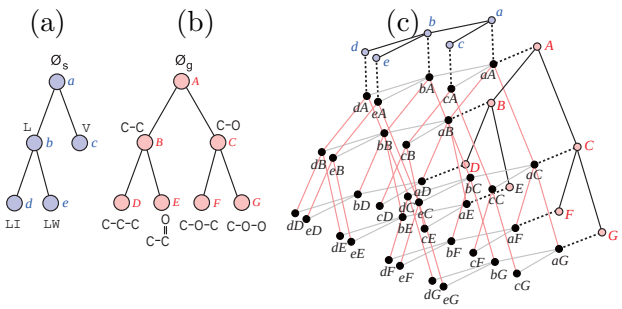


Figure 2: An example of the search space. The search space (c) is defined as the graph product of two enumeration trees for subsequences (a) and for subgraphs (b). (c) covers all possible frequent subgraph-subsequence pairs.

trees. Fig. 2 (a) and (b) show examples of enumeration trees for subsequences and subgraphs, respectively, which can be combined into Fig. 2 (c), where each subgraph-subsequence pair has two parent nodes, and hence this is no longer a tree. In this case, theoretically, we can compute the support of each subgraph-subsequence pair in a dynamic-programming manner.

Proposition 2 (Dynamic programming for subgraph-subsequence pairs)**.** $\mathcal{Q}_{(g,s)}$ *can be iteratively computed as follows:*

1. $\mathcal{Q}_{(g,s)} = \mathcal{Q}_{(\mathrm{parent}(g),s)} \cap \mathcal{Q}_{(g,\mathrm{parent}(s))}$.
2. $\mathcal{Q}_{(g,\varnothing_s)} = \{(G,S) \in \mathcal{Q} \mid g \in G\}$.
3. $\mathcal{Q}_{(\varnothing_g,s)} = \{(G,S) \in \mathcal{Q} \mid s \in S\}$.

In practice, all combinations of frequent subgraphs and frequent subsequences may have a lot of infrequent subgraph-subsequence pairs, and so we can use the downward closure property on the product graph of the two enumeration trees, which can be clearly stated as follows:

Proposition 3 (Two-way downward closure for subgraph-subsequence pairs)**.** *If a subgraph-subsequence pair* (g,s) *is infrequent, then subgraph-subsequence pairs* (g',s') *(where* $g \subseteq g'$ *and* $s \subseteq s'$*) are all infrequent. Thus if* $\mathrm{support}((g,s)|\mathcal{Q}) = |\mathcal{Q}_{(g,s)}| < \sigma$*, then there is no need to extend* (g,s) *further.*

For example, if (C-C,L) (at node bB) in Fig. 2 (c) is infrequent, patterns at nodes (C-C-C,L), (C-C=O,L), (C-C,LI), (C-C,LW), (C-C-C,LI), (C-C-C,LW), (C-C=O,LI) and (C-C=O,LW) must be all infrequent.

The recursion rules in Proposition 2 make us keep all instances explicitly in the graph product $\mathcal{T}_g \times \mathcal{T}_s$. That is, $\mathcal{Q}_{(g,s)}$ must be kept and be passed to subsequent nodes. This is a space-consuming procedure, because two enumeration trees are practically very huge. Thus we can

consider a depth-first traversal of the graph product $\mathcal{T}_g \times \mathcal{T}_s$ by simplifying recursion rules in Proposition 2 into those in the following Proposition 4 (See **Note 6**).

Proposition 4 (A simplified recursion rule)**.** *The recursion rules in Proposition 2 can be simplified to*

$$\mathcal{Q}_{(g,s)} = \mathcal{Q}_{(\mathrm{parent}(g),s)} \cap \{(G,S) \in \mathcal{Q} \mid g \in G\} \quad \textit{and} \quad \mathcal{Q}_{(\varnothing_g,s)} = \{(G,S) \in \mathcal{Q} \mid s \in S\}.$$

We can obtain $\mathcal{Q}_{(g,s)}$ efficiently by using Proposition 4 as follows: we can first traverse $\mathcal{T}_s$ until subsequence s is found, with computing marginal support, $\mathrm{support}((*,s)|\mathcal{Q})$ (See **Note 7**). We can then have $\mathcal{Q}_{(\varnothing_g,s)}$. We can further traverse $\mathcal{T}_g$ from node $(\varnothing_g, s)$, keeping $\mathcal{Q}_{(\cdot,s)}$. Note here that in this traversal, we can reduce the size of $\mathcal{Q}_{(\cdot,s)}$ by using the first rule: $\mathcal{Q}_{(x,s)} = \mathcal{Q}_{(\mathrm{parent}(x),s)} \cap \mathcal{Q}_{(x,\varnothing_s)}$, each time when $\mathrm{parent}(x)$ is extended to x. In this way, we can trace $\mathcal{Q}_{(g,s)}$ along with the path from $\mathcal{Q}_{(g,\varnothing_s)}$ to $\mathcal{Q}_{(g,s)}$. This procedure can be applied to subgraph g, since g and s are symmetric. In addition a larger enumeration tree should be examined first for this procedure, and in reality, we can examine $\mathcal{T}_s$ first, since practically $\mathcal{T}_s$ is expected to be larger than $\mathcal{T}_g$ in drug-target pairs. Finally we can present a pseudocode of the *in silico* mining method as follows:

Proposition 5 (Pseudocode for enumerating all frequent subgraph-subsequence pairs)**.**

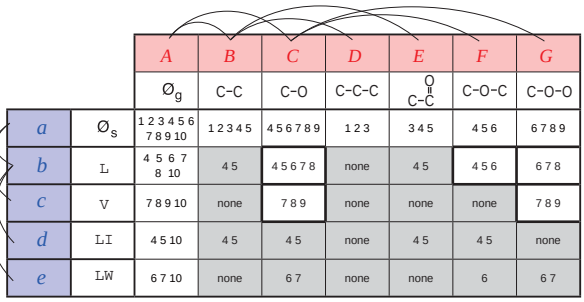
1. *Compute* $\mathcal{Q}_{(g,\varnothing_s)}$ *for all possible* g *using a frequent subgraph mining algorithm in terms of* $\mathrm{support}((g,*)|\mathcal{Q})$.

2. *Start a frequent subsequence mining algorithm from* $(\varnothing_g, \varnothing_s)$*:*
   For each $s \in \mathcal{T}_s$ *in a depth-first traversal order:*

   For each $g \in \mathcal{T}_g$ *in a depth-first traversal order:*

   - *continue if* $g = \varnothing_g$ *or* $s = \varnothing_s$.
   - *reduce the size of* $\mathcal{Q}_{(g,s)}$ *by* $\mathcal{Q}_{(g,s)} = \mathcal{Q}_{(\mathrm{parent}(g),s)} \cap \mathcal{Q}_{(g,\varnothing_s)}$.
   - *compute* $\mathrm{support}((g,s)|\mathcal{Q}) = |\mathcal{Q}_{(g,s)}|$.
   - *break if* $\mathrm{support}((g,s)|\mathcal{Q}) < \sigma$.

We can explain this algorithm more by using a toy sample shown in Fig. 3, which has 10 graph-sequence pairs numbered as $1, 2, \cdots, 10$. Each cell of Fig. 3 shows graph-sequence pairs having the corresponding subgraph-subsequence pair, such as that only graph-sequence pairs 4 and 5 have (C-C, L). Edges of the two enumeration trees in Fig. 2 are also shown by curves at the outside of both rows and columns. The objective here is to find all frequent pairs colored in white: (C-O,L), (C-O,V), (C-O-C,L), (C-O-O,L), and (C-O-O,V).



		A	B	C	D	E	F	G
		$\varnothing_g$	C-C	C-O	C-C-C	C-C(=O)	C-O-C	C-O-O
a	$\varnothing_s$	1 2 3 4 5 6 7 8 9 10	1 2 3 4 5	4 5 6 7 8 9	1 2 3	3 4 5	4 5 6	6 7 8 9
b	L	4 5 6 7 8 10	4 5	4 5 6 7 8	none	4 5	4 5 6	6 7 8
c	V	7 8 9 10	none	7 8 9	none	none	none	7 8 9
d	LI	4 5 10	4 5	4 5	none	4 5	4 5	none
e	LW	6 7 10	none	6 7	none	none	6	6 7

Figure 3: A sample for enumerating all subgraph-subsequence pairs with the support of 3 or larger under 10 graph-sequence pairs $(1, 2, \ldots, 10)$. This table corresponds to two enumeration trees of Fig. 2 (a) and (b).

We first build enumeration tree $\mathcal{T}_g$, which corresponds to generating all subgraphs in the top row of Fig. 3, and then starts traversing enumeration tree $\mathcal{T}_s$ from the root. By traversing $\mathcal{T}_s$ in a depth-first manner, the first pattern to be found is (C-C,L). Since (C-C,L) is infrequent (i.e. $|\mathcal{Q}_{(\text{C-C,L})}| < 3$), we do not have to proceed to subsequent (C-C-C,L) and (C-C=O,L). Then, the next subgraph-subsequence pair is (C-O,L), which turns out to be frequent. We then move on to (C-O-C,L). $\mathcal{Q}_{(\text{C-O-C,L})}$ is obtained by $\mathcal{Q}_{(\text{C-O-C,L})} = \mathcal{Q}_{(\text{C-O,L})} \cap \mathcal{Q}_{(\text{C-O-C},\varnothing_s)} = \{4, 5, 6\}$. Similarly, we can move to (C-O-O,L) where $\mathcal{Q}_{(\text{C-O-O,L})} = \mathcal{Q}_{(\text{C-O,L})} \cap \mathcal{Q}_{(\text{C-O-C},\varnothing_s)} = \{6, 7, 8\}$. We now finished traversing all nodes of $\mathcal{T}_g$ for $\text{L} \in \mathcal{T}_s$, and then we proceed to the next subgraph-subsequence pair (C-C,LI) by traversing $\mathcal{T}_s$ from L to LI. However, subsequent nodes, (C-C,LI), (C-O,LI), (C-C,LW), (C-O,LW) and (C-C,V) are all infrequent, and then the next frequent subgraph-subsequence pair becomes (C-O,V). Then subsequent (C-O-C,V) and (C-O-O,V) are examined in this order, and we can find that (C-O-C,V) is infrequent but (C-O-O,V) is frequent. Then we have no nodes to proceed in $\mathcal{T}_g \times \mathcal{T}_s$, and the procedure is terminated. Finally we obtain all five frequent patterns (C-O,L), (C-O-C,L), (C-O-O,L), (C-O,V) and (C-O-O,V) in this order.

3.2 Evaluating significance of subgraph-subsequence pairs

This statistical test is the same as that for detecting 'epistasis' in genetics [11], called *likelihood ratio test with logistic regression*. We first explain this test, focusing on drug-target pairs, being followed by the method for maximizing the likelihood of logistic regression from given drug-target pairs.

3.2.1 Likelihood ratio test with logistic regression

Logistic regression can be defined as the probability p that an event occurs given d explanatory variables $x_1, x_2, \ldots, x_d$, as follows:

$$p = \text{Prob}\{\text{the event occurs} \mid x_1, x_2, \ldots, x_d\} = \frac{\exp(\eta)}{1+\exp(\eta)} = \frac{1}{1+\exp(-\eta)},$$

where $\eta = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$ is a (linear) composite variable. Note that p takes a value between zero and one due to the logistic function, even though η ranges from $-\infty$ to ∞. Note further that this equation can be transformed into $\log\{\frac{p}{1-p}\} = \eta$ where $\frac{p}{1-p} = \text{Prob}\{\text{the event occurs} \mid x_1, x_2, \ldots, x_d\}/\text{Prob}\{\text{the event does not occur} \mid x_1, x_2, \ldots, x_d\}$.

Let $Y \in \{0, 1\}$ be a binary response variable, where the probability of $Y = 0$ (and $Y = 1$) is modeled by

$$\text{Prob}\{Y = 0\} = 1 - p_{\boldsymbol{\theta}}(\boldsymbol{X}) \quad \text{and} \quad \text{Prob}\{Y = 1\} = p_{\boldsymbol{\theta}}(\boldsymbol{X}), \quad p_{\boldsymbol{\theta}}(\boldsymbol{X}) = \frac{\exp(\boldsymbol{\theta}'\boldsymbol{Z})}{1+\exp(\boldsymbol{\theta}'\boldsymbol{Z})},$$

where $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_d)'$ and $\boldsymbol{Z} = (1, \boldsymbol{X}')' = (1, X_1, X_2, \ldots, X_d)'$. To fit this model to n given drug-target pairs for Y and $\boldsymbol{X}$, $\{(y^{(1)}, \boldsymbol{x}^{(1)}), (y^{(2)}, \boldsymbol{x}^{(2)}), \ldots, (y^{(n)}, \boldsymbol{x}^{(n)})\}$ suffices to maximize the likelihood $\ell(\boldsymbol{\theta})$ in terms of parameters $\boldsymbol{\theta}$. The likelihood of n given pairs is defined by

$$\ell(\boldsymbol{\theta}) := \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})^{y^{(i)}} (1 - p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))^{1-y^{(i)}}. \tag{1}$$

For subgraph-subsequence pair (g, s), we can consider two explanatory variables X_1 and X_2 for subgraph g and subsequence s, respectively, each taking 1 if a graph-target pair has the corresponding substructure; otherwise zero. We use two logistic regression models for Y where $Y = 1$ for drug-target pairs (and $Y = 0$ for non-interacting pairs), i.e. the probability that $Y = 1$:

$$p_{\theta\text{:0-2}}(X_1, X_2) = \frac{\exp(\eta)}{1+\exp(\eta)} \quad \text{and} \quad p_{\theta\text{:0-3}}(X_1, X_2) = \frac{\exp(\eta + \theta_3 X_1 X_2)}{1+\exp(\eta + \theta_3 X_1 X_2)},$$

where $\eta = \theta_0 + \theta_1 X_1 + \theta_2 X_2$. Note that the second model has interaction term $\theta_3 X_1 X_2$ while the first model has no interaction terms. Parameters of these two models are independently fitted by maximizing the likelihood. Then, the significance of pair (g, s) can be statistically measured by testing whether $\theta_3 = 0$ is kept or not. Note that this can be conducted by using the likelihood ratio test of two maximum likelihoods $\widehat{L}_{\theta\text{:0-2}}$ for $p_{\theta\text{:0-2}}(X_1, X_2)$ and $\widehat{L}_{\theta\text{:0-3}}$ for $p_{\theta\text{:0-3}}(X_1, X_2)$. The test statistic $-2\log(\widehat{L}_{\theta\text{:0-2}}/\widehat{L}_{\theta\text{:0-3}})$ follows the chi-squared distribution with one degree of freedom under the hypothesis that $\theta_3 = 0$. Thus, we can compute the p-value of the observed statistic from the chi-squared distribution.

3.2.2 Computing likelihood ratio test numerically

Given drug-target pairs, to maximize the likelihood (or fit the logistic regression model to given pairs), we can use the Newton-Raphson method, which is a typical and standard manner for parameter estimation of logistic regression. Explanatory variables X_1 and X_2, as well as response variable Y, are all binary ($Y \in \{0,1\}, X_1 \in \{0,1\}$ and $X_2 \in \{0,1\}$), and thus drug-target pairs for (Y, X_1, X_2) have eight possible combinations only, as shown in Table 1 (a). Thus, only what we have to do is to count how many times each of the eight combinations occurs in given drug-target pairs.

Table 1: Tables for counting eight values.

(a) model without the interaction term

explanatory			response	
	X_1	X_2	$\#\{Y=1\}$	$\#\{Y=0\}$
$\boldsymbol{x}_{00}$	0	0	P_{00}	N_{00}
$\boldsymbol{x}_{01}$	0	1	P_{01}	N_{01}
$\boldsymbol{x}_{10}$	1	0	P_{10}	N_{10}
$\boldsymbol{x}_{11}$	1	1	P_{11}	N_{11}

(b) model with the interaction term

explanatory				response	
	X_1	X_2	X_1X_2	$\#\{Y=1\}$	$\#\{Y=0\}$
$\boldsymbol{x}_{000}$	0	0	0	P_{00}	N_{00}
$\boldsymbol{x}_{010}$	0	1	0	P_{01}	N_{01}
$\boldsymbol{x}_{100}$	1	0	0	P_{10}	N_{10}
$\boldsymbol{x}_{111}$	1	1	1	P_{11}	N_{11}

In fact, we can compute the likelihood ratio test using logistic regression from the counts of eight possible combinations: P_{00}, P_{01}, P_{10}, P_{11}, N_{00}, N_{01}, N_{10}, and N_{11} in Table 1. Denoting $(X_1 = 0, X_2 = 0)$ by $\boldsymbol{x}_{00}$, the probability that $\boldsymbol{x}_{00}$ is observed P_{00} times (from observations with $Y = 1$) and N_{00} times (from observations with $Y = 0$) can be written:

$$\overbrace{p(\boldsymbol{x}_{00}) \times \cdots \times p(\boldsymbol{x}_{00})}^{P_{00}} \times \overbrace{(1 - p(\boldsymbol{x}_{00})) \times \cdots \times (1 - p(\boldsymbol{x}_{00}))}^{N_{00}} = p(\boldsymbol{x}_{00})^{P_{00}} (1 - p(\boldsymbol{x}_{00}))^{N_{00}},$$

and the entire likelihood $\ell(\boldsymbol{\theta})$ of Eq. (1) is implicitly given for binary variables Y, X_1, and X_2 by

$$p(\boldsymbol{x}_{00})^{P_{00}} (1-p(\boldsymbol{x}_{00}))^{N_{00}} p(\boldsymbol{x}_{01})^{P_{01}} (1-p(\boldsymbol{x}_{01}))^{N_{01}} p(\boldsymbol{x}_{00})^{P_{10}} (1-p(\boldsymbol{x}_{00}))^{N_{10}} p(\boldsymbol{x}_{11})^{P_{11}} (1-p(\boldsymbol{x}_{00}))^{N_{11}}.$$

Thus, letting an index set be $\Lambda := \{00, 01, 10, 11\}$, the log-likelihood can be written as follows:

$$\begin{aligned} L(\boldsymbol{\theta}) := \log \ell(\boldsymbol{\theta}) &= \log \left\{ \prod_{\lambda \in \Lambda} p(\boldsymbol{x}_\lambda)^{P_\lambda} (1 - p(\boldsymbol{x}_\lambda))^{N_\lambda} \right\} = \sum_{\lambda \in \Lambda} \{ P_\lambda \log p(\boldsymbol{x}_\lambda) + N_\lambda \log(1 - p(\boldsymbol{x}_\lambda)) \} \\ &= \sum_{\lambda \in \Lambda} \left\{ P_\lambda \log \frac{p(\boldsymbol{x}_\lambda)}{1 - p(\boldsymbol{x}_\lambda)} + (P_\lambda + N_\lambda) \log(1 - p(\boldsymbol{x}_\lambda)) \right\}, \end{aligned}$$

which means that we can compute the p-value of likelihood ratio test using logistic regression by using only P_{00}, P_{01}, P_{10}, P_{11}, N_{00}, N_{01}, N_{01}, N_{11}. To maximize $L(\boldsymbol{\theta})$ by changing $\boldsymbol{\theta}$, the Newton-Raphson method repeats the following update:

$$\boldsymbol{\theta}^{[k+1]} \leftarrow \boldsymbol{\theta}^{[k]} + (\nabla^2 L(\boldsymbol{\theta}))^{-1} \nabla L(\boldsymbol{\theta}) \qquad \text{until} \quad \left| \frac{L(\boldsymbol{\theta}^{[k+1]}) - L(\boldsymbol{\theta}^{[k]})}{L(\boldsymbol{\theta}^{[k]})} \right| < \epsilon,$$

where score $\nabla L(\boldsymbol{\theta}) = \nabla \log \ell(\boldsymbol{\theta})$ and the Hessian matrix (asymptotic Fisher information matrix) are given as

$$\nabla L(\boldsymbol{\theta}) = \begin{bmatrix} \partial L(\boldsymbol{\theta})/\partial\theta_0 \\ \partial L(\boldsymbol{\theta})/\partial\theta_1 \\ \partial L(\boldsymbol{\theta})/\partial\theta_2 \end{bmatrix} = \begin{bmatrix} \sum_{\lambda\in\Lambda}(P_\lambda - (P_\lambda + N_\lambda)p(\boldsymbol{x}_\lambda)) \\ \sum_{\lambda\in\Lambda}(x_1)_\lambda(P_\lambda - (P_\lambda + N_\lambda)p(\boldsymbol{x}_\lambda)) \\ \sum_{\lambda\in\Lambda}(x_2)_\lambda(P_\lambda - (P_\lambda + N_\lambda)p(\boldsymbol{x}_\lambda)) \end{bmatrix}$$

$$\nabla^2 L(\boldsymbol{\theta}) = \begin{bmatrix} \dfrac{\partial^2 L(\boldsymbol{\theta})}{\partial\theta_0\theta_0} & \dfrac{\partial^2 L(\boldsymbol{\theta})}{\partial\theta_0\theta_1} & \dfrac{\partial^2 L(\boldsymbol{\theta})}{\partial\theta_0\theta_2} \\ \dfrac{\partial^2 L(\boldsymbol{\theta})}{\partial\theta_1\theta_0} & \dfrac{\partial^2 L(\boldsymbol{\theta})}{\partial\theta_1\theta_1} & \dfrac{\partial^2 L(\boldsymbol{\theta})}{\partial\theta_1\theta_2} \\ \dfrac{\partial^2 L(\boldsymbol{\theta})}{\partial\theta_2\theta_0} & \dfrac{\partial^2 L(\boldsymbol{\theta})}{\partial\theta_2\theta_1} & \dfrac{\partial^2 L(\boldsymbol{\theta})}{\partial\theta_2\theta_2} \end{bmatrix}, \quad \frac{\partial^2 L(\boldsymbol{\theta})}{\partial\theta_i\theta_j} = \sum_{\lambda\in\Lambda}(P_\lambda + N_\lambda)(x_i)_\lambda(x_j)_\lambda p(\boldsymbol{x}_\lambda)(1 - p(\boldsymbol{x}_\lambda)).$$

Hence, the Newton-Raphson update can be written in a matrix form:

$$\boldsymbol{\theta}^{[k+1]} \leftarrow \boldsymbol{\theta}^{[k]} + (\boldsymbol{X}'\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}'(\boldsymbol{y} - \boldsymbol{p}), \tag{2}$$

where when we use a logistic model $p_{\theta:0\text{-}2}(\cdot)$ for $p(\cdot)$,

$$\boldsymbol{X} := \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad \boldsymbol{W} := \begin{bmatrix} d_{00} & 0 & 0 & 0 \\ 0 & d_{01} & 0 & 0 \\ 0 & 0 & d_{10} & 0 \\ 0 & 0 & 0 & d_{11} \end{bmatrix}, \quad \boldsymbol{y} := \begin{bmatrix} P_{00} \\ P_{01} \\ P_{10} \\ P_{11} \end{bmatrix}, \quad \boldsymbol{p} := \begin{bmatrix} (P_{00} + N_{00})\, p_{\theta:0\text{-}2}(\boldsymbol{x}_{00}) \\ (P_{01} + N_{01})\, p_{\theta:0\text{-}2}(\boldsymbol{x}_{01}) \\ (P_{10} + N_{10})\, p_{\theta:0\text{-}2}(\boldsymbol{x}_{10}) \\ (P_{11} + N_{11})\, p_{\theta:0\text{-}2}(\boldsymbol{x}_{11}) \end{bmatrix},$$

$$d_{00} = (P_{00} + N_{00})\, p_{\theta:0\text{-}2}(\boldsymbol{x}_{00})(1 - p_{\theta:0\text{-}2}(\boldsymbol{x}_{00})), \qquad d_{01} = (P_{01} + N_{01})\, p_{\theta:0\text{-}2}(\boldsymbol{x}_{01})(1 - p_{\theta:0\text{-}2}(\boldsymbol{x}_{01})),$$
$$d_{10} = (P_{10} + N_{10})\, p_{\theta:0\text{-}2}(\boldsymbol{x}_{10})(1 - p_{\theta:0\text{-}2}(\boldsymbol{x}_{10})), \qquad d_{11} = (P_{11} + N_{11})\, p_{\theta:0\text{-}2}(\boldsymbol{x}_{11})(1 - p_{\theta:0\text{-}2}(\boldsymbol{x}_{11})),$$

and when we use a logistic model $p_{\theta:0\text{-}3}(\cdot)$ for $p(\cdot)$,

$$\boldsymbol{X} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad \boldsymbol{W} := \begin{bmatrix} d_{000} & 0 & 0 & 0 \\ 0 & d_{010} & 0 & 0 \\ 0 & 0 & d_{100} & 0 \\ 0 & 0 & 0 & d_{111} \end{bmatrix}, \quad \boldsymbol{y} := \begin{bmatrix} P_{00} \\ P_{01} \\ P_{10} \\ P_{11} \end{bmatrix}, \quad \boldsymbol{p} := \begin{bmatrix} (P_{00} + N_{00})\, p_{\theta:0\text{-}3}(\boldsymbol{x}_{000}) \\ (P_{01} + N_{01})\, p_{\theta:0\text{-}3}(\boldsymbol{x}_{010}) \\ (P_{10} + N_{10})\, p_{\theta:0\text{-}3}(\boldsymbol{x}_{100}) \\ (P_{11} + N_{11})\, p_{\theta:0\text{-}3}(\boldsymbol{x}_{111}) \end{bmatrix},$$

$$d_{000} = (P_{00} + N_{00})\, p_{\theta:0\text{-}3}(\boldsymbol{x}_{000})(1 - p_{\theta:0\text{-}3}(\boldsymbol{x}_{000})), \qquad d_{010} = (P_{01} + N_{01})\, p_{\theta:0\text{-}3}(\boldsymbol{x}_{010})(1 - p_{\theta:0\text{-}3}(\boldsymbol{x}_{010})),$$
$$d_{100} = (P_{10} + N_{10})\, p_{\theta:0\text{-}3}(\boldsymbol{x}_{100})(1 - p_{\theta:0\text{-}3}(\boldsymbol{x}_{100})), \qquad d_{111} = (P_{11} + N_{11})\, p_{\theta:0\text{-}3}(\boldsymbol{x}_{111})(1 - p_{\theta:0\text{-}3}(\boldsymbol{x}_{111})).$$

The deviance of maximum likelihood $\hat{\boldsymbol{\theta}}$ can be defined by

$$D := -2(L(\hat{\boldsymbol{\theta}}) - L^*)$$

where L^* is the log-likelihood by the so-called full model (or saturated model) where probabilities can be given in the following:

$$p(\boldsymbol{x}_{00}) = \frac{P_{00}}{P_{00} + N_{00}}, \quad p(\boldsymbol{x}_{01}) = \frac{P_{01}}{P_{01} + N_{01}}, \quad p(\boldsymbol{x}_{10}) = \frac{P_{10}}{P_{10} + N_{10}}, \quad p(\boldsymbol{x}_{11}) = \frac{P_{11}}{P_{11} + N_{11}},$$

resulting in that the log-likelihood L^* can be obtained as

$$L^* = \sum_{\lambda \in \Lambda} \left\{ P_\lambda \log \frac{P_\lambda}{P_\lambda + N_\lambda} + N_\lambda \log \frac{N_\lambda}{P_\lambda + N_\lambda} \right\}.$$

Finally the deviance thus can be written by

$$D(\hat{\boldsymbol{\theta}}) = 2(L^* - L(\hat{\boldsymbol{\theta}})) = 2 \cdot \sum_{\lambda \in \Lambda} \left\{ P_\lambda \log \frac{P_\lambda}{(P_\lambda + N_\lambda) p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}_\lambda)} + N_\lambda \log \frac{N_\lambda}{(P_\lambda + N_\lambda)(1 - p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}_\lambda))} \right\}.$$

4 Notes

1) Unknown drug-target pairs may be in non-interacting pairs. However we think that they are statistically negligible, since the number of non-interacting pairs is huge.

2) 2D structures of drugs were converted into hydrogen-suppressed molecular graphs, where nodes were labeled with atom types except hydrogens and edges are labeled with bond types.

3) Drug substructures and target substructures mean connected subgraphs and consecutive subsequences, respectively.

4) The support of a subgraph-subsequence pair is monotonically decreasing with increasing the size of the subgraph or the subsequence, meaning that a subgraph on a deeper level in an enumeration tree has a smaller support.

5) In the literature of mining frequent subsequences (or subgraphs), there already exist established algorithms, such as the PrefixSpan algorithm [12] for frequent subsequences and the gSpan algorithm [13] for frequent subgraphs. Here the original PrefixSpan algorithm allows any size of gaps in subsequences, but we restrict to only consecutive subsequences. This is because input sequences are amino acid sequences, which are usually long and consist of only twenty amino acids, meaning that if we allow any size of gaps, small subsequences are likely to be frequent, by which mining subsequences in protein sequences will be infeasible.

7) This depth-first traversal, which is similar to the gSpan and PrefixSpan algorithms, gives a practically efficient algorithm.

8) We can use any algorithm for mining frequent subgraphs (subsequences) to compute the marginal support in $\mathcal{Q}$ with the traversal over $\mathcal{T}_g$ (and $\mathcal{T}_s$). As mentioned above, for this purpose, we use gSpan and PrefixSpan for graphs and sequences, respectively.

References

[1] Apsel, B., Blair, J., Gonzalez, B., Nazif, T., Feldman, M., Aizenstein, B., Hoffman, R., Williams, R., Shokat, K., Knight, Z. (2008) Targeted polypharmacology: discovery of dual inhibitors of tyrosine and phosphoinositide kinases. *Nat. Chem. Biol.*, 4, 691-699.

[2] Campillos, M., Kuhn, M., Gavin, A., Jensen, L., Bork, P. (2008) Drug target identification using side-effect similarity. *Science*, 321, 263-266.

[3] Frantz, S. (2005) Drug discovery: playing dirty. *Nature*, 437, 942-943.

[4] Yildirim, M., Goh, K., Cusick, M., Barabasi, A., Vidal, M. (2007) Drug-target network. *Nat. Biotechnol.*, 25, 1119-1126.

[5] Morphy, R., Rankovic, Z. (2007) Fragments, network biology and designing multiple ligands. *Drug Discov. Today*, 12, 156-160.

[6] Hopkins, A. (2008) Network pharmacology: the next paradigm in drug discovery. *Nat. Chem. Biol.*, 4, 682-690.

[7] Paolini, G., Shapland, R., van Hoorn. W., Mason. J., Hopkins. A (2006) Global mapping of pharmacological space. *Nat. Biotechnol.*, 24, 805-815.

[8] Takigawa, I., Tsuda, K. and Mamitsuka, H. (2011) Mining significant substructure pairs for interpreting polypharmacology in drug-target network. *PLoS One*, 6(2), e16999.

[9] Wishart, D., Knox, C., Guo, A., Cheng, D., Shrivastava, S., Tzur, D., Gautam, B., Hassanali, M. (2008) DrugBank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic Acids Res.* 36, D901-906.

[10] Han, J., Cheng, H., Dong, X. and Yan, X. (2007) Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1), 55-86.

[11] Cordell, H. (2002) Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. *Hum. Mol. Genet.* 11, 2463-2468.

[12] Pei, J., Han, J., Mortazavi-Asl, B., Wan, J., Pinto, H., Chen, Q., Dayal, U. and Hsu, M-C. (2004) Mining sequential patterns by pattern-growth: the PrefixSpan approach. *IEEE Trans. Knowl. Data Eng.*, 16(11), 1424–1440.

[13] Yan, X. and Han, J. (2002) gSpan: Graph-based substructure pattern mining. *IEEE International Conference on Data Mining (ICDM'02)*, 721–724, Washington, DC, USA, 2002.