INEXACT GRAPH MATCHING THROUGH GRAPH COVERAGE

Lorenzo Livi, Guido Del Vescovo and Antonello Rizzi
Department of Information Engineering, Electronics and Telecommunications, SAPIENZA University, Via Eudossiana 18, 00184 - Rome, Italy

Keywords: Inexact graph matching, Graph kernels, Tensor product, Graph classification system.

Abstract: In this paper we propose a novel inexact graph matching procedure called graph coverage, to be used in supervised and unsupervised data driven modeling systems. It relies on tensor product between graphs, since the resulting product graph is known to be able to encode the similarity of the two input graphs. The graph coverage is defined on the basis of the concept of graph weight, computed on the weighted adjacency matrix of the tensor product graph. We report the experimental results concerning two distinct performance evaluations. Since for practical applications the computing time of any inexact graph matching procedure should be feasible, the first tests have been conceived to measure the average computing time when increasing the average size of a random sample of fully-labeled graphs. The second one aims to evaluating the accuracy of the proposed dissimilarity measure when used as the core of a classification system based on the k-NN rule. Overall the graph coverage shows encouraging results as a dissimilarity measure.

1 INTRODUCTION

A great number of Pattern Recognition problems coming from real world applications must cope with structured patterns, such as digital images, audio signals and chemical compounds, for instance. As a result, each pattern can be represented as a *labeled graph*, where vertices and edges are equipped with complex labels, able to encode different kind of information. Indeed, developing classification systems able to cope with labeled graphs is a fundamental step. Consequently, the field of *Graph-based Pattern Recognition* is growing fast, and is aimed to the establishment of efficient and effective Pattern Recognition techniques on the domain of graphs. The generalization capability of any classification system (being a particular data driven modeling problem) strictly depends on the way the inductive logic inference is defined and computed, which in turn is fixed by choosing a dissimilarity (or similarity) measure between input patterns. Such a measure is therefore the most important procedure in any inductive modeling system. When dealing with graphs as input patterns, the way we define a dissimilarity measure is not trivial, especially in the case of *fully-labeled* graphs, where, in general, vertices and edges can be even labeled with complex data structures (text excerpts, audio sequences, images, and so on). Therefore, any dissimilarity measure for graphs must be able to cope with both topological and labels related information.

A labeled graph is a tuple $G = (V, E, \mu, \nu)$, where V is the (finite) set of vertices, $E \subseteq V \times V$ is the set of edges, $\mu : V \to \mathcal{L}_V$ is the vertex labeling function, with $\mathcal{L}_V$ the vertex-labels set and $\nu : E \to \mathcal{L}_E$ is the edge labeling function, with $\mathcal{L}_E$ the edge-labels set. The cardinalities of V and E are called the *order* and the *size* of the graph, respectively. The adjacency matrix of the graph is denoted with $\mathbf{A}$, and if vertices, say v_i, v_j, are connected by an edge e_{ij}, we have $A_{ij} = 1$, otherwise 0. A (labeled) graph is said to be *weighted* if $\mathcal{L}_E \subseteq \mathbb{R}$, with $W_{ij} = \nu(e_{ij})$ known as the weighted adjacency matrix. In the current scientific literature it is possible to distinguish three mainstream approaches for the inexact graph matching problem: Graph Edit Distance (Neuhaus et al., 2006; Riesen and Bunke, 2009), Graph Embedding (Del Vescovo and Rizzi, 2007; Riesen and Bunke, 2010) and Graph Kernels (Borgwardt et al., 2005; Gärtner, 2008). In the following we propose a dissimilarity measure between graphs named *Graph Coverage* by defining a particular way to perform an inexact graph matching procedure belonging to the graph kernels based family.

The paper is organized as follows. We explain the proposed inexact graph matching procedure in Section 2. In Section 3 we describe and discuss the tests performed to evaluate performances considering both

Livi L., Del Vescovo G. and Rizzi A. (2012).
INEXACT GRAPH MATCHING THROUGH GRAPH COVERAGE.
In *Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, pages 269-272
DOI: 10.5220/0003732802690272
Copyright © SciTePress

qualitative and quantitative indexes. Conclusions are drawn in Section 4, where we give also possible future directions for enhancing the computation efficiency.

2 THE PROPOSED INEXACT GRAPH MATCHING MEASURE

In this section we will explain the proposed inexact graph matching method, that fits well in the domain of graph kernels. For this purpose, we will give a briefly introduce the context of graph kernels and tensor product of graphs.

A symmetric continuous function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is called a *positive definite kernel* if $\sum_{i,j}^{n} c_i c_j k(x_i, x_j) \geq 0$ holds, with $c_i, c_j \in \mathbb{R}$, for each finite $n \geq 2$. Graph kernels are functions defined over the domain of graphs, $\mathcal{X} = \mathcal{G}$. They rely on the representation of the graph in an *implicitly defined high dimensional* feature space. The *reproducing property* of *valid* kernel functions is of fundamental importance in Machine Learning and Pattern Recognition domains, and the relation $k(x,z) = \langle \Phi(x), \Phi(z) \rangle,\ \forall x, z \in \mathcal{X}$, is referred as the *kernel trick* (Schölkopf and Smola, 2002). So, the effort should be focused only to the definition of a valid kernel function for the specific input domain $\mathcal{X}$ (*e.g.* graphs, $\mathcal{X} = \mathcal{G}$).

The tensor product (also called direct product) (Imrich and Klavžar, 2000) of two graphs G_1 and G_2, denoted with $G_\times = G_1 \otimes G_2$, is defined as

$$\begin{aligned} V_\times &= \{(v_i, u_r) : v_i \in V_1, u_r \in V_2\} \qquad (1) \\ E_\times &= \{((v_i, u_r), (v_j, u_s)) : (v_i, v_j) \in E_1 \wedge (u_r, u_s) \in E_2\} \end{aligned}$$

In the following, we will call $G_\times$ the tensor product graph. In the context of Pattern Recognition, the key issue is to define a meaningful dissimilarity measure between patterns. Consequently, the standard tensor product formulation have been adapted, taking into account also the similarities of the labels of both vertices and edges (Borgwardt et al., 2005). The resulting adjacency matrix of the tensor product graph $G_\times$ can be computed using a valid kernel function, $k(\cdot,\cdot)$, defined as a product of three different valid kernel functions for vertex and edge labels

$$\begin{aligned} &k((u_i, u_j), (v_r, v_l)) = \qquad (2) \\ &= k_V(u_i, u_j) \cdot k_E((u_i, v_r), (u_j, v_l)) \cdot k_V(v_i, v_j) \end{aligned}$$

For example, $k_V(\cdot,\cdot)$ and $k_E(\cdot,\cdot)$ can be evaluated as Gaussian Radial Basis kernels of the type

$$k_V(u_i, u_j) = \exp\left(-\frac{d(\mu(u_i), \mu(u_j))^2}{2\sigma_V^2}\right) \qquad (3)$$

where $d(\cdot,\cdot)$ is a suitable dissimilarity for the specific labels domain.

2.1 Graph Coverage

Given two input arbitrarily labeled graphs, say G_1 and G_2, we compute the tensor product graph $G_\times$ as shown in Equation 2, using the product of three Gaussian kernel functions, as the basic similarity scheme. If $G_\times$ is the resulting graph of this operation, we analyze its characteristics with respect to the *optimal* tensor product graph achievable from the two input graphs. Given two input graphs G_1 and G_2, three tensor product graphs are possible, namely $G_\times^{(1,2)} = G_1 \otimes G_2$, $G_\times^{(1,1)} = G_1 \otimes G_1$ and $G_\times^{(2,2)} = G_2 \otimes G_2$. The graphs $G_\times^{(1,1)}$ and $G_\times^{(2,2)}$ are the *best matching* tensor product graphs, that is, they represent the cases where G_2 is exactly equal to G_1 and viceversa. Now we introduce the notion of *weight* for a graph.

Definition 1. (Weight of the Graph). Given a weighted graph G of order *n*, we define its *weight*, denoted with $W(G)$, as

$$W(G) = \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} \geq 0$$

Our tensor product graph $G_\times$ is actually a weighted graph, with $\nu(e_{ij}) \in [0,1],\ \forall e_{ij} \in E(G_\times)$. The maximum *theoretical* achievable weight for such a tensor product graph $G_\times$ is clearly upper bounded by its size, that is $W(G_\times) \leq |E_\times|$. Given two input labeled graphs G_1 and G_2, the maximum theoretical weight for $G_\times = G_1 \otimes G_2$ is certainly achievable if and only if $|\mathcal{L}_V| = |\mathcal{L}_E| = 1$ holds. The aim of the graph coverage measure is in understanding how much the two given input graphs G_1 and G_2 *overlap*, considering both the topologies and labels. This is done confronting the weight of the tensor product graph $G_\times^{(1,2)}$ with respect to the *real* maximum achievable weight considering G_1 and G_2. Formally,

Definition 2. (Graph Coverage). Given two arbitrarily labeled graphs G_1 and G_2, let $G_\times^{(1,2)}$, $G_\times^{(1,1)}$ and $G_\times^{(2,2)}$ be their respective possible tensor product graphs. We define the coverage of G_1 and G_2 as

$$\kappa(G_1, G_2) = \frac{W(G_\times^{(1,2)})}{\max\{W(G_\times^{(1,1)}), W(G_\times^{(2,2)})\}} \qquad (4)$$

It is easy to see that the function shown in Equation 4 is symmetric and normalized in $[0,1]$. It is symmetric because the tensor product $\otimes$ and $\max\{\cdot,\cdot\}$ operators are both commutative. This is indeed a valid kernel function, because it basically consists in products and sums between valid kernel functions of the type shown in Equation 2. It can be also converted very easily into a dissimilarity just setting, for example

$$d(G_1, G_2) = 1 - \kappa(G_1, G_2) \quad (5)$$

If two graphs are equal, their coverage is maximal, that is 1. Conversely, if they are completely different, their coverage tends to 0. A dual behavior is valid for the dissimilarity formulation shown in Equation 5. The number of parameters on which it depends is derived from the nature of the basic similarity functions, that is from the particular kernel functions used in Equation 2. The computational complexity of the graph coverage is dominated by the computation of the tensor product between the two input graphs, that is between their adjacency matrices. If n and m are, respectively, the order of G_1 and G_2, then the computational complexity of the tensor product between G_1 and G_2 is of the order $O(n^2m^2)$. Therefore, the computational complexity of the graph coverage is given by $O(d(n^4+m^4+n^2m^2))$, where d is the cost for each basic kernel computation. So, from the computational complexity point of view, the graph coverage formulation is faster than other graph kernels based on the convolution of (random) walks. In Figure 1 is shown the computing time achieved over a random sample of graphs with 10 and 20-dimensional real vectors for vertices and edges labels, respectively. As it is possible to observe, up to an order of 45, with an average size of 256, the average computation time remains under 1 second.

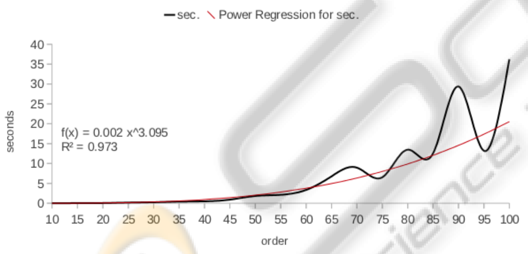

Figure 1: Time Performance Plot.

3 *K*-NN CLASSIFICATION

In this section, we provide the evaluations over the *IAM* dataset (Riesen and Bunke, 2008). The tests are executed on a machine with an Intel(R) Core(TM)2 Quad CPU Q6600 2.40GHz and 4 Gb of RAM over a Linux OS. The graph coverage is implemented in C++, as well as the other software components of the *SPARE* library (Del Vescovo et al.,). The execution time in each test is computed using the *clock()* function defined in *ctime*. To be able to evaluate the *pure* performance of the graph coverage measure, we have tested the accuracy of the classification using the *k*-NN classifier of the *SPARE* library over some *IAM* datasets. The *k*-NN classifier is a very simple tool that relies totally on the dissimilarity evaluation of the input patterns, so the classification performance of the graph coverage can be evaluated directly. In Table 1 are shown the results achieved over some *IAM* datasets, namely *Letter LOW*, *Letter MED*, *Letter HIGH*, *AIDS*, *Fingerprint* and *COIL-DEL*. The datasets *Letter LOW*, *Letter MED* and *Letter HIGH* are composed of a triple of training, validation and test sets, each of 750 patterns. The first dataset is composed of letters with a low level of distortion. The patterns of the second and the third dataset are affected by medium and high level of distortions. Each dataset contains equally-distributed patterns from 15 different classes. The *AIDS* dataset is a not-equally distributed two-class set of graphs with 250, 250 and 1500 samples for the training, validation and test set, respectively. The *Fingerprints* dataset contains graphs from four different classes with 500, 300 and 2000 samples for the training, validation and test set, respectively. The graphs are not equally distributed among the four different classes. Finally, the *COIL-DEL* dataset is composed of 2400, 500 and 1000 graphs for the training, validation and test set, respectively, that are equally distributed among 100 classes. The last column of Table 1, labeled *ms*, contains the average computation time, expressed in milliseconds, for each single graph coverage computation. The results are shown for ten different values of the k parameter of the nearest neighborhood classifier (for $k = 1 \to 10$). The results shown in Table 1 are obtained learning the three different σ parameters of the Gaussian RBF of Equation 3 with a genetic algorithm-based optimization procedure carried out over the validation set. It is possible to see a generic stable behavior for different values of k. In particular we observe very good results on the three *Letters* and the *AIDS* datasets. Conversely, on *COIL-DEL* and *Fingerprint* we observe a low classification accuracy. This result is, in some sense, not expected, considering the average achieved results. However, it is worth to perform a deeper analysis and we will proceed with further investigations aiming to better characterize the effectiveness of the proposed graph matching procedure with respect to classification problems properties.

In Table 2 are shown the best results, taken from (Riesen and Bunke, 2009, Table 2), achieved by different algorithms over some of the *IAM* graphs databases. The algorithm named *Heuristic-A** (Neuhaus et al., 2006) computes the exact GED

Table 1: Classification Accuracy with k-NN.

Dataset	K=1	K=3	K=5	K=7	K=10	ms
Letter L	98.80	98.40	97.73	98.93	98.80	0.103
Letter M	80.53	79.60	79.86	81.86	82.93	0.105
Letter H	74.00	71.86	75.60	75.87	78.40	0.153
AIDS	96.06	94.40	93.53	93.00	89.94	2.338
Fingerprints	39.67	37.38	38.95	39.15	38.24	0.573
COIL-D	37.50	12.50	12.50	12.50	12.50	32.94

using heuristic information. The algorithm named *Beams(10)* is one of its approximated variation proposed in (Neuhaus et al., 2006). Note that 10 is the value of the parameter s chosen for the algorithm execution. The algorithm named *BP* is a fast bipartite graph matching procedure proposed in (Riesen and Bunke, 2009). Finally, *GC* stands for the Graph Coverage algorithm. Note that in Table 2 the *Heuristic-A** algorithm is unable to achieve any result for some dataset, due to its computational limit.

The achieved results, using the k-NN rule over the *IAM* datasets, clearly show the validity of the proposed method, with respect to some of the state of the art methodologies.

Table 2: Classification Results over the Letter LOW (L-L), Letter MED (L-M), Letter HIGH (L-H), AIDS, Fingerprints (F) and COIL (C) Datasets.

Algorithm	Datasets					
	L-L	L-M	L-H	AIDS	F	C
Heuristic-A^*	91.0	77.9	63.0	-	-	93.3
Beam(10)	91.1	78.5	63.9	96.2	84.6	93.3
BP	91.1	77.6	61.6	97.0	78.7	93.3
GC	98.9	83.2	78.4	96.0	39.6	37.5

4 CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we have proposed a novel inexact graph matching procedure. It is simple in its formulation and at the same time effective, relatively fast and flexible. This is, indeed, the real interesting contribution introduced by the proposed method, considering the other available graph kernels. In fact, it is worth to stress that the graph coverage is able to deal also with fully-labeled graphs, where vertices and edges labels can be even complex data structures, once valid kernel functions defined in these domains, to be used as similarity measures, are provided. The proposed procedure shows interesting preliminary results, considering both classification accuracy and computational performance. We are planning to test this algorithm over more shared benchmarking graph-based datasets. Moreover, the proposed inexact graph matching procedure is based on tensor product between graphs. This product is a mathematically solid and properties-rich operation, that is basically founded on multiple product between matrices and scalars. Therefore, the procedure is well suited to be implemented in relatively inexpensive parallel computing devices, such as Graphic Processing Units (GPUs) or Field Programmable Gate Array (FPGA). Taking advantage of these technologies, our effort will be focused on the formulation of a more efficient graph coverage procedure, able to deal with graphs of order of 200 and beyond in a reasonable computing time.

REFERENCES

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S. V. N., Smola, A. J., and Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, 21:47–56.

Del Vescovo, G., Livi, L., Rizzi, A., and Frattale Mascioli, F. M. Spare: Something for pattern recognition. Submitted for publication at: Journal of Machine Learning Research, Microtome Publishing.

Del Vescovo, G. and Rizzi, A. (2007). Automatic classification of graphs by symbolic histograms. In *Proceedings of the 2007 IEEE International Conference on Granular Computing*, GRC '07, pages 410–416, San Jose, CA, USA. IEEE Computer Society.

Gärtner, T. (2008). *Kernels for structured data*. Number v. 72 in Kernels For Structured Data. World Scientific.

Imrich, W. and Klavžar, S. (2000). *Product graphs, structure and recognition*. Wiley-Interscience series in discrete mathematics and optimization. Wiley.

Neuhaus, M., Riesen, K., and Bunke, H. (2006). Fast suboptimal algorithms for the computation of graph edit distance. In *Structural, Syntactic, and Statistical Pattern Recognition. LNCS*, pages 163–172. Springer.

Riesen, K. and Bunke, H. (2008). Iam graph database repository for graph based pattern recognition and machine learning. In *Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, SSPR & SPR '08, pages 287–297, Berlin, Heidelberg. Springer-Verlag.

Riesen, K. and Bunke, H. (2009). Approximate graph edit distance computation by means of bipartite graph matching. *Image Vision Comput.*, 27:950–959.

Riesen, K. and Bunke, H. (2010). *Graph Classification and Clustering Based on Vector Space Embedding*. Series in Machine Perception and Artificial Intelligence. World Scientific Pub Co Inc.

Schölkopf, B. and Smola, A. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning. MIT Press.