The Conjunctive Disjunctive Graph Node Kernel for Disease Gene Prioritization

Dinh Tran Van, Alessandro Sperduti

Department of Mathematics, Padova University, Trieste, 63, 35121 Padova, Italy

Fabrizio Costa

Department of Computer Science, University of Exeter Exeter EX4 4QF, UK

Abstract

Biological relations encoded data are naturally represented in the form of graphs. Therefore, there is a high number of graph-based biological learning systems have been proposed and shown promissing results. One of the key points which determines the performance of graph-based learning systems is the denifition of node similarity measure. Node similarities are normally measured by graph node kernels. However, most available graph node kernels do not show high discriminative capacity since they share two common limitations. First they are based on the diffusion phenomenon which does not effectively exploit the nodes' configuration. Second they are not able to take into account the available information associated to nodes of graph.

In this paper, we propose an efficient graph node kernel that not only effectively exploits nodes' context, but also can integrate information which are available on the graph nodes. Empirical evaluation show that our proposed graph node kernel show state of the art concerning graph node kernels.

Keywords: Graph node kernels, graph decomposition, disease gene prioritization

1. Introduction

Node similarity measure definition is the key that determines the performance of graph-based biological learning systems. The state of the art graph

node kernels used to measure node similarity, are based on the notion of information diffusion, including LEDK [1], MEDK [2], MDK [3], RLK [4], etc. These graph node kernels often show relatively low discriminative capacity, especially in cases of working with sparse graphs or graphs whose high numbers of missing links, since they share following limitations. First information is processed in an additive and independent fashion which prevents them from accurately modeling the configuration of each node's context. Second, they do not take into account information associated to nodes of graphs when they are available. These additional information normally provide a complement to graph topology. Therefore, they are potential to use to improve the expressiveness of graph node kernels.

We propose an effective convolutional graph node kernel, named *Conjunctive disjunctive node kernel* (CDNK) which is able to *i*) effectively exploit the nodes' context, *ii*) process with information sticked on nodes of graphs.

Predictive systems for gene-disease associations are often based on a notion of similarity between genes. A common strategy is to encode relations between genes as a network and to use graph based techniques to make useful inferences. In the last decades, a number of graph kernel methods have been proposed that directly exploit transitive properties in biological networks. The prototypical method is the Diffusion kernel (DK) [1] inspired by the heat diffusion phenomenon. The key idea is to allow a given amount of *heat* placed on nodes to *diffuse* through the edges. The similarity between two nodes v_i, v_j is then defined as the amount of heat starting from v_i and reaching v_j within a given time interval. In DK the heat flow is proportional to the number of paths connecting two nodes, which introduces an undesired bias that penalize peripheral nodes w.r.t. central ones. This problem is tackled by a modified version of DK called Markov exponential diffusion kernel (MED) [2] where a Markov matrix replaces the Laplacian matrix. Another kernel called Markov diffusion kernel (MD) [3], exploits instead the notion of *diffusion distances*, a measure of similarity between patterns of heat diffusion. The Regularized Laplacian kernel (RL) [4] represents instead a normalized version of the random walk with

restart model and defines the node similarity as the number of paths connecting two nodes with different lengths. All these approaches can be applied to dense networks with high degree nodes. A drawback of these approaches is however their relatively low discriminative capacity. This is in part due to the fact that information is processed in an additive and independent fashion which prevents them from accurately modeling the configuration of each gene's context. To address this issue here we propose to employ a *decompositional* graph kernel (DGK) [5] technique in which the similarity function between graphs can be formed by decomposing each graph into subgraphs and by devising a valid local kernel between the subgraphs. To exploit its higher discriminative capacity we first decompose the network in a collection of connected sparse graphs and then we develop a suitable kernel, that we call Conjunctive Disjunctive Node Kernel (CDNK).

[6], [2], [7]

2. Background

In this section, we first introduce definitions and notations that are used to define our proposed method. We then present graph node kernels and problems of state of the art graph node kernel.

2.1. Definitions and Notations

A graph is a structure, notated as $G = (\mathbb{V}, \mathbb{E})$ where $\mathbb{V}$ is the set of nodes and $\mathbb{E}$ is the set of links (edges).

We define the shortest path between u and v, notated as $\mathcal{D}(u, v)$, as the number of edges on the shortest path between them. The *neighborhood* of a node u with radius r, $N_r(u) = \{v \mid \mathcal{D}(u, v) \leq r\}$, is the set of nodes at distance no greater than r from u. The corresponding *neighborhood subgraph* $\mathcal{N}_r^u$ is the subgraph induced by the neighborhood (i.e. considering all the edges with endpoints in $N_r(u)$). The *degree* of a node u, $deg(u) = |\mathcal{N}_1^u|$, is the cardinality of its neighborhood for $r = 1$. The maximum node degree in the graph G is $deg(G)$.

Definition 1. *An adjacency matrix* $\boldsymbol{A}$ *is a symmetric matrix used to characterize the direct links between vertices* v_i *and* v_j *in the graph. Any entry* A_{ij} *is equal to* w_{ij} *when there exists a link connecting* v_i *and* v_j*, and is 0 otherwise..*

Definition 2. *The Laplacian matrix* $\boldsymbol{L}$ *is defined as* $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$*, where* $\boldsymbol{D}$ *is the diagonal matrix with non-null entries equal to the summation over the corresponding row of the adjacency matrix, i.e.* $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$*.*

Definition 3. *Transition matrix of a graph* G*, notated as* P*, is a matrix whose each element* $P_{ij} = A_{ij} / \sum_i A_{ij}$ *showing the probability of stepping on node* j *from node* i*.*

2.2. Kernels defined on graphs

Kernel methods, whose kernels, have emerged as one of the most powerful framework in machine learning which has been successfully applied in various fields. Kernels can be considered as similarity functions that allow to measure the similarity between any couple of objects in feature space and with any type of data representation.

Formally, a kernel, k, on $\mathbb{X} \times \mathbb{X}$ with $\mathbb{X}$ is a set of objects, is a function defined as $k : \mathbb{X} \times \mathbb{X} \longmapsto \mathbb{R}$ such that i) k is symmetric, meaning that $k(x_1, x_2) = k(x_2, x_1)$, where $x_1, x_2 \in \mathbb{X}$, and ii) k is positive semi-definite, that is $\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j k(x_i, x_j) \geq 0$ for any $N > 0$, $c_i, c_j \in \mathbb{R}$, and $x_i, x_j \in \mathbb{X}$.

When the input of kernels is a set of graphs, we have graph kernels and when it is a set of graph nodes, we refer kernels as graph node kernels. The problem of developing kernels defined on structured data in general and graphs in particular is made possible thanks to release of decomposition kernels proposed in [5]. Examples are kernels proposed in [8, 9, 10, 11]. Following we describe NSPDK [11], a convolutional graph kernel which shows state of performance and it is later on used to define our proposed graph node kernel.

The Neighborhood Subgraph Pairwise Distance Kernel

The NSPDK [11] is an instance of convolution kernel [5] where given a graph $G \in \mathcal{G}$ and two rooted graphs A_u, B_v, the relation $R_{r,d}(A_u, B_v, G)$ is true *iff*

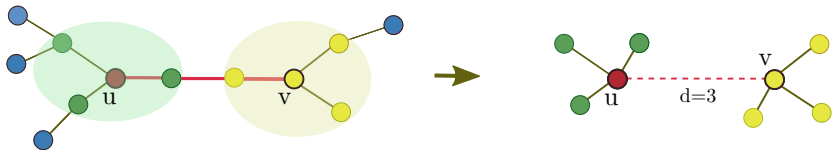

Figure 1: Pairwise neighborhood subgraphs for the "red" node with $r = 1$ and $d = 3$

$A_u \cong \mathcal{N}_r^u$ is (up to isomorphism $\cong$) a neighborhood subgraph of radius r of G and so is $B_v \cong \mathcal{N}_r^v$, with roots at distance $\mathcal{D}(u, v) = d$. Figure 2.2 illustrates a pairwise neighborhood subgraph. We denote R^{-1} as 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:

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

where $\mathbf{1}_{A \cong B}$ 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 to a given maximal r^* and d^*, respectively.

The state of the art kernels are based on diffusion phenomenon. In other words, they take paths connecting two nodes into account when measuring their similarity. In the following, we shortly introduce some of the most used graph node kernels.

2.2.1. Laplacian exponential diffusion kernel

One of the most well-known graph node kernels is the Laplacian exponential diffusion kernel **LEDK**, as it is widely used for exploiting discrete structures in general and graphs in particular. On the basis of the heat diffusion dynamics, Kondor and Lafferty proposed **LEDK** in [1]: imagine to initialize each vertex with a given amount of heat and let it flow through the edges until an arbitrary instant of time. The similarity between any vertex couple v_i, v_j is the amount of heat starting from v_i and reaching v_j within the given time. Therefore, **LEDK**

can capture the long range relationship between vertices of a graph to define the global similarities. Below is the formula to compute **LEDK** values:

$$K = e^{-\beta \mathbf{L}} = \mathbf{I} - \beta \mathbf{L} + \frac{\beta \mathbf{L}^2}{2!} - ... \tag{1}$$

where β is the diffusion parameter and is used to control the rate of diffusion and **I** is the identity matrix. Choosing a consistent value for β is very important: on the one side, if β is too small, the local information cannot be diffused effectively and, on the other side, if it is too large, the local information will be lost. **LEDK** is positive semi-definite as proved in [1].

2.2.2. Exponential diffusion kernel

In **LEDK**, the similarity values between high degree vertices are generally higher compared to those between low degree ones. Intuitively, the more paths connect two vertices, the more heat can flow between them. This could be problematic since peripheral nodes have unbalanced similarities with respect to central nodes. In order to make the strength of individual vertices comparable, a modified version of **LEDK** is introduced by Chen et al in [2], called Markov exponential diffusion kernel **MEDK** and given by the following formula:

$$\mathbf{K} = e^{-\beta \mathbf{M}} \tag{2}$$

The difference with respect to the Laplacian diffusion kernel is the replacement of **L** by the matrix $\mathbf{M} = (\mathbf{D} - \mathbf{A} - n\mathbf{I})/n$ where n is the total number of vertices in graph. The role of β is the same as for **LEDK**.

2.2.3. Markov diffusion kernel

The original Markov diffusion kernel **MDK** was introduced by Fouss et al. [3] and exploits the idea of diffusion distance, which is a measure of how similar the pattern of heat diffusion is among a pair of initialized nodes. In other words, it expresses how much nodes "influence" each other in a similar fashion. If their diffusion ways are alike, the similarity will be high and, vice versa, it will be low if they diffuse differently. This kernel is computed starting from the transition

matrix $\mathbf{P}$ and by defining $\mathbf{Z}(t) = \frac{1}{t}\sum_{\tau=1}^{t}\mathbf{P}^{\tau}$, as follows:

$$\mathbf{K} = \mathbf{Z}(t)\mathbf{Z}^{\top}(t) \tag{3}$$

2.2.4. Regularized Laplacian kernel

Another popular graph node kernel function used in graph mining is the regularized Laplacian kernel **RLK**. This kernel function is introduced by Chebotarev and Shamis in [4] and represents a normalized version of the random walk with restart model. It is defined as follows:

$$\mathbf{K} = \sum_{n=0}^{\infty}\beta^{n}(-\mathbf{L})^{n} = (\mathbf{I} + \beta\mathbf{L})^{-1} \tag{4}$$

where the parameter β is again the diffusion parameter. **RLK** counts the paths connecting two nodes on the graph induced by taking **-L** as the adjacency matrix, regardless of the path length. Thus, a non-zero value is assigned to any couple of nodes as long as they are connected by any indirect path. **RLK** remains a relatedness measure even when diffusion factor is large, by virtue of the negative weights assigned to self-loops.

3. Method

We start from the type of similarity notion computed by a neighborhood based decomposition kernel between graph instances [11], NSPDK, and adapt it to form a convolutional graph node kernel which aims at expressing the similarity between nodes in a single graph. Our intended graph node kernel takes an undirected, labeled graph as its input.

Given an input graph $G = (V, E)$, our method consists of three steps. In the first step, *node labeling*, a labeling function is proposed to assign label for every node of the graph. This step is necessary since we desire to design a convolutional kernel which deals with labeled graphs. The second step includes a graph decompostion procedure which allows to decompose the graph into a set of linked sparse connected components. In this procedure, we define two different kinds of link: *conjunctive* and *disjunctive* in which they are treated in

distinct manners later on. In the final step, the similarity between any node couple u and v is computed by adopting NSPDK on two neighborhood subgraphs rooted as u and v which extracted from the resulted graph after decomposing. Following, we describe each step in detail.

3.1. Node Labeling

We are interested in investigating biological networks in general and genetic networks in specific. Therefore, in this section, we propose two different approaches to label for genetic graphs. It is worth to notice that genetic networks typically represent genes as nodes labeled with a gene identifier. Here we take a different approach and use the node labels to encode abstract information about the genes. In this way downstream machine learning algorithms can generalize from similar examples and allow the identification of overlooked but related genes. We experiment with two types of information: *i*) topological information and *ii*) functional information.

Topological labels This information is simply based on the connectivity degree of the gene. The idea is that genes that have the same number of connections are more similar than genes with a different connectivity. Here we propose a node labeling function $\mathcal{L}$ that assigns labels for nodes by considering their degrees, such that nodes with similar degrees are assigned with a same label. Therefore, our function assigns the degree for nodes u having degree less than or equal to a user defined threshold T ($T = 5$ in our experimental evaluation). However degree values larger than T are subsequently discretized into k levels. We treat nodes differently between nodes with high degrees and low degrees since we envisage that the properties of low-degree nodes whose different node degrees are more dissimilar in comparison with the ones of high degrees. Formally, the labeling function is defined as:

$$\mathcal{L}(u) = \begin{cases} deg(u), & \text{if } deg(u) \leq T \\ T + i, & \text{if } deg(u) > T \end{cases},$$

where $i = \lceil \frac{deg(u)-T}{bin} \rceil$, $bin = \frac{deg(G)-T}{p-T}$ and p ($p > T$) is the maximum number of symbols used. The value of p depends on the degree distribution and can be tuned as a hyperparameter of the approach.

It is often that nodes in the graph are associated to extra information. These information can be considered as the complement to the graph topological information. Therefore, utilizing both information source are supposed to improve the performance of graph node kernels. In the following, we propose two different approaches to process with this kind of information. In the first approach, it is converted and used as the functional discrete labels for nodes, while it is processed and utilized as the real valued vectors in the second one. In both cases, we use *Gene Ontology* [12] as an example of extra information associated to the graph nodes.

Functional discrete labels We use a gene ontology to construct binary vectors representing a bag-of-words encoding for each gene (*i.e. each element of a binary vector is equal to 1 if its corresponding GO-term is associated to the gene, and is equal to 0 otherwise.*). The resulting vectors are then clustered using the k-means algorithm into a user defined number of classes K (tuned as a hyperparameter of the approach), so that genes with similar description profiles receive the same class identifier as label.

Functional real valued vector labels In addition to encoding the functional information as a discrete label we add a richer description by computing the similarity vector w.r.t. to each cluster. In this way we can fully exploit the latent description of the genes in terms of the different functional groups captured by the clustering procedure. Formally, given a vector $v \in I\!R^{26501}$ we compute a similarity vector $S(v) = s_1, s_2, \ldots s_K$ with entries $s_i = \frac{1}{1+\ell(v,c_i)}$ where $\ell(v, c_i)$ is the Euclidean distance of v from the center of the i^{th} cluster $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$ computed as the geometric mean of the elements in the cluster C_i.

3.2. Network Decomposition

In genetic networks it is not uncommon to find nodes with high degrees. Unfortunately these cases cannot be effectively processed by a neighborhood based decomposition kernel (see convolution kernels presented in **??**) since these are based on the notion of exact matches and the probability of finding identical neighborhoods decreases exponentially as the degree increases. This means that in a finite network it quickly becomes impossible to find any match and hence learn or generalize at all. Therefore, we propose a procedure to "sparsify" the network that is observed by the neighborhood kernel. In practice, we mostly keep the same the cardinality of the edge set. However we mark the edges with special attributes so that kernel is able to treat them differently when computing. The result is a procedure that decomposes the network in a linked collection of sparse sub-networks where each node has a reduced connectivity when considering the edges of a specific type. However the other edges are still available to connect the various sub-networks. We distinguish two types of edges: *conjunctive* and *disjunctive* edges. Nodes linked by conjunctive edges are going to be used jointly to define the notion of context and will be visible to the neighborhood graph kernel. Nodes linked by disjunctive edges are instead used to define features based only on the pairwise co-occurrence of the genes at the endpoints and are processed by our novel kernel.

Iterative k-core decomposition [13]: 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 first part contains all nodes with degree smaller than or equal to D and the second part the remaining ones. The node partition is used to induce the "conjunctive" vs "disjunctive" notion for the edge partition: edges that have both endpoints in the same part are marked as conjunctive, otherwise they are marked as disjunctive. We apply the k-core decomposition iteratively, where at each iteration we consider only the graph induced by the conjunctive edges. We stop iterating the decomposition after a user defined number of steps. Note that this decomposition does not alter the cardinality of the edge set, it is simply a procedure to mark each edge with the attribute conjunctive or disjunctive.

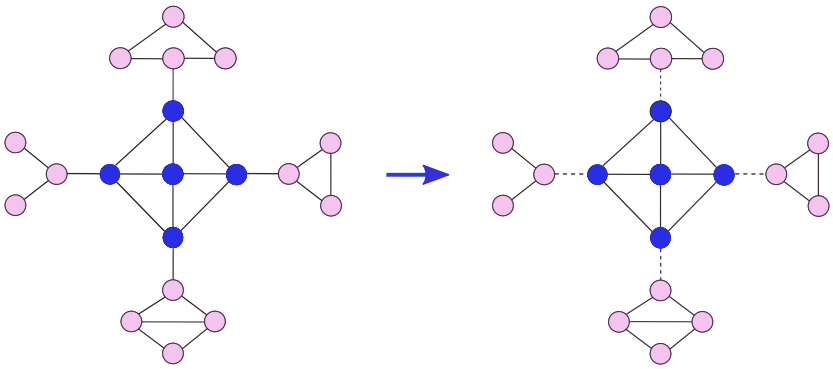

Figure 2: K-core decomposition with degree threshold $D = 4$

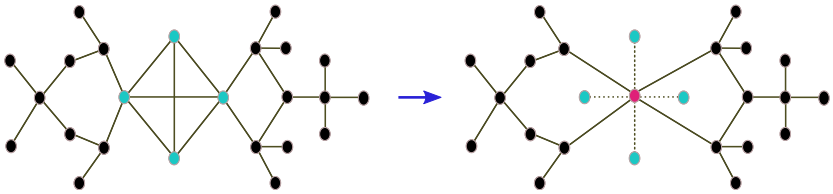

Figure 3: Clique decomposition with threshold $C = 4$

Clique decomposition [14]: To model the notion that nodes in a clique are tightly related, we summarize the whole clique with a new "representative" node. All the cliques (completely connected subgraphs) with a number of nodes greater than or equal to a given threshold size C are identified. The endpoints of all edges incident on the clique's nodes are moved 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.

In our work a network is transformed by applying first the iterative k-core decomposition and then the clique decomposition.

3.3. The Conjunctive Disjunctive Node Kernel

We start from the Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) [11] and adapt it to express the similarity between nodes in a single network. The key idea in NSPDK is to decompose graphs in small fragments and count how many pairs of fragments are shared between two instances. We introduce

two improvements: *i*) we partition the features according to the individual node's neighborhood, and *ii*) we introduce a distinction between "disjunctive" and "conjunctive" edges.

We define a node kernel $K(G_u, G_{u'})$ between two copies of the same network G where we distinguish the nodes u and u' respectively. The idea is to define the features of a node u as the subset of NSPDK features that always have the node u as one of the roots. In addition we distinguish between two types of edges, called *conjunctive* and *disjunctive* edges. 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 (Figure 4 is an illustration). 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.

Formally, we define two relations: the *conjunctive relation* $R^{\wedge}_{r,d}(A_u, B_v, G_u)$ identical to the NSPDK relation $R_{r,d}(A_u, B_v, G)$, and (ii) $\mathcal{D}(u, v) = d$; the *disjunctive relation* $R^{\vee}_{r,d}(A_u, B_v, G_u)$ is true *iff* (i) $A_u \cong \mathcal{N}^u_r$ and $B_v \cong \mathcal{N}^v_r$ are true, (ii) $\exists w$ s.t. $\mathcal{D}(w, v) = d$, and (iii) (u, w) is a disjunctive edge. We define $\kappa_{r,d}$ on the inverse relations ${R^{\wedge}_{r,d}}^{-1}$ and ${R^{\vee}_{r,d}}^{-1}$:

$$\kappa_{r,d}(G_u, G_{u'}) = \sum_{\substack{A_u, B_v \in {R^{\wedge}_{r,d}}^{-1}(G_u) \\ A'_{u'}, {B'}_{v'} \in {R^{\wedge}_{r,d}}^{-1}(G_{u'})}} \mathbf{1}_{A_u \cong A'_{u'}} \cdot \mathbf{1}_{B_v \cong B'_{v'}} + \sum_{\substack{A_u, B_v \in {R^{\vee}_{r,d}}^{-1}(G_u) \\ A'_{u'}, {B'}_{v'} \in {R^{\vee}_{r,d}}^{-1}(G_{u'})}} \mathbf{1}_{A_u \cong A'_{u'}} \cdot \mathbf{1}_{B_v \cong B'_{v'}}.$$

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

3.4. Real valued node information

In order to integrate the information of real vectors we proceed as follows. We compute a sparse vector representation for the neighborhood graph rooted in node v following [11]: for each neighborhood subgraph we calculate the quasi-isomorphism certificate hash code; we then combine the hashes for the pair

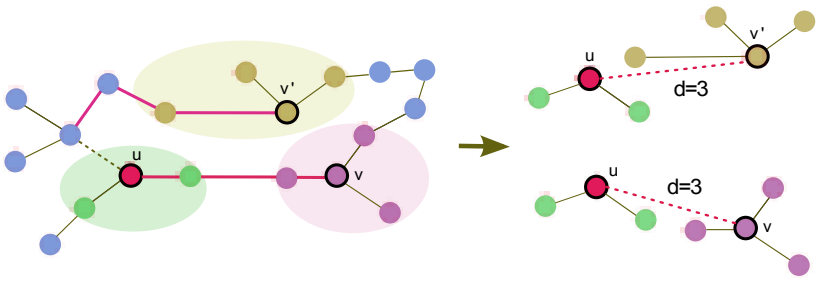

Figure 4: Pairwise neighborhood subgraphs for the "red" node with $r = 1$ and $d = 3$ using "conjuction" and "disjuctive" edges

of neighborhoods and use the resulting integer as a feature indicator. This yields a direct sparse vector representation (associated to node u in graph G) $f : G_u \longmapsto I\!R^L$ where $L \approx 10K - 1M$. Given the real valued vector information (associated to node u in graph G) $g : G_u \longmapsto I\!R^K$ computed as the multi-class similarity to the K clusters (c.f.r. Section 3.1), we update the computation of CDNK considering the discrete convolution of the discrete information with the real valued information:

$$K(G_u, G_{u'}) = \left\langle f(G_u) \bigotimes g(G_u), f(G_{u'}) \bigotimes g(G_{u'}) \right\rangle, \tag{5}$$

where the discrete convolution is defined as: $(f \bigotimes g)[n] = \sum_{m=0}^{K-1} f[n-m]g[m]$. In words, we are starting a scaled copy of the real valued vector at the position indicated by each feature computed on the basis of the discrete information. Intuitively, when both the real valued and the discrete information match, the kernel computes a large similarity, but if there is a discrepancy in either one of the sources of information, the similarity will be penalized.

4. Empirical Evaluation

In this section, we desire to evaluate the performance of CDNK and compare it with the state of the art graph node kernels. In order to do that, we employ disease gene prioritization problem where the aim is to build a learning system

which allows to prioritize candidate genes based on their probabilities of being associated to a given genetic disease.

Given a genetic graph and a list of training genes for a genetic disease (training set consists both positive and negative genes), we first apply a graph node kernel to compute kernel matrix. This kernel matrix together with training gene set are used as input of a learning algorithm to construct a system. The trained system is used to prioritize for candidate genes.

To compare the performance of graph node kernels, we fix the learning algorithm and substitute graph node kernels. The performance of the system obtained in different cases reflect the performance of graph node kernels.

Datasets: We carry out in two separate networks derived from BioGPS and Pathways datasets. Following are the short description of these networks. We carry out experiments on two separate datasets and we follow experiment's procedure carried out in [2] where 12 diseases [15] are used in each disease is associated to at least 30 positive genes. For each disease, we construct a positive set $\mathcal{P}$ with all positive (confirmed) disease genes, and a negative set $\mathcal{N}$ which contains random genes associated at least to one disease class which is not related to the class that is defining the positive set. In [2] the ratio between the dataset sizes is chosen as $|\mathcal{N}| = \frac{1}{2}|\mathcal{P}|$. The predictive performance of each method is evaluated via a leave-one-out cross validation: one gene is kept out in turn and the rest are used to train an SVM model. We compute a decision score q_i for the test gene g_i as the top percentage value of score s_i among all candidate gene scores. We collect all decision scores for every gene in the training set to form a global decision score list on which we compute the AUC-ROC.

- **BioGPS:** A gene co-expression network is constructed from BioGPS dataset, which contains 79 tissues, measured with the Affymetrix U133A array. Edges are inserted when the pairwise Pearson correlation coefficient (PCC) between genes is larger than 0.5.

- **Pathways:** Pathway information is retrieved from KEGG, Reactome, PharmGKB and the Pathway Interaction Database. If a couple of proteins

co-participate in any pathway, the two corresponding genes are linked.

Model Selection: The hyper parameters of the various methods are tuned using a k-fold strategy. However due to the non i.i.d. nature of the problem, we employ a stronger setup to ensure no information leakage. The dataset on which we are validating the performance is never subsequently used in the predictive performance estimate. The values for diffusion parameter in DK and MED are sampled in $\{10^{-3}, 10^{-3}, 10^{-2}, 10^{-1}\}$, time steps in MD in $\{1, 10, 100\}$ and RL parameter in $\{1, 4, 7\}$. For CDNK, the degree threshold values are sampled in $\{10,\ 15,\ 20\}$, clique size threshold in $\{4,\ 5\}$, maximum radius in $\{1,\ 2\}$, maximum distance in $\{2,\ 3,\ 4\}$, number of clusters K in $\{5,\ 7\}$. Finally, the regularization trade off parameter C for the SVM is sampled in $\{10^{-5},\ 10^{-4},\ 10^{-3},\ 10^{-2},\ 10^{-1}, 1,\ 10,\ 10^{2}\}$.

Table 1 and 2 show the AUC performance of the system corresponding to different cases of employing graph node kernels. Predictive models based on CDNK variations are ranked in top positions in all cases on BioGPS dataset and 9 out of 11 cases on Pathways when compared to diffusion based kernels. CDNK is ranking first when considering both the average AUC-ROC and the average rank with a difference, w.r.t. the best diffusion kernel, ranging from 5.4% to 10% and from 1.2% to 3.2% on BioGPS and Pathways, respectively. Regarding the variations of CDNK, the integration of real valued vector information improves the performance in most cases on using discrete labels only. The improvement in average ranges from 1.3% to 4.6%. Note that also in the case where only topological information is used to induce the discrete label (CDNK3 in the Table 1 and 2), the proposed approach improves on state-of-the-art.

5. Conclusions

We have shown how decomposing a network in a set of connected sparse graphs allows us to take advantage of the discriminative power of CDNK, a novel decomposition kernel, to achieve state-of-the-art results. Moreover, we have also introduced the way to integrate "side" information in form of real

Table 1: *Predictive performance on 11 gene-disease associations in percentage using network induced by the BioGPS. Best results in bold. We report the AUC ROC and the rank for each kernel method. CDNK1 = ontology for discrete labels, CDNK2 = ontology for both discrete and vector labels, CDNK3 = node degree for discrete labels and CDNK4 = node degree for discrete labels and ontology for vector label.*

	BioGPS							
Disease	DK	MD	MED	RL	CDNK1	CDNK2	CDNK3	CDNK4
1	51.9/8	57.4/7	59.0/6	59.2/5	65.1/4	69.5/2	69.3/3	**70.3/1**
2	81.7/5	78.5/6	75.2/7	75.0/8	88.3/2	**88.8/1**	85.1/4	86.8/3
3	64.3/7	59.6/8	71.6/3	71.8/2	65.5/5	**72.5/1**	64.7/6	66.4/4
4	65.3/7	58.2/8	67.8/6	67.8/5	71.9/4	**78.7/1**	73.9/3	77.0/2
5	64.0/8	64.1/7	66.5/5	66.2/6	75.9/4	76.2/3	76.9/2	**77.4/1**
6	74.6/5	70.2/8	71.0/7	71.2/6	79.3/2	**83.7/1**	76.7/4	79.0/3
7	73.0/5	66.7/8	75.4/3	75.6/2	68.8/6	73.9/4	67.3/7	**76.9/1**
8	74.4/8	76.8/3	76.2/5	76.4/4	74.7/7	**77.7/1**	76.0/6	76.8/2
9	71.5/2	65.6/8	67.7/5	69.9/3	66.8/7	**71.7/1**	67.1/6	68.1/4
10	54.0/6	50.3/8	56.1/5	51.1/7	77.6/4	**82.7/1**	80.5/2	80.0/3
11	58.2/7	51.3/8	59.3/6	59.3/5	71.8/4	**80.2/1**	75.3/3	77.1/2
$\overline{AUC}$	66.6	63.5	67.8	67.6	73.2	**77.8**	73.9	76.0
$\overline{Rank}$	6.18	7.18	5.27	4.82	4.45	**1.55**	4.18	2.36

Table 2: *Predictive performance on 11 gene-disease associations in percentage using network induced by the Pathways. Best results in bold. We report the AUC ROC and the rank for each kernel method. CDNK1 = ontology for discrete labels, CDNK2 = ontology for both discrete and vector labels, CDNK3 = node degree for discrete labels and CDNK4 = node degree for discrete labels and ontology for vector label.*

	Pathways							
Disease	DK	MD	MED	RL	CDNK1	CDNK2	CDNK3	CDNK4
1	74.7/8	76.4/7	78.7/6	78.8/5	80.2/3	**82.4/1**	80.6/2	79.4/4
2	55.1/8	64.9/7	76.6/6	76.6/5	81.1/2	80.3/3	79.8/4	**82.2/1**
3	55.0/8	62.7/7	64.1/5	65.6/4	67.1/3	63.6/6	68.5/2	**71.5/1**
4	54.3/8	65.2/7	**73.7/1**	73.7/2	66.1/5	68.1/3	67.5/4	65.6/6
5	52.9/8	55.7/7	62.7/5	62.7/6	68.3/2	**69.6/1**	66.3/4	66.8/3
6	83.4/8	92.7/7	96.5/2	**96.5/1**	93.0/6	94.1/5	94.4/4	95.5/3
7	84.5/8	88.3/7	89.4/3	89.5/2	88.5/6	88.5/5	88.7/4	**90.5/1**
8	53.7/8	65.6/7	72.0/6	72.3/5	72.5/3	72.5/2	72.3/4	**76.5/1**
9	52.5/8	64.9/5	64.2/7	64.2/6	**81.3/1**	81.0/2	79.6/3	78.8/4
10	68.8/6	65.4/8	74.4/5	74.4/4	66.9/7	76.8/2	76.1/3	**79.5/1**
11	53.7/8	69.2/7	74.6/5	74.1/6	77.0/2	**78.7/1**	75.4/4	77.0/3
$\overline{AUC}$	62.6	70.1	75.2	75.3	76.5	77.8	77.2	**78.5**
$\overline{Rank}$	7.82	6.91	4.64	4.18	3.64	2.82	3.45	**2.55**

valued vectors when it is available on graph to get even better performance of CDNK. In future work we will investigate how to *i*) decompose networks in a data driven way and *ii*) extend the CDNK approach to gene-disease association problems exploiting multiple heterogeneous information sources in a joint way.

Funding

This work was supported by the University of Padova, Strategic Project BIOINFOGEN.

References

[1] R. I. Kondor, J. Lafferty, Diffusion kernels on graphs and other discrete input spaces, in: ICML, Vol. 2, 2002, pp. 315–322.

[2] B. Chen, M. Li, J. Wang, F.-X. Wu, Disease gene identification by using graph kernels and markov random fields, Science China. Life Sciences 57 (11) (2014) 1054.

[3] F. Fouss, L. Yen, A. Pirotte, M. Saerens, An experimental investigation of graph kernels on a collaborative recommendation task, in: Data Mining, 2006. ICDM'06. Sixth International Conference on, IEEE, 2006, pp. 863–868.

[4] P. Chebotarev, E. Shamis, The matrix-forest theorem and measuring relations in small social groups, arXiv preprint math/0602070.

[5] D. Haussler, Convolution kernels on discrete structures, Tech. rep., Technical report, Department of Computer Science, University of California at Santa Cruz (1999).

[6] F. Mordelet, J.-P. Vert, Prodige: Prioritization of disease genes with multitask machine learning from positive and unlabeled examples, BMC bioinformatics 12 (1) (2011) 389.

[7] G. Valentini, A. Paccanaro, H. Caniza, A. E. Romero, M. Re, An extensive analysis of disease-gene associations using network integration and fast kernel-based gene prioritization methods, Artificial Intelligence in Medicine 61 (2) (2014) 63–78.

[8] N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, K. M. Borgwardt, Weisfeiler-lehman graph kernels, Journal of Machine Learning Research 12 (Sep) (2011) 2539–2561.

[9] T. Gärtner, A survey of kernels for structured data, ACM SIGKDD Explorations Newsletter 5 (1) (2003) 49–58.

[10] K. M. Borgwardt, H.-P. Kriegel, Shortest-path kernels on graphs, in: Data Mining, Fifth IEEE International Conference on, IEEE, 2005, pp. 8–pp.

[11] F. Costa, K. De Grave, Fast neighborhood subgraph pairwise distance kernel, in: Proceedings of the 26th International Conference on Machine Learning, Omnipress, 2010, pp. 255–262.

[12] G. O. Consortium, et al., The gene ontology (go) database and informatics resource, Nucleic acids research 32 (suppl 1) (2004) D258–D261.

[13] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, A. Vespignani, k-core decomposition: A tool for the visualization of large scale networks, arXiv preprint cs/0504107.

[14] R. E. Tarjan, Decomposition by clique separators, Discrete mathematics 55 (2) (1985) 221–232.

[15] K.-I. Goh, M. E. Cusick, D. Valle, B. Childs, M. Vidal, A.-L. Barabási, The human disease network, Proceedings of the National Academy of Sciences 104 (21) (2007) 8685–8690.