ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

A Study of Graph Kernels through Two Decades

[1] Jyoti
MCA, M. D. University, Rohtak

Abstract: **- In the real world all events are connected. There is a hidden network of dependencies that governs behavior of natural processes. Without much argument it can be said that, of all the known data- structures, graphs are naturally suitable to model such information. But to learn to use graph data structure is a tedious job as most operations on graphs are computationally expensive, so exploring fast machine learning techniques for graph data has been an active area of research and a family of algorithms called kernel based approaches has been famous among researchers of the machine learning domain. With the help of support vector machines, kernel based methods work very well for learning with Gaussian processes. In this survey we will explore various kernels that operate on graph representations. Starting from the basics of kernel based learning we will travel through the history of graph kernels from its first appearance to discussion of current state of the art techniques in practice.**

Keywords**: Graph kernels, Support vector machines, Graph similarity, Isomorphism.**

I. INTRODUCTION

Information has always been in the primary focus of researchers in the field of computer science. In our world, most of the available information is represented as networks of meaningfully connected data elements. These connections can signify some sort of interde- pendence or portray some contextual significance. This relational aspect of information is one of the main challenges for researchers. In this survey will be explored the utility of various graph kernels in this domain of relational information, but before we move on to the details of graph kernels, let us first understand the importance of ‘‘graphs’’ and ‘‘kernels’’ in the field of artificial intelligence. One of the primary tasks is sensible representation of such relational data, so that they could be used to perform machine learning tasks such as classifications, sequence predictions, density estimations and so on. Information is mainly stored using data structures for comput- ers to process them. While there are many data structures avail- able, the most generic format is a graph. All other data structures are simply some sort of specializations of a graph. As we know, graphs are characterized by their network of nodes connected by edges. Similarly, natural information in general can be broken down to smaller elements that can have some sort of semantic connection hence, this property of graph makes it most suitable for representing relational information. So, the first step of graph based learning is to actually represent the information in the form of a graph. Once that is done the second step is the learning part.

The most straightforward technique for learning is to extract meaningful features from a sample that uniquely predicts its na- ture. However, that is not always feasible given the dynamic na- ture of real world problems. Problems can be so complicated that manually extracting features can be really hectic and sometimes humanely impossible. Data in its raw form is not suitable for computational operations. A consistent input space is needed to represent the data in its actual form. The key idea behind finding features is to move the sample from the input space to another dimension where similar samples will be mapped in close proxim- ity while distance between dissimilar samples will be significantly higher. Another branch of machine learning, namely kernel based learning, views the problem from a different perspective. If we can find some metric to map this similarity between samples we can directly map them onto the feature dimension without actually having to learn the features themselves. Another way to explain this is to approximate the nature of the probability distribution of the real world process, also know as the Gaussian process, so that the similar samples stay in close proximity and vice versa. This new dimension is also called an Hilbert space. The entire goal of kernel based learning is to map the available sample space into a suitable Hilbert space. Once we know the Gaussian distribution, also termed as the posterior, it will be much easier to calculate the similarity among samples. Machine learning dived into a new paradigm through the introduction of a special function referred to as a kernel function which can directly map the input space to such feature dimensions. Throughout the next chapters, we will look into details regarding definitions, mathematical concepts and old and modern research works surrounding the application of kernels to the field of graph theory. As we finish the introductory section we will find our motiva- tion to study more about this domain in next section. Section 3 introduces us to the preliminary concepts of some Gaussian Pro- cesses, Kernel based Machine Learning, and Graph Theory. This is absolutely necessary for understanding the concepts of various graph kernels. As we move on to the fourth section, we will discuss the core

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

concepts of graph kernels, starting from the earliest point in the history of research where the first idea of structural kernels was conceived and slowly moving through time to finally analyze a couple of state of the art technologies. Utmost effort has been made to keep all explanation as simple as possible while maintaining enough mathematical formulation to ensure logical clarity.

II. MOTIVATION

Graphs provide one the most generic data structures for repre- senting information. Philosophically speaking a graph represents a network of relationships among objects. All real world phenomena can be interpreted as a system with various components that work in tandem. These relations and interdependence connect these components to form a complex network. Another interpretation may be all real world objects or events can either be described as a network or can be considered to be a part of a larger network. Philosophical arguments have been made in favor of graphs as the most ideal data structure to represent the world in the language of mathematics [1].

In computational terms it has already been mentioned that graph are the most generic form of data structure as all common datatypes can simply be referred to as an instance of a graph. For example, a scalar or a constant can be treated as single node graph, and array or matrix can be seen as a graph where each nodes represent an index in the array and their adjacency is represented by an edge. Stacks and queues have similar structure but with limitation of insertion and deletion property of the nodes. A time series can be modeled by representing time stamps as nodes and connecting each stamp with an edge to the next one.

So, with all this said, the real question is why graphs are not being used as the most common data structure for decades? The simple answer is that handling graphs is complicated. On one hand graphs provide a lot of flexibility to represent complex data in an efficient way but, the same flexibility stands in the way when computational operations are performed. Normal vectors can be easily represented in a co-ordinate space, hence allowing simple metric like euclidean distance to serve as an excellent choice for vector comparison. However, it is much more difficult to represent a graph in an n-dimensional space hence the difficulty of compar- ing them. The straightforward or brute force method would be to identify the common parts in both graphs. For this purpose we must find all sub-graphs of the graph. A graph with n nodes will always have 2n possible sub-graphs. Hence the problem shifts to an exponential search space. As aptly stated by Horst Bunke [2]:

‘‘computing the distances of a pair of objects[...] is linear in the number of data items in the case of feature vectors, quadratic in case of strings, and exponential for graphs’’

Hence, to overcome this curse of exponential time complexity, researchers have avoided graph based machine learning for long time before the introduction of stronger computational resources in the last couple of decades. Gradually, analysis revealed that these problems need crucial attention for the sake of progress of research in this field [3].

III. PRELIMINARY CONCEPTS

As we progress in our journey of rediscovering the domain of graph kernels we must equip ourselves with the proper tools and techniques to ensure proper and clear understanding of the Fig. 1 shows how μ and σ affects the Gaussian distribution. For conceptual clarity we may visualize a single dimensional Gaussian process as a set of observations that depend on a single variable. The expectation of these observations are said to exhibit the nature of a Gaussian distribution. We may notice that we have only talked about a Gaussian distribution that depends only on one random variable (X). Such a distribution is called uni- variate Gaussian distribution. When we take into consideration more than one random variable we get a multivariate Gaussian distribution which depends on a random vector (X1, X2, . . . , Xk). Any linear combination of these random variables would give us an multi-variate Gaussian distribution. The multivariate Gaussian distribution can be represented in terms of a linear combination of uni-variate distributions as shown in Fig. 2. In this case the probability density function fx(x1, . . . , xk) for a k-variate Gaussian is given by:

core concepts. This refresher section is divided into three main subsections, namely, Gaussian processes and Covariance function, Kernels and Support Vector Machines, and Graph Theory.

1.1. Gaussian process and co-variance functions

The concept of kernel deals with expressing samples in an alter- nate feature space where they exhibit some properties which allow similar samples to remain closer. Most processes in the real-world deal with random variables that govern the outcome of the process. In real-world random variables that trigger events are not always completely random but in fact exhibits a probabilistic nature or can be expressed through a probabilistic distribution. Hence the output of the process also exhibits probabilistic properties. We will see one of the most common kind of process called Gaussian process. This Gaussian process can be expressed in terms of a co- variance function which computes the relation between different components of the input

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

dimensions. Furthermore, we will see the positive semi definite property for covariance functions which is an essential property to define a kernel. Finally we will associate these kernels to a Hilbert space which is the alternate feature space that we need.

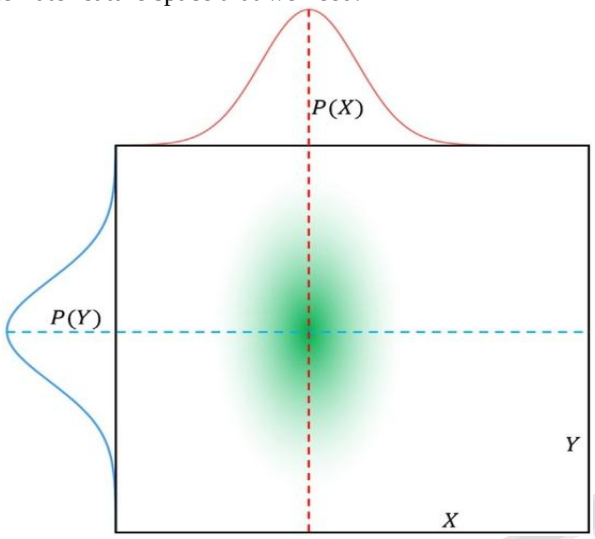

Fig. 2. A multivariate distribution (bi-variate in this case).

3.1. Gaussian process and co-variance functions

The concept of kernel deals with expressing samples in an alter- nate feature space where they exhibit some properties which allow similar samples to remain closer. Most processes in the real-world deal with random variables that govern the outcome of the process. In real-world random variables that trigger events are not always completely random but in fact exhibits a probabilistic nature or can be expressed through a probabilistic distribution. Hence the output of the process also exhibits probabilistic properties. We will see one of the most common kind of process called Gaussian process. This Gaussian process can be expressed in terms of a co- variance function which computes the relation between different components of the input dimensions. Furthermore, we will see the positive semi definite property for covariance functions which is an essential property to define a kernel. Finally we will associate these kernels to a Hilbert space which is the alternate feature space that we need.

3.1.2. Covariance functions

In the previous section, we talked about how a Gaussian dis- tribution can be expressed in terms of a mean function and a co- variance matrix. As expressed in (2), the covariance matrix can be found using a covariance function Cov(x, x′). A covariance function is crucial in the field of Gaussian process prediction, as it is a latent representation of the function we wish to predict.

3.1.3. Gaussian process regression

Gaussian process regression can be considered an interpolation method such that interpolated samples from Gaussian processes are constrained by prior covariances as per the training data points. The Gaussian process itself can be explained through different perspectives; the simplest representation may be as an infinite di- mensional Gaussian random variable with a specified co-variance structure. While this process demonstrates its properties it is not useful for practical models. A weight-space view represents Gaus- sian processes as weighted averages of training target values. An- other interpretation views Gaussian processes as distribution over functions: finite dimensional Gaussian processes are distributions
over finite dimensional vectors, while infinite dimensional Gaus- sian Processes are distributions over infinite dimensional vectors or functions.

IV. GRAPH KERNELS

Before we move on to study graph kernels we need to go through a brief summarization of different classes of kernels on structured data. Kernel methods and support vector machines especially have succeeded in various learning problems on data represented as a single table. But most of the 'real-world' data is structured, i.e., it has no default representation in a equation format. Generally, to apply such kernel methods to 'real-world' data, we need extensive pre-processing to map the data into a real vector space and therefore into a single table. Most of the datasets used can be found either from links provided by the respective authors in the papers or this website which has a good collection of Graph Kernel datasets.1 Fig. 7 shows the various types of kernels that will be taken into account in the consequent chapters. Graph Kernels can be broadly categorized into two major branches based on the principal driving force of their definition, namely, model based kernels and syntax based kernels.

Model driven kernels rely on some kind of knowledge about the sample space, i.e., about the relationships among data. There are principally two subcategories in this branch namely, generative
models and transformative models. While parameters of genera- tive models are treated as features for comparison, transformative models study the ability of the graphs to transform them in certain
way as per the problem domain. These transformed graphs can be seen as a model of the instance space; while each edge only contains local information about neighboring vertices during the process of transformation, the set of all edges contain information about the global structure of the sample space

4.1. Model driven kernels

Syntax based models focus on the semantics of the data. It comprises of the largest family of kernels, namely the convolu- tional kernels. Most of the algorithms deal with

manipulation of syntactic elements like walk, paths, cycles, subgraphs, subtrees and so on. We will see that lots of these kernels preserve both local and global features for comparison and hence are applicable in various domains

4.2. Syntax driven kernels

Syntax based models focus on the semantics of the data. It comprises of the largest family of kernels, namely the convolu- tional kernels. Most of the algorithms deal with manipulation of syntactic elements like walk, paths, cycles, subgraphs, subtrees and so on. We will see that lots of these kernels preserve both local and global features for comparison and hence are applicable in various domains.

4.4. Choosing the right kernel

Throughout the last two decades various kernels surfaced in the field of graph based algorithms as it can be seen in Fig. 13. However, the application of a graph kernel to a specific domain requires the understanding of the advantages and disadvantages of the various available techniques. While earlier versions of random walk kernels prove to be quite efficient for simple graphs, as the size and complexity increases we need to shift to other modern techniques. Normally complexity of random walk kernels are in the order of O(n6), however faster computations up to O(n3) were obtained by extending concepts of linear algebra to Reproducing Kernel Hilbert Spaces [47]; even further speedups up to O(n2) were obtained by Kang [43] considering rank approximation of the adjacency matrix.

4.5. Future of graph kernels

Graph kernels show considerable amount of promise in the future. As discussed before, many real world scenarios or events can be conceptualized through graphs. Most scientific fields show increasing use of structured data. From nuclear studies in CERN,3 to decoding our own genome,4 we have been generating tremen- dous amount of structured information. Through internet we have access to an immensely large network connecting almost 3.78 billion5 people across the world. These huge sources of structured information must be processed to extract what is essential for the progress of science. Structured data provides a different level of challenge to traditional artificial approaches because of the complexity of information. But with graph kernels it is possible handle such challenges. One of the most significant progress in machine learning is the onset of deep learning [64].

4.6. Applications

Having studied the theoretical aspects of graph kernel, it is essential to know about the possible domains where they can be applied. In the subsections below, some of the most prominent application areas will concisely discussed.

4.6.2. Bioinformatics

A major reason for the growing interest in graph-structured data is the advent of large volumes of structured data in molecular biology. This structured data comprises graph models of molecular structures, from RNA to proteins [72], and of net- works which include protein–protein interaction networks [73], metabolic networks [74], regulatory networks [75], and phyloge- netic networks [76]. Bioinformatics seeks to establish the function of these networks and structures. Currently, the most success- ful approach towards function prediction of structures is based on similarity search among structures with known function. For instance, if we want to predict the function of a new protein structure, we compare its structure to a database of functionally annotated protein structures. The protein is then predicted to exert the function of the (group of) protein(s) which it is most similar to. This concept is supported by models of evolution: proteins that have similar topological structures are more likely to share a common ancestor, and are more likely to carry out the same biochemical function [77].

4.6.3. Social network analysis

Another important source of graph structured data is social network analysis [78]. In social networks, nodes represent individ- uals and edges represent interaction between them. The analysis of these networks is both of scientific and commercial interest. On the one hand, psychologists want to study the complex social dynamics between humans, and biologists want to uncover the social rules in a group of animals. On the other hand, industries want to analyze these networks for marketing purposes. Detecting influential individuals in a group of people, often referred to as 'key-players' or 'trend-setters', is relevant for marketing, as com-panies could then focus their advertising efforts on persons known to influence the behavior of a larger group of people. In addition, telecommunication and Internet surfing logs provide a vast source of social networks, which can be used for mining tasks ranging from telecommunication network optimization to automated rec- ommender systems.

4.6.4. Internet, HTML, XML

A fourth application area for graph models is the Internet which is a network and hence a graph itself. HTML documents are nodes in this network, and hyperlinks connect these nodes. In fact, Google exploits this link structure of the Internet in its famous PageRank algorithm [79] for ranking websites. Furthermore, semi-structured data in form of XML documents is becoming very popular in the database community and in industry. The natural mathematical structure to describe semi-structured data is a graph. As the W3 Consortium puts it ''The main structure of an XML document is tree-like, and most of the lexical structure is devoted to defining that tree, but there is also a way to make connections between arbitrary nodes in a tree''.7 Consequently, XML documents should be regarded as graphs.

Various tasks of data manipulation and data analysis can be performed on this graph representation, ranging from basic opera- tions such as querying [80] to advanced problems such as duplicate detection [81].

4.6.5. Natural language processing

Language in general possesses a very generic entity relationship structure, hence graph based algorithms have always been used in this field [82–84]. Tree kernels are particularly common [85] in NLP because of parse tree representations of the language which are very precise to deal with common concepts.

4.6.6. Image processing

Images can also be treated as graphs if we consider similar components in an image as node and their semantic relations as edges. A lot of work has been seen in this area: earlier, image classi- fication was performed using methods like graph edit distance [16] or marginalized kernels [86] later much sophisticated approaches that involved point clouds [87] or segmentation graphs [88] has also been seen.

V. CONCLUSION

Graph Kernels is a beautiful concept that has immense future prospect. It can deal with problems of numerous domains and un- like many other learning techniques, it excels in learning relational models. Information around us, no matter from which source, can somehow be expressed in terms of graphs through their inherent semantic dependencies, thus making graph kernels a powerful tool for researchers.

We started from the absolute basics of Linear algebra and graph theory, and slowly climbed up the ladder to learn how kernels operate in the field of machine learning. Starting from the historical aspects of the domain, we saw that graph kernels can be computed through various walk-based, path-based, subgraph-based or sub- tree based techniques. Finally, we saw how modern researchers have been successful in creating very fast techniques that can be used for larger graphs. With all this research, graph kernel is at its pinnacle of development. Now is the time to dive into even more challenging problems with these techniques.

REFERENCES

[1] R.R. Dipert, The mathematical structure of the world: The world as graph, J. Philos. 94 (7) (1997) 329–358.

[2] H. Bunke, Graph-based tools for data mining and machine learning, in: International Workshop on Machine Learning and Data Mining in Pattern Recognition, Springer, 2003, pp. 7–19.

[3] D. Conte, P. Foggia, C. Sansone, M. Vento, Thirty years of graph matching in pattern recognition, Int. J. Pattern Recognit. Artif. Intell. 18 (03) (2004) 265– 298.

[4] T. Gärtner, A survey of kernels for structured data, ACM SIGKDD Explor. Newslett. 5 (1) (2003) 49–58.

[5] S.V.N. Vishwanathan, N.N. Schraudolph, R. Kondor, K.M. Borgwardt, Graph kernels, J. Mach. Learn. Res. 11 (Apr) (2010) 1201–1242.

[6] N. Shervashidze, Scalable Graph Kernels (Ph.D. thesis), Universität Tübingen, 2012.

[7] K.M. Borgwardt, Graph Kernels (Ph.D. thesis), lmu, 2007.

[8] N. Aronszajn, Theory of reproducing kernels, Trans. Amer. Math. Soc. 68 (3) (1950) 337–404.

[9] B. Scholkopf, A.J. Smola, Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, 2002, arXiv: arXiv:1011.1669v3, http://dx.doi.org/10.1198/jasa.2003.s269.

[10] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, A.-C. Org, Large margin methods for structured and interdependent output variables, J. Mach. Learn. Res. 6 (2005) 1453–1484. http://dx.doi.org/10.1007/s10994-008-5071-9.

[11] V. Vapnik, a. Lerner, Pattern recognition using generalized portrait method, Autom. Remote Control 24 (1963) 774–780 doi:citeulike-article-id:619639.

[12] H. Bunke, Graph matching : Theoretical foundations, algorithms, and applica- tions, Algorithmica 2000 (2) (2000) 82–88.

[13] H. Bunke, P. Foggia, C. Guidobaldi, C. Sansone, M. Vento, A comparison of algorithms for maximum common subgraph on randomly connected graphs, in: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Springer, 2002, pp. 123–132.

[14] M.R. Garey, D.S. Johnson, A Guide to the Theory of NP-Completeness, 1979, http://dx.doi.org/10.1137/1024022.

[15] B. McKay, Nauty User's Guide (version 2.4), Computer Science Dept., Australian National University, 2007, pp. 1–70.

[16] M. Neuhaus, H. Bunke, Edit distance-based kernel functions for structural pattern classification, Pattern Recognit. 39 (10) (2006) 1852–1863. http://dx. doi.org/10.1016/j.patcog.2006.04.012.

[17] H. Bunke, K. Shearer, A graph distance metric based on the maximal common subgraph, Pattern Recognit. Lett. 19 (3) (1998) 255–259.

[18] M.L. Fernández, G. Valiente, A graph distance metric combining maximum common subgraph and minimum common supergraph, Pattern Recognit. Lett. 22 (6–7) (2001) 753–758. http://dx.doi.org/10.1016/S0167-8655(01)00017-4.

[19] I. Koch, Enumerating all connected maximal common subgraphs in two graphs, Theoret. Comput. Sci. 250 (12) (2001) 1–30. http://dx.doi.org/10.1016/ S0304-3975(00)00286-3.

[20] C. Bron, J. Kerbosch, Algorithm 457: Finding all cliques of an undirected graph, Commun. ACM 16 (9) (1973) 575–577. http://dx.doi.org/10.1145/362342. 362367. arXiv:citation.cfm?doid=362342.362367.