Pattern Recognition Letters 56 (2015) 22–29

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Learning graph-matching edit-costs based on the optimality of the oracle's node correspondences[☆]

Xavier Cortés, Francesc Serratosa[*]

Universitat Rovira i Virgili Tarragona, Catalonia, Spain

A R T I C L E I N F O

Article history:
Received 5 June 2014
Available online 31 January 2015

Keywords:
Graph matching
Graph edit distance
Learning edit costs
Hamming distance

A B S T R A C T

The Graph edit distance is the most used distance between attributed graphs and it is defined as the minimum amount of required distortion to transform one graph into the other. Six Edit operations define this distortion: insertion, deletion and substitution of nodes and arcs. To quantitatively determine the degree of distortion, a penalty cost to each edit operation is defined according to the amount of distortion that it introduces in the transformation. Although a proper definition of these costs is a cornerstone of classification or clustering applications, little research has been done to automatically find them. Usually, they are established through a manual validation process. This paper presents an optimization method to learn the value of these costs such that the Hamming distance between an oracle's node correspondence and the automatically correspondence is minimized. Experimental validation shows that the clustering and classification experiments drastically increase their accuracy with the automatically learned costs respect some usual cost values.

1. Introduction

Graphs refer to a collection of nodes and edges that connect pairs of nodes. Attributed graphs are graphs in which some attributes are added on nodes and edges to represent local information or characterization. Attributes on nodes and edges represent unary and binary relations of local parts of the objects at hand. Attributed graphs have been widely used in several fields to represent objects composed of local parts and relations between these parts [1–3]. Given two graphs, error-tolerant graph matching [4] obtains a distance value between the involved graphs and a correspondence between their nodes.

Given a pair of attributed graphs, several distance measures between them have been presented [5–7] but probably the most well known distance is the graph edit distance [4,8,9]. This distance measure is defined as the minimum amount of required distortion to transform one graph into the other. To this end, a number of distortion or edit operations, consisting of insertion, deletion and substitution of both nodes and edges are defined. To quantitatively evaluate the degree of distortion, edit cost functions are introduced. The basic idea is to assign a penalty cost to each edit operation according to the amount of distortion that it introduces in the transformation.

An interesting question arises in this context: given the attributed graphs that represent some objects, how we gauge the importance of each edit operation? That is, how we decide the penalty cost? Suppose we have an application that we want to match nodes of both graphs only if their attributes are almost similar, then, the penalty cost of substituting nodes or edges would have to be higher than insertion or deletion nodes and edges. Contrarily, if we wish almost all nodes to be mapped to the nodes of the other graph, then, the penalty cost of substitution would have to be very low. Usually, these weights are manually set at the validation process and little research has been carried out to automatically decide them.

Table 1 lists the five papers that have been published related to learning these edit costs. We have found two optimization functions (also called predictors). The recognition ratio given a classified test set and the average Hamming distance between an oracle's correspondence and the automatically deducted correspondence. One of the drawbacks of the first function is that graphs in the dataset need to be classified. This is not the case of the second function but it is needed an oracle's correspondence, therefore, the learning elements in the dataset must be composed of two graphs and an oracle's correspondence between nodes. Note that the oracle's correspondence is the labeling ground truth between nodes of the involved graphs.

Another important feature is the type of costs the learning algorithm obtains. For instance, methods in [10] and [11] obtain a self organizing map (SOM) and a probability density function (PDF), respectively. Therefore, in these cases, classical graph matching algorithms have to be tuned to be applied on these learning methods since these algorithms assume edit costs are real numbers. Methods

[☆] This paper has been recommended for acceptance by R. Davies

[*] Corresponding author. Tel.: +34 977558507; fax: +34 977558710.
E-mail address: francesc.serratosa@urv.cat (F. Serratosa).

http://dx.doi.org/10.1016/j.patrec.2015.01.009
0167-8655/© 2015 Elsevier B.V. All rights reserved.

Table 1
Papers published about graph matching learning.

Ref.	Year	Optimization function	Costs	Optimization algorithm
[10]	2005	Recognition ratio	SOM	Self organizing maps (SOM) [15]
[11]	2007	Recognition ratio	PDF	Expectation minimization [16]
[12]	2009	Hamming distance	Substitution weights	Bundle method [17]
[13]	2011	Hamming distance	Insertion and deletion	Interactive labeling space [18]
[14]	2012	Hamming distance Recognition ratio	Substitution weights	Spectral [19]

[12] and [14] assume nodes and edges have several attributes (for instance features obtained by SIFT descriptors) and these methods obtain a weight for each feature. In these cases, there is not a unique substitution cost on nodes and on edges but a vector of substitution weights, one for each feature. Moreover, insertion and deletion costs are not learned. Finally, method described in [13] computes the substitution, insertion and deletion costs on nodes and edges considering the interaction of a human. Contrarily of methods [12] and [14], this method assumes a unique substitution costs on nodes and edges.

In this paper, we present a method to learn the real numbers for the insertion and deletion costs on nodes and edges without a human interaction such that the Hamming distance between a ground truth node correspondence and the automatically obtained correspondence is minimized. This is because, in some applications, graphs in the reference and test databases are not split in classes and so it is not valid to minimize the recognition ratio. This is the case of some graph retrieval applications [1], in which the aim is to find similar graphs without a previous classification. Moreover, there are some graph databases [39] such that nodes and edges have only one or any attributes. In these cases, it has non-sense to learn the substitution weights for each feature as in [12] and [14] but it is crucial to learn the best combinations of insertion and deletion costs on nodes and edges as unique real numbers. Finally, we want the classical graph matching algorithms [2,3] to be applied on our obtained costs. For this reason, the computed values have to be real numbers and therefore, methods [10] and [11] are not valid. In the next section, we summarize the graph edit distance and the Hamming distance. In Section 3, we explain our optimization method and in Section 4, we show our experimental validation. We conclude the paper in Section 5.

2. Graph edit distance and Hamming distance

2.1. Attributed graph

An attributed graph (over Δ_v and Δ_e) is defined by a tuple $G = (\Sigma_\nu, \Sigma_e, \gamma_v, \gamma_e)$, where $\Sigma_v = \{v_a \mid a = 1, \ldots, n\}$ is the set of nodes, $\Sigma_e = \{e_{ab} \mid a, b \in 1, \ldots, n\}$ is the set of edges, $\gamma_v : \Sigma_v \to \Delta_v$ assigns attribute values to nodes and $\gamma_e : \Sigma_e \to \Delta_e$ assigns attribute values to edges. The order of graph G is n.

2.2. Error-tolerant graph matching and graph edit distance between two attributed graphs

Error-tolerant graph matching is the problem of finding a distance value and a node correspondence between two attributed graphs. Let $G = (\Sigma_\nu, \Sigma_e, \gamma_v, \gamma_e)$ and $G' = (\Sigma'_v, \Sigma'_e, \gamma'_v, \gamma'_e)$ be two attributed graphs of initial order n and m. To allow maximum flexibility in the matching process, graphs are extended with null nodes [4] to be of order $n+m$. We refer to null nodes of G and G' by $\hat{\Sigma}_\nu \subseteq \Sigma_\nu$ and $\hat{\Sigma}'_v \subseteq \Sigma'_v$ respectively. Let T be a set of all possible bijections between two sets Σ_ν and Σ'_v. We define the non-existent or null edges by $\hat{\Sigma}_e \subseteq \Sigma_e$ and $\hat{\Sigma}'_e \subseteq \Sigma'_e$. Bijection $f : \Sigma_\nu \to \Sigma'_v$, assigns one node of G to only one node of G'. The bijection between edges is defined accordingly to the bijection of their terminal nodes.

Graph edit distance [4,8,9] is the most used method to solve the error-tolerant graph matching. As commented in the introduction, the distance is defined as the minimum amount of required distortion to transform one graph into the other through inserting, deleting and substituting nodes and edges. Moreover, some penalty costs are assigned to each edit operation. Section 2.2 of [20] properly explains this distance with a toy example. Deletion (Insertion) operations are transformed to assignations in f of non-null nodes of the first (second) graph to null nodes of the second (first) graph. Substitutions simply indicate node-to-node assignations in f. Using this transformation, given two graphs, G and G', and a bijection between their nodes, f, the graph edit cost is given by (Eq. 2 in [20]):

$$\begin{aligned} EditCost(G, G', f) = & \sum_{\substack{v_a \in \Sigma_\nu - \hat{\Sigma}_\nu \\ v'_i \in \Sigma'_\nu - \hat{\Sigma}'_\nu}} C_{vs}(v_a, v'_i) + \sum_{\substack{v_a \in \Sigma_\nu - \hat{\Sigma}_\nu \\ v'_i \in \hat{\Sigma}'_\nu}} C_{vd}(v_a, v'_i) \\ & + \sum_{\substack{v_a \in \hat{\Sigma}_\nu \\ v'_i \in \Sigma'_\nu - \hat{\Sigma}'_\nu}} C_{vi}(v_a, v'_i) + \sum_{\substack{e_{ab} \in \Sigma_e - \hat{\Sigma}_e \\ e'_{ij} \in \Sigma'_e - \hat{\Sigma}'_e}} C_{es}(e_{ab}, e'_{ij}) \\ & + \sum_{\substack{e_{ab} \in \Sigma_e - \hat{\Sigma}_e \\ e'_{ij} \in \hat{\Sigma}'_e}} C_{ed}(e_{ab}, e'_{ij}) + \sum_{\substack{e_{ab} \in \hat{\Sigma}_e \\ e'_{ij} \in \Sigma'_e - \hat{\Sigma}'_e}} C_{ei}(e_{ab}, e'_{ij}) \\ & \text{Being } f(v_a) = v'_i \text{ and } f(v_b) = v'_j \end{aligned} \quad (1)$$

where C_{vs} is the cost of substituting node v_a of G by node v'_i of G', C_{vd} is the cost of deleting node v_a of G and C_{vi} is the cost of inserting node v'_i of G'. Equivalently for edges, C_{es} is the cost of substituting edge e_{ab} of graph G by edge e'_{ij} of G', C_{ed} is the cost of assigning edge e_{ab} of G to a non-existing edge of G' and C_{ei} is the cost of assigning edge e'_{ij} of G' to a non-existing edge of G. The cost of mapping two null nodes or null edges is always zero. With these costs, the Graph Edit distance is defined as the minimum cost under any bijection in T [20]:

$$EditDist(G, G') = \min_{f \in T} EditCost(G, G', f) \quad (2)$$

Finally, the optimal correspondence $\dot{f}$ is the one that obtains the minimum cost,

$$\dot{f} = \operatorname*{argmin}_{f \in T} EditCost(G, G', f) \quad (3)$$

Using these definitions, the graph distance depends on C_{vs}, C_{vd}, C_{vi}, C_{es}, C_{ed} and C_{ei} costs and several definitions of these costs have been published. There are two main options to define the costs on substituting nodes and edges, C_{vs} and C_{es}. The first one considers cost $C_{vs} \in \{0, K_{vs}\}$ where $C_{vs}(v_a, v'_i) = K_{vs}$ if $dist(v_a, v'_i) > \text{Threshold}$ otherwise $C_{vs} = 0$ (similarly for C_{es}). Function $dist$ is defined as a distance over the domain of the attributes. Specific examples of this cost can be found in [21–23]. The second one corresponds to the case where $C_{vs}(v_a, v'_i) \in \mathbb{R}$ or $C_{es}(v_a, v'_i) \in \mathbb{R}$. In this case, node and edge substitution costs depend on the attributes of the nodes and edges and possibly on some other parameters as shown in [12,24,25], among others. Functions C_{vd}, C_{vi}, C_{ed} and C_{ei}, are usually defined as a constant, but in some cases depend on node or edge attributes [26–28].

Several specific definitions of C_{vs}, C_{vd}, C_{vi}, C_{es}, C_{ed} and C_{ei} costs have been studied, which are summarized in Table 2. We propose