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

Due to advances in data acquisition technologies (such as next generation sequencing), large amounts of information on biological systems are being encoded in relational data bases at an increasing rate. State of the art systems capable to deal with such structured data at scale, include graph approaches based on the notion of diffusion kernels. In these systems, information is propagated in a discounted way according to the topology of the data network. A disadvantage of these approaches is that the specific configuration of the neighborhood of each node is ignored. Moreover it would be beneficial to integrate, when available, side information, encoded as arbitrary feature vectors associated to nodes or edges. In this paper, we propose a graph decomposition approach that can exploit the information present in the local topology and that can efficiently incorporate vectorial side information. We consider the challenging problem of disease gene prioritization and show that we can achieve state-of-the-art results.

Keywords: Graph node kernels, disease gene association

1. Introduction

The release of advanced technologies is one of the main reasons for the revolution in various scientific research fields. In Biological and Medical domain, modern technologies are making it not only easier but also more economical than ever to undertake experiments and creating applications. As a consequence, a vast amount of biological data in terms of volume and type is generated through scientific experiments, published literature, high-throughput experiment technology, and computational analysis. This huge quantity of data are saved as biological datasets and made discoverable through web browsers, application programming interfaces, scalable search technology and extensive cross-referencing between databases. Biological databases normally contain information about gene function, structure, localization, clinical effects of mutations and similarities of biological sequences and structures.

In Biomedical, disease-gene association recovery is a major goal in molecular that has received much attention from many researchers. Despite that fact that a big progress has been made in the last decades, a number of genes known to be related to a genetic disease is normally limited. In order to find out the complement set of the known disease gene set, one way is to search for whole genome or specific regions that often contain a large number of suspected genes (candidate genes). This is obvious not a good idea as it is expensive not only in term of time consuming but also from financial aspect. For this reason, a considerable number of gene prioritization methods have been proposed. 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 [1], [2], [3].

One of the key points that determines the performance of graph-based biological learning systems in general and graph-based disease gene prioritization systems in particular is the definition of node similarity measure. The node similarity is often measured by graph node kernels [4], [2], [5], [6]. However, the state of the art graph node kernels used to measure node similarity, are based on the notion of information diffusion. These graph node kernels often show relatively low discriminative capacity, especially in cases of working with sparse graphs, graphs have 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.

2. Background

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

2.1. Definitions and notations

A graph is a structure $G = (\mathbb{V}, \mathbb{E}, \mathcal{L}_1, \mathcal{L}_2)$ where $\mathbb{V}$, $\mathbb{E}, \mathcal{L}_1, \mathcal{L}_2$ are the vertex (node) set, link (edge) set, discrete labeling fuction and real vector labeling function, respectively. The functions $\mathcal{L}_1, \mathcal{L}_2$ are defined as:

- $\mathcal{L}_1 : \mathbb{V} \longmapsto \mathbb{L}$, where $\mathbb{L}$ is a set of discrete labels. $\mathcal{L}_1$ assigns a single discrete label $\ell \in \mathbb{L}$ for each ndoe $v \in \mathbb{V}$, $\mathcal{L}_1(v) = \ell$.

- $\mathcal{L}_2 : \mathbb{V} \longmapsto \mathbb{R}^n$. $\mathcal{L}_2$ assigns a single real vector label $(v_1, v_2, \ldots, v_n) \in \mathbb{R}^n$ for each ndoe $v \in \mathbb{V}$, $\mathcal{L}_2(v) = (v_1, v_2, \ldots, v_n)$.

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 value, $P_{ij} = A_{ij}/\sum_i A_{ij}$, is proportional to the probability of stepping on node j from node i.*

2.2. Kernels on graphs

Kernel methods have emerged as one of the most powerful framework in machine learning, which has been successfully applied in various domains, due to their modularity. In kernel methods, the definition of kernel functions is independent from the design of the learning algorithm. A kernel function can be considered as the similarity measure between example couples in the feature space. Interestingly, It can be defined 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 $x_i, x_j \in \mathbb{X}$.

Canonical machine learning methods take vectorial data, data that are represented by vectors of features, as their input. However, there are many fields where data are not naturally represented by vectors, but structured forms in which graph is one of the most popular representation. Therefore, the task of developing methods which are able to learn from structured data in general or graphs in particular is very important. In this thesis, we focus on graphs, a special type of structured data representation.

The problem of developing kernels defined on structured data in general and graphs in particular is made possible thanks to the release of decomposition kernels proposed in [7]. If the input of a kernel is a set of graphs, it is called a graph kernel and if it is a set of graph nodes, the kernel is referred as graph node kernel. Both graph kernels and graph node kernels are widely applied to build graph-based learning systems ranging from social networks, to recommendation systems, to biology. In the following, we shortly describe some of the kernels in each category.

2.2.1. Graph kernels

The task of designing effient and expressive graph kernels play an important role in the development of graph-based predictive systems. Existing graph kernels are decompositional kernels and can be classified into two categories: sequence-based graph kernels and subgraph-based graph kernels. The sequence-based graph kernels decompose graphs into "parts" in sequence-based forms, such as paths and walks. Typical examples of sequence-based graph kernels are product graph kernel, [8], shortest path kernels [9]. The subgraph-based graph kernels dissolve graphs into subgraphs. Examples include Weisfeiler-Lehman kernels [10, 11], The Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) [12]. Following we describe NSPDK since it is later adopted to develop our graph node kernel (presented in Section 3).

The NSPDK [12] is an instance of convolution kernel [7] 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* $A_u \cong \mathcal{N}_r^u$ 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 1 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, $R_{r,d}^{-1}(G) = \{A_u, B_v | R_{r,d}(A_u, B_v, G) = \mathit{true}\}$. The kernel $\kappa_{r,d}$ over $\mathcal{G} \times \mathcal{G}$, counts the number of such fragments in common in two input graphs:

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

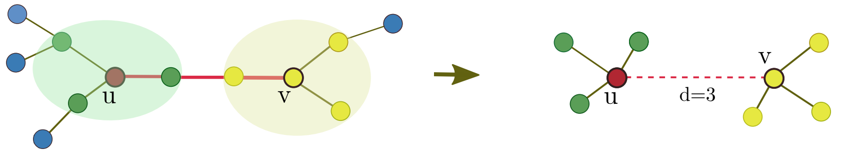


Figure 1: Pairwise neighborhood subgraphs for the "red" node with $r = 1$ and $d = 3$

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.

2.2.2. Graph node kernels

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.

- 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 [4]: 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:

$$K = e^{-\beta L} = I - \beta L + \frac{\beta L^2}{2!} - ... \qquad (1)$$

  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 [4].

- 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:

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

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

- Markov diffusion kernel: The original Markov diffusion kernel MDK was introduced by Fouss et al. [5] 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 P and by defining $Z(t) = \frac{1}{t}\sum_{\tau=1}^{t} P^{\tau}$, as follows:

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

- 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 [6] and represents a normalized version of the random walk with restart model. It is defined as follows:

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

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 [12], 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 labeled graph $G = (V, E, \mathcal{L}_1, \mathcal{L}_2)$, our kernel consists of two phases. In the first phase, a network decomposition procedure is applied to transform 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 we treat them in distinct manners. In the second step, the similarity between any node couple (u, v) is computed by adopting NSPDK on two neighborhood subgraphs rooted as u and v which extracted from the resulted graph after decomposing. In the following, we describe each phase in detail.

3.1. 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 Section 2.2.1) since these are based on the notion of exact matches. Neighborhood subgraphs rooted at high degree nodes are relatively big due to a high number of neighbors and edges. It leads to a low likelihood of finding neighborhood subgraphs rooted at high degree nodes which are isomorphic. Thus, the probability of having 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.

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.

Clique decomposition [14]: to model the notion that nodes in a clique are

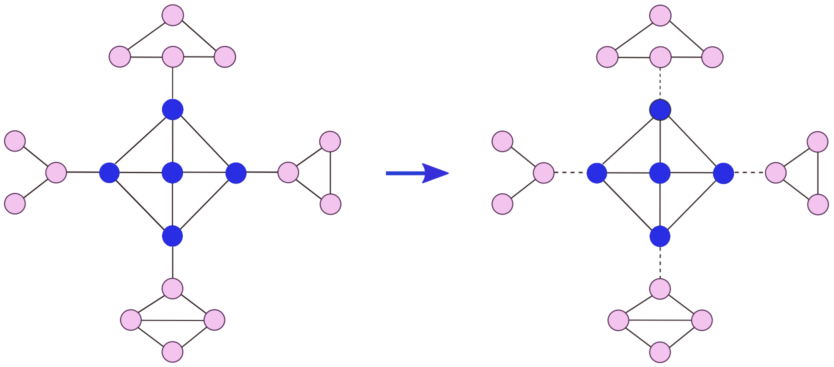


Figure 2: K-core decomposition with degree threshold $D = 4$

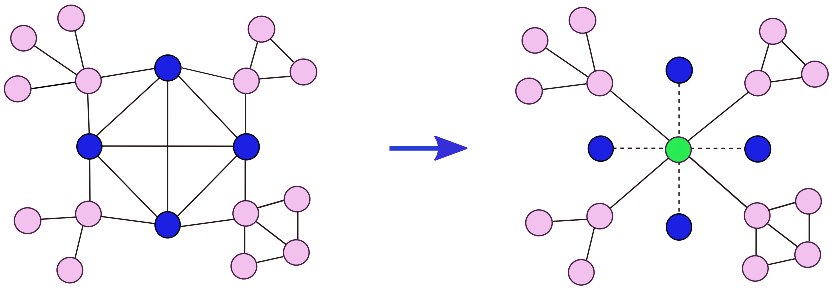


Figure 3: Clique decomposition with threshold $C = 4$

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.

3.2. The conjunctive disjunctive node kernel

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

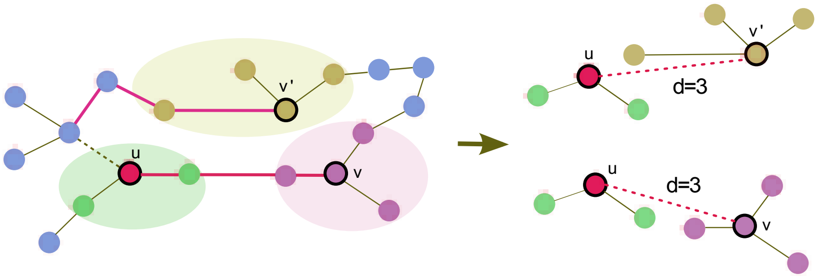


Figure 4: Pairwise neighborhood subgraphs for the "red" node with $r = 1$ and $d = 3$ using "*conjunctive*" and "*disjunctive*" edges

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^*.

In order to integrate the information of real vector labels, we proceed as follows. We compute a sparse vector representation for the neighborhood graph rooted in node v following [12]: for each neighborhood subgraph we calculate the quasi-isomorphism certificate hash code; we then combine the hashes for the pair 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) $f : G_u \longmapsto IR^N$ where $N \approx 10K-1M$. Given the real valued vector information (associated to node u in graph G) $g : G_u \longmapsto IR^P$ computed as the multi-class similarity to the P clusters (c.f.r. Section 5.2), 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. Parameter space

Our kernel consists of five parameters: the threshold degree D, clique size threshold C, maximal radius r^*, maximal distance d^* and number of clusters P clusters in which their values are in $\mathbb{N}^*$. In order to choose the optimal tuple of parameters for kernel in a specific setting, a model selection procedure is normally adopted from a determined subset of parameter space. The parameter space of our kernel seems large due to the relatively high number of parameters. However, most parameter values are supposed to be in limited ranges. Therefore, it is actually not too big. In particular, the clique size threshold C is recommended to be in $\{4, 5\}$. The maximal radius r^* is set with a value smaller than or equal to 3 since it will lead to big neighborhood subgraphs (as discussed in Section 3.1). The maximal distance d^* is assigned with values less than or equal to 4 because the value of the maximal shortest distance between nodes in a connected component is often not too high. The values of the threshold degree D should neither be too high and too low. If it is high, we have to face with high degree node problem, and if it is too low, the obtained graph consists of too sparse connected components. Therefore, we suggest the value for D is in $[6, 20]$.

5. Empirical evaluation

In this section, we desire to evaluate the performance of CDNK and other graph node kernels to answer for the two following questions:

- Does CDNK show better performance comparing to other graph node kernels?
- Does the use of side information (real vector labels) help to improve the performance of CDNK?

5.1. Experimental settings and evalution methods

We carry out experiments in the context of disease gene prioritization 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 known to cause a genetic disease. We first apply a graph node kernel to compute kernel matrix. This kernel matrix together with training gene set are used as the input of a learning algorithm to construct a model. The obtained model is then used to prioritize for candidate genes.

The experiments are performed on two separate networks derived from BioGPS and Pathways datasets in which we follow the experiment procedure in [2] where 12 diseases [15] are used in which each disease is associated to at least 30 positive genes (see *Appendix* [1] for the list of genetic diseases and the number of

[1] https://github.com/dinhinfotech/Neurocoputing/blob/master/appendix.pdf

positive genes in each disease). 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 $|\mathcal{N}| = \frac{1}{2}|\mathcal{P}|$. This is due to the fact that genes known to be related to at least a genetic disease, but not to the considered one are well studied, so they have low probability to be associated to the current disease.

- **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.

To compare the performance of graph node kernels, we fix the learning algorithm and sequentially substitute diffirent graph node kernels. The performance of the system obtained in different cases are used to compare the performance of the employed graph node kernels: LEDK, MEDK, MDK, RLK and CDNK.

The predictive performance of the system using each kernel 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 test set to form a global decision score list on which we compute the AUC-ROC.

5.2. Node labeling

The graph node kernel takes the labled graphs as its input in which labels can be discrete and/or real valued vectors. Therefore, to carry out experiments on graphs derived from BioGPS and Pathways, we need ways to label for these genetic graphs. Following are our proposed node labeling methods for genetic networks. Note that we allow the user to define and use any node labeling functions which are informative.

Discrete labels: We cast two different approaches to associate genes with dicrete labels.

The first approach, we use a same label for every node of graphs. In this case, the configuration of nodes expressed by the pairwise neighborhood subgraphs in CDNK now turn to pairwise neighborhoods. Besides, since labels are identical, the cardinality of neighborhoods are took into account.

The second approach aims to use nodes 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 employ a gene ontology [16] 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, P, so that genes with similar description profiles receive the same class identifier as label.

Real 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 $v \in IR^{26501}$ we compute a similarity vector $S(v) = s_1, s_2, \dots s_P$ 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.

5.3. Model selection

The hyper parameters of the various methods are tuned using a k-fold cross validation. 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 estimation. The values for diffusion parameter β in LEDK and MEDK are sampled in $\{10^{-3}, 10^{-3}, 10^{-2}, 10^{-1}\}$, time steps t in MDK in $\{1, 10, 100\}$ and RLK parameter β in $\{1, 4, 7\}$. For CDNK, the degree threshold values, D, are sampled in $\{10,\ 15,\ 20\}$, clique size threshold, C, in $\{4,\ 5\}$, maximum radius, r^*, in $\{1,\ 2\}$, maximum distance, d^*, in $\{2,\ 3,\ 4\}$, number of clusters P in $\{5,\ 7\}$. Finally, the regularization trade off parameter C for the SVM is sampled in $\{10^{-5},\ 10^{-4},\ 10^{-3},\ 10^{-2},\ 10^{-1}, 1,\ 10,\ 10^2\}$.

5.4. Parameter impact

In order to evaluate the stability of kernel w.r.t the change of each parameter values, we perform experiments with disease-gene association 2 and on the BioGPS network. For each parameter, first fix the values of all other parameters, but allow the value of this papameter to change. More precisely, we use the optimal tuple $C = 5,\ d^* = 1,\ D = 20,\ P = 15$ for the fixed of parameters' values. We let the changes of C in $\{4,\ 5,\ 6,\ 7,\ 8,\ 9\}$, d^* in $\{1,\ 2,\ 3,\ 4\}$, D in $\{8,\ 10,\ 15,\ 20,\ 30,\ 40,\ 50\}$ and real vector size, P in $\{8,\ 10,\ 15,\ 20,\ 30,\ 50,\ 70,\ 90\}$

5.5. Results and discussion

Table 1 and 2 show the AUC-ROC performance of different models using state of the art graph node kernels and different variations of CDNK on BioGPS and Pathways networks, respectively. In what follows, we analyze the results to answer for the two questions which are previously raised.

Concering the performance of different graph node kernels, CDNK variations outperform diffusion-based graph node kernels in all cases on BioGPS dataset and 8 out of 11 cases on Pathways when compared to diffusion-based kernels. CDNK is ranked first when considering both the average AUC-ROC and the average rank with a difference, w.r.t. the best diffusion-based kernel, ranging from 5.4% to 10% and from 1.2% to 3.4% on BioGPS and Pathways, respectively. As a consequence, we can conclude that CDNK show state of the art performance in graph node kernels.

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 = uniform discrete labels and CDNK4 = uniform 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	69.5/3	72.0/2	**72.3/1**
2	81.7/5	78.5/6	75.2/7	75.0/8	88.3/2	**88.8/1**	83.2/4	84.8/3
3	64.3/6	59.6/8	71.6/3	71.8/2	65.5/5	**72.5/1**	61.4/7	67.9/4
4	65.3/7	58.2/8	67.8/6	67.8/5	71.9/4	**78.7/1**	73.8/3	76.7/2
5	64.0/8	64.1/7	66.5/5	66.2/6	75.9/4	76.2/3	77.8/2	**78.2/1**
6	74.6/5	70.2/8	71.0/7	71.2/6	79.3/3	**83.7/1**	77.2/4	80.0/2
7	73.0/5	66.7/7	75.4/3	75.6/2	68.8/6	73.9/4	66.1/8	**77.4/1**
8	74.4/8	76.8/3	76.2/5	76.4/4	74.7/7	77.7/1	76.1/6	76.9/2
9	71.5/2	65.6/8	67.7/6	69.9/3	66.8/7	**71.7/1**	67.9/5	68.1/4
10	54.0/6	50.3/8	56.1/5	51.1/7	77.6/4	**82.7/1**	77.7/3	78.3/2
11	58.2/7	51.3/8	59.3/6	59.3/5	71.8/4	**80.2/1**	74.8/2	74.0/3
$\overline{AUC}$	66.6	63.5	67.8	67.6	73.2	**77.8**	73.5	75.9
$\overline{Rank}$	6.09	7.09	5.36	4.82	4.45	**1.64**	4.18	2.27

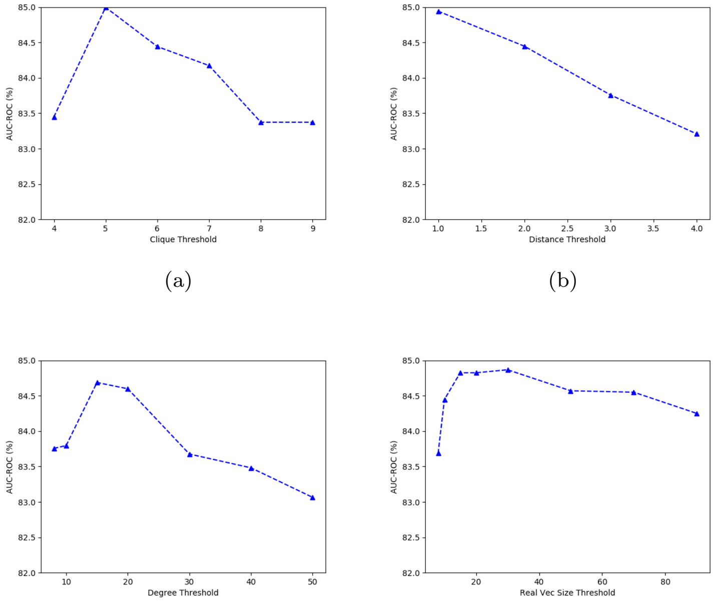
Regarding the variations of CDNK, the integration of real valued vectors improves the performance in most cases. In particular, considering the use of discrete labels based on ontology, using side information helps to increase the performance in all diseases on BioGPS and in 9 out of 11 diseases on Pathways. Similarly, by employing auxiliary information associated to graph nodes, the performance of CNDK in case of using uniform discrete labels is also improved in 9 and 8 out of 11 on BioGPS and Pathways, respectively.

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 = uniform discrete labels and CDNK4 = uniform 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	80.2/4	82.4/2	**82.8/1**	82.3/3
2	55.1/8	64.9/7	76.6/6	76.6/5	81.1/2	80.3/4	80.1/3	**81.9/1**
3	55.0/8	62.7/7	64.1/5	65.6/4	67.1/3	63.6/6	69.7/2	**71.4/1**
4	54.3/8	65.2/6	**73.7/1**	73.7/2	66.1/5	68.1/3	64.0/7	67.3/4
5	52.9/8	55.7/7	62.7/5	62.7/6	68.3/2	**69.6/1**	66.2/4	68.1/3
6	83.4/8	92.7/7	96.5/2	**96.5/1**	93.0/6	94.1/5	94.5/4	94.9/3
7	84.5/7	88.3/8	89.4/2	**89.5/1**	88.5/6	88.5/5	89.0/3	88.7/4
8	53.7/8	65.6/7	72.0/6	72.3/5	72.5/4	72.5/3	74.9/2	**75.3/1**
9	52.5/8	64.9/5	64.2/7	64.2/6	**81.3/1**	81.0/3	80.1/2	79.8/4
10	68.8/6	65.4/8	74.4/5	74.4/4	66.9/7	76.8/2	75.2/3	**78.7/1**
11	53.7/8	69.2/7	74.6/5	74.1/6	77.0/3	**78.7/1**	75.1/4	77.3/2
$\overline{AUC}$	62.6	70.1	75.2	75.3	76.5	77.8	77.4	**78.7**
$\overline{Rank}$	7.73	6.82	4.55	4.09	3.91	**2.27**	3.18	2.45

6. Conclusion and future work

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 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.



(a)

(b)

(c)

(d)

Figure 5: The performance of the system w.r.t the change of each hyper-parameter: (a) clique threshold, (b) distance threshold, (c) degree threshold, (d) real vec size threshold

Funding

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

References

[1] 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.

[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] 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.

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

[5] 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.

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

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

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

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

[10] N. Shervashidze, K. M. Borgwardt, Fast subtree kernels on graphs, in: Advances in neural information processing systems, 2009, pp. 1660–1668.

[11] 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.

[12] 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.

[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.

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