University of Padua
Department of Mathematics
Doctorate Degree in Brain, Mind and Computer Science
Curriculum in Computer Science

KERNEL METHODS FOR GRAPH-BASED DATA INTEGRATION

Candidate
Dinh Tran Van

Supervisor
Prof. Alessandro Sperduti
University of Padova, Italy

August 28, 2017

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor Prof. Alessandro Sperduti 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. Alberto Testolin 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, August 28, 2017

Abstract

We have been witnessing the explosive growth of biological data in terms of volume and variety, thanks to the development of technologies related to web services and embeded systems. Such data can naturally be represented as graphs. It is common that entities are encoded in different data sources. On the one hand, this provides a great opportunity for us to have unified views about entities of interest. On the other hand, this poses challenges for scientists 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 need of developing smart learning systems which support experts to form and assess hypotheses in biology and medicine. The problem of data integration in general or graph-based data integration in specific needs to be efficiently solved if we desire to build high performance learning systems. The contributions of this thesis focus on addressing challenges for graph-based data integration problem with the aim of achieving high performance systems: *node similarity measure*, *graph sparsity*, *scalability*, *effectiveness and efficiency*.

The first contribution is the definition of an efficient graph node kernel named Conjunctive disjunctive graph node kernel which aims at defining the similarity between any graph node couple. The kernel first decomposes the input graph into a collection of connected sparse graphs. It then develops a suitable kernel that explicitly models the configuration of each node's context to measure node proximity. The proposed kernel shows state of the art performance for graph node kernels.

The second contribution of this thesis aims to deal with graph sparsity problem by introducing a link prediction method whose objective is to enrich 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 is a proposed method to enrich the performance of diffusion-based graph node kernels when working with sparse graphs caused by the lack of information. By adding link enrichment phase before employing diffusion-based kernels, we empirically show a robust and large effect for combination of a number of link prediction and a number of diffusion-based kernels on several real gene-disease association problems.

The fourth contribution copes with the scalability problem 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 outperforms and enhances the scalability, efficacy comparing with existing methods for disease gene prioritization.

The last contribution is an approach for graph integration targeting to solve disease gene prioritization problem. In this method, 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

1.1 Graph-based Data Integration

1.2 Challenges

1.2.1 Node Similarity Measure

1.2.2 Sparsity

1.2.3 Scalability

1.2.4 Effectiveness and Efficiency

1.3 State of the Art Graph-based Data Integration

1.4 Contributions

1.4.1 Conjuctive Disjuctive Graph Node Kernel

1.4.2 Joint Neighborhood Subgraphs for Link Prediction

1.4.3 Link Enrichment for Diffusion-based Graph Node Kernels

1.4.4 Scalable Kenrel-based Gene Prioritization

1.4.5 Disjunctive Interconnection Graphs for Disease Gene Prioritization

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 application 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 [2], 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 $\mathcal{X}$ 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, $\mathcal{D}$, resulted from any observation, measurement or recording apparatus such that $\mathcal{D} \cup \mathcal{X}$. The set $\mathcal{D}$ is normally referred as training dataset. Machine Leanring techniques desire to exploit $\mathcal{D}$ 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, $\mathcal{U} = \mathcal{X} - \mathcal{D}$.

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 la-

bels, 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 consier a training set $\mathcal{D}$ generated by an unknown probability distribution $\mathcal{P}$, $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ where $x_i \in \mathcal{X}$ are examples and $y_i \in \mathcal{Y}$ are labels. The relations between x_i and y_i are defined by a true function (target function) $f : \mathcal{X} \longmapsto \mathcal{Y}$. 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 $\mathcal{D}$ to be as close to f as possible. Depending on the domain of $\mathcal{Y}$, we can further group supervised learning methods into following sub-groups:

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

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, $\mathcal{H}$. 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.

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

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

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

Unfortunately, it is impossible to directly the optimization 2.1 since the probability distribution $\mathcal{P}$ in true loss function 2.2 is an unknown function and we only have access to a finite training set $\mathcal{D}$. 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:

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

However, in order to use $R_{emp}(h)$, we need to guarantee that the value of $R_{emp}(h)$ converges to the value of $R(h)$?. Using the law of large numbers, authors prove in [3] that the convergence happens when the number of examples is high enough. For more details we refer the reader an amazing book [3].

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, $x \in X \longrightarrow \phi(x) \in \mathcal{F}$. 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 [1] 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: $k : \mathcal{X} \times \mathcal{X} \longmapsto R$. 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.

2.3 Kernel functions

As stated in 2.2, in kernel methods, the definition of kernel functions is independent from the definition general learning algorithms. Therefore, different kernel functions have been proposed for different types of data. In this section, we formally define what is a kernel function.

2.3.1 Perceptron Method

2.3.2 Support Vector Machine

2.4 Graph Node Kernels

Let us consider an undirected, weighted graph $G = (V, E)$ representing genes and relationships among them. The adjacency matrix $\mathbf{A}$ 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 an edge of weight $w_{ij} > 0$ connecting v_i and v_j, and is 0 otherwise. The Laplacian matrix $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal matrix with non-null entries equal to the summation over the corresponding row of the adjacency matrix, i.e. $D_{ii} = \sum_j A_{ij}$. The following graph node kernels are described under this notation convention.

Laplacian exponential diffusion kernel

One of the most well-known kernels for graphs is the Laplacian exponential diffusion kernel $\mathbf{K}_{LED}$, 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 $\mathbf{K}_{LED}$ in [**?**]: 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, $\mathbf{K}_{LED}$ can capture the long range relationship between vertices of a graph to define the global similarities. Below is the formula to compute $\mathbf{K}_{LED}$ values:

$$\mathbf{K}_{LED} = e^{-\beta \mathbf{L}} = \mathbf{I} - \beta \mathbf{L} + \frac{\beta \mathbf{L}^2}{2!} - ... \tag{2.4}$$

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. $\mathbf{K}_{LED}$ is positive semi-definite as proved in [**?**].

Exponential diffusion kernel

In $\mathbf{K}_{LED}$, similarity values between high degree vertices is generally higher compared to that 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 $\mathbf{K}_{LED}$ was introduced by Chen et al in [**?**],

called Markov exponential diffusion kernel $\mathbf{K}_{MED}$ and given by the following formula:

$$\mathbf{K}_{MED} = e^{-\beta \mathbf{M}} \tag{2.5}$$

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 $\mathbf{K}_{LED}$.

Markov diffusion kernel

The original Markov diffusion kernel $\mathbf{K}_{MD}$ was introduced by Fouss et al. [**?**] 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:

$$\mathbf{K}_{MD} = \mathbf{Z}(t)\mathbf{Z}^{\top}(t) \tag{2.6}$$

Regularized Laplacian kernel

Another popular graph node kernel function used in graph mining is the regularized Laplacian kernel $\mathbf{K}_{RL}$. This kernel function was introduced by Chebotarev and Shamis in [**?**] and represents a normalized version of the random walk with restart model. It is defined as follows:

$$\mathbf{K}_{RL} = \sum_{n=0}^{\infty} \beta^n (-\mathbf{L})^n = (\mathbf{I} + \beta\mathbf{L})^{-1} \tag{2.7}$$

where the parameter β is again the diffusion parameter. $\mathbf{K}_{RL}$ counts the paths connecting two nodes on the graph induced by taking $-\mathbf{L}$ 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. $\mathbf{K}_{RL}$ remains a relatedness measure even when diffusion factor is large, by virtue of the negative weights assigned to self-loops.

2.5 Disease Gene Prioritization

Let us formally define the problem of disease gene prioritization 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 $U_i = \mathcal{G} \setminus P_i$ contains genes that are not a priori related to the disease, but

we assume that inside U_i some positive genes are hidden. Therefore, our aim is to build an efficient disease gene prioritization/classification model that allows to prioritize/classify the genes in U_i based on their likelihood to be related to P_i. In particular, G1 will take a set of data sources $\mathcal{S} = \{S_1, S_2, ..., S_m\}$ and a set of disease genes P_i as the input. The output is a list of scores, each score representing the proximity of a gene in U_i to the genes in P_i.

Chapter 3

Conjunctive Disjunctive Graph Node Kernel

Chapter 4

Joint Neighborhood Subgraphs for Link Prediction

Chapter 5

Link Enrichment for Diffusion-based Graph Node Kernel

Chapter 6

Scuba - Scalable Kernel-based Gene Prioritization

6.1 Motivation

6.2 Method

6.3 Experiments

6.4 Analysis

6.5 Conclusion

Chapter 7

Disjuctive Interconnection Graphs for Disease Gene Prioritization

Chapter 8

Conclusion and Future Work

Bibliography

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

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

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