University of Padua

Department of Mathematics

Doctorate Degree in Brain, Mind and Computer Science

Curriculum in Computer Science

KERNEL METHODS FOR LARGE-SCALE GRAPH-BASED HETEROGENEOUS BIOLOGICAL DATA INTEGRATION

Candidate
Dinh Tran Van

Supervisor
Prof. Alessandro Sperduti

Co-supervisor
Prof. Franca Stablum
University of Padova, Italy

October 9, 2017

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor Prof A for the continuous support of my Ph.D study, for his patience, motivation and enthusiasm.

Besides my advisor, I would like to thank the rest of my thesis committee for their insightful and valuable comments and suggestions.

I would like to thank Dr. B for all his help and for his guidance before and during my Ph.D studies. Also, I would like to thank Prof. Peter Tino for the meaningful discussions and inputs he provided during my visit to University of Birmingham.

I would like to thank all my friends, colleagues and officemates (Moreno, Riccardo, Alberto, Daniele, Hossein and Ding Ding) here at the University of Padua for the stimulating discussions, and for all the fun we have had in the last three years.

Last but not the least, a huge thank goes to my family, that always supported me in these years. Words cannot express how grateful I am to my mother, and father for all of the sacrifices that they have made on my behalf.

Dinh Tran Van
Padova, October 9, 2017

Abstract

The last decades have been experiencing a rapid growth in volume and diversity of biological data, thanks to the development of high-throughput technologies related to web services and embeded systems. It is common that information related to a given biological phenomenon is encoded in multiple data sources. On the one hand, this provides a great opportunity for biologists and data scientists to have more unified views about phenomenon of interest. On the other hand, this presents challenges for scientists to find optimal ways in order to wisely extract knowledge from such huge amount of data which normally cannot be done without the help of automated learning systems. Therefore, there is a high need of developing smart learning systems, whose input as set of multiple sources, to support experts to form and assess hypotheses in biology and medicine. In these systems, the problem of combining multiple data sources or data integration needs to be efficiently solved to achieve high performances.

Biological data can naturally be represented as graphs. By taking graphs for data representation, we can take advantages from the access to a solid and principled mathematical framework for graphs, and the problem of data integration is converted into graph-based integration. In recent years, the machine learning community has witnessed the tremendous growth in the development of kernel-based learning algorithms. Kernel methods whose kernel functions allow to separate between the representation of the data and the general learning algorithm. Interestingly, kernel representation can be applied to any type of data, including trees, graphs, vectors, etc. For this reason, kernel methods are a reasonable and logical choice for graph-based inference systems. However, there is a number of challenges for graph-based systems using kernel methods need to be effectively solved, including: *definition of node similarity measure*, *graph sparsity*, *scalability*, *integration methods*. The contributions of this thesis aim at investigating to propose solutions that overcome the challenges faced when constructing graph-based data integration learning systems.

The first contribution of the thesis is the definition of a novel decompositional graph node kernel, named conjunctive disjunctive node kernel, to measure graph node similarities. We first employ a network decomposition procedure to transform the network into a set of linked connected components in which we distinguish between *conjunctive* links whose endpoints are in the same connected components and *disjunctive* links that connect nodes located in different connected components. We then propose a graph node kernel that explicitly models the configuration of each nodes context.

The second contribution aims at dealing with the sparsity problem of graphs by introducing a link prediction method whose objective is to recover missing links of a graph. In this method we first represent each link connecting two nodes by a graph composed of their neighborhood subgraphs. We then cast the link prediction problem as a binary classification task over obtained graphs in which we employ an efficient decompositional graph kernel for graph similarity. Empirical evaluation proves the promissing of the method.

The third contribution targets to boost the performance of diffusion-based graph node kernels when the graph structure is affected by noise in the form of missing links, similarities are distorted proportionally to the sparsity of the graph and to the fraction of missing links. As a consequence, we propose a method named link enrichment for diffusion-based graph node kernels with the idea of carrying out the computation of information diffusion on a graph that contains edges identified by link prediction approaches. We discover a surprisingly robust signal that indicates that diffusion-based node kernels consistently benefit from the coupling with similarity-based link prediction techniques on large scale datasets in biological domains.

The fourth contribution copes with the scalability problem of graph-based data integration learning systems by proposing scalable kernel-based gene prioritization method (Scuba). Scuba is optimized to deal with strongly unbalanced setting and is able to deal with both large amount of candidate genes and arbitrary number of data sources. It enhances the scalability and efficacy and outperforms existing methods for disease gene prioritization.

The last contribution is another approach for graph-based data integration, targeting to solve disease gene prioritization problem. In this approach, the common genes between graph layers, derived from biological sources, are connected by disjunctive links. Then a particular graph node kernel is adopted to exploit topological graph features from all layers for measuring gene similarities. The state of the art performance on different experimental settings confirms the strength of the method.

Contents

Chapter 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.

The abundance of biological data, on the one hand, creates a golden chance for biologists to extract useful information. However, it, on the other hand, poses the challenge for scientists to wisely and effectively extract knowledge from such amount of data that normally cannot be done without the help of automated learning systems. Hence, the task of developing high performance learning systems, which help sciencists to form and assess hypotheses, plays an important role in the development of biology and medicine.

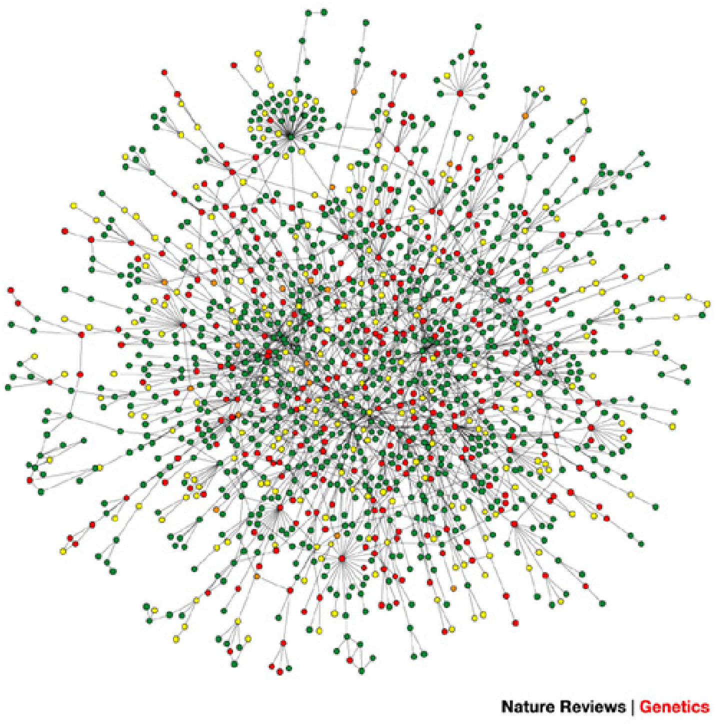

Figure 1.1: Yeast protein interaction network

1.1 Why graph-based biological data integration?

Biological knowledge is distributed among general and specialized sources, such as gene expression, protein interaction, gene ontology, etc. It is common that information of a biological phenomenon is encoded over various heterogeneous sources. Each source captures different aspects of the phenomenon. The distribution of information over sources provides us an unprecedented opportunity to understand the phenomenon from multiple angles. Therefore, the idea of data integration which allows multiple sources of information to be treated in a unified way is potential to improve the performance of biological learning systems. Despite the fact that data integration is a promising solution, it exposes a challenge for machine learning experts and data scientists to find out optimal solutions for combining multiple sources in a big space of solutions.

Relations between entities encoded in biological sources can be naturally represented in form of graphs (networks) whose vertices describe for biological entities and links characterize the relations between entities. An example is the protein-protein interaction network fig. 1.1 where each vertex represents for a protein and a link connecting two vertices if they interact. Graph theory provides a mathematical abstraction for the description of such relationships. Thus, using graphs to represent for biological data allows us to *i*) access to a principled and solid mathematical framework built for graphs that most scientists are familiar, *ii*) develop concepts and tools which are independent of the concrete applications. By using graph presentation, the problem of biological data integration now can be converted into graph-based data integration. The final representation of data obtained by an optimal integration way is used as the input to construct inference systems (graph-based learning systems).

1.2 Kernel methods for graph-based data integration and challenges

Recently, *kernel methods* whose best known member is support vector machine (SVM) [8], has emerged as one of the most famous and powerful frameworks in machine learning. Kernel methods with the use of kernels allow to decouple the representation of the data (via kernel function) from the specific learning algorithm. Kernel representation is flexible and efficient and it provides a principled framework that allows universal type of data to be represented, including images, graphs, vectors, strings, etc. As a consequence, kernel methods are a reasonable and logical choice for graph-based inference systems. However, there are a number of challenges need to be efficiently solved, if we desire to have high performance graph-based data integration learning systems. Following are the main challenges.

1.2.1 Definition of node similarity measure

In machine learning, one of the main factors which impacts on the performance of learning systems is the definition of example similarity measure. In our context, large-scale graph-based inference systems, examples are nodes of graphs. Hence, it is necessary to have a good definition of node similarity measure. Node similarity are normally measured by *graph node kernels*. However, there is not a clear way to define a graph node kernel which is efficiently applied to a wide range of graphs.

1.2.2 Sparsity

The input of a graph-based data integration system is a set of graphs which often contain sparse graphs whose number of present links are much less comparing number of actually links, due to the lack of information. For instance, in disease gene network, links connecting genes are formed when genes are involved in same diseases. New genes associated to a certain disease could be discovered over time. It means that new links could be added into networks from time to time. At a give time, a number of links discovered can be very limited, so it causes the sparsity problem. When working with sparse graphs, systems cannot effectively learn since they have access to a limited information. As a consequence, an effective solution helps to overcome the sparsity problem is crucial and needs to be proposed at hand.

1.2.3 Scalability

Given an adopted learning algorithm, the complexity of a large-scale graph-based data integration learning system incurs with the growth of the input graph set. In other words, the complexity of a graph-based learning system strongly depends on the size and the number of graphs used as its input. It is often that the complexity of a graph-based system changes at a fast pace, sometimes it exponentially changes, with respect to the cardinality of the input graph set. Therefore, scalability is an important property that a graph-based learning system is supposed to possess. It allows systems to run in reasonable time and with a reasonable memory consumption.

1.2.4 Data integration methods

Information encoded in multiple sources (graphs) provides a complementary views of phenomenon of interest. Combination information from collective sources helps to form a complete picture of the phenomenon or problem. However, optimal ways of integration are resided on a big space of solutions. Therefore, we need an effective approach to search for the optimal solutions which lead to high performance of graph-based data integration learning systems.

1.3 Contributions

The contributions of this thesis focus on solutions to overcome the challenges faced when working with large-scale graph-based biological data integration.

In the first contribution, we take the problem of defining node similarity measure into account by introducing an effective graph node kernel named conjunctive disjunctive node kernel (CDNK). Most existing graph node kernels are based on a notion of information diffusion which can be applied to dense networks with high value of average node degrees. However, a drawback of these approaches is 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 genes context. To address this issue, we propose to employ a decompositional graph kernel 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 into a collection of connected sparse graphs and then we develop a suitable kernel, CDNK.

The second contribution is an introduction of a link prediction method, which is adopted later on for link enrichment with the aim of solving for the problem of graph sparsity. We get the motivation from the current link prediction methods that do not effectively exploit the contextual information available in the neighborhood of each edge. In our method, we propose to cast the problem as a binary classification task over the union of the pair of subgraphs located at the endpoints of each edge. We model the classification task using a support vector machine endowed with an efficient graph kernel and achieve state-of-the-art results on several benchmark datasets.

In the third contribution, we propose a method that boosts performance of diffusion-based kernels when working with sparse graphs by tackling them with link enrichments methods. In particular, given a sparse graph, our proposed method consists of two phases. In the first phase, a link prediction method is employed to rank unobserved links based on their probabilities to be related to missing links. The top links in the ranking are then added into graph. In the second phase, diffusion-based graph node kernels are applied to the graph obtained from the first phase to compute the kernel matrix.

The fourth contribution is a proposed method for disease gene prioritization named scalable kernel-based gene prioritization (Scuba). In Scuba, the scalability problem of graph-based data integration is effectively solved. Besides, Scuba is optimized to deal with strongly unbalanced setting. In particular, the method first employs different graph node kernels to compute a set of kernels from each input genetic graph. It then adopts an effective multiple kernel learning to find an optimal linear combination from these obtained kernels. The final kernel is then fed into a kernel machine (SVM) to output a ranking for candidate genes. Results from different empirical experiments show the outperformance of our method comparing with state-of-the-art ones.

In the last contribution, we propose another approach for graph-based data integration that we again target to solve for disease gene prioritization problem. Unlike Scuba whose the integration happens when multiple kernels computed from input graphs are combined, in this approach, data integration is performed by employing disjunctive interconnection graph procedure. In this procedure, the common genes between graph layers, derived from biological sources, are connected by disjunctive links. We then use CDNK to compute a kernel matrix that is later used as inpute of kernel machine to build model. The evaluation through two particular experimental settings proves that it is the state of the art method in disease gene prioritization.

1.4 Thesis roadmap

This thesis is organized as follows.

Chapter 2 presents preliminary concepts, notations. The rest of the thesis will follow these notations and conventions. Besides, it provides a comprehensive review of the state-of-the-art in the field.

Chapter 3 proposes conjunctive disjunctive graph node kernel.

Chapter 4 introduces a novel link prediction method.

Chapter 6 and 7 describe two different proposed approaches for large-scale graph-based biological data integration.

Chapter 8 summarizes the contributions of the thesis and discusses the directions for future work.

Chapter 2

Background

In this chapter, we describe preliminary knowledge and notaions used for the remaining parts of this thesis to make it easy for readers to follow.

2.1 Machine Learning

Recently, *machine learning* has become a must-know term not only in academia but also in daily life due to the popularity of it's applications in various fields. Machine learning can be considered as a branch of Artificial Intelligence which aims at providing systems the ability to automatically adapt to their environment and learn from experience without being explicitly programmed. According to [19], machine learning is formally defined as:

Definition 2.1.1. *A computer program is said to learn from experience E with respect to some task T and some performance measure P if its performance on T, as measured by P, improves with experience E.*

We denote $\mathbb{X}$ as domain dataset which encodes the complete information of a domain. For each domain, however, we are only able to collect a small fraction of domain dataset, $\mathbb{D}$, resulted from any observation, measurement or recording apparatus such that $\mathbb{D} \cup \mathbb{X}$. The set $\mathbb{D}$ is normally referred as the training dataset. Machine Leanring techniques desire to exploit $\mathbb{D}$ to get useful information for constructing a model that generalizes nature of data source. The model is then used to make prediction or inference in unseen dataset, $\mathbb{U} = \mathbb{X} - \mathbb{D}$.

Machine Learning algorithms can be classified into three groups: supervised learning, unsupervised learning and reinforcement learning. Supervised learning proceed with datasets whose objects are associated to labels, while unsupervised learning works with datasets consisting of input data without labeled responses. Reinforcement Learning aims at designing machines and software agents that can automatically determine the ideal behaviour within a specific context, in order to maximize its performance. Simple reward feedback is required for the agent to learn its behaviour; this is known as the reinforcement signal. In this thesis, we focus on supervised learning scenario.

We consider a training set $\mathbb{D}$ generated by an unknown probability distribution $\mathcal{P}$, $\mathbb{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ where $x_i \in \mathbb{X}$ are examples and $y_i \in \mathbb{Y}$ are labels. The relations between x_i and y_i are defined by a true function (target function) $f : \mathbb{X} \longmapsto \mathbb{Y}$. What we desire to do is to learn the function f. However, the only information we can access is from the training set. Therefore, a supervised learning method aims at estimating a function g based on $\mathbb{D}$ to be as close to f as possible. Depending on the domain of $\mathbb{Y}$, we can further group supervised learning methods into following sub-groups:

- if $\mathbb{Y} \subseteq \mathbb{R}$, the problem is called regression
- if $|\mathbb{Y}| = 2$, we have a binary classification problem
- if $|\mathbb{Y}| = n$ with $n > 2$, we have multi-class classification problem

Besides, a supervised learning algorithm is called multi-labels classification if an example could have more than one label associated with that. It is worth highlighting that there is normally more than one possible choice for g. We refer each choice of g as a hypothesis (h) or model and the set of all possible g as hypothesis space, $\mathbb{H}$. A hypothesis space need to be define in advance and it is necessary to contains good approximations to target function. In order to find the optimal hypothesis, h^*, one way is to employ a true loss function, $\mathcal{L}$, which measures how much a hypothesis fails to correctly map between examples and their corresponding labels.

$$\mathcal{R}(h) = \int_{\mathbb{X}\times\mathbb{Y}} \mathcal{L}(h(x), y) dP(x, y) \tag{2.1}$$

The optimal function with the least of mis-mappings (errors) is then the solution of following optimization:

$$h^* = \arg\min_{h\in\mathbb{H}} \mathcal{R}(h) \tag{2.2}$$

Unfortunately, it is impossible to directly solve the optimization 2.1 since the probability distribution $\mathcal{P}$ in the true loss function 2.2 is an unknown function and we only have access to a finite training set $\mathbb{D}$. In this case, an alternative approach is to use empirical loss instead of true loss function. The empirical loss function is defined over the training set as follow:

$$\mathcal{R}_{emp}(h) = \frac{1}{n}\sum_{n=1}^{n} |h(x_i) - y_i| \tag{2.3}$$

However, in order to use $\mathcal{R}_{emp}(h)$, we need to guarantee that the value of $\mathcal{R}_{emp}(h)$ converges to the value of $\mathcal{R}(h)$?. Using the law of large numbers, authors prove in [28] that the convergence happens when the number of examples is high enough. For more details we refer the reader an amazing book [28].

2.2 Kernel Methods

In classical machine learning techniques for binary classification, first the data presentation form is defined, strings, vectors for instances. It then is used to represent for each example, $x \in \mathbb{X} \longrightarrow \phi(x) \in \mathbb{F}$. After a linear function is learnt to separate positive examples from negative ones. Although these approaches have sucessfully applied in some cases, they share two common limitations: i) the high comlexity when working with high dimensional spaces. ii) the difficulty or impossiblity to find the vectorical form to represent data in many cases.

Recently, a new framework named Kernel method has been proposed and shown the state-of-the-art results in many cases of various fields. SVM [8] is a typical example of kernel methods. Unlike the presentation of data in traditional machine learning, data are not individually represented in kernel methods, but through a set of pairwise similarities. More precisely, a matrix whose each element is a real-valued comparision between two examples is used to represent for a data set. These real-valued elements are computed by using a kernel function: $k : \mathbb{X} \times \mathbb{X} \longmapsto \mathbb{R}$. By using matrix to represent for data set, the presentation of data in kernel methods does not depend on the nature of objects. That means the presentation of strings, images,... are the same. More interesting, it allows kernel machines to modularize into two components: the design of a specific kernel function and the design of a general learning algorithm (kernel machine).

2.3 Kernel functions

As stated in 2.2, in kernel methods, the definition of kernel functions is independent from the definition of general learning algorithms. Therefore, it provides kernel machines more options when employing kernels. A number of kernel functions have been proposed for different types of data. In this section, we first formally define what is a kernel function. We then introduce some kernels defined on graphs that later on are used in our experiments.

Definition 2.3.1. *Given a set of object $\mathbb{X}$, a function $k : \mathbb{X} \times \mathbb{X} \longmapsto \mathbb{R}$ is called a kernel on $\mathbb{X} \times \mathbb{X}$ iff k is*

- *symmetric: it means $k(x_1, x_2) = k(x_2, x_1)$, where $x_1, x_2 \in \mathbb{X}$.*
- *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 $x_i, x_j \in \mathbb{X}$.*

A kernel is usually represented as a matrix K which is called Kernel matrix (Gram matix) and K needs to be symmetric and positive semi-definite, i.e. its eigen values are non-negative. In this thesis, kernels and kernel matrcies are identical.

$$K = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & k(x_1,x_3) & \dots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & k(x_2,x_3) & \dots & k(x_2,x_n) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & k(x_n,x_3) & \dots & k(x_n,x_n) \end{bmatrix}$$

One of the most simple kernel, called Linear kernel which is defined on vectors, $\mathbb{X} \subseteq \mathbb{R}^n$.

$$k_L(x_1, x_2) = x_1^{\intercal} x_2, \tag{2.4}$$

where $x_1, x_2 \in \mathbb{X}$. This kernel suggests a systematic way to define kernels. Given a general set of object $\mathbb{X}$, we first project each element in $\mathbb{X}$ into a vector space, $x \in \mathbb{X} \longrightarrow \phi(x) \in \mathbb{R}^n$. Next, we define a kernel as:

$$k(x_1, x_2) = \phi(x_1)^{\intercal} \phi(x_2) \tag{2.5}$$

Interestingly, any kernels defined on $\mathbb{X}$, there exists a Hilbert space, $\mathbb{F}$ and a mapping $\phi : \mathbb{X} \longrightarrow \mathbb{F}$ such that $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$, where $x_1, x_2 \in \mathbb{X}$.

There are two problems we might face with if we would like to explicitly embed objects into a vector space. i) We need to deal the high computation if we embed objects into high demensional spaces. ii) We do not have

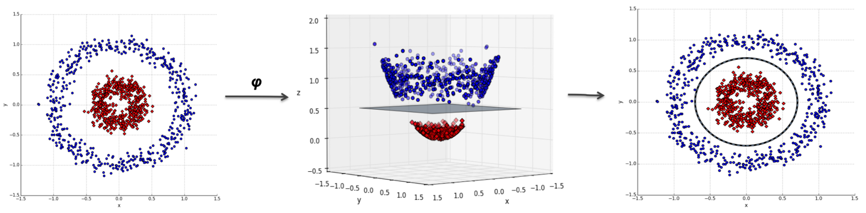

Figure 2.1: The kernel trick transforms the data in a feature space where the instances from the two classes may be linearly separable.

clear way to represent objects in vectorial forms. These limitations are effectively solved by using *Kernel trick*. The kernel trick 2.1 avoids the explicit mapping. Instead, it allows the operations (dot product) between vectors in the feature space to be done by computing only in the input space.

It is worth to notice that a kernel is considered as a similarity (proximity) measure since its value computed for two objects is proportional to their similarity.

Most kernels are defined on vectorial form of data in which Radial basis function kernel (RBF) [30] is the most used one. However, real-world data often cannot be represented in the vectorial form without lossing important information. Therefore, a high number of kernels are proposed to deal with structured data, including trees, graphs, etc.

Convolution kernels

On the development of kernels for structured data, R-convolution kernels [14] proposed by David Haussler can be considered as one of the most important frameworks. The basic idea of convolution kernels is that the semantics of composite objects can often be captrued by a relation R between the object and its parts. The kernel on the object is then made up from kernels defined on different parts.

Let x be a composite structure whose $x_1, x_2, \ldots, x_N = \overrightarrow{x}$ are parts of x, such that $x \in \mathbb{X}$, $x_i \in \mathbb{X}_{\daleth}$, $i = \overline{1, N}$ and $\mathbb{X}, \mathbb{X}_1, \mathbb{X}_2, \ldots, \mathbb{X}_N$ are non-empty and separable metric spaces. We define a relation $R(\overrightarrow{x}, x)$ on $\mathbb{X}_1 \times \mathbb{X}_2 \times \ldots \times \mathbb{X}_N \times \mathbb{X}$ is true iif $x_1, x_2, \ldots, x_N$ are the parts of x. We denote R^{-1} as the inverse relation of R and it is defined as $R^{-1} = \{\overrightarrow{x} | R(\overrightarrow{x}, x)\}$.

If there exists kernel k_i defined on X_i, the similarity between $x, y \in \mathbb{X}$ is defined on $\mathbb{S} \times \mathbb{S}$, where $\mathbb{S} = \{x|R^{-1}(x) \neq \emptyset\}$, as:

$$K(x,y) = \sum_{\overrightarrow{x} \in R^1(x),\ \overrightarrow{y} \in R^1(y)} \prod_{i=1}^{N} k_i(x_i, y_i) \tag{2.6}$$

K is referred as finite convolution, if R is finite. The zero expansion of K to $\mathbb{X} \times \mathbb{X}$ is called R-convolution and it is denoted as $K_1 \star K_2 \star \ldots K_N(x,y)$.

Theorem 1. *If $K_1, K_2, \ldots, K_N$ are kernels on $\mathbb{X}_1, \mathbb{X}_2, \ldots, \mathbb{X}_N$, respectively, and R is a finite relation on $\mathbb{X}_1 \times \mathbb{X}_2 \times \ldots \times \mathbb{X}_N$, then $K_1 \star K_2 \star \ldots K_N(x,y)$ is a kernel on $\mathbb{X} \times \mathbb{X}$.*

Constructing kernels

Kernels can be constructed from predefined kernels. Let k_1, k_2 be kernels over $\mathbb{X} \times \mathbb{X}$, $\mathbb{X} \subseteq \mathbb{R}^n$, $\alpha_1, \alpha_2 \in \mathbb{R}^+$, $f(.)$ is a real valued function on $\mathbb{X}$, $\phi : \mathbb{X} \longmapsto \mathbb{R}^N$ with k_3 a kernel over $\mathbb{R}^N \times \mathbb{R}^N$, and $\mathbf{B}_{n \times n}$ is a symmetric, positive demi-definite. The following functions are kernels.

- $k(x,y) = \alpha_1 k_1(x,y) + \alpha_2 k_2(x,y)$
- $k(x,y) = k_1(x,y) k_2(x,y)$
- $k(x,y) = f(x) f(y)$
- $k(x,y) = k_3(\phi(x), \phi(y))$
- $k(x,y) = x^{'} \mathbf{B} y$

For the proof of above kernels and other ways to form kernels from predefined kernels, we highly recommend a great book titled ”Kernel methods for pattern analysis” [22] to readers.

2.4 Kernel machine

Traditional machine learning techniques aim at finding linear relations in datasets which are presented in vectorical forms. However, there are many cases where expected linear relations does not exist in input dataset. A solution to overcome these situations is to first explicitly transform data into a higher dimensional space and then search for linear relations in that space. Unlike traditional machine methods, kernel methods with the use of kernel functions are able to operate in a high-dimensional space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates.

A number of kernel methods have been proposed. Examples of kernel methods include Perceptron, support vector machines (SVM), Gaussian processes, principal components analysis (PCA), etc. In the next sections, we will introduce in detail two famous algorithms: Perceptron and SVM. For the simplicity, we describe these algorithms in the context of binary classification.

2.4.1 Perceptron kernel algorithm

Perceptron is an old, online leanring algorithm which is based on error-driven learning. It desires to learn a hyper-plane, $wx + b$ or wx for simplicity, to separate positive examples from the negative ones in the training set. It is then used to predict a label for each unseen example, x, through *sign* function. If $wx \geq 0$, $\hat{y} = sign(x) = 1$, otherwise $\hat{y} = sign\,(x) = -1$.

Perceptron works by first initilizing values for weight vector, w. It then iteratively improves the performance by updating the weight vector whenever a misclassification is found in the training set. Consider y_i and $\hat{y}_i$ are true label and predicted label for x_i, if $y_i \neq \hat{y}_i$, w is updated as follow:

$$w \longleftarrow w + \alpha y_i x_i,$$

where $\alpha \in (0, 1]$ is the learning rate. Suppose that n misclassified examples are observed, the weight vector w can be presented as:

$$w = \sum_{i=1}^{n} \alpha y_i x_i.$$

The interation is done when there is no error found. This algorithm guarantees that a linear separation is found if it exists. When the linear separation does not exist, a possible solution is to embed input data into a higher dimensional space. By virtue of doing so, there is a higher chance to have linear separation. However, a problem of high computation is raised when the algorithm operates in a high dimensional space. As a consequence, Perceptron kernel method, an extension of original Perceptron, is proposed to deal with high dimensional space.

Suppose that $\phi(x)$ is the image of x in feature space. We rewrite the formula to compute the weight vector in feature space as

$$w = \sum_{i=1}^{n} \alpha y_i \phi(x_i).$$

The *sign* function is presented as:

$$sign(w\phi(x)) = \sum_{i=1}^{n} \alpha y_i x_i x_j$$

One limitation of both Perceptron and Perceptron kernel method is that they are not able to find the optimal linear separation. Normally, a certain linear separation is supposed to not show promissing predicting ability for unseen examples. In the next section, we describe SVM, a kernel method, which aims at finding an optimal hyperplane to separate between positive and negative examples.

2.4.2 Support vector machine

The original Support vector machine is a linear classifier and it was invented by Vladimir N. Vapnik [27]. SVM became popular when Vladimir et al introduced in [3] a way to create nonlinear classifiers by employing the notion of kernel trick. In particular, a SVM searches for an optimal hyperplane in feature space, $\mathbb{H}$, through operations in input space only.

Given a set of training examples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ in which $x_i \in \mathbb{R}^n$ and $y_i \in \{\pm 1\}$, we first assume that the examples in the training set are linear separable. SVM tries to learn a linear function

$$f(x) = w^\intercal x + b \tag{2.7}$$

where $b \in \mathbb{R}$ is the bias and w is the norm vector. This function forms two half-spaces $h^+ = \{x : w^\intercal x \geq 1\}$ and $h^- = \{x : w^\intercal x \leq -1\}$. The distance between these two half-spaces is referred as *margin* and equal to $\frac{2}{\|w\|}$. To have all examples are correctly classified, the following condition needs to be satisfied

$$y_i(wx_i + 1) \geq 1 \tag{2.8}$$

The optimal hyperplane is the solution of the below quadratic optimization problem (primal form):

$$\begin{aligned} \underset{w,b}{\text{maximize}} \quad & \frac{2}{\|w\|} \\ \text{subject to} \quad & y_i(wx_i + 1) \geq 1 \end{aligned} \tag{2.9}$$

It is equivalent to

$$\begin{aligned} \underset{w,b}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i(wx_i + 1) \geq 1 \end{aligned} \tag{2.10}$$

The resulting Lagrange multiplier equation we desire to optimize is

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i(wx_i + b) - 1 \right], \tag{2.11}$$

where $\alpha_i \geq 0$ are Lagrange multipliers. Solving Lagrangian optimization 2.11, we obtain values for w, b and α which determin a unique hyperplane. The x_is corresponding to α_is which differ from 0 are called support vectors.

The formula of hyperplane decision function 2.7 can be rewritten as:

$$f(x) = sign(\sum_{i=1}^{N} y_i\alpha_i\langle x, x_i\rangle + b) \tag{2.12}$$

In the case that the training set is not linearly separable, we can apply kernel trick to let SVM operates in Hilbert space through calculating in the input space only. We achieve the following decision function:

$$f(x) = sign(\sum_{i=1}^{N} y_i\alpha_i\langle \phi(x), \phi(x_i)\rangle + b) \tag{2.13}$$

There are usually very few α_is which are equal to 0. Therefore, it requires a low computation to predict for unseen examples.

In practice, there are two problems that we need to take into account. First, in many cases, the separating hyperplane does not exist due to the high level of noise in data, that is, a large part of examples in one side are located in the other side. Second, the learning function is so complex that it not only fits the examples, but it also fits the noise. Therefore, the learning function is able to classify well for training examples, but it fails to generalize for unseen data. The latter problem is called overfitting. In order to solve such problems, a solution one may think is to allow examples to violate 2.8.

A soft margin SVM is introduced in which it allows to make a trade off between the mistakes on the training set and the complexity of the hypothesis. The optimization 2.10 is modified by introducing slack variables:

$$\begin{aligned} \underset{w,b,\xi}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & y_i(wx_i + 1) \geq 1 - \xi_i \end{aligned} \tag{2.14}$$

where $\xi_i \geq 0$ and C is a constant which determines the trade-off between margin maximazation and the training error minimization.

2.5 Kernels on Graphs

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. An example of data represented by graph is the genetic network where each node represents for a gene and each link is formed between two genes if they encode common protein(s). Another example is the social network whose nodes are users and links depict friendship between users. There are many systems have been proposed that take graphs as their input. These systems can be called as graph-based systems.

One of the key points that determines the performance of a learning system is the similarity measure definition. In our context, that is the definition of similarity measure between graphs. An idea is to find ways that map graphs into vectorial forms: $\mathbb{X} \longmapsto \mathbb{R}^n$, and then employ similarity functions defined on vectors. However, the task of designing these mappings that are able to encapsulate all information in a vectorial form, is a difficult task since the they need to:

- map isomorphic graphs into the same vector;
- non-isomorphic graphs into different vectors;
- be efficient in terms of time computation and memory consumption.

Recently, kernel methods, whose kernel functions, have emerged as one of the most powerful framework in machine learning. Kernel functions are considered as similarity functions which can be defined on any type of data representation. Therefore, similarity measure defined on graphs are mostly based on kernels. There are two groups of kernels defined on graphs. The first group consists of kernels that aim at measuring the similalrities between graphs and they are referred as graph kernels. To have an overview of graph kernels, we recommend readers to a survey on graph kernels presented in [31]. The second one includes kernels which intend to measure the similarities between nodes inside graphs and are called graph node kernels or node kernels in short. For analysis of different graph node kernels, we suggest to read the work proposed in [10].

In the following, we first give formal definitions and notations related to a graph. We then give an overview of graph kernels followed and graph node kernels.

Definition 2.5.1. *A graph, notated as $G = (\mathbb{V}, \mathbb{E})$, is a structure which consists of a set of nodes (vertices), $V = \{v_1, v_2, \dots, v_n\}$, and a set of links (edges), $\mathbb{E} = \{(v_i, v_j)\} \subseteq (\mathbb{V} \times \mathbb{V})$.*

Definition 2.5.2. *An undirected graph is a graph in which edges have no orientation. The edge (u, v) is identical to the edge (v, u), i.e., they are not ordered pairs, but sets $\{u, v\}$ (or 2-multisets) of vertices. The maximum number of edges in an undirected graph without a loop is $n \times (n-1)/2$.*

Definition 2.5.3. *The graph G is unweighted, if $w_{ij} \in \{0, 1\}$, otherwise it is weighted graph.*

Definition 2.5.4. *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 2.5.5. *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 2.5.6. *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.*

We define the *distance* $\mathcal{D}(u, v)$ 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, $N_r(u) = \{v \mid \mathcal{D}(u, v) \leq r\}$, 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. The maximum node degree in the graph G is $deg(G)$.

2.5.1 Graph Kernels

The research on graph kernels has been made possible thanks to the results of Haussler research, convolution or decomposition kernel, [14] (see 2.3). A decomposition kernel is sum over all possible ways to decompose a structured instance of the product of valid kernels over the "parts" of the instance. The "parts" are referred as features. Graph kernels can preliminarily be classified into two classes: global and local. The global graph kernels consider glocal features, for example a path from one node to another one in a graph. Meanwhile, local graph kernels take into account local features such as subgraphs. Following we first introduce some global graph kernels: product graph kernel, shorstest path kernels, we then describe some local graph kernels: Weisfeiler-Lehman kernels, The neighborhood subgraph pairwise distance kernel.

2.5.1.1 Product graph kernel

Product graph kernel was originaly proposed in [12] with the aim to measure the similarity between two labeled graphs by counting their common walks. To compute the similarity between two graphs (factor graphs), first a graph, called direct product graph, is constructed from two factor graphs. Then the similarity is then computed based on the obtained graph.

Formally, we consider two factor graphs G_1 and G_2 and $\mathcal{L}_{n1}$, $\mathcal{L}_{n2}$, $\mathcal{L}_{e1}$, $\mathcal{L}_{e2}$ as the node and edge labeling functions of G_1 and G_2, respectively. We define the direct product graph of G_1, G_2 as a graph $G_\times = (V_\times, V_\times)$ where

- $V_\times = \{(u, v) : u \in E(G_1) \wedge v \in V(G_2) \wedge \mathcal{L}_{n1}(u) = \mathcal{L}_{n2}(v)\}$
- $E_\times = \{((u, v), (u', v')) \in V_\times \times V_\times : (u, v) \in E(G_1) \wedge (u', v') \in E(G_2)$ $\wedge\, \mathcal{L}_{e1}((u, v)) = \mathcal{L}_{e2}((u', v'))\}$.

Given $\lambda_1, \lambda_2, \ldots (\lambda_i \in \mathbb{R}, \lambda_i \geq 0,\ \forall i \in \mathbb{N})$, the direct product kernel is defined as follow

$$k_(G_1, G_2) = \sum_{i,j=1}^{|V_\times|} \left[\sum_{n=0}^{\infty} \lambda_n A_\times^n \right]_{ij}, \tag{2.15}$$

if the limit exists, in which $A_\times$ is the adjacency matrix of the direct product graph $G_\times$. The computation of this limit is high, $O(n^6)$. There are different modifications of this kernel that can be more efficient to compute, such as the method proposed in [32] that has complexity of $O(n^3)$.

2.5.1.2 Shortest path kernels

The simple idea behind shortest path kernels is that they consider the common shortest paths between two graphs to measure their similarity. Although the computation of shortest paths in a graph can be computed in polinomial time, taking into account shortest paths leads to some problems. First, the shortest paths between two nodes normally are not unique and there has been a way to deterministically choose one shortest path among others. Second, if we keep all shortest paths into account, it could lead to NP-hard kernel. Although the shorstest paths are not unique, the length of them is unique. As the consequence, a shortest path graph kernel is proposed in [2] with total runtime of $O(n^4)$.

Given two graphs G_1, G_2, we first construct their two corresponding shortest graphs G_{s1}, G_{s2}, respectively. The shortest graph of a graph, G, is a labeled graph defined as $G_s = (V_s, E_s)$ where

- $V_s = V(G)$
- $E_s = \{(u, v) : u, v \in V(G) \wedge |p(u, v)| \geq 1\}$, where $p(u, v)$ is the number of paths connecting u and v.
- $\mathcal{L}_e((u, v)) = \mathcal{D}(u, v)$

We then define the shortest graph kernel for G_1 and G_2 as follow:

$$k_s(G_1, G_2) = \sum_{(u_1,v_1)\in E(G_{s1})} \sum_{(u_2,v_2)\in E(G_{s2})} k^1_{walk}((u_1, v_1), (u_2, v_2)), \tag{2.16}$$

where k^1_{walk} is a positive semi-definite kernel on 1-length walks.

2.5.1.3 Weisfeiler-Lehman kernels

The Weisfeiler-Lehman kernel framework is proposed in [24]. It is derived from the kernel proposed in [23]. The idea is to first decompose each graph into a sequence of graphs. Then the similarity between two graphs is the summation of values computed by a given kernel defined on the two corresponding sequences graphs.

Formally, the Weisfeiler-Lehman graph sequence of a given graph $G = (V, E, \mathcal{L})$ up to the height h is defined as:

$$\{G_0, G_1, \dots, G_h\} = \{(V, E, \mathcal{L}_0), (V, E, \mathcal{L}_1), \dots, (V, E, \mathcal{L}_h)\},$$

where $G_0 = G$, $\mathcal{L}_0 = \mathcal{L}$, $\mathcal{L}_i$s $(i = \overline{1,h})$ are different labeling function on G.

Given two graphs G, G' and a graph kernel k, the Weisfeiler-Lehman kernel, k_{WL} is defined as:

$$k_{WL}(G, G') = k(G_0, G_0^{'}) + k(G_1, G_1^{'}) + \ldots + k(G_h, G_h^{'}) \tag{2.17}$$

2.5.1.4 The Neighborhood Subgraph Pairwise Distance Kernel

The NSPDK is an instance of convolution kernel [14] 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 $\mathcal{D}(u, v) = d$ (see the Fig 2.2). 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:

$$\kappa_{r,d}(G, G^{'}) = \sum_{\substack{A_u, B_v \, \in \, R_{r,d}^{-1}(G) \\ A'_{u'}, B'_{v'} \, \in \, R_{r,d}^{-1}(G')}} \mathbf{1}_{A_u \cong A'_{u'}} \cdot \mathbf{1}_{B_v \cong B'_{v'}} \tag{2.18}$$

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

$$K(G, G') = \sum_r \sum_d \kappa_{r,d}(G, G'), \tag{2.19}$$

where for efficiency reasons, the values of r and d are upper bounded to a given maximal r^* and d^*, respectively.

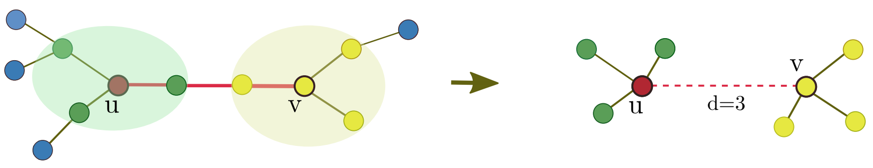

Figure 2.2: Example of a pairwise neighborhood subgraphs rooted at u with radius $r = 1$ and distance $d = 3$

2.5.2 Graph Node Kernels

It is different from graph kernels which aims at measuring similarities between graphs, graph node kernels intend to measure proximities between nodes in graphs. The ideas behind graph node kernels are similar to graph kernels. However, graph node kernels attempt to exploit the configuration concerning two nodes in the graph in order to define their similarity. Most available graph node kernels are global kernels and are based on the diffusion phenomenon. In other words, they consider paths connecting two given nodes in order to form their similarity. In the following, we introduce some of the most used graph node kernels.

2.5.2.1 Laplacian exponential diffusion kernel

One of the most well-known kernels for graphs 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 [17]: 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:

$$K = e^{-\beta \mathbf{L}} = \mathbf{I} - \beta \mathbf{L} + \frac{\beta \mathbf{L}^2}{2!} - ... \tag{2.20}$$

where β is the diffusion parameter and is used to control the rate of diffusion and $\mathbf{I}$ 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 [17].

2.5.2.2 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 [6], called Markov exponential diffusion kernel **MEDK** and given by the following formula:

$$\mathbf{K} = e^{-\beta \mathbf{M}} \tag{2.21}$$

The difference with respect to the Laplacian diffusion kernel is the replacement of $\mathbf{L}$ 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**.

2.5.2.3 Markov diffusion kernel

The original Markov diffusion kernel **MDK** was introduced by Fouss et al. [11] 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:

$$\mathbf{K} = \mathbf{Z}(t)\mathbf{Z}^{\top}(t) \tag{2.22}$$

2.5.2.4 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 [5] and represents a normalized version of the random walk with restart model. It is defined as follows:

$$\mathbf{K} = \sum_{n=0}^{\infty} \beta^n (-\mathbf{L})^n = (\mathbf{I} + \beta\mathbf{L})^{-1} \tag{2.23}$$

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.

2.6 Disease Gene Prioritization

Disease-gene association recovery is a major goal in molecular biology and medical 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. A gene prioritization method aims at ordering candidate genes from the most to the least probable to be associated to the disease. The top genes in the ranking are then sent to biologists and medical scientists for further studies to determine whether each gene is related to considered disease.

Let us formally define the problem of disease gene prioritization which is later on employed in our empirical experiments to evaluate of different adopted methods. We consider a list of genes $\mathcal{G} = \{g_1, g_2, ..., g_n\}$ that could either be the full list of human genes or a subset of it. Considering a specific disease, there exists a set $P_i \subseteq \mathcal{G}$ of genes known to be associated with it. Its complementary set $U_i = \mathcal{G} - P_i$ contains genes that are not a priori related to the disease, but we assume that inside U_i some positive genes are hidden. Gene priorization is a task that allows to rank the genes in U_i based on their likelihood to be related to P_i.

2.7 Biological Datasets

The development of computational biology makes a high number of biological datasets available. Many biological datasets can be naturally represented as networks which are later on used as the input of graph-based biological systems. In biological networks, vertices are biological entities (genes and proteins, etc) and links describe the relation between entities. The relations can be discovered by either physical experiments and results from inferring methods (systems). Following we describe the way to extract information from some biological datasets and transform them into unweighted, undirected networks, represented in the forms of adjacency matrix. These networks will be employed in our experiments for evaluating the performance of different algorithms.

Human Protein Reference Database (**HPRD**) a database of curated proteomic information pertaining to human proteins. It is derived from [16] with 9,465 vertices and 37,039 edges. We employ the HPRD version used in [4] that forms a graph which contains 7311 vertices (represent for genes) and 30503 links. In the graph, two vertices are linked if proteins encoded by their corresponding genes interact.

BioGPS [34]. It contains expression profiles for 79 human tissues, which are measured by using the Affymetrix U133A array. Gene co-expression, defined by pairwise Pearson correlation coefficients (PCC), is used to build an unweighted graph. A pair of genes are linked by an edge if the PCC value is larger than 0.5.

Pathways. Pathway datasets are obtained from the database of KEGG [20], Reactome [29], PharmGKB [33] and PID [21], 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.

String [15]. 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 from which we extracted functional links among 17078 human genes.

Phenotype similarity: we use the OMIM [18] dataset and the phenotype similarity notion introduced by Van Driel et al. [26] based on the relevance and the frequency of the Medical Subject Headings (MeSH) vocabulary terms in OMIM documents. We built the graph linking those genes whose associated phenotypes have a maximal phenotypic similarity greater than a fixed cut-off value. Following [26], we set the similarity cut-off to 0.3. The resulting graph has 3393 nodes and 144739 edges.

Biogridphys: this dataset encodes known physical interactions among proteins. The idea is that mutations can affect physical interactions by changing the shape of proteins and their effect can propagate through protein graphs. We introduce a link between two genes if their products interact. The resulting graph has 15389 nodes and 155333 edges.

Omim: OMIM is a public database of disease-gene association. Genes implicated in the same disease are more likely to be involved in other similar diseases as well. Therefore, Omim network is formed by connecting genes which are involved in common disease(s).

2.8 Link prediction

We are witnessing a constant increase of the rate at which data is being produced and made available in machine readable formats. Interestingly it is not only the quantity of data that is increasing, but also its complexity, i.e. not only are we measuring a number of attributes or features for each data point, but we are also capturing their mutual relationships, that is, we are considering non independent and identically distributed (non i.i.d.) data. This yields collections that are best represented as graphs or relational data bases and requires a more complex form of analysis. As cursory examples of application domains that are social networks, where nodes are people and edges encode a type of association such as friendship or co-authorship, bioinformatics, where nodes are proteins and metabolites and edges represent a type of chemical interaction such as catalysis or signaling, and e-commerce, where nodes are people and goods and edges encode a "buy" or "like" relationship. A key characteristic of this type of data collections is the sparseness and dynamic nature, i.e. the fact that the number of recorded relations is significantly smaller than the number of all possible pairwise relations, and the fact that these relations evolve in time. A crucial computational task is then the "link prediction problem" which allows to suggest friends, or possible collaborators for scientists in social networks, or to discover unknown interactions between proteins to explain the mechanism of a disease in biological networks, or to suggest novel products to be bought to a customer in a e-commerce recommendation system. Many approaches to link prediction that exist in literature can be partitioned according to *i*) whether additional or "side" information is available for nodes and edges or rather only the network topology is considered and *ii*) whether the approach is unsupervised or supervised.

Unsupervised methods are non-adaptive (i.e. they do not have parameters that are tuned on the specific problem instance), and can therefore be computationally efficient. In general they define a score for any node pair that is proportional to the existence likelihood of an edge between the two nodes. *Adamic-Adar* [?] computes the weighted sum over the common neighbors where the weight is inversely proportional to the (log of) each neighbor node degree. The *preferential attachment* method computes a score simply as the product of the node degrees in an attempt to exploit the "rich get richer" property of certain network dynamics. *Katz* [?] takes into account the number of common paths with different lengths between two nodes, assigning more weight to shorter paths. The *Leicht-Holme-Newman* method [?] computes the number of intermediate nodes. In [?] the score is derived from the singular value decomposition of the adjacency matrix.

Supervised link prediction methods convert the problem into a binary classification task where links present in the network (at a given time) are considered as positive instances and a subset of all the non links are considered as negative instances. Following [?], we can further group these methods into four classes: feature-based models, graph regularization models, latent class models and latent feature models. A Bayesian nonparametric approach is used in [?] to compute a nonparametric latent feature model that does not need a user defined number of latent features but rather induces it as part of the training phase. In [?] a matrix factorization approach is used to extract latent features that can take into consideration the output of an arbitrary unsupervised method. The authors show a significant increase in predictive performance when considering a ranking loss function suitable for the imbalance problem, i.e. when the number of negative is much larger than the number of positive instances.

In general supervised methods exhibit better accuracies compared to unsupervised methods although incurring in much higher computational and memory complexity costs. Moreover, most approaches implicitly represent the link prediction problem and the inference used to tackle it as a disjunction over the edges, that is, information on edges is propagated in such a fashion so that for a node to have k neighbors or $k + 1$ does not make a drastic difference. We claim that this hypothesis is likely putting a cap on the discriminative power of classifiers and therefore we propose a novel supervised method that employs a conjunctive representation. We call the method "joint neighborhood subgraphs link prediction" (JNSL). The key idea here is to transform the link prediction task into a binary classification on suitable small subgraphs which we then solve using an efficient graph kernel method.

Chapter 3

Conjunctive Disjunctive Graph Node Kernel

In this chapter, we present our proposed graph node kernel which aims at measuring the similarities between nodes in large-scale graphs.

3.1 Motivation

As discussed before, node similarity measure definition is the key that determines the performance of graph-based 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 [17], MEDK [6], MDK [11], RLK [5], 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.

3.2 Method

We start from the type of similarity notion computed by a neighborhood based decomposition kernel between graph instances [9], 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 employed 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.

3.2.1 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 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 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:

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

where $i = \lceil \frac{deg(u)-T}{bin} \rceil$, $bin = \frac{deg(G)-T}{\lambda - T}$ and λ ($\lambda > T$) is the maximum number of symbols used. The value of λ 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* [7] as an example of extra information associated to the graph nodes.

Functional discrete labels We use the ontology to construct binary vectors representing a bag-of-words encoding for each gene (i.e. if one of the 26501 GO-terms is associated with the gene). 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 $s_i = \frac{1}{1+d(v,c_i)}$ where $d(v, c_i)$ is the euclidean distance of v from the center of the ith 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.

3.2.2 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 2.3) 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 [1]: 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 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.

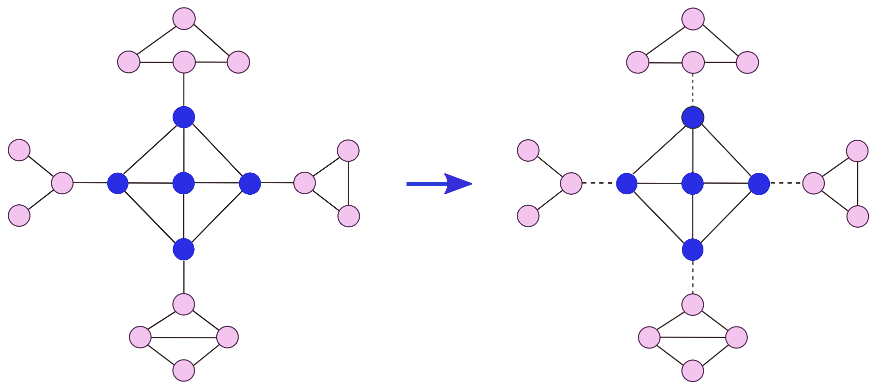
Figure 3.1: K-core decomposition with degree threshold $D = 5$

Clique decomposition [25]: 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 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.

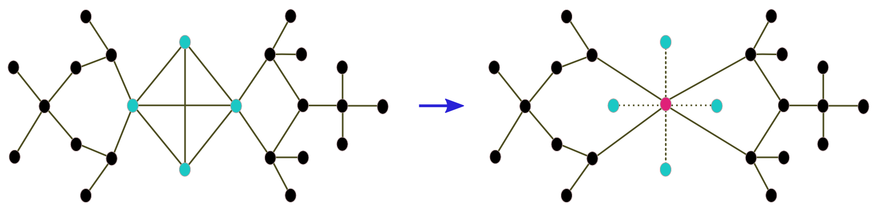
Figure 3.2: Clique decomposition with threshold $C = 4$

3.2.3 The Conjunctive Disjunctive Node Kernel

We start from the Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) [9] 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.

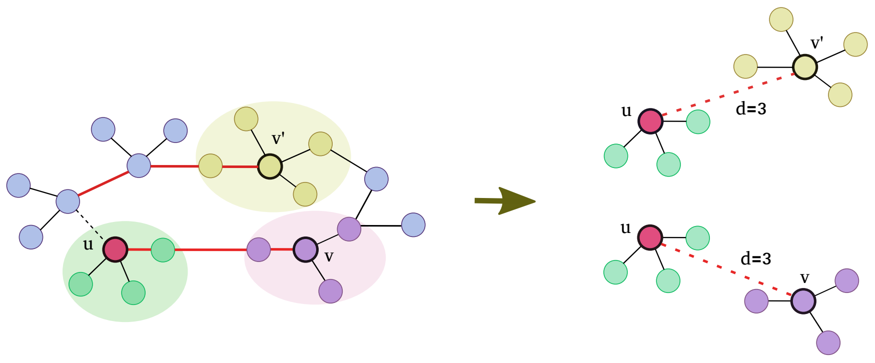

Figure 3.3: Pairwise neighborhood subgraphs for the "red" node with $r = 1$ and $d = 3$ using "conjuction" and "disjuctive" edges

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 3.3 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 $B_v \cong \mathcal{N}^u_r$ 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^*.

3.2.4 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 [9]: 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) $f : G_u \longmapsto I\!R^L$ 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 3.2.1), we update the computation of CDNK considering the discrete convolution of the discrete information with the real valued information:

$$K(G_u, G_{u'}) = \left\langle f(G_u) \bigotimes g(G_u), f(G_{u'}) \bigotimes g(G_{u'}) \right\rangle \tag{3.1}$$

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.

3.3 Empirical Experiments

In this section, we describe the procedure to carry out empirical experiments with the aim to compare the performance of graph node kernels.

In order to evaluate, we employ the disease gene prioritization problem on two different genetic graphs constructed from two biological datasets BioGPS and Pathways described in 2.7. The aim of disease gene prioritization is to rank candidate genes based on their probabilities to be related to a specific disease given a set of genes known to be associated to the disease. We follow [6] and analyze 12 diseases [13] for which it is known that at least 30 genes are involved. For each disease, we construct a positive set $\mathcal{P}$ with all 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 [6] 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.

Model Selection In order to choose the best model for each method, the optimal values of hyper parameters 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 $\{2,\ 3,\ 4\}$, nuber of clusters K in $\{5,\ 7\}$. Finally, the regularization trade off parameter C for the SVM is sampled in $\{10^{-5},\ 10^{-4},\ 10^{-3},\ 10^{-2},\ 10^{-1}, 1,\ 10,\ 10^{2}\}$.

3.4 Results and discussion

Figure 3.4 and 3.5 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. 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 (see results presented in tables in the Appendices) 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 Figure 3.4 and 3.5), the proposed approach improves on state-of-the-art.

3.5 Conclusion

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

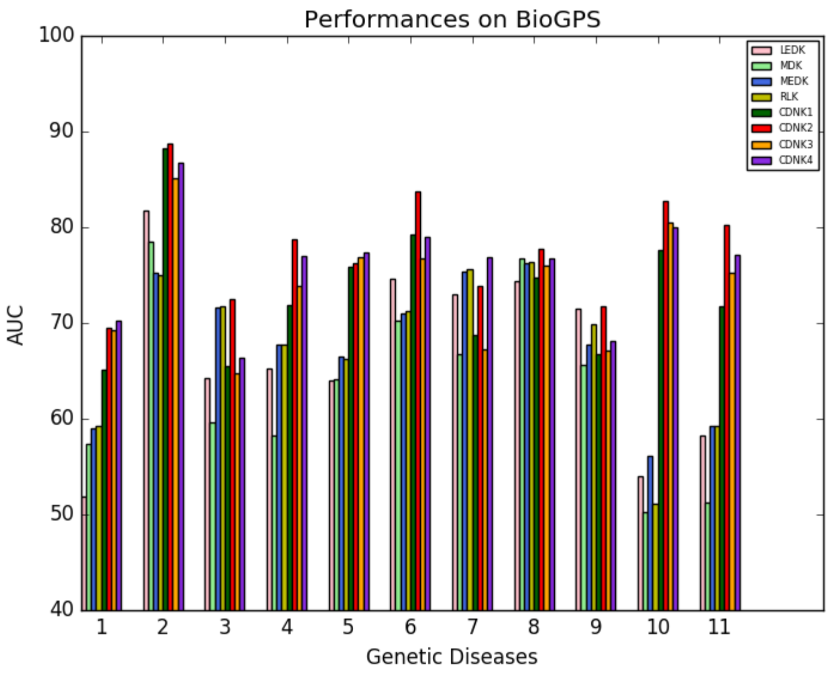

Figure 3.4: *AUC-ROC performance on 11 genetic diseases using network induced by the BioGPS. CDNK1: ontology for discrete labels, CDNK2: ontology for both discrete and vector labels, CDNK3: node degree for discrete labels and CDNK4: node degree for discrete labels and ontology for vector label.*

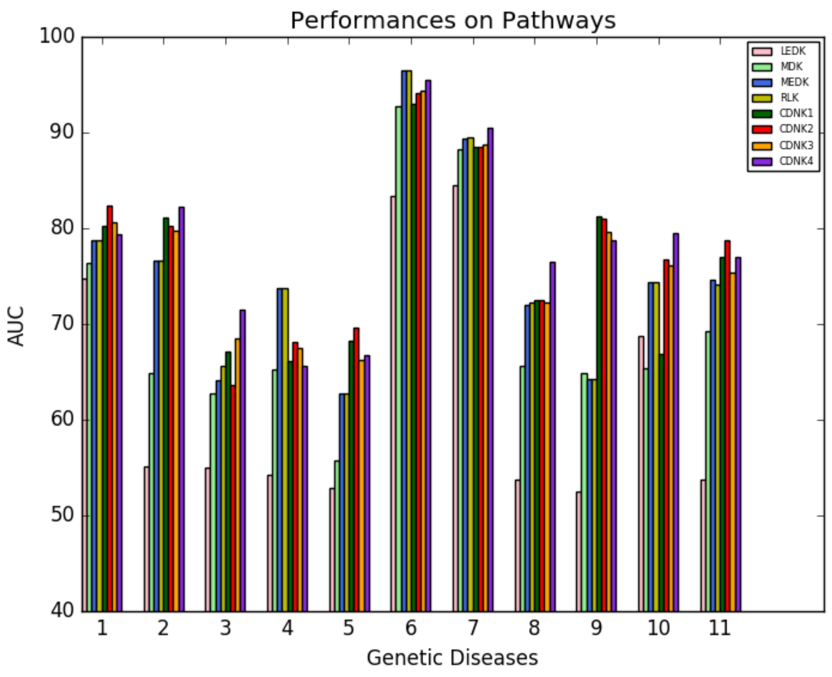

Figure 3.5: *AUC-ROC performance on 11 genetic diseases using network induced by the Pathways. CDNK1: ontology for discrete labels, CDNK2: ontology for both discrete and vector labels, CDNK3: node degree for discrete labels and CDNK4: node degree for discrete labels and ontology for vector label.*

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.

Chapter 4

Joint Neighborhood Subgraphs for Link Prediction

4.1 Motivation

We are witnessing a constant increase of the rate at which data is being produced and made available in machine readable formats. Interestingly it is not only the quantity of data that is increasing, but also its complexity, i.e. not only are we measuring a number of attributes or features for each data point, but we are also capturing their mutual relationships, that is, we are considering non independent and identically distributed (non i.i.d.) data. This yields collections that are best represented as graphs or relational data bases and requires a more complex form of analysis. As cursory examples of application domains that are social networks, where nodes are people and edges encode a type of association such as friendship or co-authorship, bioinformatics, where nodes are proteins and metabolites and edges represent a type of chemical interaction such as catalysis or signaling, and e-commerce, where nodes are people and goods and edges encode a “buy” or “like” relationship. A key characteristic of this type of data collections is the sparseness and dynamic nature, i.e. the fact that the number of recorded relations is significantly smaller than the number of all possible pairwise relations, and the fact that these relations evolve in time. A crucial computational task is then the “link prediction problem” which allows to suggest friends, or possible collaborators for scientists in social networks, or to discover unknown interactions between proteins to explain the mechanism of a disease in biological networks, or to suggest novel products to be bought to a customer in a e-commerce recommendation system. Many approaches to link prediction that exist in literature can be partitioned according to *i*) whether additional or "side" information is available for nodes and edges or rather only the network topology is considered and *ii*) whether the approach is unsupervised or supervised.

Unsupervised methods are non-adaptive (i.e. they do not have parameters that are tuned on the specific problem instance), and can therefore be computationally efficient. In general they define a score for any node pair that is proportional to the existence likelihood of an edge between the two nodes. *Adamic-Adar* [?] computes the weighted sum over the common neighbors where the weight is inversely proportional to the (log of) each neighbor node degree. The *preferential attachment* method computes a score simply as the product of the node degrees in an attempt to exploit the "rich get richer" property of certain network dynamics. *Katz* [?] takes into account the number of common paths with different lengths between two nodes, assigning more weight to shorter paths. The *Leicht-Holme-Newman* method [?] computes the number of intermediate nodes. In [?] the score is derived from the singular value decomposition of the adjacency matrix.

Supervised link prediction methods convert the problem into a binary classification task where links present in the network (at a given time) are considered as positive instances and a subset of all the non links are considered as negative instances. Following [?], we can further group these methods into four classes: feature-based models, graph regularization models, latent class models and latent feature models. A Bayesian nonparametric approach is used in [?] to compute a nonparametric latent feature model that does not need a user defined number of latent features but rather induces it as part of the training phase. In [?] a matrix factorization approach is used to extract latent features that can take into consideration the output of an arbitrary unsupervised method. The authors show a significant increase in predictive performance when considering a ranking loss function suitable for the imbalance problem, i.e. when the number of negative is much larger than the number of positive instances.

In general supervised methods exhibit better accuracies compared to unsupervised methods although incurring in much higher computational and memory complexity costs. Moreover, most approaches implicitly represent the link prediction problem and the inference used to tackle it as a disjunction over the edges, that is, information on edges is propagated in such a fashion so that for a node to have k neighbors or $k + 1$ does not make a drastic difference. We claim that this hypothesis is likely putting a cap on the discriminative power of classifiers and therefore we propose a novel supervised method that employs a conjunctive representation. We call the method “joint neighborhood subgraphs link prediction” (JNSL). The key idea here is to transform the link prediction task into a binary classification on suitable small subgraphs which we then solve using an efficient graph kernel method.

4.2 Method

4.2.1 Definitions and notation

We represent a problem instance as a graph $G = (V, E)$ where V is the set of nodes and E is the set of links. The set E is partitioned into the subset of observed links (O) and the subset of unobserved links (U). Like other approaches we assume that all un-observed links are indeed “non-links” and we therefore define the link prediction problem as the task of ranking candidate links from the most to the least probable to recover links in O but not in U exploiting only the network topology.

We define the *distance* $\mathcal{D}(u, v)$ 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, $N_r(u) = \{v \mid \mathcal{D}(u, v) \leq r\}$, 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, $d(u) = |\mathcal{N}_1^u|$, is the cardinality of its neighborhood. The maximum node degree in the graph G is $d(G)$.

4.2.2 Link encoding as subgraphs union

Most methods for link prediction compute pairwise nodes similarities treating the nodes defining the candidate edge independently. Instead we propose to jointly consider both candidate endpoint nodes together with their extended “context”. To do so we build a graph starting from the two nodes and the underlying network. Given nodes u and v, we first extract the two neighborhood sets with a user defined radius R rooted at u and v to obtain $N_R(u)$ and $N_R(v)$, respectively. We then consider the graph $\mathcal{J}$ induced by the set union $N_R(u) \cup N_R(u)$. Finally we add an auxiliary node w and the necessary edges to connect it to u and v (see Fig.4.1).

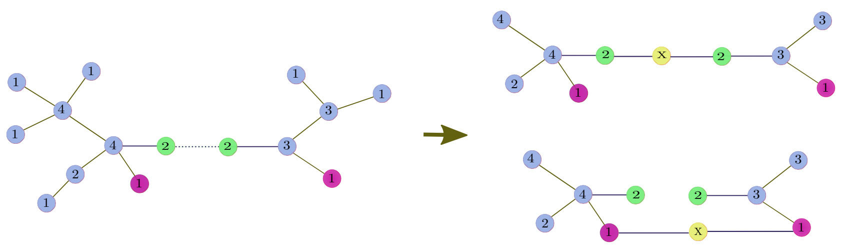

Figure 4.1: Left) We represent with solid lines edges belonging to the training material and with a dotted line edges belonging to the test material. Right) joint neighborhood subgraphs for an existing (green endpoints) (top) and a non existing (red endpoints) link (bottom). These graphs will receive respectively a positive and a negative target.

4.2.3 Node labeling

We propose to use a graph kernel approach to classify the subgraphs encoding each link. In our setup nodes are not endowed with any "side" information. However to increase the discriminative power of the similarity notion induced by the graph kernel, instead of assuming a dummy, non-informative label on each node, we propose to use a node labeling function ℓ which assigns as the discrete label the node degree. More precisely, for nodes having degree less than or equal than a user defined threshold T ($T = 5$ in our experimental evaluation) we use as label the degree value. Degree values larger than T are subsequently discretized into k levels. Here the implicit assumption is that nodes with similar degrees have common properties. Formally, the labeling function is defined as:

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

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

4.2.4 The graph kernel

Here we briefly describe an efficient graph kernel called the Neighborhood Subgraph Pairs Distance kernel (NSPDK) introduced in [?]. NSPDK is an instance of "decompositional" kernels [?] based on the idea of counting the number of common small subgraphs between two graphs. The subgraphs are pairs of neighborhoods whose roots are at a short distance.

Given a labeled graph $G \in \mathcal{G}$ and two rooted graphs A_u, B_v, we first define the relation $R_{r,d}(A_u, B_v, G)$ to be true *iff* $A_u \cong \mathcal{N}_r^u$ is (up to isomorphism $\cong$) a neighborhood subgraph with radius r of G and so is $B_v \cong \mathcal{N}_r^v$, such that v is a distance d from u: $\mathcal{D}(u, v) = d$. We then define the inverse relation R^{-1} 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 $\mathcal{G} \times \mathcal{G}$ is 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.

4.2.5 Joint neighborhood subgraphs link prediction

In the link prediction problem we are given a graph $G(V, E)$ and a binary target vector $Y = \{y_{(0,0)}, y_{(0,1)}, \cdots, y_{(|V|,|V|)}\}$ where $y_{(u,v)} = 1$ if $(u, v) \in E$ and 0 otherwise. The training data is obtained considering a random subset of edges in $E^{tr} \in E$ and inducing a training graph $G^{tr} = (V, E^{tr})$. Note that the graph used for training does not contain any of the edges that will be queried in the test phase. The remaining edges $E^{ts} = E \setminus E^{tr}$ are used to partition the target vectors: $Y^{tr} = \{y_{(u,v)} | (u, v) \in E^{tr}\}$, $Y^{ts} = \{y_{(u,v)} | (u, v) \in E^{ts}\}$. We can now cast the problem as a standard classification problem in the domain of graphs. Given G^{tr} we build a train and test set as the corresponding joint neighborhood subgraphs as detailed in Section 4.2.2. We can now compute a Gram matrix of the instances and solve the classification task using for example the efficient LinearSVC library [?].

4.3 Experiments

4.4 Empirical evaluation

To compare the performance of the JNSL method with other link prediction approaches we follow [?] and use 6 datasets belonging to different domains.

- *Protein* [?]: nodes are proteins and edges encodes a thresholded interaction confidence between proteins. It has 2617 nodes and 11855 links with an average degree of 9.1.
- *Metabolic* [?]: nodes are enzyme and metabolites, edges are present if the enzyme catalyzes for a reaction that include those chemical compounds. It has 668 nodes and 2782 links with an average degree of 8.3.
- *Nips* [?]: nodes are authors at the NIPS conference from the first to the 12^{th} edition. Links encode the co-authorhip relation, i.e. if two authors have published a paper together. This network contains 2865 nodes and 4733 links with an average degree of 3.3.
- *Condmat* [?]: nodes are scientists working in condensed matter physics, edges encode co-authorship. This network has 14230 nodes and 1196 links with an average degree of 0.17.
- *Conflict* [?], [?]: nodes are countries and edges encode a conflict or a dispute. We have 130 nodes and 180 links in total in this network with an average degree of 2.5.
- *Powergrid* citepowergrid: a network of electric powergrid in US. It has 4941 nodes and 6594 links. The average degree is 2.7.

We evaluate the performance of employed methods by splitting 10 times the data in a train and a test part. For *Protein*, *Metabolic*, *Nips* and *Conflict* networks, we use 10% of the edges to induce the training set while for *Condmat* and *Powergrid* we use 90% of the links. The performance of each method is computed as the average of the AUC-ROC over the 10 rounds.

Model Selection: The values of different hyper-parameters are set by using a 3-fold on the training set, that is, always considering only the training network, we use one fold for fitting the parameters and the rest two folds for validating the effect of the hyper parameter choice. We tune the values of radius for extracting subgraphs R in $\{1, 2\}$, λ in node label function in $\{10, 15\}$, for r and d parameters of NSPDK in $\{1, 2\}$ and $\{1, 2, 3\}$, respectively. Finally, the regularization tradeoff C for the SVM is picked up in $\{10^{-4}, 10^{-3}, 10^{-2},\ 10^{-1}, 1,\ 10,\ 10^{2}, 10^{3},\ 10^{4}\}$.

4.5 Results and discussion

In Table **??**, we report the performance of link prediction methods measured as the AUC-ROC value on 6 datasets. From the results on the table, we

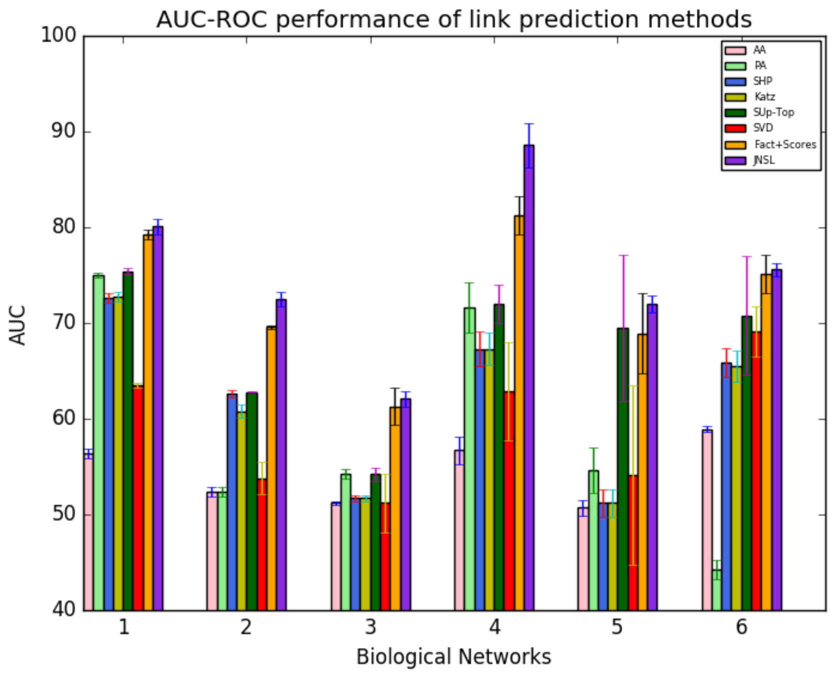

Figure 4.2: AUC-ROC performance on 6 datasets. Legend: AA: Adamic-Adar, PA [?] preferential Attachment, SHP: Shortest Path, Sup-Top [?]: Liear regression running on unsupervised scores, SVD [?]: Singular value decomposition, Fact+Scores [?]: Factorization with unsupervised scores, JNSL: joint neighborhood subgraph link (our method).

can group methods into two groups based on their performances: supervised methods and unsupervised methods. The performance of supervised methods are considerably higher than unsuperivsed ones in most cases, except in the Conflict dataset where Sup-Top outperforms Fact+Scores, but with a very small difference. Concerning supervised methods, JNSL outperforms Fact-Scores in all cases. The difference between their performance is small in PowerGrid and Protein datasets with 0.5% and 0.8%, respectively. And the big gap is in the Condmat dataset with 7.4%.

4.6 Conclusion

We have presented a novel approach to link prediction in absence of side information that can effectively exploit the topological contextual information available in the neighborhood of each edge. We have empirically shown that this approach achieves very competitive results compared to other state-of-

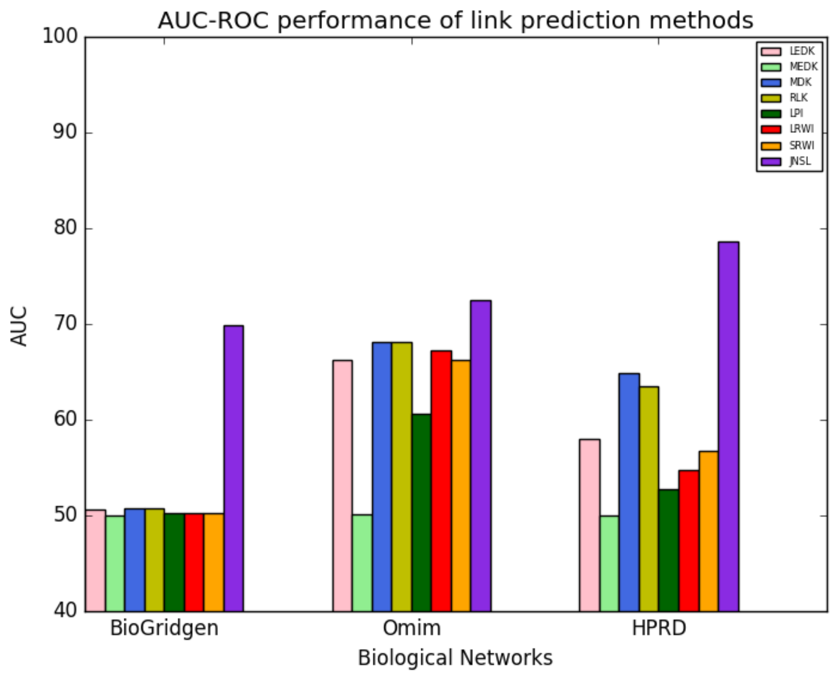

Figure 4.3: AUC-ROC performance on 3 datasets.

the-art methods. In future work, we will investigate how to make use of multiple and heterogeneous information sources when these are available for nodes and edges.

Chapter 5

Link Enrichment for Diffusion-based Graph Node Kernel

5.1 Motivation

A powerful approach to process large heterogeneous sources of data is to use graph encodings [?], [?] and then use graph-based learning systems. In these systems the notion of node similarity is key. A common approach is to resort to graph node kernels such as diffusion-based kernels [?] where the graph node kernel measures the proximity between any pair of nodes by taking into account the paths that connect them. However, when the graph structure is affected by noise in the form of missing links, node similarities are distorted proportionally to the sparsity of the graph and to the fraction of missing links. Two of the main reasons for this are that 1) the lower the average node degree is, the smaller the number of paths through which information can travel, and 2) missing links can end up separating a graph into multiple disconnected components. In this case, since information cannot travel across disconnected components, the similarity of nodes belonging to different components is null. To address these problems we propose to solve a link prediction task prior to the node similarity computation and start studying the question: *how can we improve node similarity using link prediction?* In this work we review both the link prediction literature and the diffusion kernel literature, select a subset of approaches in both categories that seem well suited, focus on a set of node predicting problems in the bioinformatics domain and empirically investigate the effectiveness of the combination of these approaches on the given predictive tasks. The encouraging result that we find is that all the strategies for link prediction we examined consistently enhance the performance on downstream predictive tasks, often significantly improving state of the art results.

5.2 Method

Often the relational information that defines the graph structure is incomplete because certain relations are not known at a given moment in time or have not been yet investigated. When this happens the resulting graphs tend to become sparse and composed of several disconnected components. Diffusion-based kernels are not suited in these cases and show a degraded predictive capacity. Our key idea is to introduce a *link enrichment phase* that can address both issues and enhance the performance of diffusion-based systems.

Given a link prediction algorithm M, a diffusion-based graph node kernel K and a sparse graph $G = (V, E)$ in which $|V| = n$ and $|E| = m$, with $m \approx n$ the link enrichment method consists of two phases:

- enrichment: the link prediction algorithm M is used to score all possible $\frac{n(n-1)}{2} - m$ missing links. The top scoring t links are added to E to obtain E' that defines the new graph $G' = (V, E')$.
- kernel computation: the diffusion-based graph node kernel K is applied to graph $G^{'}$ to compute the kernel matrix K' which captures the similarities between any couple of nodes, possibly belonging to different components in the graph G.

The kernel matrix K' can be used directly by a kernelized learning algorithm, such as a support vector machine, to make predictive inferences.

5.3 Experiments

To empirically study the answer to the question: *how can we improve node similarity using link prediction?* we would need to define a taxonomy of prediction problems on graphs that make use of the notion of node similarity and analyze which link prediction strategies can be effectively coupled with specific node similarity computation techniques for each given class of problems. In addition we should also study the quantitative relation between the degree of missingness and the size of the improvement offered by prepending the link prediction to the node similarity assessment. In this paper we start such endeavor restricting the type of predictive problems to that of node ranking in the sub-domain of gene-disease association studies with a fixed but unknown degree of missingness given by the current medical knowledge. More in detail, the task, known as *gene prioritization*, consists in ranking candidate genes based on their probabilities to be related to a disease on the basis of a given a set of genes experimentally known to be associated to the disease of interest. We have studied the proposed approach on the following 4 datasets:

BioGPS: a gene co-expression graph (7311 nodes and 911294 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 (PCC) between genes is larger than 0.5.

HPRD: a database of curated proteomic information pertaining to human proteins. It is derived from [?] with 9,465 vertices and 37,039 edges. We employ the HPRD version used in [?] that contains 7311 nodes and 30503 edges.

Phenotype similarity: we use the OMIM [?] dataset and the phenotype similarity notion introduced by Van Driel et al. [?] based on the relevance and the frequency of the Medical Subject Headings (MeSH) vocabulary terms in OMIM documents. We built the graph linking those genes whose associated phenotypes have a maximal phenotypic similarity greater than a fixed cut-off value. Following [?], we set the similarity cut-off to 0.3. The resulting graph has 3393 nodes and 144739 edges.

Biogridphys: this dataset encodes known physical interactions among proteins. The idea is that mutations can affect physical interactions by changing the shape of proteins and their effect can propagate through protein graphs. We introduce a link between two genes if their products interact. The resulting graph has 15389 nodes and 155333 edges.

5.3.1 Evaluation Method

To evaluate the performance of the diffusion kernels, we follow [?]: we choose 14 diseases with at least 30 confirmed genes. For each disease, we construct a positive set $\mathcal{P}$ with all confirmed disease genes. To build the negative set $\mathcal{N}$, we randomly sample a set of genes that are associated at least to one disease class, but not related to the class which defines the positive set such that $|\mathcal{N}| = \frac{1}{2}|\mathcal{P}|$. We replicate this procedure 5 times[1]. We assess the performance of kernels via a 3-fold CV, where, after partitioning the

[1]Note that the positive set is held constant, while the negative set varies.

dataset $\mathcal{P} \cup \mathcal{U}$ in 3 folds, we use one fold for training model using SVM and the two remaining folds for testing. For each test gene g_i, the model returns a score s_i proportional to the likelihood of being associated to the disease. Next a decision score q_i is computed as the top percentage value of s_i among all candidate gene scores. We collect all decision scores for every test genes to compute the area under the curve for the receiver operating characteristic (AUC-ROC). The final performance on the disease class is obtained by taking average over 3 folds $\times$ 5 trials.

Model Selection: The hyper-parameters of the various methods are set using a 3-fold on training set in which one fold is used for training the model and two remaining folds are used for validation. We try the values for LEDK and MEDK in $\{0.01, 0.05, 0.1\}$, time steps in MDK in $\{3, 5, 10\}$ and RLK parameter in $\{0.01, 0.1, 1\}$. For CDNK, we try for the degree threshold value in $\{10,\ 15,\ 20\}$, clique size threshold in $\{4,\ 5\}$, maximum radius in $\{1,\ 2\}$, maximum distance in $\{2,\ 3,\ 4\}$. The number of links used for the enrichment are chosen in $\{40\%,\ 50\%,\ 60\%,\ 70\%\}$ of the number of existing links. Finally, the regularization tradeoff C for the SVM is chosen in $\{10^{-4}, 10^{-3}, 10^{-2},\ 10^{-1}, 1,\ 10,\ 10^2,$
$10^3,\ 10^4\}$.

5.4 Results and discussion

5.5 Results and Discussion

In Table 5.1 we report a synthesis of all the experiments. Each row represent a different disease, in the columns we consider the different sources of information used to build the underlying graph (BioGPS, Biogridphys, Hprd, Omim). Note that each resource yields a graph with different characteristic sparsity and number of components. We compare the average AUC-ROC scores in two cases: plain diffusion kernel (denoted by a "-" symbol) and diffusion kernel on a modified graph G' (denoted by a "+" symbol) which includes the novel edges identified by a link prediction system. Here we report the aggregated results (a detailed breakdown is available in *Appendix*[2]) where we have averaged not only across a random choice of negative genes, but also among the type of diffusion kernel and the type of link prediction. The noteworthy result is how consistent the result is: each link prediction method improves each diffusion kernel algorithm, and on average using link prediction yields a 15% to 20% relative error reduction for diffusion-based

[2]https://github.com/dinhinfotech/ICANN/blob/master/appendix.pdf

Table 5.1: *Predictive performance on 14 gene-disease associations using four different graphs induced by the BioGPS, Biogridphys, Hprd and Omim. We report the average AUC-ROC (%) and standard deviations for all difussion-based kernels with (+) and without (-) link enrichment.*

	BioGPS		**Biogridphys**		**Hprd**		**Omim**	
Disease	-	+	-	+	-	+	-	+
1	60.3±1.5	63.4±1.0	73.1±4.1	77.1±2.9	75.5±0.2	77.5±0.9	85.3±1.1	86.9±1.5
2	53.7±1.4	63.4±3.8	56.6±3.4	61.3±4.1	57.1±0.9	60.2±1.8	75.0±2.2	76.5±2.4
3	50.2±0.4	58.6±3.0	58.9±5.9	67.5±7.7	61.8±3.6	70.7±3.8	77.3±1.8	83.1±0.9
4	61.5±0.9	72.2±2.2	65.7±4.1	74.6±4.2	67.3±1.1	71.9±2.2	90.2±1.2	92.1±1.2
5	55.1±0.4	61.7±0.9	54.2±4.8	60.7±4.0	57.7±1.6	67.0±1.8	76.4±0.8	81.9±1.5
6	60.8±0.9	67.9±2.2	60.6±3.6	65.9±3.5	66.8±1.3	71.9±2.3	79.9±2.4	83.3±1.2
7	68.1±1.4	73.4±0.7	57.7±3.2	63.7±4.0	68.9±2.1	72.5±1.2	81.0±1.2	84.1±1.0
8	69.2±2.3	74.0±2.2	68.1±3.6	72.6±2.5	76.6±2.2	80.3±2.8	85.4±2.2	91.0±1.0
9	62.0±1.6	64.5±1.4	68.7±4.6	71.7±4.3	68.4±2.5	75.0±3.2	78.5±0.2	80.6±0.6
10	67.5±2.9	72.9±1.8	58.8±3.2	66.1±3.8	65.8±3.4	74.4±2.6	86.1±0.6	87.8±0.3
11	58.7±1.8	62.3±1.5	58.2±1.2	61.6±1.7	60.1±1.1	64.2±1.5	82.0±1.4	83.6±0.9
12	64.0±1.3	73.6±1.7	59.3±2.1	67.0±2.8	60.8±1.1	68.8±2.8	82.0±1.8	85.9±1.7
13	56.5±0.9	63.3±2.4	55.8±1.1	65.1±4.2	66.4±1.3	71.8±1.7	83.1±2.8	87.5±2.5
14	55.2±0.3	62.3±1.2	55.6±1.6	63.5±4.0	66.3±2.3	71.1±2.8	97.4±0.1	99.0±0.4
$\overline{AUC}$	60.2±0.3	66.7±1.2	60.8±1.6	67.0±4.0	65.7±2.3	71.2±2.8	82.8±0.1	86.0±0.4

methods. What varies is the amount of improvement, which depends on the coupling between the four elements: the disease, the information source, the link prediction method and the diffusion kernel algorithm. In specific we obtain that the largest improvement is obtained for disease 3 (connective) where we have a maximum improvement of 20% ROC points, while the minimum improvement is for disease 8 (immunological) with a minimal improvement of 0% ROC points (see in the detailed report). On average the largest improvement is of 13% ROC points, while the smallest improvement is on average of 1% ROC point. Such stable results are of interest since diffusion-based methods are currently state-of-the-art for gene-disease prioritization tasks, and hence a technique that can offer a consistent and relatively large improvement can have important practical consequences in the understanding of disease mechanisms.

5.6 Conclusion

In this paper we have proposed the notion of *link enrichment* for diffusion kernels, that is, the idea of carrying out the computation of information diffusion on a graph that contains edges identified by link prediction approaches. We have discovered a surprisingly robust signal that indicates that diffusion-based node kernels consistently benefit from the coupling with similarity-based link prediction techniques on large scale datasets in biological domains.

In future work we will carry out a more fine grained analysis, defining a taxonomy of prediction problems on graphs that make use of the notion of node similarity and analyze which link prediction strategies can be effectively coupled with specific node similarity computation techniques for a given problem class. In addition we will study the quantitative relation between the degree of missingness and the size of the improvement offered by prepending the link prediction to the node similarity assessment. Finally, we will extend the analysis to the more complex case of kernel integration and data fusion, i.e. when multiple heterogeneous information sources are used jointly to define the predictive task.

Chapter 6

Scuba - Scalable Kernel-based Gene Prioritization

6.1 Motivation

The identification of the genes underlying human diseases is a major goal in current molecular genetics research. Dramatic progresses have been made since the 1980s, when only a few DNA loci were known to be related to disease phenotypes. Nowadays opportunities for the diagnosis and the design of new therapies are progressively growing, thanks to several technological advances and the application of statistical or mathematical techniques. For instance, positional cloning has allowed to map a vast portion of known Mendelian diseases to their causative genes [?, ?]. However, despite the huge advances, much remains to be discovered. On December 21st 2016, the Online Mendelian Inheritance in Man database (OMIM) registered 4,908 Mendelian phenotypes of known molecular basis and 1,483 Mendelian phenotypes of unknown molecular origin [?]. Moreover, 1,677 more phenotypes were suspected to be Mendelian. But it is among oligogenic and poligenic (and multifactorial) pathologies that the most remains to be elucidated: for the majority of them, only a few genetic loci are known [?, ?].

Independently of the type of disease, the search of causative genes usually concerns a large number of suspects. It is therefore necessary to recognise the most promising candidates to submit to additional investigations, as experimental procedures are often expensive and time consuming. Gene prioritization is the task of ordering genes from the most promising to the least. In traditional genotype-phenotype mapping approaches - as well as in genome-wide association studies - the first step is the identification of the genomic region(s) wherein the genes of interest lie. Once the candidate region is identified, the genes there residing are prioritized and finally analysed for the presence of possible causative mutations [?]. More recently, in new generation sequencing studies this process is inverted as the first step is the identification of mutations, followed by prioritization and final validation [?]. Prioritization criteria are usually based on functional relationships, co-expression and other clues linking genes together. In general, all of them follow the "guilt-by-association" principle, i.e. disease genes are sought by looking for similarities to genes already associated to the pathology of interest [?].

In the last few years, computational techniques have been developed to aid researchers in this task, applying both statistics and machine learning [?]. Thanks to the advent of high-throughput technologies and new generation sequencing, a huge amount of data is in fact available for this kind of investigations. In particular, computational methods are essential for multi-*omics* data integration, that has been recognised as a valuable strategy for understanding genotype-phenotype relationships [?]. In fact, clues are often embedded in different data sources and only their combination leads to the emergence of informative patterns. Furthermore, incompleteness and noise of the single sources can be overcome by inference across multiple levels of knowledge.

Several popular algorithms for pattern analysis are based on *kernels*, which are mathematical transformations that permit to estimate the similarity among items (in our case genes) taking into account complex data relations [?]. Importantly, kernels provide a universal encoding for any kind of knowledge representation, e.g. vectors, trees or graphs. When data integration is required, a multiple kernel learning (MKL) strategy allows a data-driven weighting/selection of meaningful information [?]. The goal of MKL is indeed to learn optimal kernel combinations starting from a set of predefined kernels obtained by various data sources. Through MKL the issue of combining different data types is then solved by converting each dataset in a kernel matrix.

Numerous MKL approaches have been proposed for the integration of genomic data [?, ?] and some of them have been applied to gene prioritization [?, ?, ?, ?]. De Bie *et al* formulated the problem as a one-class support vector machine (SVM) optimization task [?], while Mordelet and Vert tackled it through a biased SVM in a *positive-unlabelled* framework [?, ?]. Recently, Zakeri *et al* proposed an approach for learning non-linear log-euclidean kernel combinations, showing that it can more effectively detect complementary biological information compared to linear combinations-based approaches [?]. However, as highlighted in a recent work by Wang *et al* [?], current methods share two limitations: high computational costs - given by a (at least) quadratic complexity in the number of training examples - and the difficulty to predefine optimal kernel functions to be fed to the MKL machine.

In this work we tackle these issues by proposing a novel scalable gene prioritization method based on a particular MKL approach [?]. By this approach, the optimal kernel is efficiently computed by maximizing the distance between positive and negative examples and optimizing the margin distribution [?]. This permits to obtain a high scalability relatively to the number of kernels, with a linear time complexity and a practically constant memory requirement. However, this approach assumes comparable label noise in the two example distributions, which does not reflect the case in consideration. Moreover, it does not scale with the number of training examples. Here we introduce a new algorithm, specifically adapted to a *positive-unlabelled* unbalanced framework and we apply it to gene prioritization for the first time. The new learning algorithm has an additional gain in scalability that comes particularly useful when large numbers of genes have to be prioritized. This scalability allows us to transform each data source by multiple kernels and alleviates the issue of defining appropriate base kernels for each source. We called the proposed method Scuba (SCalable UnBAlanced gene prioritization).

From an experimental point of view, here we focus on the integration of multiple gene networks whose edges symbolize functional relationships from heterogeneous sources and we employ two different test settings. In the first setting, we reproduce the procedure presented in a previous work by Chen *et al* [?], built upon cross-validation experiments [?] on collections of known disease genes. This kind of evaluation is useful to compare different methods, but results may suffer from overestimation due to the reliance of many data repositories on medical literature or external data sources like OMIM [?]. Such dependence introduces a bias that may favour the retrieval of known disease genes. Thus, as a second validation we employ a more realistic setting, following a previous evaluation of gene prioritization tools by Börnigen *et al* [?]. Here performance measures focus on the ability of predicting disease genes discovered subsequently to the last update of datasets.

Overall, we compare Scuba with other 14 gene prioritization systems, including other 2 kernel-based methods and 8 web tools. We find that

Scuba has competitive accuracy and in particular yields the best results in genome-wide prioritizations, showing its value for large-extent applications.

6.2 Method

In this section, we first introduce and formalize some concepts that will be used throughout this paper. Then, we present the proposed approach in detail.

Disease gene prioritization: Let us consider a set of genes $\mathcal{G} = \{g_1, g_2, \ldots, g_N\}$ that represents either the global set of genes in the genome or a subset of it. Given another set $\mathcal{P} = \{g_1, g_2, \ldots, g_m\}$, $\mathcal{P} \subset \mathcal{G}$ containing genes known to be associated to a genetic disease, gene prioritization is the task that aims to rank genes in the set of candidates $\mathcal{U} = \mathcal{G} \setminus \mathcal{P}$ according to their likelihood of being related to that disease. Genes in $\mathcal{P}$ are labelled as *positive* and represent a secure source of information. In contrast, candidate genes in $\mathcal{U}$ are technically *unlabelled*, as we expect that some of them may be associated to the disease but we do not know which ones. Under this notation, this problem can be posed as a *positive-unlabelled* (PU) learning task [**?**, **?**].

Kernel: *Kernels* can be informally seen as similarity measures between pairs of data examples. Mathematically, such similarities are defined by inner products between vectors of corresponding examples in a Hilbert space $\mathcal{H}$, without the need of an explicit transformation to that space. A kernel function k on $\mathcal{X} \times \mathcal{X}$ is then formally defined as:

$$k : \mathcal{X} \times \mathcal{X} \longrightarrow \mathcal{R}$$
$$k(x_1, x_2) = <\phi(x_1), \phi(x_2)> ,$$

where $x_1, x_2 \in \mathcal{X}$, ϕ is a mapping $\phi : \mathcal{X} \longrightarrow \mathcal{H}$ and k needs to be (1) symmetric, i.e. $k(x_1, x_2) = k(x_2, x_1)$ (2) semi-definite, i.e. the kernel matrix defined by $k_{ij} = k(x_i, x_j)$ has all eigenvalues ≥ 0. Kernels can be used to define similarities starting from various data types, like graph nodes.

Graph node kernel: A graph $G = (V, E)$ is a structure consisting of a node set $V = \{v_1, \ldots, v_N\}$ and an edge set $E = \{(v_i, v_j) | v_i, v_j \in V)\}$. A graph node kernel aims at defining a similarity between any couples of nodes in a graph. A considerable number of graph node kernels have been introduced. The most popular is the diffusion kernel [**?**] which is based on the heat diffusion phenomenon. The key idea is to allow a given amount of *heat* on each node and let it *diffuse* through the edges. The similarity between two nodes v_i, v_j is then measured as the amount of heat starting from v_i and reaching v_j over an infinite time interval. In the diffusion kernel the heat flow is proportional to the number of paths connecting two nodes, introducing a bias that penalizes peripheral nodes with respect to central ones. This problem is tackled by a modified version called Markov exponential diffusion kernel (MEDK) [?] where a Markov matrix replaces the adjacency matrix. Another kernel called Markov diffusion kernel (MDK) [?], exploits the notion of *diffusion distance*, a measure of similarity between patterns of heat diffusion. The regularized Laplacian kernel (RLK) [?] implements 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.

Scalable Multiple Kernel Learning: EasyMKL

We approach the problem of disease gene prioritization by employing a graph-based integration in which we use graph node kernels to extract gene information and encode it in the form of kernel matrices. However, a big challenge is how to effectively combine kernels when building predictive systems. This challenge can be solved by MKL. In the following, we first formalize the MKL problem and we then briefly introduce a scalable MKL algorithm named EasyMKL [?].

Given a set of pre-defined kernels, multiple kernel learning is a task that aims at finding an optimal kernel combination:

$$\mathbf{K} = \psi(\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_R). \tag{6.1}$$

Recently, many MKL methods have been proposed [?, ?]. However, most of them require a long computation time and a high memory consumption, especially when the number of pre-defined kernels is high. To tackle these limitations, a scalable multiple kernel learning named EasyMKL has been proposed in [?]. This method focuses on learning a linear combination of the input kernels with positive linear coefficients, namely

$$\mathbf{K} = \sum_{r=1}^{R} \eta_r \mathbf{K}_r, \ \eta_r \geq 0\,, \tag{6.2}$$

where $\eta = (\eta_1, \ldots, \eta_R)$ is the coefficient vector. In particular, EasyMKL computes the optimal kernel by maximizing the distance between positive and negative examples. Its formulation is based on a previous kernel-based approach for the optimization of the margin distribution in binary classification or ranking tasks [?]. Let us define then the probability distribution

$\gamma \in \mathbb{R}_+^N$ representing weights assigned to training examples and living in the domain $\Gamma = \{\gamma \in \mathbb{R}_+^N | \sum_{i \in \oplus} \gamma_i = 1, \sum_{i \in \ominus} \gamma_i = 1\}$. Under this notation, the EasyMKL task can be posed as a min-max problem over variables γ and η as follows:

$$\max_{\eta : \|\eta\|_2 \leq 1} \min_{\gamma \in \Gamma} (1-\lambda)\gamma^\top \mathbf{Y}(\sum_r \eta_r \mathbf{K}_r)\mathbf{Y}\gamma + \lambda \gamma^\top \gamma \,. \tag{6.3}$$

Here $\mathbf{Y}$ is a diagonal matrix containing the vector of example labels. The optimization of the first term alone leads to the two nearest points in the convex hulls of positive and negative examples and is equivalent to a hard SVM task using a kernel $\mathbf{K}$ [?]. The second term represents a quadratic regularization over γ whose objective solution is the squared distance between positive and negative centroids in the feature space. The regularization parameter $\lambda \in [0, 1]$ permits to tune the objective to optimize, by balancing between the two critical values $\lambda = 0$ and $\lambda = 1$. When $\lambda = 0$ the regularization disappears, while when $\lambda = 1$ it makes the whole objective.

It can be shown that this problem has analytical solution in the η variable, so that the previous expression can be reshaped into:

$$\min_{\gamma \in \Gamma} (1-\lambda)\gamma^\top \mathbf{Y}\mathbf{K}^s\mathbf{Y}\gamma + \lambda \gamma^\top \gamma \,, \tag{6.4}$$

where $\mathbf{K}^s = \sum_r^R \mathbf{K}_r$ is the sum of the pre-defined kernels.

This minimization can be efficiently solved and only requires the sum of the kernels. The computation of the kernel summation can be easily implemented incrementally and only two matrices need to be stored in memory at a time. As shown in [?], EasyMKL can deal with an arbitrary number of kernels using a fixed amount of memory and a linearly increasing computation time.

Once the problem in Eq. 6.4 is solved, we are able to obtain the optimal weights η_r^* by using the formula:

$$\eta_r^* = \frac{\gamma^* \mathbf{Y}\mathbf{K}_r\mathbf{Y}\gamma^*}{\sum_{r=1}^R \gamma^* \mathbf{Y}\mathbf{K}_r\mathbf{Y}\gamma^*}. \tag{6.5}$$

The optimal kernel is thus evaluated as $\mathbf{K} = \sum_r^R \eta_r^* \mathbf{K}_r$. Finally, by replacing $\mathbf{K}^s$ with $\mathbf{K}$ in Eq. 6.4, we can get the final probability distribution γ^*.

Unbalanced Multiple Kernel Learning: Scuba

In the previous section we introduced EasyMKL, a scalable, efficient kernel integration approach. However, the gene prioritization task has two additional issues that complicate the work. First, our learning setting is not fully supervised: an assumption is that there are some positive examples hidden among the negatives and we want to retrieve them. Thus, we have the certainty about positive examples but not about negative ones. Second, the number of known disease genes is typically much smaller than the number of candidates, making the problem strongly unbalanced. For these reasons, inspired by a previous work [?] we propose a new MKL algorithm based on EasyMKL that not only inherits its scalability, but also efficiently deals with an unbalanced setting.

In order to clearly present our method, we first need to highlight the different contributions given by positive and unlabelled examples. Therefore, we define $\mathbf{K}^{+}$, $\mathbf{K}^{-}$ and $\mathbf{K}^{+-}$ the sub-matrices of $\mathbf{K}^{s}$ pertaining to positive-positive, unlabelled-unlabelled and positive-unlabelled example pairs, respectively. Schematically, we have:

$$\mathbf{K}^{s} = \begin{pmatrix} \mathbf{K}^{+} & \mathbf{K}^{+-} \\ \mathbf{K}^{-+} & \mathbf{K}^{-} \end{pmatrix} .$$

being $\mathbf{K}^{-+}$ the transpose of $\mathbf{K}^{+-}$. In other words, $\mathbf{K}^{+}$ contains similarities among positive examples (known disease genes), $\mathbf{K}^{-}$ contains similarities among unlabelled examples (candidate genes) and $\mathbf{K}^{+-}$ includes similarities between positive-unlabelled example pairs. In the same way, we define γ_{+} and γ_{-} as the probability vectors associated to positive and unlabelled examples, respectively.
Under this change of variables, we reformulate the problem as:

$$\min_{\gamma\in\Gamma} \gamma_{+}^{\top}\mathbf{K}^{+}\gamma_{+} - 2\,\gamma_{+}^{\top}\mathbf{K}^{+-}\gamma_{-} + \gamma_{-}^{\top}\mathbf{K}^{-}\gamma_{-} + \lambda_{+}\gamma_{+}^{\top}\gamma_{+} + \lambda_{-}\gamma_{-}^{\top}\gamma_{-} \, .$$

In this new formulation, the original EasyMKL problem is obtained by setting $\lambda_{+} = \lambda_{-} = \frac{\lambda}{1-\lambda}$. However, due to the unbalanced PU nature of the problem, we are interested in using two different regularizations among positive and unlabelled examples. In our case, we decide to fix *a priori* the regularization parameter $\lambda_{-} = +\infty$, corresponding to fixing $\lambda = 1$ over unlabelled examples only. Then, the solution of part of the objective function is defined by the uniform distribution $\gamma_{-} = (\frac{1}{n}, \frac{1}{n}, \dots \frac{1}{n}) \equiv u$, where n is the

number of unlabelled examples.
We inject this analytic solution of part of the problem in our objective function as

$$\min_{\gamma \in \Gamma^+} \gamma_+^\top \mathbf{K}^+ \gamma_+ - 2\,\gamma_+^\top \mathbf{K}^{+-} u + u^\top \mathbf{K}^- u \\ + \lambda_+ \gamma_+^\top \gamma_+ + \lambda_- u^\top u\,,$$

where $\Gamma^+ = \{\gamma \in \mathcal{R}_+^m | \sum_{i:y_i=1} \gamma_i = 1, \gamma_j = 1/n\, \forall j : y_j = -1\}$ is the probability distribution domain where the distributions over the unlabelled examples correspond to the uniform distribution. It is trivial that $u^\top \mathbf{K}^- u$ and $\lambda_- u^\top u$ are independent from the γ_+ variable. Then, they can be removed from the objective function obtaining

$$\min_{\gamma \in \Gamma^+} \gamma_+^\top \mathbf{K}^+ \gamma_+ - 2\,\gamma_+^\top \mathbf{K}^{+-} u + \lambda_+ \gamma_+^\top \gamma_+\,. \tag{6.6}$$

In this expression, we only need to consider the entries of the kernel $\mathbf{K}^s$ concerning the positive set, avoiding all the entries with indices in the unlabelled set. The complexity becomes quadratic in the number of positive examples m, which is always much smaller than the number of examples to prioritize. Moreover, this algorithm still depends linearly on the number of kernels R and the overall time complexity is then $\mathcal{O}(m^2 \cdot R)$. In this way, we greatly simplify the optimization problem, while being able to take into account the diverse amount of noise present in positive and unlabelled example sets.
Like in the previous section, after solving the problem of Eq. 6.6 we use Eq. 6.5 to compute the optimal kernel weights. Next, we solve again the Scuba optimization problem to get the final optimal probability distribution γ^*.
The likelihood of association to disease for every gene is given by the vector of scores s defined as

$$s = \mathbf{K}\mathbf{Y}\gamma^*\,, \tag{6.7}$$

where $\mathbf{K}$ is the final kernel matrix. Candidate genes are then prioritized on the basis of the score computed through this scoring function.

Base kernels selection

We leverage the scalability achieved by the new algorithm to ease the optimization of base kernels. As a general practical case, we start from a set of data sources $\mathcal{S} = \{S_1, S_2, \ldots, S_L\}$ representing various levels of biological information. We apply kernels with different parameter values on each

$S_i \in \mathcal{S}$. As a consequence, for each source S_i, we get a set of kernel matrices $\mathcal{K}_i = \{K_{i1}, K_{i2}, \ldots, K_{iH}\}$. By collecting all kernels from all $\mathcal{K}_i$, we achieve a final kernel matrix set $\mathcal{K}$ comprising $L \cdot H$ matrices. Next, all matrices in $\mathcal{K}$ and gene sets $\mathcal{P}$ and $\mathcal{U}$ are fed into Scuba to obtain the optimal kernel K. In this way, we directly use MKL to perform an automatic selection of optimal kernel parameters. The final kernel and the disease gene set $\mathcal{P}$ are then employed to train a model, which is used to generate a score list for candidate genes in $\mathcal{U}$ through Eq. 6.7. The score assigned to a candidate expresses the likelihood of it being associated to the disease.

Experimental workflow

We employ Scuba to prioritize candidate genes starting from multiple gene networks, obtained by various data sources. We transform every network by means of multiple graph node kernels as explained in the previous section. In the cross-validation experimental setting we use MEDK to estimate the similarity among genes, just like in [?]. In the unbiased setting we use MDK and RLK, selected by validating on training sets.

In both settings, we fix the number of kernel matrices per data source $H = 3$ and learn the regularization parameter λ_+ by employing k-fold cross validation on the training set, using the the grid of values $\{0,\ 0.1,\ 0.2,\ \ldots\ 1\}$. Kernel parameter values are set as follows: $\{0.01,\ 0.04,\ 0.07\}$ for MEDK, $\{2,\ 4,\ 6\}$ for MDK and $\{1,\ 10,\ 100\}$ for RLK.

Data sources

We employ several biological data sources to test Scuba, presented in the following.

- **Human Protein Reference Database** (**HPRD**) [?]. The HPRD resource provides protein interaction data which we implement as an unweighted graph, where genes are linked if their corresponding proteins interact.

- **BioGPS** [?]. It contains expression profiles for 79 human tissues, which are measured by using the Affymetrix U133A array. Gene co-expression, defined by pairwise Pearson correlation coefficients (PCC), is used to build an unweighted graph. A pair of genes are linked by an edge if the PCC value is larger than 0.5.

- **Pathways**. Pathway datasets are obtained from the database of KEGG [?], Reactome [?], PharmGKB [?] and PID [?], which contain 280, 1469, 99 and 2679 pathways, respectively. A pathway coparticipation network is constructed by connecting genes that coparticipate in any pathway.

- **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, from which we extracted functional links among 17078 human genes.

The first three datasets were obtained directly from Chen *et al* [?], already preprocessed in such a way that all of them represent exactly the same 7311 genes. We employed this data without any further processing.

Known gene-disease associations employed in the cross-validation experimental setting were taken from a work of Goh *et al* which defines classes of related diseases [?]. Training and candidate gene sets used in the second set of experiments (section *Unbiased evaluation*) were obtained from the supplementary material of the unbiased evaluation of gene prioritization tools performed by Börnigen *et al* [?]. Finally, gene-disease associations from the Human Phenotype Ontology were used, belonging to builds 29 and 117 [?].

Other kernel-based gene prioritization methods

We compare Scuba with other two kernel methods for gene prioritization. The first one implements a one class approach to MKL, slightly modifying the formulation of the method of De Bie *et al* [?]. In the corresponding work [?], authors show that this newer approach reaches higher performances in ranking. In the following, we refer to it as MKL1class. The second method we consider is ProDiGe, a PU approach that combines MKL and multitask learning [?]. We focus on its first version without multitask learning, as our purpose is to study performances in terms of the MKL framework. We ran ProDiGe using the default parameters indicated in the corresponding paper: number of bagging iterations $B = 30$ and regularization parameter $C = 1$. In the same way, we set the regularization parameter $\nu = 0.5$ for MKL1class.

6.3 Experiments

We employ Scuba to prioritize candidate genes starting from multiple gene networks, obtained by various data sources. We transform every network by means of multiple graph node kernels as explained in the previous section. In the cross-validation experimental setting we use MEDK to estimate the similarity among genes, just like in [?]. In the unbiased setting we use MDK and RLK, selected by validating on training sets.

In both settings, we fix the number of kernel matrices per data source $H = 3$ and learn the regularization parameter λ_+ by employing k-fold cross validation on the training set, using the the grid of values $\{0,\ 0.1,\ 0.2,\ \ldots\ 1\}$. Kernel parameter values are set as follows: $\{0.01,\ 0.04,\ 0.07\}$ for MEDK, $\{2,\ 4,\ 6\}$ for MDK and $\{1,\ 10,\ 100\}$ for RLK.

Data sources

We employ several biological data sources to test Scuba, presented in the following.

- **Human Protein Reference Database** (**HPRD**) [?]. The HPRD resource provides protein interaction data which we implement as an unweighted graph, where genes are linked if their corresponding proteins interact.

- **BioGPS** [?]. It contains expression profiles for 79 human tissues, which are measured by using the Affymetrix U133A array. Gene co-expression, defined by pairwise Pearson correlation coefficients (PCC), is used to build an unweighted graph. A pair of genes are linked by an edge if the PCC value is larger than 0.5.

- **Pathways**. Pathway datasets are obtained from the database of KEGG [?], Reactome [?], PharmGKB [?] and PID [?], 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.

- **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, from which we extracted functional links among 17078 human genes.

The first three datasets were obtained directly from Chen *et al* [?], already preprocessed in such a way that all of them represent exactly the same 7311 genes. We employed this data without any further processing.

Known gene-disease associations employed in the cross-validation experimental setting were taken from a work of Goh *et al* which defines classes of related diseases [?]. Training and candidate gene sets used in the second set of experiments (section *Unbiased evaluation*) were obtained from the supplementary material of the unbiased evaluation of gene prioritization tools performed by Börnigen *et al* [?]. Finally, gene-disease associations from the Human Phenotype Ontology were used, belonging to builds 29 and 117 [?].

Other kernel-based gene prioritization methods

We compare Scuba with other two kernel methods for gene prioritization. The first one implements a one class approach to MKL, slightly modifying the formulation of the method of De Bie *et al* [?]. In the corresponding work [?], authors show that this newer approach reaches higher performances in ranking. In the following, we refer to it as MKL1class. The second method we consider is ProDiGe, a PU approach that combines MKL and multitask learning [?]. We focus on its first version without multitask learning, as our purpose is to study performances in terms of the MKL framework. We ran ProDiGe using the default parameters indicated in the corresponding paper: number of bagging iterations $B = 30$ and regularization parameter $C = 1$. In the same way, we set the regularization parameter $\nu = 0.5$ for MKL1class.

In this section, we describe the tests made to evaluate our proposed method, which follow two different experimental procedures. In the first setting, we aim at estimating Scuba performance in a standard validation framework. In the second setting we evaluate it by an unbiased approach, making a comparison with prioritization tools available on the web and with two state-of-the-art kernel-based methods.

Cross-validation

As a first evaluation of Scuba, we followed the experimental protocol used by Chen *et al* to test predictive performance of other prioritization methods

[?]. In this setting, we employed three data sets: BioGPS, HPRD and Pathways, which we borrowed from the authors of the work. To perform the experiments, we employed known gene-disease associations from OMIM, grouped into 20 classes on the basis of disease relatedness by Goh *et al* [?]. Among those classes we selected the 12 with at least 30 confirmed genes. We then built a training set consisting of a positive set P and a reliably negative set N (still unlabelled in practice) for each of them. P contains all its disease gene members. N is constructed by randomly picking genes from known disease genes such that $|N| = \frac{1}{2}|P|$. The unknown genes relate to at least one disease class, but do not relate to the current class. We chose the genes in N from the other disease genes because we assume that they were less likely to be associated to the considered class. In fact, disease genes are generally more studied and a potential association has more chances to have already been identified.

After that, leave-one-out cross validation was used to evaluate the performance of the algorithm. Iteratively, every gene in the training set was selected to be the test gene and the remaining genes in P and N were used to train the model. Once the model was trained, a score list for the test gene and the candidate genes was computed. Then, we computed a decision score for each test gene representing the percentage of candidate genes ranked lower than it. We collected all decision scores for every gene in all disease classes to form a global decision score list. The performance of Scuba was measured by calculating the area under the curve (AUC) in the receiver-operating-characteristic plot obtained from the decision score list. The AUC expresses the probability that a randomly chosen disease gene is ranked above a randomly picked non-disease gene for any disease class.

Table **??** illustrates the performance of different techniques in this experimental setting reported by Chen *et al* [?], and the performance of our proposed method. In the second column we show the significance of the difference between reported AUCs and Scuba AUC[?]. Scuba performs significantly better than all other methods, getting an AUC around 3.6% greater than the second best performing technique, F3PC.

Unbiased evaluation

Although the previous evaluation is useful to compare Scuba with other methods, predictive performance in cross-validation experiments may be inflated compared to real applications. Indeed, the retrieval of known disease genes can be facilitated by various means. One mean is the crosstalk between data repositories: for example, KEGG [?] draws its information also from medical literature. Moreover, often the discovery of the link between a gene and a disease coincides with the discovery of a functional annotation or of a molecular interaction. In practice, instead, researchers are interested in novel associations, which in most cases are harder to find due to a lack of information around them.

In order to achieve a thorough evaluation of Scuba, we tested it in a more realistic setting, following the work of [?]. In this study, several gene prioritization tools were benchmarked as follows. Newly discovered disease-gene associations were collected over a timespan of six months, gathering 42 associations. As soon as a new association was discovered, eight pre-selected gene prioritization web tools were queried with a proper training set in order to mimic the discovery through each of them. These 42 predictions were used to assess the ability of the tools to successfully prioritize disease genes. The idea behind this procedure is to anticipate the integration of the associations in the data sources and so avoid biased predictions.

In order to test Scuba in this setting, we backdated our data to a time prior to May 15, 2010 by employing String v8.2 data [?]. The candidate sets were constructed by considering all genes with Ensembl [?] gene identifier within the chromosomal regions around the test genes, in order to get on average 100 candidates for each trial. Then, we performed prioritizations for each test gene in two distinct cases - genome-wide and candidate set-based prioritizations. In genome-wide prioritizations all coding genes in the genome were prioritized, while in candidate set-based tests only the genes belonging to the candidate groups were ranked. In both cases, we normalized ranking positions over the total number of genes in order to get the median and the standard deviation of the normalized ranks for test genes. We also computed the true positive rate (TPR) relatively to some representative thresholds (5%, 10% and 30% of the ranking) and the AUC obtained by averaging over the 42 prioritizations.

Along with Scuba, we evaluated in this setting also MKL1class [?] and ProDiGe [?], two state-of-the-art kernel based gene prioritization methods. In Table **??** it is possible to see performances for all three methods and their significance assessed by Wilcoxon signed rank tests. With a significance threshold of 0.05, Scuba achieves the significantly higher performances in genome-wide tasks compared to both baselines. In the candidate set-based setting, it performs significantly better than ProDiGe and better, although not significantly, than MKL1class.

6.4 Results and discussion

In Table **??** we show results for Scuba compared with the methods considered in the work of Börnigen *et al* [**?**]. In genome-wide predictions, Scuba dominates over the other tools. On predictions over smaller candidate sets, it is still competitive although best results are achieved by GeneDistiller [**?**], Endeavour [**?**] and ToppGene [**?**]. It is important to underline that in this case considered tools rely on different data sources, so we are comparing different prioritization systems rather different algorithms. Furthermore, tools are in some cases unable to provide an answer to a given task, depending on the underlying data sources (for more details see the original work [**?**]). We report the fraction of prioritizations on which tools are actually evaluated as response rate. This table has the purpose of showing the potentiality of Scuba relatively to what is easily accessible by non-bioinformaticians. However, since we used the String data for instance Scuba is directly comparable with Pinta [**?**].

Next, we expanded this validation by employing gene-disease annotations derived from the Human Phenotype Ontology (HPO) [**?**]. This resource gathers information from several databases and makes available its monthly updates, permitting to trace the annotations history. We downloaded the HPO build 29 - dating March 2013 - and build 117 of February 2017. We compared the two annotations corresponding to these versions of HPO and extracted the gene-disease associations that were added in this time gap. We concentrated on the multifactorial diseases covered in the previous analysis, that could possibly have some previously undiscovered associations. We thus analyzed how the obtained genes are ranked in genome-wide prioritizations of the previous analysis, applying the same performance measures as before. The outcome is an analogous evaluation, but this time target genes are those extracted from HPO.

In Table **??** results for Scuba, MKL1class and ProDiGe are shown. We can observe a slightly different trend compared to previous results, with Scuba and ProDiGe having very close performance and MKL1class being significantly worse than Scuba.

Gene prioritization is progressively becoming essential in molecular biology studies. In fact, we are assisting to a continuous proliferation of a variety of *omic* data brought by technological advances. In the near future it is then likely that more heterogeneous knowledge will have to be combined. Moreover, the classes of biological agents to be prioritized are going to enlarge. For instance, we are only beginning to understand the complex regulation machinery involving non-coding RNA and epigenetic agents. It is estimated that around 90.000 human long non coding genes exist, whose functional implications are progressively emerging [?]. Facing this challenge, the development of novel methods is still strongly needed in order to enhance predictive power and efficiency.

Compared to the considered benchmark kernel methods - MKL1class and ProDiGe - Scuba has some important advantages. ProDiGe is one of the first proposed kernel-based PU learning method for gene prioritization [?]. It implements a PU learning strategy based on a biased SVM, which over-weights positive examples during training. In order to reach scalability to large datasets, it leverages a bagging procedure. Like ProDiGe, Scuba implements a learning strategy based on a binary classification set up, but from a different perspective. In a PU problem, the information on positive examples is assumed secure, while the information on negative examples is not - which indeed are unlabelled. In terms of margin optimization, this translates in unbalanced entropy on the probability distributions associated to the two sets of training examples. It is then required to regularize more on the unlabelled class - having higher entropy - and in the limit of maximum uncertainty we get the uniform distribution.

MKL1class implements another effective approach for data integration, namely single class learning. This means that the model is obtained solely based on the distribution of known disease genes, disregarding unlabelled ones. Scuba has enhanced scalability compared to MKL1class, as it involves the optimization of the 1-norm of the margin vector from the different kernels. In contrast, MKL1class optimizes its 2-norm, which is more computationally demanding. Importantly, another distinctive feature of Scuba is a time complexity dependent on the number of positive examples and not on the number of total examples. This may be of great advantage as typically disease genes are orders of magnitude less numerous than the candidates.

Results from two different evaluation settings show that our proposed method Scuba outperforms many existing methods, particularly in genome-wide analyses. Compared to the two considered existing kernel-based methods, Scuba performances (considering AUC) are always higher, and often significantly higher. Moreover, Scuba has two main levels of scalability that make it particularly suitable for gene prioritization:

- **Scalability on number of kernels**: Scuba is able to deal with a large number of kernels defined on different data sources. As a consequence, it can be useful to get a more unified view of the problem and to build more powerful predicting models.

- **Scalability on number of training examples**: In typical gene prioritization problems, the number of known disease genes is much smaller than the number of candidates. Scuba is designed to efficiently deal with unbalanced settings and at the same time take advantage of the whole candidates distribution.

Altogether, our results show that Scuba is a valuable tool to achieve efficient prioritizations, especially in large-scale investigations. A detailed overview on the validation results for single diseases is available in Supplementary Tables 1, 3, 4.

Finally, as it is visible in Supplementary Table 2, performance with multiple kernels might be close to those with single kernels. Nevertheless, feeding multiple kernels into Scuba alleviates the issue of choosing appropriate kernels for each data source, as implemented in our work. Importantly, this strategy can also provide multiple views on the same data and possibly increase performance. Nevertheless caution must be paid since the more kernels are combined and the more parameters have to be learned, thus increasing the risk of over-fitting. We advice then to moderate the number of kernel matrices generated from each data source.

6.5 Conclusion

In this work, we propose a novel computational kernel-based method to guide the identification of novel disease genes. Our method takes advantage of complementary biological knowledge by combining heterogeneous data sources. Every source can be transformed by appropriate kernel functions in order to take full advantage of its information. Our original algorithm is scalable relatively to the size of input data, number of kernel transformations employed and number of training examples. Experimental results support the thesis that Scuba is an effective approach and can be applied in various disease domains.

Scuba only requires a collection of input genes and optionally a set of candidate genes. The simple requirements make it applicable to a wide range of laboratory investigations. Furthermore, Scuba can be potentially employed also in other prioritization problems, as long as a PU approach and the integration of heterogeneous biological knowledge are needed.

Chapter 7

Disjuctive Interconnection Graphs for Disease Gene Prioritization

7.1 Motivation

Disease-gene association recovery is a major goal in molecular biology and medical that has received much attention from many researchers. As a consequence, a big progress has been made in the last decade. Concerning a genetic disease, there normally a small number of genes known to be related to it. 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). It 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. A gene prioritization method aims at ordering candidate genes from the most to the least probable to be associated to the disease. The top genes in the ranking are then sent to biologists and medical scientists for further studies to determine whether each gene is related to considered disease.

Genetics data are growing at a fast pace thanks to the development of High-throughput technologies. Different data sources are made available in which each data source focuses on a specific aspect of genes. Therefore, only integration of different heterogeneous data sources is supposed to lead to high performance of gene inference systems. It is due to combining sources brings more unified views about genes. Graphs are known the best data structures for genetic data where gene relations are encoded as links. So the data integration can now turn to graph integration.

One of the key points which determins the performance of machine learning systems is the definition of the entity similarity, graph node similarity in our case. Kernel is well-known as a universal paradigm used to capture similarity between entities any any kind of form. Thus, many graph node kernels are proposed to measure similarity between graph nodes and applied to graph-based data integration systems solving for disease gene prioritization. However, most methods/systems are based on multiple kernel learning that first compute kernel for each graph separately and then try to learn a final normally based on a data driven way. It shows a disadvantage since data driven only can impact to graph unit level or all nodes (genes) in a graph are assigned a same weight when combining.

In this paper, we first propose a novel graph integration which allows genes not only linked with genes in the same graph but also are able to be linked with genes from all other graphs. We then employ a relevant compositional graph kernel named conjuctive disjuctive graph node kernel (CDNK).

7.2 Related work

In the last decade, there is a number of gene prioritization methods which are based on data integration have been proposed. They can be classified into two groups by considering their integration stage.

The first group contains methods that first separately define the similarities between genes from each data source. They then combine these obtained similarities to have the final gene similarities.

[?] proposed a method named EPU which is a modification of PUDI [?]. Unlike most other methods that treat unknown genes homogeneously, in EPU, for each data source, the unknown gene set is partitioned into multiple positive and negative sets with different confidence scores (weights). It then employs an ensemble procedure to compute the final weight for unweighted genes in the unlabelled set. Besides, three weighted classifiers are applied. Finally, ensamble PU learning is used to decide whether an unlabelled gene belongs to positive or negative set. Experiments show that EPU gets significantly higher performance compared to many methods.

[?] proposed a candidate gene prioritization approach, DIR, that can integrate multiple data sources by taking advantage of a unified graphic representation of information. DIR employs a diffusion kernel to define a global similarity for all pairs of genes in every source. Later it defines a data integration rank score to select the most informative evidence among a set of data sources.

ProDiGe ([?]) is an efficient method that uses MKL to combine data sources [8]. The global similarity between genes are computed by using the inner product between feature vectors and a diffusion kernel for PPI. Then a MKL algorithm is used to combine pre-defined kernels and learn an optimal kernel. Next, it uses a biased SVM to identify potential gene-disease associations. In ProDiGe, the set of data sources can be expanded easily. However, the use of a single kernel type and the employed MKL algorithm could be considered as limitations.

[?] a kernel-based Markov random field algorithm is proposed. In this method, graph kernels are used to define global similarity between genes. From a prior probability vector, MRF algorithm computes the posterior probability vector. This vector gives the probability of each candidate gene to be related to the considered disease. The use of an adaptation of the Gibbs sampling procedure allows to assign weights for different data sources. MRF method is flexible in terms of data integration and it outperforms other previous methods that use Markov random field. However, during the Gibbs sampling process, it takes a long time in order to maintain a long Markov chain for every gene.

A method proposed in [?], notated as F3PC, shows better results compared with many disease gene prioritization tools. The strength of this method is the use of modified conditional random field model, that simultaneously utilizes both gene annotations and gene interactions while preserving their original representation.

[?] introduced a scalable kernel-based method for gene prioritization in which an efficient of multiple kernel learning is proposed to find an optimal kernel given a set of predefined ones. Scuba can deal with a high number of data sources with a linear time complexity w.r.t number of sources and a constant memory consumption.

The second group includes algorithms that merge graphs derived from different sources to form a single one. It then computes gene similarities underlying the achieved graph.

[?] introduces a method named RWR that merges different networks into a single one. Then a global similarity measure, the random walk with restart algorithm, is applied to rank the candidate genes. This method often shows better performance compared to some other methods in terms of prediction accuracy and time consumption. Although it integrates useful information from different data sources, it also integrates noise and it does not take the individual weight of sources into account.

In order to facilitate for users, some methods for disease gene prioritization are made available as web tools such as Suspects [?], ToppGene [?], GeneDistiller [?], GeneWanderer [?], Posmed [?], Candid [?], Endeavour [?] and Pinta [?]. In [?], many web tools are described and evaluated.

Regarding first group, the similarities of nodes are computed separately from different graphs. They are then combined by assigning a weight for each source. It means, every nodes in a same graph are assigned with a same weight. Therefore, it is not able to distingush the impact level of different genes on a considered disease. The fact is that some genes are more relevant the the disease (task) than others. For the second group, graphs are combined before a single similarity measure is carried out. In this case, methods cannot effectively take the advantage from the use of multiple graphs since the properties of individual graphs could not be preserved.

In this paper, we propose a method that can overcome the aboved analysed limitations from these two groups. More specifying, by proposing a novel paradigm for graph integration and employing a logical, effective node kernel, in our method the features of a node in a graph are extracted from not only in the graph it is placed, but also from other graphs. Therefore, it preserves the characteristics of single graphs in the one hand and it allows genes in a graphs to have different weights when combining for the final similarities on the other hand. We show the outperformance of our method comparing with other methods in two different setting.

7.3 Method

In this section, we first introduce definitions and notations used for the rest of the paper. We then describe our proposed method for disease gene prioritization.

7.3.1 Background, definition and notations

A graph is a structure notated as $G = (V, E)$ where V is the set of nodes and E is the set of links. We define the *distance* $\mathcal{D}(u, v)$ 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, $N_r(u) = \{v \mid \mathcal{D}(u, v) \leq r\}$, 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*. The maximum node degree in the graph G is $deg(G)$.

Disease gene prioritization: Let us consider a set of genes $\mathcal{G} = \{g_1, g_2, \ldots, g_N\}$ that represents either the global set of genes in the genome or a subset of it. Given another set $\mathcal{P} = \{g_1, g_2, \ldots, g_m\}$, $\mathcal{P} \subset \mathcal{G}$ containing genes known to be associated to a genetic disease, gene prioritization is the task that aims to rank genes in the set of candidates $\mathcal{U} = \mathcal{G} \setminus \mathcal{P}$ according to their likelihood of being related to that disease.

The Conjunctive Disjunctive Node Kernel: In [?], a modification of NSPDK kernel [?] named CDNK is proposed to define the similarity for graph nodes. CDNK 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. This kernel extracts subset of NSPDK features whose u as one of the roots and uses as features for u. Interestingly, it distingushes between between two types of edges, called *conjunctive* and *disjunctive* edges. Hence, it is relevant to apply on graphs resulted from the network decomposition presented in 7.3.2. Regarding the edge types, it considers conjunctive edges only when computing distances to induce neighborhood subgraphs. When choosing the pair of neighborhoods to form a single feature, it additionally considers 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. 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.

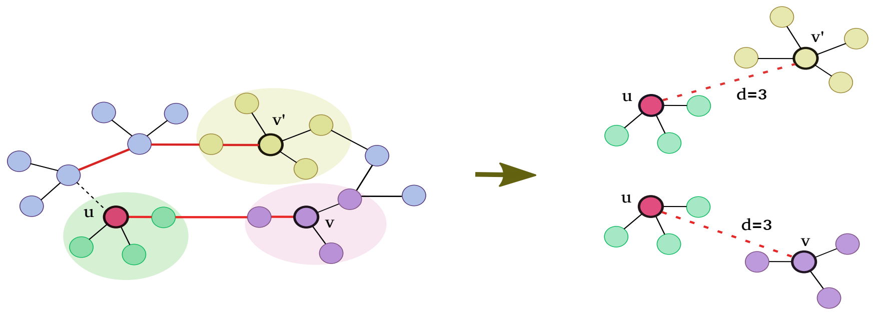

Figure 7.1: CDNK

Formally, It defines 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 $B_v \cong \mathcal{N}^u_r$ are true, (ii) $\exists w$ s.t. $\mathcal{D}(w, v) = d$, and (iii) (u, w) is a disjunctive edge. Besides, it defines $\kappa_{r,d}$ 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 $K(G_u, G_v) = \sum_r \sum_d \kappa_{r,d}(G_u, G_v)$, where once again for efficiency reasons, the values of r and d are upper bounded to a given maximal r^* and d^*.

7.3.2 Disjunctive interconnection graph integration

In this section, we first introduce the flow of our proposed method for disease gene prioritization. We then describe each part in detail.

We consider a set of graphs $\mathcal{G} = \{g_1, g_, \ldots, g_n\}$ where we refer each graph as a layer. Our method consists of following steps:

- Graph decomposition: Each graph is transformed into a collection of sparse sub-networks.
- Graph labeling: Different ways are introduced to label for node and edges of each graph.
- Graph union: we connect the set of graphs obtained from previous steps to form a single graph whose layers are linked with disjunctive links.
- Similarity definition: we apply a particular graph node kernel, CDNK [**?**], on the union graph to define node similarities which will be fed into a kernel machine to construct a model for disease gene inference.

Graph Decomposition

Graph decompositional kernels can only efficiently work on graphs whose nodes with low degree. However, it is common to find nodes with degree in gene-based networks. It prevents decompositional kernels from showing promising performance due to the high number of neighborhood subgraphs. To overcome this problem, in [**?**], a network decomposition procedure is proposed with the aim of transforming a given network into a collection of sparse sub-networks whose *conjunctive* links only are linked by *disjunctive* links. The decompostion works by first applying the k-core decomposition

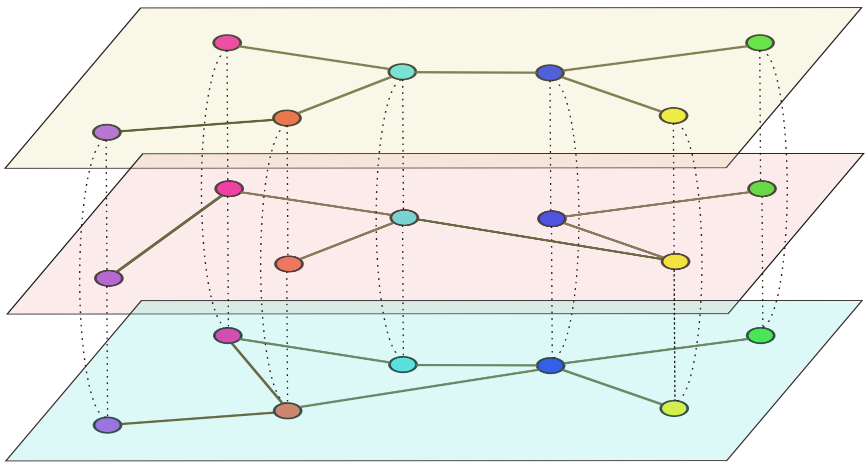
Figure 7.2: Graph Disjunctive Interconnection Illustration

iteratively and then the clique decomposition. Following we describe in detail each decomposition type in detail.

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 vs 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 [1]greater than D.

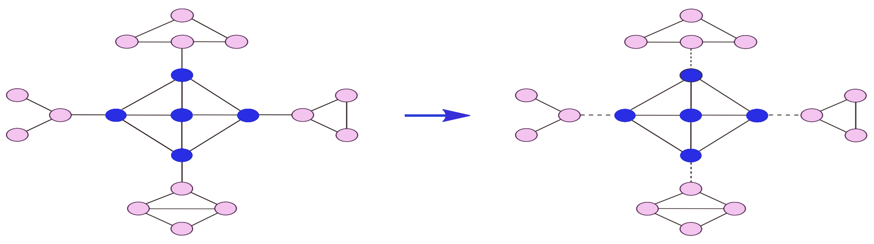
Figure 7.3: K-core decomposition

Clique decomposition: 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 a 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

[1]The degree is defined by only considering incident conjunctive edges.

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

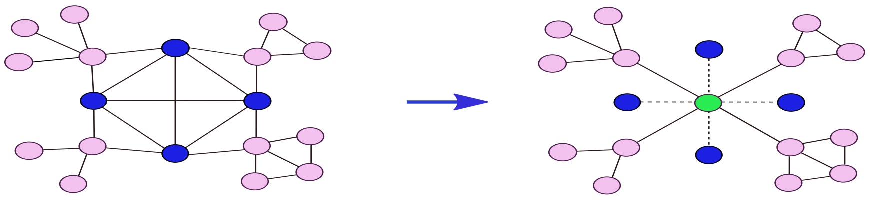
Figure 7.4: Clique decomposition

iterative k-core decomposition and then the clique decomposition.

Graph labeling

Node labeling: we employ two methods to label for nodes in a graph. Both methods are based on the idea that genes with similar number of neighborhoods tend to have similar properties. We consider both conjunctive and disjunctive edges to compute the node degree.

- The first node labeling function, ℓ_1, assigns the degree for nodes u having degree less than or equal a user defined threshold T ($T = 5$ in our experimental evaluation). However degree values larger than T are subsequently discretized into k levels. Formally, the labeling function is defined as:

$$\ell_1(u) = \begin{cases} deg(u), & \text{if } deg(u) \leq T \\ T + i, & \text{if } deg(u) > T \end{cases},$$

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

- The second node labeling function, ℓ_2 implements the idea of equal frequency discretization. Given number of bins, k, we assign nodes into bins such that two criteria. First, nodes with similar degree are assigned into the same bin. Second, every bin needs to have similar number of nodes. We then use the bin ID as label for its nodes.

Edge labeling: We assign labels for every edges in a layer as its layer ID.

Graph union

Consider a layer tuple g_i, g_j ($g_i, g_j \in \mathcal{G}$) and two nodes $u \in g_i$, $v \in g_j$ such that u and v are both represent for a same gene, we connect u and v by a disjuctive link. As the consequence, we achieve a graph g whose nodes represeting same genes in all graph layers are linked by disjuctive links. Figure 7.2 is a visualization for our graph integration idea.

Similarity definition

We employ CDNK to define similarity for the similarity for any couple of genes. The similarity between two genes g_i and g_j is the summation of similarity of between their corresponding nodes on all layers.

7.4 Experiments

In order to evaluate the performance of our proposed method, we conduct two separate experiments. In the first experiment, *cross-validation evaluation*, we desire to compare our method with state-of-the-are methods for disease gene prioritization in the first experiment, while we intend to compare the performance our proposed method with web-tools for disease gene identification in the second experiment, *unbiased evaluation*. Following, we first describe data sources employed for our experiments, we then present the procedures to carry out them.

Data sources

- **Human Protein Reference Database** (**HPRD**): a database of curated proteomic information pertaining to human proteins. It is derived from [?] with 9,465 vertices and 37,039 edges. We employ the HPRD version used in [?] that contains 7311 nodes and 30503 edges.

- **BioGPS** [?, ?] a gene co-expression graph (7311 nodes and 911294 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 (PCC) between genes is larger than 0.5.

- **Pathways**: Pathway datasets are obtained from the database of KEGG [?], Reactome [?], PharmGKB [?] and PID [?], 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.

- **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 [**?**], from which we extracted functional links among 17078 human genes.

- **Omim**: OMIM is a public database of disease-gene association. Genes implicated in the same disease are more likely to be involved in other similar diseases as well. Therefore, Omim network is formed by connecting genes which are involved in common disease(s).

Cross-validation evaluation

We follow the experimental setting used in [**?**]. In this experiment, three datasets are employed: BioGPS, HPRD and Pathways. To perform the experiment, 12 disease-gene associations are selected from OMIM data source with at least 30 confirmed genes. Concerning each disease, we construct a a positive set (P) which contains its all confirmed disease genes. For the negative set (N), we randomly sample from the set of genes which are related to at least one disease, but not with the disease that defines P, such that $|N| = \frac{1}{2}|P|$.

For evaluation, a leave-one-out cross validation is used. Iteratively, each turn a gene in the training set ($P \cup N$) is selected to be the test gene and the remaining genes used to train the model. For each test gene g_i, the model returns a score s_i proportional to the likelihood of being associated to the disease. Next a decision score q_i is computed as the top percentage value of s_i among all candidate gene scores. We collect all decision scores for every test genes to compute the area under the curve for the receiver operating characteristic (AUC-ROC).

Unbiased evaluation

When a new gene-based discovery is made, most related data sources are updated. Therefore, the performance of methods evaluated through the previous experiment could be over-optimistic. To void this inflation, we conduct our experiment following the setting used in [?] with the aim at comparing our method performance with different web tools for disease gene prioritization. Newly discovered disease-gene associations were collected over a timespan of six months, gathering 42 associations. As soon as a new association was discovered, eight pre-selected gene prioritization web tools were queried with a proper training set in order to mimic the discovery through each of them. These 42 predictions were used to assess the ability of the tools to successfully prioritize disease genes. The idea behind this procedure is to anticipate the integration of the associations in the data sources and so avoid biased predictions.

To evaluate the performance of DIGI, we backdated our datasets to a time prior to May 15, 2010 by employing String v8.2 data [?] and OMIM. The candidate sets were constructed by considering all genes with Ensembl [?] gene identifier within the chromosomal regions around the test genes, in order to get on average 100 candidates for each trial. Then, we performed prioritizations for each test gene in two distinct cases - genome-wide and candidate set-based prioritizations. In genome-wide prioritizations all coding genes in the genome were prioritized, while in candidate set-based tests only the genes belonging to the candidate groups were ranked. In both cases, we normalized ranking positions over the total number of genes in order to get the median and the standard deviation of the normalized ranks for test genes. We also computed the true positive rate (TPR) relatively to some representative thresholds (5%, 10% and 30% of the ranking) and the AUC obtained by averaging over the 42 prioritizations.

Model Selection

The hyper-parameters of our method are set using a 3-fold on training set in which one fold is used for training the model and two remaining folds are used for validation. For CDNK, we try for the degree threshold value in $\{10,\ 15,\ 20\}$, clique size threshold in $\{4,\ 5\}$, maximum radius in $\{1,\ 2\}$, maximum distance in $\{2,\ 3,\ 4\}$. The number of bins, k, in node labling function ℓ_1 and ℓ_2 is set in $\{7, 10, 12\}$. Finally, the regularization tradeoff C for the SVM is chosen in $\{10^{-4}, 10^{-3}, 10^{-2},\ 10^{-1}, 1,\ 10,\ 10^{2},$
$10^{3},\ 10^{4}\}$.

7.5 Results and discussion

Table 7.1 and 7.2 show the performance of our proposed method together with different methods and tools in unbiased evaluation and cross validation setting, respectively. Overall, DIGI outperforms all compared methods.

In the Unbiased evaluation setting, it can be seen from the table 7.1 that our method shows impressive performance which considerably higher than other's in both cases: Genome-wide and candidate genes. In particular, It shows big difference with the second best one, Scuba, with around 15% higher in both top 5% and 10%. For the time and finance constraints, we normally take genes at very top in the ranking to have further tests. Therefore, having high accuracy in top 5% and 10% is really meaningful in practice. It is also competive in top 30% with 85.7% and 88.1% for Genome-Wide and Candidate-Genes, respectively. It only shows a bit lower, around 2%, than Endeavour with 90.5% in Candidate-Genes. Similarly for AUC measure, it show the best results in both cases with 87 and 86. Concerning the median rank, a huge gap between our method and the rest of compared methods can be observed from the table. We also report the relative position over 42 genes in the rank of DIGI in Figure **??** and 7.6 in the two cases. We can see that most test genes are ranked in the top of the rank, especially 22 and 17 genes are in in top 5% with 22 and 17 genes for Genome-Wide and Candidate, respectively. There is only 1 gene for each case in a very low position, top 80% and 90%.

In the Cross validation setting, DIGI together with Scuba show the best performance with high gaps, at least 4.6% difference, compared with the rest of methods. Between DIGI and Scuba, DIGI presents a slightly higer than Scuba with 88.1% and 87.6%, respectively. The worse performance methods in this setting are DIR and GeneWanderer.

7.6 Conclusion

We have proposed an efficient method for graph integration for disease gene prioritization. In our method, first graph layers are decomposed and disjunctively interconnected to let information traversing between layers. It then employ a perticular graph node kernel, which is able to efficiently exploit the graph structure resulted from the graph integration paradigm, to compute the capture the node similarities. As a consequence, it shows promissing performance and outforms all methods/tools employed in two experimental setttings.

Table 7.1: 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						
DIGI	100	**4.73**	**52.4**	**59.5**	**85.7**	**87**
Scuba	100	10.55	33.3	47.6	78.6	80
Candid	100	18.10	21.4	33.3	64.3	73
Endeavour	100	15.49	28.6	38.1	71.4	79
Pinta	100	19.03	26.2	31.0	71.4	77
Candidate-Genes						
DIGI	100	**6.10**	**50.0**	**59.5**	88.1	**86**
Scuba	100	12.95	28.6	45.2	73.8	78
Suspects	88.9[a]	12.77[a]	33.3[a]	33.3[a]	63.0[a]	76[a]
ToppGene	97.6	16.80	35.7	42.9	52.4	66
GeneWanderer-RW	88.1	22.10	16.7	26.2	61.9	71
Posmed-KS	47.6	31.44	4.7	7.1	23.8	58
GeneDistiller	97.6	11.11	26.2	47.6	78.6	85
Endeavour	100	11.16	26.2	42.9	**90.5**	82
Pinta	100	18.87	28.6	31.0	71.4	75

Table 7.2: Performance of different methods on Cross validation setting

Method	ROC-AUC (%)
DIR	71.6
F3PC	83.0
MRF	73.1
GeneWanderer	71.1
Scuba	87.6
DIGI	**88.1**

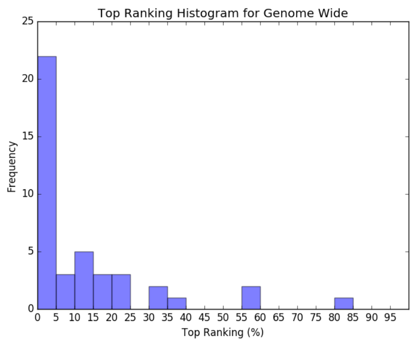

Figure 7.5: Top Rank Histogram for Genome-Wide Setting

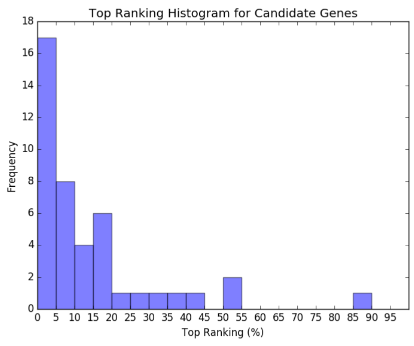

Figure 7.6: Top Rank Histogram for Candidate Setting

Chapter 8

Conclusion and Future Work

Bibliography

[1] José Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. k-core decomposition: A tool for the visualization of large scale networks. *arXiv preprint cs/0504107*, 2005.

[2] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[3] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[4] Andrew Chatr-Aryamontri, Bobby-Joe Breitkreutz, Rose Oughtred, Lorrie Boucher, Sven Heinicke, Daici Chen, Chris Stark, Ashton Breitkreutz, Nadine Kolas, Lara O'donnell, et al. The biogrid interaction database: 2015 update. *Nucleic acids research*, 43(D1):D470–D478, 2014.

[5] Pavel Chebotarev and Elena Shamis. The matrix-forest theorem and measuring relations in small social groups. *arXiv preprint math/0602070*, 2006.

[6] BoLin Chen, Min Li, JianXin Wang, and Fang-Xiang Wu. Disease gene identification by using graph kernels and markov random fields. *Science China. Life Sciences*, 57(11):1054, 2014.

[7] Gene Ontology Consortium et al. The gene ontology (go) database and informatics resource. *Nucleic acids research*, 32(suppl 1):D258–D261, 2004.

[8] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[9] Fabrizio Costa and Kurt De Grave. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 26th International Conference on Machine Learning*, pages 255–262. Omnipress, 2010.

[10] François Fouss, Kevin Francoisse, Luh Yen, Alain Pirotte, and Marco Saerens. An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification. *Neural networks*, 31:53–72, 2012.

[11] Francois Fouss, Luh Yen, Alain Pirotte, and Marco Saerens. An experimental investigation of graph kernels on a collaborative recommendation task. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 863–868. IEEE, 2006.

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

[13] Kwang-Il Goh, Michael E Cusick, David Valle, Barton Childs, Marc Vidal, and Albert-László Barabási. The human disease network. *Proceedings of the National Academy of Sciences*, 104(21):8685–8690, 2007.

[14] David Haussler. Convolution kernels on discrete structures. Technical report, Technical report, Department of Computer Science, University of California at Santa Cruz, 1999.

[15] Lars J Jensen, Michael Kuhn, Manuel Stark, Samuel Chaffron, Chris Creevey, Jean Muller, Tobias Doerks, Philippe Julien, Alexander Roth, Milan Simonovic, et al. String 8a global view on proteins and their functional interactions in 630 organisms. *Nucleic acids research*, 37(suppl_1):D412–D416, 2008.

[16] TS Keshava Prasad, Renu Goel, Kumaran Kandasamy, Shivakumar Keerthikumar, Sameer Kumar, Suresh Mathivanan, Deepthi Telikicherla, Rajesh Raju, Beema Shafreen, Abhilash Venugopal, et al. Human protein reference database2009 update. *Nucleic acids research*, 37(suppl_1):D767–D772, 2008.

[17] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, volume 2, pages 315–322, 2002.

[18] Victor A McKusick. Mendelian inheritance in man and its online version, omim. *The American Journal of Human Genetics*, 80(4):588–604, 2007.

[19] Tom MITCHELL and McGraw HILL. *Machine Learning.* McGraw-Hill Science/Engineering/Math, 1997.

[20] Hiroyuki Ogata, Susumu Goto, Kazushige Sato, Wataru Fujibuchi, Hidemasa Bono, and Minoru Kanehisa. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 27(1):29–34, 1999.

[21] Carl F Schaefer, Kira Anthony, Shiva Krupa, Jeffrey Buchoff, Matthew Day, Timo Hannay, and Kenneth H Buetow. Pid: the pathway interaction database. *Nucleic acids research*, 37(suppl_1):D674–D679, 2008.

[22] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis.* Cambridge university press, 2004.

[23] Nino Shervashidze and Karsten M Borgwardt. Fast subtree kernels on graphs. In *Advances in neural information processing systems*, pages 1660–1668, 2009.

[24] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

[25] Robert E Tarjan. Decomposition by clique separators. *Discrete mathematics*, 55(2):221–232, 1985.

[26] Marc A Van Driel, Jorn Bruggeman, Gert Vriend, Han G Brunner, and Jack AM Leunissen. A text-mining analysis of the human phenome. *European journal of human genetics: EJHG*, 14(5):535, 2006.

[27] Vladimir Vapnik. Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780, 1963.

[28] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[29] Imre Vastrik, Peter D'Eustachio, Esther Schmidt, Geeta Joshi-Tope, Gopal Gopinath, David Croft, Bernard de Bono, Marc Gillespie, Bijay Jassal, Suzanna Lewis, et al. Reactome: a knowledge base of biologic pathways and processes. *Genome biology*, 8(3):R39, 2007.

[30] Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. A primer on kernel methods. *Kernel Methods in Computational Biology*, pages 35–70, 2004.

[31] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242, 2010.

[32] SVN Vishwanathan, Karsten M Borgwardt, and Nicol N Schraudolph. Fast computation of graph kernels. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, pages 1449–1456. MIT Press, 2006.

[33] Michelle Whirl-Carrillo, EM McDonagh, JM Hebert, Li Gong, K Sangkuhl, CF Thorn, RB Altman, and Teri E Klein. Pharmacogenomics knowledge for personalized medicine. *Clinical Pharmacology & Therapeutics*, 92(4):414–417, 2012.

[34] Chunlei Wu, Camilo Orozco, Jason Boyer, Marc Leglise, James Goodale, Serge Batalov, Christopher L Hodge, James Haase, Jeff Janes, Jon W Huss, et al. Biogps: an extensible and customizable portal for querying and organizing gene annotation resources. *Genome biology*, 10(11):R130, 2009.

Appendices

Chapter A

First appendix

A.1 First section

A.2 Second section

Chapter B

Second appendix

B.1 First section

B.2 Second section