Bioinformatics, 36(9), 2020, 2649–2656
doi: 10.1093/bioinformatics/btaa008
Advance Access Publication Date: 28 January 2020
Original Paper

Genome analysis

Heterogeneous networks integration for disease–gene prioritization with node kernels

Van Dinh Tran[1], Alessandro Sperduti[2], Rolf Backofen[1,3] and Fabrizio Costa[4,*]

[1]Bioinformatics Group, Department of Computer Science, University of Freiburg, Freiburg im Breisgau, Germany, [2]Department of Mathematics, University of Padova, Padua, Italy, [3]Signalling Research Centres BIOSS and CIBSS, University of Freiburg, Germany and [4]Department of Computer Science, University of Exeter, Exeter, UK

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on July 22, 2019; revised on December 19, 2019; editorial decision on January 2, 2020; accepted on January 23, 2020

Abstract

Motivation: The identification of disease–gene associations is a task of fundamental importance in human health research. A typical approach consists in first encoding large gene/protein relational datasets as networks due to the natural and intuitive property of graphs for representing objects' relationships and then utilizing graph-based techniques to prioritize genes for successive low-throughput validation assays. Since different types of interactions between genes yield distinct gene networks, there is the need to integrate different heterogeneous sources to improve the reliability of prioritization systems.

Results: We propose an approach based on three phases: first, we merge all sources in a single network, then we partition the integrated network according to edge density introducing a notion of edge type to distinguish the parts and finally, we employ a novel node kernel suitable for graphs with typed edges. We show how the node kernel can generate a large number of discriminative features that can be efficiently processed by linear regularized machine learning classifiers. We report state-of-the-art results on 12 disease–gene associations and on a time-stamped benchmark containing 42 newly discovered associations.

Contact: f.costa@exeter.ac.uk

Availability and implementation: Source code: https://github.com/dinhinfotech/DiGI.git.

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The identification of causal links between genes and diseases is one of the main goals in human health research. To this end, biologists use high-throughput techniques to obtain vast amounts of data from which to derive some initial hypothesis. These hypotheses are then refined until a carefully selected subset is finally experimentally validated using low-throughput techniques to ascertain the mechanistic and causal relationships. One of the main challenges is how to efficiently exploit the large amount of genomics data to accelerate the pace of discoveries. A common approach involves starting from a set of genes which are believed to be causally related to a specific disease and then identify, in the set of remaining genes, those missing genes that are potentially involved. This is done by exploiting associations between genes (or the proteins they code for) based on protein–protein interactions, shared biochemical pathways, etc. One can, therefore, formulate a *ranking problem*, where the desired output is the prioritization of genes according to their probability of being associated with the disease. Ideally, limiting the expensive experimental validation to the most probable genes only, we can increase the overall efficiency of the gene–disease discovery process.

Recently, several approaches have been proposed that encode large amounts of relational information using the formalism of networks (Moreau and Tranchevent, 2012), motivated by (i) the ease with which graphs can encode relational data and (ii) the observation that genes responsible for similar diseases are often found in the linked neighborhood of one another (the *guilt-by-association* hypothesis). A successful family of prioritization approaches is based on graph kernels, which is a flexible way to induce predictive estimators well suited for discrete structures, such as graphs.

Integrating heterogeneous sources of information is, however, not an easy task. The heterogeneity arises from the different type of relationships that can be used to build the network. These can range from genomics proximity, to co-expression, from experimentally validated physical interactions, to associations mined from medical literature. It is, therefore, desirable to integrate different heterogeneous data sources in a flexible way so as to improve the accuracy and reliability of the predictive estimate by compensating the noise present in one source with more reliable information in another. When we consider the stage at which the data sources are combined, multiple sources gene prioritization approaches can be divided into two main groups.

The first group consists of methods that define similarities between genes from each data source separately and then combine the similarities in a single gene similarity notion. EPU (Yang *et al.*, 2014) [a modification of PUDI (Yang *et al.*, 2012)] is an approach that, differently from other methods that treat unknown genes homogeneously, partitions the unknown gene set in multiple positive and negative sets with different confidence scores (weights). It then employs an ensemble procedure to compute the final score. In Chen *et al.* (2011), a method named DIR is proposed that integrates multiple data sources using a unified representation. It then adopts a diffusion kernel to define a pairwise gene similarity. Finally, it integrates all rank scores to select the most informative evidence among a set of data sources. ProDiGe (Mordelet and Vert, 2011) is an efficient method that uses multiple kernel learning (MKL) to combine several data sources. The global pairwise similarity between genes is computed combining the similarity between auxiliary gene information, encoded as a feature vector and the diffusion kernel. Then a MKL algorithm is used to combine pre-defined kernels and learn an optimal combined kernel. Note that while the number of data sources can be arbitrary, only a single type of kernel is used. In Chen *et al.* (2014), a kernel-based Markov random field (MRF) algorithm is proposed. From a prior probability vector, the MRF algorithm computes the posterior which is interpreted as the probability of each candidate gene to be related to the selected disease. A modification of the Gibbs sampling procedure allows to weight differently each data source. This approach is flexible and accurate but has a high time complexity. F3PC (Chen *et al.*, 2015) uses a modified conditional random field model, that simultaneously utilizes both gene annotations and gene interactions while preserving their original representation. Scuba (Zampieri *et al.*, 2018) is a scalable MKL method that can deal with a high number of data sources in linear time complexity w.r.t number of sources and a constant memory consumption. A common drawback of these approaches is that, while different sources can be weighted differently, all nodes (genes) in a given network are considered equally important. This hinders the capacity to finely discriminate some genes as more relevant for a specific disease than others.

The second group includes methods that merge networks obtained from different sources and then define a notion of gene similarity on the resulting global network. An example of a system that belongs to the second group is RWR (Köhler *et al.*, 2008) that merges all networks and then uses a random walk with restart to induce a pairwise gene similarity notion. Another example is SmuDGE (Alshahrani and Hoehndorf, 2018), where information on genes, phenotypes and diseases are combined in a single knowledge graph. Genes and diseases are then vectorized using skipgrams and the resulting encoding is finally processed by a neural network to learn a prioritization model. This type of approaches does not distinguish the importance of individual sources and combines them before the definition of a similarity notion. As a result, the characteristics of each network are lost and the properties of individual graphs are generally not further exploited.

A number of methods are available also as web servers, such as Suspects (Adie *et al.*, 2006), ToppGene (Chen *et al.*, 2007), GeneDistiller (Seelow *et al.*, 2008), GeneWanderer (Köhler *et al.*, 2008), Posmed (Kobayashi and Toyoda, 2011), Candid (Hutz *et al.*, 2008), Endeavor (Aerts *et al.*, 2006) and Pinta (Nitsch *et al.*, 2010). See Börnigen *et al.* (2012) for a review.

Integration system based on graph kernels often employ a MKL strategy (Aiolli and Donini, 2015; Gönen and Alpaydin, 2011; Wang *et al.*, 2015; Zampieri *et al.*, 2018), where specific kernel similarities are defined for each source of information separately and the overall similarity notion is then expressed as weighted combination of the individual kernels, with weights that need to be specifically tuned for each disease.

In this article, we propose a method, named Disjunctive Graph Integration (DiGI), that does not suffer from these disadvantages. We develop an efficient node kernel that considers features from all pairs of sources once these have been merged in a single network. The node kernel is not based on a diffusion notion, as in most of the other proposals, but rather operates via a decomposition approach.

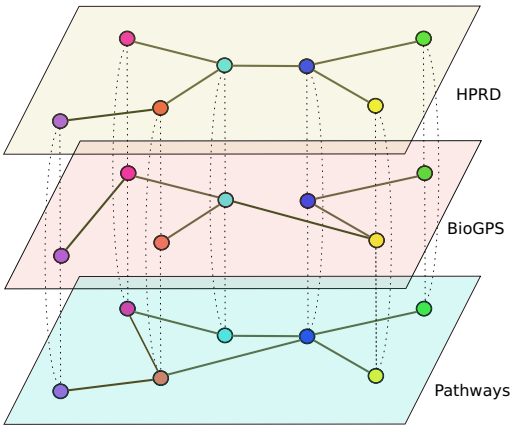

Fig. 1. Information integration: each information source [HPRD (Chatr-Aryamontri *et al.*, 2015), BioGPS (Wu *et al.*, 2009) and Pathways (Kanehisa and Goto, 2000; Schaefer *et al.*, 2009; Vastrik *et al.*, 2007; Whirl-Carrillo *et al.*, 2012)] is encoded as a graph in a distinct layer with nodes representing genes. All pairs of nodes (marked with the same color) that refer to the same gene are connected to form a single unified network

This yields a large number of highly specific features that can be exploited by regularized estimators, such as support vector machines, to obtain significantly improved predictive accuracy. We show that we can achieve state-of-the-art results on a time-stamped benchmark in which predictions are made and only benchmarked later on when enough new data have accumulated.

2 Materials and methods

DiGI system introduces some novel key concepts in order to integrate heterogeneous networks in an effective way: firstly, we distinguish relations that are important when considered jointly (conjunctive), from relations that are useful to give contextual hints (disjunctive); secondly, we introduce decomposition techniques to partition the set of edges in the combined network in these two types; and lastly, we introduce an efficient node graph kernel that is aware of and can exploit the disjunctive/conjunctive distinction.

Sources combination: we first build a network representation of the information source by considering all genes as nodes and materializing an edge between two genes if the relationship is considered sufficiently certain (the thresholds will depend on the type of data source). Note that, here, we consider the set intersection of all genes, i.e. all genes that are in common among all data sources. All nodes are then uniformly labeled using only the information source identifier (i.e. the database name) and no auxiliary information or gene identifier is employed at this stage. Using a more complex gene representation (e.g. using a vector to encode functional descriptors) is left for future work. We then combine the various networks by linking all pairs of nodes that represent the same gene in different sources (see Fig. 1). Note that considering sources that have a different 'density' (i.e. average number of relations per gene) is not an issue, since our approach will ultimately operate on a single combined graph.

Node kernel: a common strategy to build node kernels for gene networks is to use the notion of *diffusion*, i.e. spreading the information available on whether a gene is known to be related or not, in a discounted fashion along the network connections. The disadvantage of this approach is that information on the exact topological configuration of various parts of the network is lost and only distances are considered. Here, we expand the recent work presented in Van *et al.* (2017) and propose instead to use a more expressive decomposition kernel. The idea is to extract fragments from the local neighborhood around genes of interest and consider those as features. These can represent notions, such as 'at two hops away there is a gene with a high degree connected to two genes with very low degree', and many others. To be more precise, we consider as

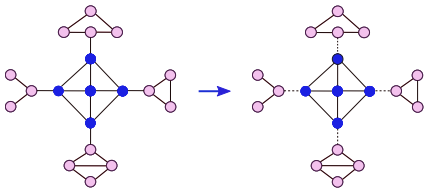

Fig. 2. *K*-core decomposition: nodes are partitioned according to their degree: given a degree threshold (here $D = 3$) all nodes with degree less or equal than D are considered in part 1 (nodes with light color) and the others in part 2 (nodes with dark color). Edges between nodes in different parts are marked as disjunctive (represented with dashed line) (center), the others as conjunctive. The procedure is iterated until there are no nodes with degree exceeding the threshold (right)

features all nearby pairs of fragments, where we distinguish the structure of each fragment and their exact distance (a formal explanation of the feature extraction procedure is given below). The idea is to extract tens of thousands of features and then let regularized predictive methods learn their relative importance for each specific disease prediction task.

Decomposition issues: one drawback of decomposition methods, such as the one employed here, is that, in order to be efficient, these methods often employ exact subgraph matching, i.e. two subgraphs are tested for isomorphism and are not tested for a 'softer' notion of similarity that allows a successful match of two fragments when they share, say, 90% of the nodes. In our approach, we consider as fragments only *neighborhood* subgraphs, i.e. given a node, we consider the subgraph formed by all nodes that can be reached in a predetermined number of hops (the *radius* of the neighborhood). This strategy allows us to generate a reduced number of fragments and hence features (there are only as many neighborhoods of a fixed size as there are nodes in a graph) compared to the exponential number of possible subgraphs. Our approach is very efficient and works very well (see e.g. Costa and De Grave, 2010) on sparse graphs with small maximal node degree (like molecular graphs); however, biological networks often exhibit 'small-world' properties where the presence of a few 'hub' nodes (i.e. nodes with very high degree), causes short-circuits in the majority of paths between any two nodes. Our notion of features, i.e. of pairs of fragments at various distances, becomes in these cases *degenerate* since (i) only few integer distance values are possible and (ii) neighborhood subgraphs will often be very large as they will likely include high-degree nodes. In these cases, we end up obtaining only few fragments, which in addition are very large and very specific and will likely never match exactly anywhere else in the network, making learning and generalization next to impossible.

Edge type: to address the aforementioned issues, we have developed a decomposition kernel that distinguishes two edge types, called the *conjunctive* and the *disjunctive* edge type. The idea is to mark edges with an attribute that tells the kernel if the edge has to be considered in a 'soft/contextual' way (disjunctive) or it is mandatory (conjunctive), i.e. if the matching is to be tolerant to the absence of the edge or not. A *hub* node could then have all its edges marked as disjunctive and allow for a softer context match with other hub nodes without the need to share exactly all of them. The way we do this is to consider only conjunctive edges when we are building the neighborhood subgraphs, but to consider the disjunctive edges when computing distances between pairs of fragments (see Fig. 4).

Network decomposition: Tthe aim of network decomposition is to decompose a given network into a collection of liked subgraphs. There is an implicit trade-off when assigning the conjunctive/disjunctive type to edges: on the one hand, disjunctive edges are useful to allow soft matches, on the other hand, conjunctive edges are useful to build complex features. If we use only disjunctive edges, we obtain as features the occurrences of node labels (i.e. the label histogram) because our neighborhood can never extend beyond each individual node; if we use only conjunctive edges and we have nodes with high degree, as mentioned earlier, we end up obtaining features that occur only once in the whole dataset and no useful generalization can take place. To strike a balance, we employ two (non-

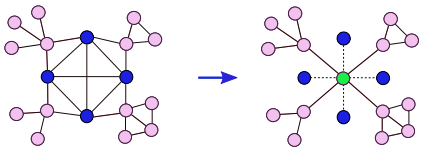

Fig. 3. Clique decomposition: a four-clique is abstracted by a new node. All edges incident on the original nodes are transferred to the new node. Original nodes are linked by disjunctive edges to the abstract node

adaptive) strategies: (i) the *k*-core decomposition (Fig. 2) and (ii) the clique decomposition (Fig. 3). In future work, we will devise adaptive strategies to identify edge types in a task dependent way. The *k*-core-decomposition strategy (see Fig. 2) consists in identifying nodes with high degree and separates them from the low-degree nodes. This partitioning forms connected components that can be iteratively decomposed. The edges in each component are marked as conjunctive while the edges across components (i.e. those edges that connect low degree nodes with high degree nodes) are marked as disjunctive. The clique-decomposition (see Fig. 3) strategy consists in replacing a clique (i.e. a complete subgraph where each node is connected to every other node) with a representative node and assigns to that node all incoming edges. The original nodes are still available but are now connected via disjunctive nodes and can therefore be matched in a soft way. These strategies allow us to decompose networks that have high degree nodes in low-degree components interconnected by disjunctive edges and hence preserving the original relational information but in a way that can now be conveniently processed by a highly discriminative node kernel.

Procedure pipeline: in summary, our approach consists of the following steps: (i) we define a node decomposition kernel that distinguishes between conjunctive and disjunctive edge types; (ii) we encode each information source as a network with genes as nodes and reliable relations as edges; (iii) we decompose each network applying the *k*-decomposition core first and the clique decomposition after; (iv) we join all information networks by linking all corresponding genes with disjunctive edges; (v) we use the node kernel as a feature constructor to extract a direct feature encoding for each node; (vi) for each disease, we fit a classifier (a regularized linear SVM in our case) to predict if a node belongs to the positive class (associated to the disease) or to the negative class (not associated); and (vii) we use the prediction confidence (i.e. the distance from the separating hyperplane) to impose a total order on the test instances and we rank them accordingly.

In the following, we provide the formal definitions of the notions previously introduced.

2.1 Background, definition and notations

We consider a graph as a tuple $G = (V, E, f_d, f_c)$, where V, E are the set of nodes and edges, respectively; $f_d : V \to \mathbb{L}$ is the discrete label function that assigns each node of graph a discrete label in $\mathbb{L}$. Discrete labels are used for graph isomorphism checking; $f_c : V \to \mathbb{R}^k$ is the real label function assigning each node a k-dimensional real attribute vector. We define the *distance* $D(u, v)$ between two nodes u and v, as the number of edges on the shortest path between them. The *neighborhood* of a node u with radius r, $N_r(u) = \{v \mid D(u, v) \leq r\}$, is the set of nodes at distance not greater than r from u. The corresponding *neighborhood subgraph* N_r^u is the subgraph induced by the neighborhood (i.e. the subgraph obtained considering all the edges with endpoints in the node set$N_r(u)$). The *degree* of a node u, $\deg(u) = |N_1^u|$, is the cardinality of its *neighborhood*. The maximum node degree in the graph G is indicated as $deg(G)$.

Disease–gene prioritization**:** Ggiven a set of genes associated to a genetic disease and a set of candidate genes, disease–gene prioritization is a task which aims at ranking the candidate genes based on their likelihood of being related to the disease.

Given a set of genes $\mathcal{G} = \{g_1, g_2, \ldots, g_N\}$ and a set of diseases $\mathcal{D} = \{d_1, d_2, \ldots, d_k\}$, we define the set of genes associated with

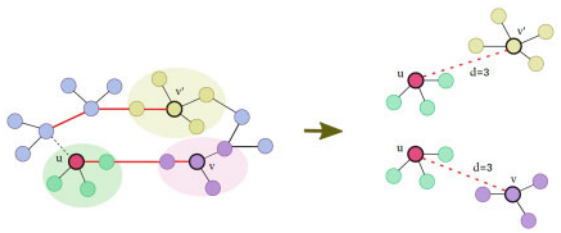

Fig. 4. Given a graph (left) a CDNK feature is defined as a pair of near neighborhood subgraphs. Each feature is defined by two neighborhood subgraphs with mutual distance d. Each neighborhood is defined by a root vertex and a radius r ($r = 1$ in this case). Two features are depicted in the figure (right): one has roots u and v and the second has roots u and v'. The distance between the neighborhood subgraphs is the length of the path (marked in red) between the respective roots. Note that a disjunctive edge (dashed line) works as a displacement: from u, we move to the node connected by the disjunctive edge and we consider that as the starting point for the path. (Color version of this figure is available at *Bioinformatics* online.)

disease d as $\mathcal{P}^d = \{g_1, g_2, \ldots, g_p\} \subset \mathcal{G}$, the set of genes not associated with disease d as $\mathcal{N}^d = \{g_1, g_2, \ldots, g_n\}$ and the set of candidate genes as $\mathcal{U}^d = \mathcal{G} \setminus (\mathcal{P}^d \cup \mathcal{N}^d)$. We call *gene prioritization* the task of ranking a candidate-set of genes $\mathcal{U}^d$ according to their likelihood of being related to disease d.

The conjunctive disjunctive node kernel (CDNK): Iin Van *et al.* (2017, 2018), we proposed the CDNK as an extension of the decomposition graph kernel, neighborhood subgraph pairwise distance kernel (NSPDK) (Costa and De Grave, 2010), to define a similarity notion between nodes within graphs rather than between entire graphs.

In NSPDK, a graph G is decomposed in features (pairwise neighborhood subgraphs) constituted by couples of subgraphs of radius r rooted at nodes of G which are at distance d. Given two rooted graphs A_u, B_v, where u and v are nodes of G, the relation $R_{r,d}(A_u, B_v, G)$ is true *iff* $D(u,v) = d$ and $A_u \cong N_r^u$ is (up to isomorphism $\cong$) a neighborhood subgraph of radius r of G as well as $B_v \cong N_r^v$. We denote with R^{-1} the inverse relation that returns all pairs of neighborhoods of radius r at distance d in G, $R_{r,d}^{-1}(G) = \{(A_u, B_v) | R_{r,d}(A_u, B_v, G) = true\}$. The kernel $\kappa_{r,d}$ over $\mathcal{G} \times \mathcal{G}$, counts the number of such fragments in common in two input graphs:

$$\kappa_{r,d}(G, G') = \sum_{\substack{(A_u, B_v) \in R_{r,d}^{-1}(G) \\ (A'_{u'}, B'_{v'}) \in R_{r,d}^{-1}(G')}} 1_{A_u \cong A'_{u'}} \cdot 1_{B_v \cong B'_{v'}},$$

where $1_{A \cong B}$ is the *exact matching function* that returns 1 if A is isomorphic to B and 0 otherwise. Finally, the NSPDK is defined as $K(G, G') = \sum_r \sum_d \kappa_{r,d}(G, G')$, where for efficiency reasons, the values of r and d are upper bounded by given maximal values r^* and d^*, respectively.

CDNK extends NSPDK and defines a node kernel $K(G_u, G_{u'})$ between two copies of the same network G where it distinguishes the nodes u and u', respectively. In addition, we consider a bipartition of the edge set in the conjunctive ($\wedge$) and disjunctive ($\vee$) part, i.e. $E = E^{\wedge} \cup E^{\vee}$, to induce an edge type notion. We denote a graph with $\wedge, \vee$ edges with $G^{\wedge\vee}$. We define the $K(G_u^{\wedge\vee}, G_{u'}^{\wedge\vee})$ between nodes u and u' belonging to $G^{\wedge\vee}$ as follows. When computing distances to induce neighborhood subgraphs, only conjunctive edges are considered. When choosing the pair of neighborhoods to form a single feature, we additionally consider roots u and v that are not at distance d but such that u is connected to w via a disjunctive edge and such that w is at distance d from v (see Fig. 4). In this way, disjunctive edges can still allow an *information flow* even if their endpoints are only considered in a pairwise fashion and not jointly.

In order to obtain a formal definition of the proposed kernel, we start by defining new specific notions of distance and neighborhood subgraph, which only depends on conjunctive edges. With $D^{\wedge}(u,v)$, we denote the length of a shortest path between u and v (belonging to $G^{\wedge\vee}$) where all edges are conjunctive edges. We can then define $N_r^{\wedge}(u) = \{v \,|\, D^{\wedge}(u,v) \leq r\}$ and the *conjunctive neighborhood subgraph* $N_r^{\wedge u}$ as the subgraph induced by $N_r^{\wedge}(u)$ only considering conjunctive edges. We can now define two relations: the *conjunctive relation* $R_{r,d,u}^{\wedge}(A_w, B_v, G^{\wedge\vee})$, which is true *iff* $w = u$ and $D^{\wedge}(w,v) = d$ and $A_w \cong N_r^{\wedge w}$, is (up to isomorphism $\cong$) a conjunctive neighborhood subgraph of radius r of $G^{\wedge\vee}$ as well as $B_v \cong N_r^{\wedge v}$; the *disjunctive relation* $R_{r,d,u}^{\vee}(A_w, B_v, G^{\wedge\vee})$ which is true *iff* $w = u$ and $A_w \cong N_r^{\wedge w}$ and $B_v \cong N_r^{\wedge v}$ are true and $\exists z$ s.t. $(D^{\wedge}(z,v) = d) \wedge ((w,z)$ is a disjunctive edge). We define $\kappa_{r,d}^{\wedge\vee}$ on the inverse relations $R_{r,d,u}^{\wedge -1}$ and $R_{r,d,u}^{\vee -1}$ as $\kappa_{r,d}^{\wedge\vee}(G_u^{\wedge\vee}, G_{u'}^{\wedge\vee}) = \mathbf{C} + \mathbf{D}$, where

$$\mathbf{C} = \sum_{\substack{A'_{u'}, B'_{v'} \in R_{r,d,u'}^{\wedge -1}(G^{\wedge\vee}) \\ A_u, B_v \in R_{r,d,u}^{\wedge -1}(G^{\wedge\vee})}} 1_{A_u \cong A'_{u'}} \cdot 1_{B_v \cong B'_{v'}},$$

$$\mathbf{D} = \sum_{\substack{A'_{u'}, B'_{v'} \in R_{r,d,u'}^{\vee -1}(G^{\wedge\vee}) \\ A_u, B_v \in R_{r,d,u}^{\vee -1}(G^{\wedge\vee})}} 1_{A_u \cong A'_{u'}} \cdot 1_{B_v \cong B'_{v'}}.$$

The CDNK is defined as $K(G_u^{\wedge\vee}, G_{u'}^{\wedge\vee}) = \sum_r \sum_d \kappa_{r,d}^{\wedge\vee}(G_u^{\wedge\vee}, G_{u'}^{\wedge\vee})$, where once again for efficiency reasons, the values of r and d are upper bounded to a given maximal r^* and d^*.

Direct feature extraction: Rrather than working in the kernel space with quadratic complexity, we follow Costa and De Grave (2010) and extend the explicit feature generation technique that can express $K(G, G')$ directly as $\langle \psi(G), \psi(G') \rangle$. We then build $\phi(G_u^{\wedge\vee})$ to express $K(G_u^{\wedge\vee}, G_{u'}^{\wedge\vee}) = \langle \phi(G_u^{\wedge\vee}), \phi(G_{u'}^{\wedge\vee}) \rangle$. In this article, $\phi(G_u^{\wedge\vee})$ is a feature constructor procedure that can take a typed network with a distinguished node and return a sparse vector representation in $\mathbb{R}^n$ with $n = 2^{30}$. To do so the key idea is to calculate a quasi-isomorphism certificate hash codes for the graph fragments. These hashes can then be used to compute codes for the pairs of neighborhood subgraphs at given distances and ultimately yield a direct feature indicator [see Costa and De Grave (2010) for full details]. The feature description associated to an individual node is finally obtained as the union of all the pairwise features that have one of the roots centered in that node.

Auxiliary node information integration: To integrate auxiliary information available on nodes, f_c, we upgrade the feature constructor procedure via $\Phi(G_u^{\wedge\vee}) = \phi(G_u^{\wedge\vee}) * f_c(u)$, where '*' is the convolution operation. In words, the vector representation for each node is obtained convolving the associated real vector with the CDNK sparse vector resulting from structural information as defined previously. The kernel, now suitable for graphs with real-valued vector labels, is defined as $K(G_u^{\wedge\vee}, G_{u'}^{\wedge\vee}) = \langle \Phi(G_u^{\wedge\vee}), \Phi(G_{u'}^{\wedge\vee}) \rangle$.

2.2 Information source integration

Each information source is encoded independently as a graph with genes as the node set and edges as reliable pairwise relations. The relations in each information source need to be thresholded as an independent and domain specific pre-processing step, i.e. edges are not associated to real-valued reliability scores, but rather edges that are above the threshold are materialized and those below are removed. Each resulting graph is called a 'layer' and is indicated as G^l where $l \in \{1, \ldots, L\}$ and L is the number of layers. Given two layers G^i and G^j and two nodes $g_u^i \in G^i$, $g_v^j \in G^j$ such that g_u^i and g_v^j identify the same gene $g_k \in \mathcal{G}$, i.e. $g_u^i \equiv g_v^j \equiv g_k$, we add a disjunctive edge with endpoints g_u^i and g_v^j (see Fig. 1).

2.3 Network decomposition

Decompositional graph kernels based on neighborhoods do not work well when nodes have high degrees. Unfortunately, in biological networks high degree nodes are likely to occur. To tackle this issue, Van *et al.* (2017) introduces a network decomposition procedure to obtain a collection of sparse sub-networks connected by disjunctive edges. The proposed decomposition is a two-step procedure

where the iterative k-core decomposition is followed by the clique decomposition.

Iterative k-core decomposition: The node set is partitioned in two groups on the basis of the degree of each node w.r.t. a threshold degree D. The node partition is used to induce the conjunctive versus disjunctive edge partition: edges that have endpoints in the same part are marked as conjunctive, while edges with endpoints in different parts are marked as disjunctive. We apply the k-core decomposition iteratively considering only the graph induced by the conjunctive edges until no node has a degree greater than D (the degree is defined by only considering incident conjunctive edges).

Clique decomposition: Cliques are abstracted in a new 'representative' node. All the cliques with a size (i.e. number of nodes) greater than a user defined threshold size C are identified. The endpoints of all edges incident on the clique's nodes are transferred to the representative node. Disjunctive edges are introduced to connect each node in the clique to the representative. Finally, all edges with both endpoints in the clique are removed.

2.4 Gene feature vectors

For each node/gene g_k, we employ CDNK to extract its corresponding explicit feature encoding. We then average all vectors across information layers, which correspond to nodes encoding the same gene:

$$\phi(g_k) = \frac{1}{L} \sum_{i,u|\mathcal{G}_u^i \equiv g_k} \phi(G_u^i)$$

for each $g_k \in \mathcal{G}$. The resulting vectors are then used to build the data matrix. This can then be processed by any machine learning classifier that can receive a sparse input format, such as most implementations of support vector machines.

3 Empirical evaluation

We evaluate the performance of DiGI in two experimental settings. In the first setting, *cross-validation*, we compare our method with other state-of-the-art methods for disease–gene prioritization on 12 disease–gene associations using a leave-one-gene-out setup. In the second setting, *unbiased*, we compare our method with popular web servers for disease–gene identification on a time-stamped benchmark containing 42 newly discovered associations.

3.1 Data sources

In the experiments, we use the following sources of gene information.

- Human Protein Reference Database (HPRD): a database of curated proteomic information pertaining to human proteins. It is derived from Keshava Prasad *et al.* (2009) with 9465 vertices and 37 039 edges. We employ the HPRD version used in Chatr-Aryamontri *et al.* (2015) that contains 7311 nodes and 30 503 edges.
- BioGPS (Van Dam *et al.*, 2015; Wu *et al.*, 2009): a gene co-expression graph (7311 nodes and 911 294 edges) constructed from the BioGPS dataset, which contains 79 tissues, measured with the Affymetrix U133A array. Edges are inserted when the pairwise Pearson correlation coefficient between genes is larger than 0.5.
- Pathways: pathway datasets are obtained from the database of KEGG (Kanehisa and Goto, 2000), Reactome (Vastrik *et al.*, 2007), PharmGKB (Whirl-Carrillo *et al.*, 2012) and PID (Schaefer *et al.*, 2009), which contain 280, 1469, 99 and 2679 pathways, respectively. A pathway co-participation network is constructed by connecting genes that co-participate in any pathway. It contains 7311 nodes and 2 254 822 edges.
- String: the String database gathers protein information covering seven levels of evidence: genomic proximity in procaryotes, fused genes, co-occurrence in organisms, co-expression, experimentally validated physical interactions, external databases and text mining. Overall, these aspects focus on functional relationships that can be seen as edges of a weighted graph, where the weight is given by the reliability of that relationship. To perform the unbiased evaluation, we employed the version 8.2 of String (Jensen *et al.*, 2009), from which we extracted functional links genes to construct a network with 16 113 nodes and 298 719 edges.
- Phenotype: the phenotype similarity dataset (Van Driel *et al.*, 2006) contains the similarities between phenotypes. For each phenotype, we collected from OMIM (version 2009) a set of genes which were known to be involved in the phenotype. We constructed a network in which each node is a gene that is related to at least one phenotype and a link between two genes is formed if there exists two of their corresponding phenotypes having a similarity $\geq$0.3. This yields a network with 2089 nodes and 126 294 edges.
- HumanNet-CF (Hwang *et al.*, 2019): co-functional network in which links are co-functional links from co-essentiality, co-expression, pathway database, protein domain profile associations, gene neighborhood and phylogenetic profile associations. This network includes 14 739 genes and 252 590 links.
- HumanNet-PI (Hwang *et al.*, 2019): protein–protein interaction network in which nodes are genes and links indicate that the corresponding proteins interact, according to high-throughput assays and literature. It contains 15 352 genes and 158 499 links.

3.2 Cross-validation evaluation

We follow the experimental setup used in Chen *et al.* (2015) and Zampieri *et al.* (2018) and select BioGPS, HPRD and Pathways as the information sources. In this experiment, 12 disease–gene association classes are selected based on the disease classification proposed in Goh *et al.* (2007) using OMIM data source, with at least 30 confirmed genes. The number of known causing disease genes regarding a single disease is often limited. Therefore, in order to have a sufficient number of positive genes for model training, we consider disease classes instead of individual diseases. For the sake of simplicity, we refer to each disease class as a disease.

For each disease $d \in \mathcal{D}$, a positive set, $\mathcal{P}^d$, is constructed containing all confirmed genes related to the disease. The negative set, $\mathcal{N}^d$, is a random sample from the set of known genes $\mathcal{G}$ of size $|\mathcal{N}^d| = \frac{1}{2}|\mathcal{P}^d|$, such that each gene is associated with at least one other disease (i.e. not the one that defines the corresponding $\mathcal{P}^d$), i.e. elements in $\mathcal{N}^d$ are sampled from $\cup_{i \neq d} \mathcal{P}^i$. In disease–gene prioritization tasks, often, given a specific genetic disease, the associated negative genes are not easily accessible, i.e. only the positive set is available. In our experimental setup, we consider as negative the set (of the same size as the positive set) of genes that are causing *other* diseases. The ratio here is that those genes that are known to cause a disease are well studied and their relation to the disease under consideration would have likely been uncovered if present.

We build a pair train-test set using a leave-one-out cross-validation strategy: a single gene g_k is extracted at random from the set $\mathcal{P}^d \cup \mathcal{N}^d$ and it is added to the candidate gene set $\mathcal{U}^d$; we train a classifier f on the training set $T^d = (\mathcal{P}^d \cup \mathcal{N}^d) \setminus \{g_k\}$ and estimate the classifier's predictive performance on the test set $\hat{T}^d = \mathcal{U}^d \cup \{g_k\}$.

We rank all genes $g_k \in \hat{T}^d$ using the classifier's output and compute the decision score $q_k = \frac{|\{g_l | f(g_k) \geq f(g_l)\}|}{|\hat{T}^d|}$. The score expresses the fraction of genes that are considered less related to disease d than the leave-one-out-gene under consideration.

Under this cross-validation setting, we perform two experiments. In the first, we select BioGPS, HPRD and Pathways as information

Table 1. Predictive performance comparison for a leave-one-gene-out cross-validation setting for disease–gene prioritization on 12 disease–gene associations problems with three information sources

	DIR	F3PC	MRF	GeneWanderer	Scuba	DiGI
AUC	71.6	83.0	73.1	71.1	**87.6**	**88.1**

Note: The highest performance in bold [DIR (Chen *et al.*, 2011), F3PC (Chen *et al.*, 2015), MRF (Chen *et al.*, 2014), GeneWanderer (Köhler *et al.*, 2008), Scuba (Zampieri *et al.*, 2018)].

sources. We then employ the same experimental setting as in Zampieri *et al.* (2018) in order to compare our method to those evaluated in Zampieri *et al.* (2018). We then collect all decision scores for all genes for all diseases and in Table 1, we report the average area under the receiver operating characteristic curve (AUC ROC).

In the second experiment, we use two recent data sources: HumanNet-CF (Hwang *et al.*, 2019) and HumanNet-PI (Hwang *et al.*, 2019). Table 2 shows the average performance over 12 disease–gene classes using a range of different performance metrics. Detailed experimental results and descriptions of the used metrics are available in Supplementary Table S1.

3.3 Unbiased evaluation

Cross-validated experimental setups in situations where instances cannot be considered independent (e.g. instances embedded in a network of pairwise dependencies) can yield over-optimistic estimates. A better approximation of the true predictive capacity of a method (an unbiased estimate) can be achieved when time stamps, the time when information is encoded into the datasets, for instances are available. In this way, one can estimate the answer for the question: what is it going to be the prediction accuracy of a given method in the future, when more instances from the same population are going to be available? We perform this type of experimental evaluation following the unbiased setting proposed in Börnigen *et al.* (2012) and similar in spirit to the CAFA strategy (Radivojac *et al.*, 2013). Here, we compare our method to approaches for disease–gene prioritization that are accessible as web servers.

In Börnigen *et al.* (2012), new disease–gene associations discovered during a 6-month period beginning in May 15, 2010 have been identified. These were used to build a set of 42 newly discovered disease–gene associations over 35 diseases. As soon as a new disease–gene association was uncovered, a series of gene prioritization web tools have been queried with a known positive gene set (∼17 genes for each query). Using this protocol, the researchers have been able to guarantee that the information regarding the disease–gene associations in the test set was not available during the training phase of the various tools.

To be able to compare DiGI on a fair basis, for all sources of information we consider a version release that is prior to May 15, 2010. In this setting, we use two sources: String v8.2 (Jensen *et al.*, 2009) and Phenotype (Van Driel *et al.*, 2006).

We now consider two cases that we call *genome-wide* and *candidate-set*. Formally, for each of the novel gene–disease association $i \in \{1, 2, \ldots, 42\}$, we are given access to a set $\mathcal{P}^i$ of genes that are known to be related to the given disease. Note that in 42 novel associations, we have 35 diseases with an average of ≈17 associated genes per disease.

In the genome-wide case, we build a negative training set considering 10 times the number of positive instances sampled at random from all the remaining genes, i.e. $\mathcal{N}^i \subset \mathcal{G} \setminus \mathcal{P}^i : |\mathcal{N}^i| = 10|\mathcal{P}^i|$. The training set is, therefore, $T^i = \mathcal{P}^i \cup \mathcal{N}^i$. Finally, the test set is built as the novel gene together with the remaining genes, i.e. $\hat{T}^d = \mathcal{G} \setminus T^i \cup g_i$. On average, each test set contains more than 16 K genes.

In the candidate-set case, we first construct a *candidate-set*, $\mathcal{U}^i$, for each novel association. The genes in the candidate-set are located within the chromosomal regions around the gene involved in the novel association (this is estimated using the Ensembl gene identifier). On average, the candidate-set contains 100 genes, i.e. $\hat{T}^d = \mathcal{U}^i \cup g_i$ with $|\mathcal{U}^i| \approx 100$. The negative training set is built as a random sample from the remaining genes up to 10 times the positive training set, i.e. $\mathcal{N}^i = \mathcal{G} \setminus (\mathcal{P}^i \cup \hat{T}^d) : |\mathcal{N}^i| = 10|\mathcal{P}^i|$.

Table 2. Leave-one-gene-out comparison on 12 disease–gene associations' problems with recent HumanNet-CF and HumanNet-PI information sources

	Scuba	SmuDGE	NSPDK	DiGI
$\overline{\text{fmax}}$	72.6	71.3	72.7	**75.4**
$\overline{\text{auc} - \text{roc}}$	73.7	67.1	73.1	**78.4**
$\overline{\text{aupr}}$	75.5	67.4	74.5	**79.7**
$\overline{\text{p} - \text{qscores}}$	30.0	29.2	29.0	**24.8**
$\overline{\text{rank}}$	2.73	3.08	2.62	**1.56**

Note: Best performance in bold.

In both cases, we normalize the ranks over the total number of genes (i.e. in the genome-wide case over 16 K and in the candidate-set case over 100). In Table 3, we report the median and the SD of the normalized ranks for the novel genes. We also report the true positive rate (TPR) at 5, 10 and 30% and the AUC ROC achieved by averaging over the 42 prioritizations.

In this last setting, we also tested our method augmented with auxiliary node information (denoted as DiGI_RV). We followed Van *et al.* (2018) and used the Gene Ontology to construct binary vectors representing bag-of-words encoding for each gene. The resulting representations are then clustered in k groups and each gene is finally equipped with a $k-$dimensional vector which expresses the distances to each cluster center (for details see Supplementary Material).

4 Results and discussion

In Table 1, we report a comparison of DiGI with other disease–gene association prediction approaches for the cross-validation setting and observe consistent improvements in the AUC ROC metric. Under the same setting, Table 2 shows comparison between DiGI and other methods using recent data sources and more evaluation measures. As can be seen from the table, DiGI shows best performances in all considered measures. It is worth to notice that, SmuDGE is trained using neural networks. This model normally shows potential results in case of having a sufficient number of training examples. However, in this setting, notwithstanding we consider disease class, the number of positive genes is still limited. This could be a reason for the relatively low performances of SmuDGE.

In Table 3, we compare DiGI with competitive approaches for disease–gene association prediction. We observe improved results in all metrics and in all settings (i.e. genome-wide and candidate-set). The rank and precision results are noteworthy, with a 30% relative error reduction (the error is computed as $(e_1 - e_2)/e_1$ where in our case, $e = 1-$TPR) on the best alternative (Scuba) in the TPR@5% and 26% in the TPR@10%. Considering the application case of these prioritization tools, i.e. the identification of a small subset of genes to be validated via expensive and time consuming low-throughput lab experiments, these improvements could potentially yield significant savings and conversely important speed up in the discovery rate of novel associations. Regarding the performance of the proposed method with the use of real node labels, DiGI_RV, it shows similar results as DiGI. However, in the median rank and TPR@5%, it reaches better results. It can be noticed that the better performance in TPR@5% is particularly relevant for practical use. In fact, already the top 5% of candidate genes constitutes a large number of genes to validate, so we can conclude that the use of real vector labels can be considered particularly beneficial.

Table 3. Performance of DiGI comparing with Scuba and other web tools on unbiased setting

Tool/method	Response rate (%)	Rank median (%)	TPR in top 5%	TPR in top 10%	TPR in top 30%	AUC (%)
Genome-wide						
Candid (Hutz *et al.*, 2008)	100	18.10	21.4	33.3	64.3	73
Endeavor (Aerts *et al.*, 2006)	100	15.49	28.6	38.1	71.4	79
Pinta (Nitsch *et al.*, 2010)	100	19.03	26.2	31.0	71.4	77
Scuba (Zampieri *et al.*, 2018)	100	10.55	33.3	47.6	78.6	80
DiGI	100	6.50	40.48	**64.28**	**88.1**	**87**
DiGI_RV	100	**4.73**	**52.4**	59.5	85.7	**87**
Candidate-genes						
Suspects (Adie *et al.*, 2006)	88.9[a]	12.77[a]	33.3[a]	33.3[a]	63.0[a]	76[a]
ToppGene (Chen *et al.*, 2007)	97.6	16.80	35.7	42.9	52.4	66
GeneWanderer-RW (Köhler *et al.*, 2008)	88.1	22.10	16.7	26.2	61.9	71
Posmed-KS (Kobayashi and Toyoda, 2011)	47.6	31.44	4.7	7.1	23.8	58
GeneDistiller (Seelow *et al.*, 2008)	97.6	11.11	26.2	47.6	78.6	85
Endeavor (Aerts *et al.*, 2006)	100	11.16	26.2	42.9	**90.5**	82
Pinta (Nitsch *et al.*, 2010)	100	18.87	28.6	31.0	71.4	75
Scuba (Zampieri *et al.*, 2018)	100	12.95	28.6	45.2	73.8	78
DiGI	100	6.47	47.62	**64.70**	88.1	**87**
DiGI_RV	100	**6.10**	**50.0**	59.5	88.2	86

Note: DiGI_RV: DiGI with real vector labels. The highest performance in bold.
[a]Values for Suspects were computed on the first 27 associations.

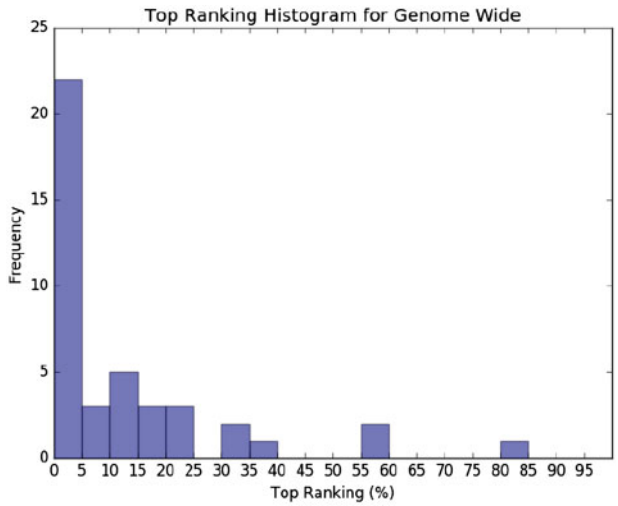

Fig. 5. Rank histogram for the 42 novel genes in the genome-wide setting

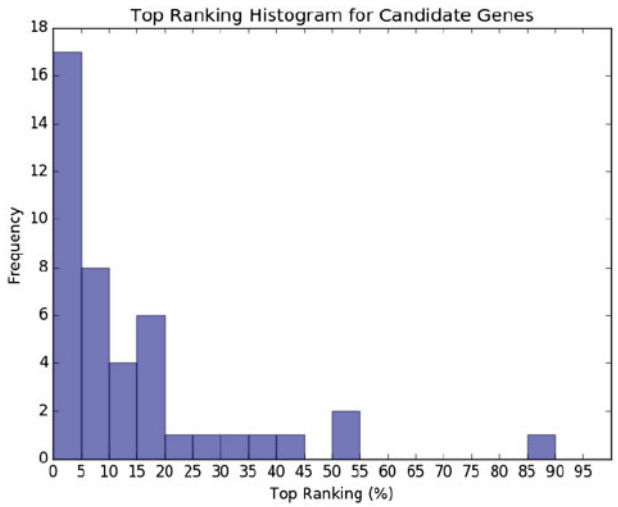

Fig. 6. Rank histogram for the 42 novel genes in the candidate-set setting

Finally, in Figures 5 and 6, we report the exact relative rank position of the 42 genes in the genome-wide and candidate cases. Most of the novel genes are highly prioritized with 22 (resp. 17) genes ranking in top 5% with 22 (resp. 17) genes in the genome-wide (candidate-set) case. Only one gene is not prioritized above the 50% threshold.

Web server: we provide a web server within the Freiburg RNA tools framework (Raden *et al.*, 2018) at http://rna.informatik.uni-freiburg.de/DiGI. The server accepts in input a list of identifiers for the genes that are believed to be associated with the disease of interest. The server uses the data sources introduced in to output a list of candidate genes ranked according to the method we proposed. We allow further analysis possibilities by providing links for each gene to the set of its neighbors according to each available data source.

5 Conclusion

We have proposed a novel approach to tackle disease–gene prioritization when multiple information sources are available. Our kernel method is not based on a notion of information diffusion but rather on the idea of extracting a large number of discriminative topological features from local neighborhood in a unified network. In future work, we will investigate how to incorporate additional real-valued node information.

Funding

This project was in part supported by the University of Padova, Strategic Project BIOINFOGEN; German Research Foundation (DFG) grant [BA 2168/3-3]; and Germanyer Excellence Strategy [CIBSS-EXC-2189 ID 390939984].

Conflict of Interest: none declared.

References

Adie,E.A. *et al.* (2006) SUSPECTS: enabling fast and effective prioritization of positional candidates. *Bioinformatics*, **22**, 773–774.

Aerts,S. *et al.* (2006) Gene prioritization through genomic data fusion. *Nat. Biotechnol.*, **24**, 537–544.

Aiolli,F. and Donini,M. (2015) EasyMKL: a scalable multiple kernel learning algorithm. *Neurocomputing*, **169**, 215–224.

Alshahrani,M. and Hoehndorf,R. (2018) Semantic Disease Gene Embeddings (SmuDGE): phenotype-based disease gene prioritization without phenotypes. *Bioinformatics*, **34**, i901–i907.

Börnigen,D. *et al.* (2012) An unbiased evaluation of gene prioritization tools. *Bioinformatics*, **28**, 3081–3088.

Chatr-Aryamontri,A. *et al.* (2015) The BioGRID interaction database: 2015 update. *Nucleic Acids Res.*, **43**, D470–D478.

Chen,B. *et al.* (2014) Identifying disease genes by integrating multiple data sources. *BMC Med. Genomics*, **7**, S2.

Chen,B. *et al.* (2015) A fast and high performance multiple data integration algorithm for identifying human disease genes. *BMC Med. Genomics*, **8**, S2.

Chen,J. *et al.* (2007) Improved human disease candidate gene prioritization using mouse phenotype. *BMC Bioinformatics*, **8**, 392.

Chen,Y. *et al.* (2011) In silico gene prioritization by integrating multiple data sources. *PLoS One*, **6**, e21137.

Costa,F. and De Grave,K. (2010) Fast neighborhood subgraph pairwise distance kernel. In: *Proceedings of the 26th International Conference on Machine Learning*. pp. 255–262. Omnipress, Madison, WI.

Goh,K.-I. *et al.* (2007) The human disease network. *Proc. Natl. Acad. Sci. USA*, **104**, 8685–8690.

Gönen,M. and Alpaydin,E. (2011) Multiple kernel learning algorithms. *J. Mach. Learn. Res.*, **12**, 2211–2268.

Hutz,J.E. *et al.* (2008) CANDID: a flexible method for prioritizing candidate genes for complex human traits. *Genet. Epidemiol.*, **32**, 779–790.

Hwang,S. *et al.* (2019) HumanNet v2: human gene networks for disease research. *Nucleic Acids Res.*, **47**, D573–D580.

Jensen,L.J. *et al.* (2009) STRING 8a global view on proteins and their functional interactions in 630 organisms. *Nucleic Acids Res.*, **37**, D412–D416.

Kanehisa,M. and Goto,S. (2000) KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, **28**, 27–30.

Keshava Prasad,T. *et al.* (2009) Human protein reference database2009 update. *Nucleic Acids Res.*, **37**, D767–D772.

Kobayashi,N. and Toyoda,T. (2011) Prioritising genes with an artificial neural network comprising medical documents to accelerate positional cloning in biological research. In: Suzuki, K. (ed.) *Artificial Neural Networks-Methodological Advances and Biomedical Applications*. IntechOpen, pp. 173–196.

Köhler,S. *et al.* (2008) Walking the interactome for prioritization of candidate disease genes. *Am. J. Hum. Genet.*, **82**, 949–958.

Mordelet,F. and Vert,J.-P. (2011) ProDiGe: prioritization of disease genes with multitask machine learning from positive and unlabeled examples. *BMC Bioinformatics*, **12**, 389.

Moreau,Y. and Tranchevent,L.-C. (2012) Computational tools for prioritizing candidate genes: boosting disease gene discovery. *Nat. Rev. Genet.*, **13**, 523–536.

Nitsch,D. *et al.* (2010) Candidate gene prioritization by network analysis of differential expression using machine learning approaches. *BMC Bioinformatics*, **11**, 460.

Raden,M. *et al.* (2018) Freiburg RNA tools: a central online resource for RNA-focused research and teaching. *Nucleic Acids Res.*, **46**, W25–W29.

Radivojac,P. *et al.* (2013) A large-scale evaluation of computational protein function prediction. *Nat. Methods*, **10**, 221–227.

Schaefer,C.F. *et al.* (2009) PID: the pathway interaction database. *Nucleic Acids Res.*, **37**, D674–D679.

Seelow,D. *et al.* (2008) GeneDistiller distilling candidate genes from linkage intervals. *PLoS One*, **3**, e3874.

Van,D.T. *et al.* (2017) The conjunctive disjunctive node kernel. In *ESANN, Bruges, Belgium*.

Van,D.T. *et al.* (2018) The conjunctive disjunctive graph node kernel for disease gene prioritization. *Neurocomputing*, **298**, 90–99.

Van Dam,S. *et al.* (2015) GeneFriends: a human RNA-seq-based gene and transcript co-expression database. *Nucleic Acids Res.*, **43**, D1124–D1132.

Van Driel,M.A. *et al.* (2006) A text-mining analysis of the human phenome. *Eur. J. Hum. Genet.*, **14**, 535–542.

Vastrik,I. *et al.* (2007) Reactome: a knowledge base of biologic pathways and processes. *Genome Biol.*, **8**, R39.

Wang,X. *et al.* (2015) Kernel methods for large-scale genomic data analysis. *Brief. Bioinform.*, **16**, 183–192.

Whirl-Carrillo,M. *et al.* (2012) Pharmacogenomics knowledge for personalized medicine. *Clin. Pharmacol. Ther.*, **92**, 414–417.

Wu,C. *et al.* (2009) BioGPS: an extensible and customizable portal for querying and organizing gene annotation resources. *Genome Biol.*, **10**, R130.

Yang,P. *et al.* (2012) Positive-unlabeled learning for disease gene identification. *Bioinformatics*, **28**, 2640–2647.

Yang,P. *et al.* (2014) Ensemble positive unlabeled learning for disease gene identification. *PLoS One*, **9**, e97079.

Zampieri,G. *et al.* (2018) Scuba: scalable kernel-based gene prioritization. *BMC Bioinformatics*, **19**, 23.