Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Matching Node Embeddings for Graph Similarity

Giannis Nikolentzos
École Polytechnique and AUEB
nikolentzos@aueb.gr

Polykarpos Meladianos
École Polytechnique and AUEB
pmeladianos@aueb.gr

Michalis Vazirgiannis
École Polytechnique and AUEB
mvazirg@lix.polytechnique.fr

Abstract

Graph kernels have emerged as a powerful tool for graph comparison. Most existing graph kernels focus on local properties of graphs and ignore global structure. In this paper, we compare graphs based on their global properties as these are captured by the eigenvectors of their adjacency matrices. We present two algorithms for both labeled and unlabeled graph comparison. These algorithms represent each graph as a set of vectors corresponding to the embeddings of its vertices. The similarity between two graphs is then determined using the Earth Mover's Distance metric. These similarities do not yield a positive semidefinite matrix. To address for this, we employ an algorithm for SVM classification using indefinite kernels. We also present a graph kernel based on the Pyramid Match kernel that finds an approximate correspondence between the sets of vectors of the two graphs. We further improve the proposed kernel using the Weisfeiler-Lehman framework. We evaluate the proposed methods on several benchmark datasets for graph classification and compare their performance to state-of-the-art graph kernels. In most cases, the proposed algorithms outperform the competing methods, while their time complexity remains very attractive.

1 Introduction

In recent years, graph-based representations have become extremely popular for modelling real-world data. Some examples of data represented as graphs include social networks, protein or gene regulation networks and textual documents. Graph kernels have been successfully used for comparing graphs in a variety of fields such as Computational Biology (Schölkopf, Tsuda, and Vert 2004), Chemistry (Mahé and Vert 2009), Information Retrieval (Hermansson et al. 2013) and Medicine (Feragen et al. 2013).

In the heart of graph kernels lies a positive semidefinite kernel function k. Once we define such a function $k : X \times X \rightarrow \mathbb{R}$ on a set X, it is known that there exists a map $\phi : \mathcal{X} \rightarrow \mathbb{H}$ into a Hilbert space $\mathbb{H}$, such that $k(x, x') = \langle \phi(x), \phi(x') \rangle$ for all $x, x' \in \mathcal{X}$ where $\langle \cdot, \cdot \rangle$ is the inner product in $\mathbb{H}$. Graph kernels have traditionally employed functions that compare substructures of graphs that are computable in polynomial time. More specifically, there is a plethora of graph kernels, each focusing on a different structural aspect of graphs. There are kernels based on random walks (Kashima, Tsuda, and Inokuchi 2003; Gärtner, Flach, and Wrobel 2003), shortest paths (Borgwardt and Kriegel 2005), subtrees (Gärtner, Flach, and Wrobel 2003; Mahé and Vert 2009), cycles (Horváth, Gärtner, and Wrobel 2004) and graphlets (Shervashidze et al. 2009). The performance of these kernels can be considerably improved by using the Weisfeiler-Lehman test of isomorphism (Shervashidze et al. 2011) or by taking into account similarities between the extracted instances of the considered substructures (Yanardag and Vishwanathan 2015b; 2015a).

Most existing graph kernels hence compare specific substructures of graphs (trees, cycles, paths, etc.). These substructures correspond either to small subgraphs present in the graphs or to relationships between very small subsets of vertices. Hence, these algorithms focus on local properties of graphs and ignore global structure. For graphs of small size, such local approaches may be appropriate. However, for larger graphs, they may suffer from their local nature and fail to perform equivalently, as several interesting properties of these graphs may not be captured in local substructures. Hence, the need for graph kernels that are built upon global properties of graphs is clear. Some very recent work in this direction has resulted in the development of two graph kernels that are based on the Lovász number and the corresponding orthonormal representation (Johansson et al. 2014). However, these kernels are defined only on unlabeled graphs. A second approach compares pairs of graphs by computing the optimal matching between the embeddings of their vertices generated from various graph-related matrices (adjacency matrix, Laplacian matrix, etc.) (Johansson and Dubhashi 2015). However, the emerging kernel matrix is not always positive semidefinite (Vert 2008) and the authors resort to the Balcan-Blum-Srebro theory of classification with similarity functions.

The goal of this paper is to address the aforementioned problems of graph kernels that focus on local substructures of graphs. By employing algorithms that utilize features describing global properties of graphs, we can overcome these inherent limitations of "local" kernels. It is also desirable that these algorithms produce valid kernel matrices so that they enjoy the attractive properties of kernel methods.

Main Contributions We present two novel algorithms designed to compare pairs of graphs based on their global

properties. The algorithms are applicable to both labeled and unlabeled graphs. Each graph is first represented as a collection of the embeddings of its vertices. The vertices of the graphs are embedded in the Euclidean space using the eigenvectors of the corresponding adjacency matrices. The similarity between pairs of graphs is measured by computing a matching between their sets of embeddings. Two algorithms are employed. The first algorithm casts the problem as an instance of the *Earth Mover's Distance* (Rubner, Tomasi, and Guibas 2000), a well-known transportation problem and for a set of graphs builds a similarity matrix. However, the similarity matrix that emerges is not always positive semidefinite. To account for this problem, we capitalize on a well-established algorithm for Support Vector Machine (SVM) classification using indefinite kernels (Luss and d'Aspremont 2008) which treats the indefinite similarity matrix as a noisy observation of the true positive semidefinite kernel. The second algorithm corresponds to a technique adapted from the *Pyramid Match* kernel (Grauman and Darrell 2007) and yields a positive semidefinite matrix. We call this method the *Pyramid Match* graph kernel. In the case of labeled graphs, to further improve the proposed kernel, we employ the Weisfeiler-Lehman framework (Shervashidze et al. 2011). We show via experimental evaluation on several graph classification datasets that the proposed algorithms produce better or competitive results with state-of-the-art graph kernels and that are also efficient in terms of running time.

The rest of this paper is organized as follows. Section 2 introduces some preliminary concepts and gives details about how the embeddings of vertices are generated. Section 3 presents the formulation of the problem as an instance of the Earth Mover's Distance and how we train an SVM using the emerging indefinite similarity matrix. Section 4 presents the Pyramid Match graph kernel. Section 5 evaluates the proposed algorithms and compares them with existing methods. Finally, Section 6 concludes.

2 Preliminaries

In this section, we first introduce basic concepts from graph theory, and define our notation. We then present how the nodes of the graphs are embedded in the vector space. We finally give details regarding the representation of graphs as bag-of-vectors.

Graph Concepts

Let $G = (V, E)$ be an undirected and unweighted graph consisting of a set V of vertices and a set E of edges between them. In this paper, we will denote by n the number of vertices and by m the number of edges. A graph G can be represented by its adjacency matrix $\mathbf{A}$. If $\mathbf{A}_{ij}$ is the element in the i^{th} row and j^{th} column of matrix $\mathbf{A}$, the adjacency matrix $\mathbf{A}$ can be defined as follows:

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise} \end{cases}$$

where (v_i, v_j) is an edge between vertices v_i and v_j of V.

A labeled graph is a graph with labels on vertices and/or edges. In this paper, we will consider unlabeled graphs and graphs with labeled vertices. Given a set of labels $\mathcal{L}$, $\ell : V \to \mathcal{L}$ is a function that assigns labels to the vertices of the graph.

Embeddings of nodes

Given a graph $G = (V, E)$, we can represent its vertices as points in a vector space using a node embedding algorithm. In this paper, we generate embeddings for the vertices of a graph using the eigenvectors of its adjacency matrix $\mathbf{A}$. Given the eigenvalue decomposition of $\mathbf{A}$, $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, the i^{th} row $\mathbf{u}_i$ of $\mathbf{U}$ corresponds to the embedding of vertex $v_i \in V$. Such embeddings capture global properties of graphs and offer a powerful and flexible mechanism for performing machine learning tasks on them. Since $\mathbf{A}$ is real and symmetric, its eigenvalues $\lambda_1, \ldots, \lambda_n$ are real. The considered graphs contain no self-loops, hence, $\mathrm{Tr}(\mathbf{A}) = 0$, and since $\mathrm{Tr}(\mathbf{A}) = \lambda_1 + \ldots + \lambda_n$, the eigenvalues sum to zero. Therefore, some eigenvalues will be positive, while others will be negative. The eigenvectors corresponding to the largest (in magnitude) eigenvalues share some interesting properties. For example, according to the Perron-Frobenius theorem, since $\mathbf{A}$ has only nonnegative entries, it has a nonnegative real eigenvalue λ_{max} which has maximum absolute value among all eigenvalues. This eigenvalue λ_{max} has a nonnegative real eigenvector. The i^{th} component of the related eigenvector gives the eigenvector centrality score of vertex v_i in the graph (Jackson 2010). Eigenvector centrality is a measure of global connectivity of a graph which is captured by the spectrum of the adjacency matrix. Other global properties of the vertices of a graph are captured by the eigenvectors corresponding to the next largest eigenvalues of $\mathbf{A}$. Note that the signs of these eigenvectors are arbitrary, hence, we replace all their components by their absolute values. In this paper, we generate low-dimensional representations for the vertices of each graph $v_i \in V$. Specifically, we embed all vertices in the d-dimensional vector space $\mathbb{R}^d$ using the eigenvectors of the d largest in magnitude eigenvalues. We can thus think of each vertex as a point in the d-dimensional unit hypercube.

Bag-of-vectors

Let $\mathcal{G} = \{G_1, G_2, \ldots, G_N\}$ be a population of N graphs. Assume we have embedded the nodes of each graph $G_i \in \mathcal{G}$ in the d-dimensional space and $\mathbf{U}_{G_i} \in \mathbb{R}^{n_i \times d}$ is the embedding matrix for the vertices of G_i. The i^{th} row, $\mathbf{u}_i \in \mathbb{R}^d$ represents the embedding of the i^{th} node in the d-dimensional space.

In several domains, it is useful to represent a data object with a collection of its parts. For instance, in Natural Language Processing, documents are usually represented as bags-of-words. Likewise, in Image Recognition, images may be described by sets of features, where each feature is extracted from a single pixel. In this paper, we focus on representing graphs as bags-of-vectors. For example, a graph G can be represented as the following set: $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$, where each vector of the set corresponds to the representation of each vertex $u_i \in V$ in $\mathbb{R}^d$. Since there is no canonical ordering for the nodes of a graph, this is a natural represen-

tation and is also meaningful as each graph can be seen as "a sum of its parts".

3 Earth Mover's Distance

A natural way to measure the similarity between pairs of graphs is by comparing their building blocks, i. e. their vertices. As mentioned above, we represent each graph as a bag-of-vectors, where the vectors correspond to the embeddings of its vertices. This enables us to formalize the comparison of two graphs as a transportation problem and solve it using the Earth Mover's Distance (EMD) (Rubner, Tomasi, and Guibas 2000). More specifically, given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the minimum "travel cost" between the two graphs is provided by the solution of the following linear program:

$$\begin{aligned} \min & \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \mathbf{T}_{ij} d(v_i, u_j) \\ & \text{subject to} \\ & \sum_{i=1}^{n_1} \mathbf{T}_{ij} = \frac{1}{n_2} \quad \forall j \in \{1,\dots,n_2\} \\ & \sum_{j=1}^{n_2} \mathbf{T}_{ij} = \frac{1}{n_1} \quad \forall i \in \{1,\dots,n_1\} \\ & \mathbf{T}_{ij} \geq 0 \quad \forall i \in \{1,\dots,n_1\}, \forall j \in \{1,\dots,n_2\} \end{aligned} \tag{1}$$

where $d(v_i, u_j)$ is a measure of vertex dissimilarity between nodes $v_i \in V_1$ and $u_j \in V_2$, and $\mathbf{T} \in \mathbb{R}^{n1 \times n2}$ is a flow matrix. In our case, the vertex dissimilarity $d(v_i, u_j)$ between nodes v_i and u_j is provided by their Euclidean distance in the embedding space. Hence, $d(v_i, u_j) = ||\mathbf{u}_i - \mathbf{u}_j||_2$. As regards to the flow matrix $\mathbf{T}$, its elements $\mathbf{T}_{ij} \geq 0$ denote how much of vertex $v_i \in V_1$ travels to vertex $u_j \in V_2$. The above formulation allows each vertex $v_i \in V_1$ to be transformed into any vertex $u_j \in V_2$ in total or in parts. The outgoing flow from each graph is equal to 1 and is equally divided among all vertices. Therefore, to transform G_1 entirely into G_2, we ensure that the entire outgoing flow from vertex v_i equals $\frac{1}{n_1}$, i. e. $\sum_j \mathbf{T}_{ij} = \frac{1}{n_1}$. Further, the amount of incoming flow to vertex u_j must match $\frac{1}{n_2}$, i. e. $\sum_i \mathbf{T}_{ij} = \frac{1}{n_2}$. Hence, the distance between the two graphs is defined as the minimum cumulative cost required to move all vertices from G_1 to G_2.

EMD is a well-studied transportation problem which has been successfully used in Computer Vision (Rubner, Tomasi, and Guibas 2000) and in Text Categorization (Kusner et al. 2015). There exist several fast specialized algorithms in the literature for solving EMD instances (Pele and Werman 2009). Specifically, solving the above optimization problem requires $\mathcal{O}(n^3 log n)$ time where n is the number of vertices of the two graphs.

The above methodology applies in the case of simple, undirected, unlabeled graphs. However, our method is evaluated on datasets consisting of graphs with labeled vertices. Hence, neglecting these labels is not the best practice. To account for these labels, we can simply set the distances between pairs of vertices with different labels to the largest possible value. The nodes of each graph lie in the interior of the unit hypercube $[0, 1]^d$ in $\mathbb{R}^d$, since the norm of each eigenvector is equal to 1 and we have set all the components of the eigenvectors to their absolute values. The length of the longest diagonal of the d-dimensional unit hypercube is $\sqrt{d}$, hence, the maximum possible Euclidean distance between two points in that space is also $\sqrt{d}$. Hence, we set the distance between nodes of different labels equal to that value:

$$d'(v_i, u_j) = \begin{cases} d(v_i, u_j) & \text{if } \ell(v_i) = \ell(u_j), \\ \sqrt{d} & \text{otherwise} \end{cases} \tag{2}$$

where $d(v_i, u_j)$ is the Euclidean distance between the embeddings of the two vertices.

Let $\mathbf{D} \in \mathbb{R}^{N \times N}$ be the distance matrix that is built from all the pairwise distances between graphs $G_i, G_j \in \mathcal{G}$. It would be very convenient if $\mathbf{D}$ was a Euclidean Distance Matrix (EDM) (Gower 1982; Gower and Legendre 1986). In such a case, we could use $\mathbf{D}$ to define a positive semidefinite kernel matrix $\mathbf{K} \in \mathbb{R}^{N \times N}, \mathbf{K} \succeq 0$ as follows:

$$\mathbf{K} = -\frac{1}{2}\mathbf{J}\mathbf{D}\mathbf{J} \tag{3}$$

where $\mathbf{J}$ is the centering matrix $\mathbf{J} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T \in \mathbb{R}^{N \times N}$ and $\mathbf{I}$ is the identity matrix. A matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ is called a Euclidean Distance Matrix (EDM), when its entries, d_{ij}^2 are the Euclidean distance-squares between pairs of $\mathbf{x}_i$ and $\mathbf{x}_j$, i. e. $\mathbf{D}_{ij} = d_{ij}^2 = ||\mathbf{x}_i - \mathbf{x}_j||^2$. An alternative definition by Schoenberg (1935) states that a matrix $\mathbf{D}$ is an EDM if and only if it belongs to the symmetric hollow subspace $\mathbf{D} \in \mathbb{S}_h^N$, i. e. the subspace of symmetric matrices $\mathbb{S}^N$ of size N with a zero diagonal and is negative semidefinite $\mathbf{D} \preceq 0$. However, the distance matrices generated by EMD are not necessarily negative semidefinite (Naor and Schechtman 2007), and cannot thus be used directly to define positive semidefinite kernels. In our setting, although matrix $\mathbf{K}$ is not positive semidefinite (since $\mathbf{D}$ is not Euclidean), it corresponds to a similarity matrix and we can consider it as a noisy observation of an unknown positive semidefinite kernel in order to derive a valid kernel. Let $\mathbf{K}_0$ be the indefinite similarity matrix and $\mathbf{y} \in \mathbb{R}^N$ be the vector of class labels, with $\mathbf{Y} = diag(\mathbf{y})$ the matrix with diagonal $\mathbf{y}$. We can formulate the following extension of the SVM for indefinite kernels (Luss and d'Aspremont 2008):

$$\begin{aligned} \max_{\boldsymbol{\alpha}^T\mathbf{y}=0, 0 \leq \mathbf{a} \leq C\mathbf{1}} \quad & \min_{\mathbf{K} \succeq 0} \quad \boldsymbol{\alpha}^T\mathbf{1} - \frac{1}{2}\operatorname{Tr}(\mathbf{K}(\mathbf{Y}\boldsymbol{\alpha})(\mathbf{Y}\boldsymbol{\alpha})^T) \\ & \qquad + \rho||\mathbf{K} - \mathbf{K}_0||_F^2 \end{aligned} \tag{4}$$

where we solve in the variables $\mathbf{K} \in \mathbb{S}^N$ and $\boldsymbol{\alpha} \in \mathbb{R}^N$, and where parameter $\rho > 0$ controls the magnitude of the penalty on the distance between $\mathbf{K}$ and $\mathbf{K}_0$. While the original SVM classification problem with indefinite kernel is nonconvex, the above formulation corresponds to a convex problem, and hence, can be solved efficiently with guaranteed complexity bounds. What the above convex problem essentially does is to simultaneously learn the support vector weights and a proxy positive semidefinite kernel matrix, while trying to minimize the distance between this proxy kernel and the indefinite similarity matrix.

4 Pyramid Match Graph Kernel

Although we can use the above scheme to perform SVM classification using the similarity matrix generated by EMD, the existence of negative eigenvalues with large magnitude may lead to proxy kernel matrices that are not in accordance with the data. It would be desirable to employ an algorithm that produces positive semidefinite kernel matrices so that we can capitalize on the theoretical properties associated with kernel methods.

We present below an algorithm that is based on the Pyramid Match kernel (PM), a very popular algorithm in Computer Vision (Grauman and Darrell 2007; Lazebnik, Schmid, and Ponce 2006). The algorithm generates positive semidefinite kernel matrices (hence, it is a graph kernel). Very recently, a graph kernel that employs the Vocabulary-Guided Pyramid Match kernel for matching the vector representations of the vertices of two graphs was proposed (Su et al. 2016). However, these representations are generated based on the labels and the neighbors of the vertices and are thus in contrast to the proposed approaches local in nature. As mentioned above, each graph is represented as a bag-of-vectors. The basic idea of the algorithm is to map these vectors to multi-resolution histograms, and then compare the histograms with a weighted histogram intersection measure in order to find an approximate correspondence between the two sets of vectors.

PM works by partitioning the feature space into regions of increasingly larger size and taking a weighted sum of the matches that occur at each level. Two points are said to match if they fall into the same region. Matches made within larger regions are weighted less than those found in smaller regions. Recall that in our setting, each vertex corresponds to a point lying in the d-dimensional unit hypercube. We repeatedly fit a grid with cells of increasing size to the d-dimensional unit hypercube. Each cell is related only to a specific dimension and its size along that dimension is doubled at each iteration, while its size along the other dimensions stays constant and equal to 1. Given a sequence of levels from 0 to L, then at level l, the d-dimensional unit hypercube has 2^l cells along each dimension and $D = 2^l d$ cells in total. Given a pair of graphs $G_1, G_2 \in \mathcal{G}$, let $H^l_{G_1}$ and $H^l_{G_2}$ denote the histograms of G_1 and G_2 at level l, and $H^l_{G_1}(i)$, $H^l_{G_2}(i)$, the number of vertices of G_1, G_2 that lie in the i^{th} cell. The number of points in two sets which match at level l is then computed using the histogram intersection function:

$$I(H^l_{G_1}, H^l_{G_2}) = \sum_{i=1}^{D} \min\left(H^l_{G_1}(i), H^l_{G_2}(i)\right) \tag{5}$$

The matches that occur at level l also occur at levels $0, \ldots, l-1$. We are interested in the number of new matches found at each level which is given by $I(H^l_{G_1}, H^l_{G_2}) - I(H^{l+1}_{G_1}, H^{l+1}_{G_2})$ for $l = 0, \ldots, L-1$. The number of new matches found at each level in the pyramid is weighted according to the size of that level's cells. Matches found within smaller cells are weighted more than those made in larger cells. Specifically, the weight for level l is set equal to $\frac{1}{2^{L-l}}$. Hence, the weights are inversely proportional to the length of the side of the cells that varies in size as the levels increase. The pyramid match kernel is then defined as follows:

$$k_\Delta(G_1, G_2) = I(H^L_{G_1}, H^L_{G_2}) + \sum_{l=0}^{L-1} \frac{1}{2^{L-l}} \big(I(H^l_{G_1}, H^l_{G_2}) - I(H^{l+1}_{G_1}, H^{l+1}_{G_2})\big) \tag{6}$$

The pyramid match kernel is a Mercer kernel (Grauman and Darrell 2007), hence, by computing it for all pairs of graphs, we can build a positive semidefinite kernel matrix. This matrix guarantees an optimal solution to kernel-based algorithms such as SVMs. The complexity of the pyramid match kernel is $\mathcal{O}(dnL)$ where n is the number of nodes of the graphs under comparison.

The above graph kernel is defined for unlabeled graphs. All vertices of the first graph can match to any vertex of the second. However, in the case of labeled graphs, we would like to restrict matchings to occur only between vertices that share same labels. Instead of representing each graph as a set of vectors, we can represent it as a set of sets of vectors, where each internal set corresponds to a specific label and contains the embeddings of the vertices with that label. We can then match sets of the two graphs corresponding to the same label using the pyramid match kernel. The emerging kernel for labeled graphs corresponds to the sum of the separate kernels:

$$k^{lab}_\Delta = \sum_{i=1}^{c} k^i_\Delta(G_1, G_2) \tag{7}$$

where c is the number of distinct labels and $k^i_\Delta(G_1, G_2)$ is the pyramid match kernel between the sets of vertices of the two graphs for which $\ell(v) = i$. We can further enhance the PM kernel by employing the Weisfeiler-Lehman framework (Shervashidze et al. 2011).

5 Experiments

In this section, we first describe the datasets that we used for our experiments. We next present the graph kernels against which we compared our algorithms, and give details about the experimental settings. We last report on the performance of our methods and the baselines.

Datasets

We evaluated the performance of our methods on the following 6 bioinformatics datasets: MUTAG, ENZYMES, NCI1, NCI109, PTC-MR and D&D. The MUTAG dataset (Debnath et al. 1991) consists of 188 mutagenic aromatic and heteroaromatic nitro compounds. Each chemical compound is labeled according to whether or not it has mutagenic effect on the Gram-negative bacterium *Salmonella typhimurium*. The ENZYMES dataset (Borgwardt et al. 2005) contains 600 graphs representing enzymes from the BRENDA enzyme database. Each enzyme is a member of one of the 6 Enzyme Commission top level enzyme classes (EC classes) and the task is to correctly assign the enzymes to their

Dataset	MUTAG	ENZYMES	NCI1	NCI109	PTC-MR	D&D
Max # vertices	28	126	111	111	109	5748
Min # vertices	10	2	3	4	2	30
Average # vertices	17.93	32.63	29.87	29.68	25.56	284.32
Max # edges	33	149	119	119	108	14267
Min # edges	10	1	2	3	1	63
Average # edges	19.79	62.14	32.30	32.13	25.96	715.66
# labels	7	3	37	38	19	82
# graphs	188	600	4110	4127	344	1178
# classes	2	6	2	2	2	2

Table 1: Summary of the 6 datasets that were used in our experiments.

classes. The NCI1 and NCI109 datasets (Wale, Watson, and Karypis 2008) are two balanced subsets of chemical compounds which are classified based on whether they are active or inactive against non-small cell lung cancer and ovarian cancer cell lines respectively. The PTC-MR dataset (Toivonen et al. 2003) consists of 344 organic molecules marked according to their carcinogenicity on male rats. The D&D dataset (Dobson and Doig 2003) contains 1178 protein structures. Each protein is a graph whose nodes correspond to amino acids and a pair of amino acids are linked by an edge if they are less than 6 Ångstroms apart. The task is to predict if a protein is an enzyme or not. Table 5 shows statistics of the five datasets that were used for the evaluation.

Baselines

We compare our methods against several state-of-the-art graph kernels. The first three are applicable both to labeled and unlabeled graphs, the next two only to labeled graphs, while the last one can be computed only for unlabeled graphs. More specifically, our baseline comparators are (1) the random walk kernel (RW) that counts common walks in two graphs (Vishwanathan et al. 2010), (2) the graphlet kernel (GR) that counts common connected subgraphs of size 3 (Shervashidze et al. 2009), (3) the shortest path kernel (SP) that counts shortest paths of equal length between pairs of nodes (Borgwardt and Kriegel 2005), (4) the Weisfeiler-Lehman subtree kernel (WL ST) that for a number of iterations counts pairs of matching subtree patterns in two graphs, while at each iteration updates the labels of the nodes of the graph (Shervashidze et al. 2011), (5) the Weisfeiler-Lehman shortest path kernel (WL SP) that does the same for shortest paths (Shervashidze et al. 2011) and (6) the Lovász ϑ kernel (Lo-ϑ) that captures global graph properties using the Lovász orthonormal representation (Johansson et al. 2014). We also compare our methods against (7) an approach that computes the optimal assignment between the embeddings of the vertices of two graphs (OA) (Johansson and Dubhashi 2015). For a fair comparison, we used embeddings generated from the adjacency matrix. Like EMD, this method does not always yield positive semidefinite matrices.

We set the parameter λ of the random walk kernel equal to the largest power of 10 smaller than the inverse of the square maximum degree (Shervashidze et al. 2011). As regards the parameter h of the Weisfeiler-Lehman kernels, it was chosen by cross-validation on the training set for $h \in \{0, 1, \ldots, 10\}$ in the case of the subtree kernel, for $h \in \{0, 1, 2\}$ in the case of the shortest path kernel, and for $h \in \{0, 1, \ldots, 5\}$ in the case of the proposed pyramid match kernel and the optimal assignment method. The proposed methods are written in Matlab[1]. As regards the baselines, we used their publicly available implementations.

Experimental Setup

To perform graph classification, we employed a C-Support Vector Machine (SVM) classifier and in particular, the LIBSVM (Chang and Lin 2011) implementation and performed 10-fold cross-validation. The whole process was repeated 10 times for each dataset and each method. The parameter C of the SVM was optimized on the training set only. We report in Table 2 average prediction accuracies, standard deviations and CPU runtimes for computing each kernel/similarity matrix as measured in Matlab R2013a on a 3.4GHz Intel Core i7 with 16Gb of RAM.

Results

Our experiments show that embedding-based graph comparison algorithms have the potential to contribute to advances in real-world graph classification problems. The EMD-based approach and the PM and WL PM kernels achieved better classification accuracies on all datasets for unlabeled graphs and on four out of six datasets for labeled graphs. Specifically, in the case of unlabeled graphs, EMD outperformed all methods on all datasets except from the PTC-MR and D&D datasets on which the PM kernel led to better performance. As regards the kernel methods, the PM kernel led to best results on all datasets. It is worth noting that on all datasets except ENZYMES, it significantly improved the best accuracy attained by the other kernels. In the case of labeled graphs, EMD reached again the highest accuracy on MUTAG, while WL PM was the best performing method on the NCI1, NCI109 and PTC-MR datasets. EMD, PM and WL PM were all outperformed by the WL SP and GR kernels on the ENZYMES and D&D datasets respectively. Overall, the proposed methods performed better or comparable to state-of-the-art graph kernels. The OA method which is closly related to the proposed approaches was outperformed by at least one of EMD, PM and WL PM on all datasets. However, its performance was close to that of the proposed approaches. As regards the good classification performance demonstrated by EMD, this may be due to the fact that the generated similarity matrices were either positive semidefinite or close to being positive semidefinite.

In terms of runtime, we observe that on all datasets except D&D, the EMD and PM approaches are slower than the SP, GR, WL ST and WL SP kernels, however, they are faster than the RW, Lo-ϑ kernels and the OA approach. It is worth mentioning that even if PM is slower than the above-mentioned four kernels, its computing times are by no means prohibitive. Between the three methods that utilize embeddings of the adjacency matrix, PM is the fastest followed by EMD and OA. More specifically, there is a significant difference in runtime between PM and the other

[1]Our code is available at http://www.db-net.aueb.gr/nikolentzos/code/matchingnodes.zip

Method \ Datasets	MUTAG	ENZYMES	NCI1	NCI109	PTC-MR	D&D
Without node labels						
SP	82.22 (± 1.14) 0.16”	28.17 (± 0.64) 1.26”	62.02 (± 0.17) 7.55”	61.41 (± 0.32) 7.32”	56.18 (± 0.56) 0.47”	> 1 day
RW	77.78 (± 0.98) 17.01”	20.17 (± 0.83) 4’ 46”	56.89 (± 0.24) 2h 40’ 15”	56.13 (± 0.31) 2h 44’ 29”	56.18 (± 1.12) 1’ 23”	58.71 (± 0.43) 6h 45’ 1”
GR	66.11 (± 1.31) 0.07”	18.16 (± 0.47) 2.01”	47.37 (± 0.15) 4.42”	48.39 (± 0.18) 4.37”	57.05 (± 0.83) 0.14”	63.67 (± 0.57) 1’ 04”
Lo-ϑ	82.78 (± 0.89) 15’ 26”	26.33 (± 0.44) 2h 11’ 31”	62.68 (± 0.24) 17h 41’ 57”	62.42 (± 0.27) 17h 45’ 27”	55.00 (± 0.62) 1h 9’ 58”	> 1 day
OA	79.44 (± 1.08) 6.56”	36.33 (± 0.71) 3’ 19”	67.81 (± 0.18) 2h 21’ 57”	66.94 (± 0.21) 2h 20’ 13”	56.17 (± 0.95) 42.94”	> 1 day
EMD	**86.11** (± 0.84) 4.5”	**36.83** (± 0.78) 1’ 57”	**72.65** (± 0.24) 1h 11’ 31”	**71.70** (± 0.16) 1h 10’ 37”	57.65 (± 0.59) 24.28”	> 1 day
PM	85.55 (± 0.63) 1.26”	28.17 (± 0.37) 8.07”	69.73 (± 0.11) 3’ 19”	68.37 (± 0.14) 3’ 17”	**59.41** (± 0.68) 3.51”	**75.55** (± 0.62) 41.23”
With node labels						
SP	87.78 (± 0.44) 0.16”	41.00 (± 0.26) 1.42”	72.85 (± 0.18) 9.86”	73.20 (± 0.16) 9.46”	60.00 (± 0.72) 0.48”	> 1 day
RW	81.11 (± 1.23) 3’ 51”	19.33 (± 0.62) 12’ 58”	> 1 day	> 1 day	57.06 (± 0.86) 1h 33’ 32”	> 1 day
GR	71.67 (± 0.81) 0.79”	32.00 (± 0.46) 17.77”	65.52 (± 0.35) 40.03”	66.70 (± 0.15) 41.36”	59.41 (± 0.94) 2.21”	**79.40** (± 0.39) 15’ 13”
OA	82.22 (± 0.68) 6.03”	43.16 (± 0.56) 1’ 57”	69.53 (± 0.20) 1h 33’ 35”	68.76 (± 0.15) 1h 32’ 51”	58.23 (± 0.82) 28.28”	77.52 (± 0.43) 7h 45’ 5”
WL ST	83.33 (± 0.86) 2.91”	52.16 (± 0.61) 17.12”	84.72 (± 0.16) 1’ 57”	84.26 (± 0.22) 2’ 00”	57.64 (± 0.66) 7.35”	76.83 (± 0.49) 5’ 06”
WL SP	84.55 (± 1.08) 1.27”	**61.00** (± 0.69) 12.23”	84.64 (± 0.21) 1’ 15”	84.29 (± 0.17) 1’ 13”	56.76 (± 0.74) 2.81”	> 1 day
WL OA	85.55 (± 1.02) 27.46”	53.66 (± 0.72) 7’ 2”	85.35 (± 0.18) 5h 7’ 57”	84.51 (± 0.14) 5h 4’ 5”	59.70 (± 1.01) 2’ 3”	> 1 day
EMD	**89.44** (± 0.76) 5.61”	43.17 (± 0.48) 1’ 53”	76.76 (± 0.17) 1h 15’ 44”	73.88 (± 0.18) 1h 16’ 10”	58.82 (± 0.83) 25.79”	> 1 day
PM	86.67 (± 0.60) 3.52”	40.33 (± 0.34) 21.50”	72.91 (± 0.53) 5’ 48”	71.97 (± 0.15) 5’ 50”	60.22 (± 0.86) 9.58”	77.78 (± 0.48) 4’ 30”
WL PM	87.77 (± 0.81) 9.20”	55.55 (± 0.56) 1’ 25”	**86.40** (± 0.20) 50’ 35”	**85.34** (± 0.23) 50’ 42”	**61.41** (± 0.81) 29.33”	78.63 (± 0.26) 16’ 15”

Table 2: Classification accuracy (± standard deviation) and CPU runtime for kernel/similarity matrix computation of the random walk kernel (RW), shortest path kernel (SP), graphlets of size 3 kernel (GR), Lovász ϑ kernel (Lo-ϑ), optimal assignment similarity (OA), Weisfeiler-Lehman subtree kernel (WL ST), Weisfeiler-Lehman shortest path kernel (WL SP), Weisfeiler-Lehman optimal assignment similarity (WL OA), earth mover's distance similarity (EMD), pyramid match kernel (PM) and Weisfeiler-Lehman pyramid match kernel (WL PM) on the 6 graph classification datasets. > 1 day indicates that the computation did not finish after 1 day.

two methods. When dealing with datasets containing large graphs such as the D&D dataset, the behaviour of the approaches completely changes. Specifically, on that dataset, half of the methods failed to compute the kernel/similarity matrix within 1 day, while from the remaining methods, only the PM, WL PM, WL ST and GR kernels managed to complete their computations within a reasonable time. PM is the fastest method on D&D for both the labeled and unlabeled case. Overall, the low computational complexity of the PM kernel makes it an attractive choice for large-scale graph classification problems. Furthermore, in contrast to EMD and OA, PM always yields a positive semidefinite kernel matrix, and enjoys the theoretical guarantees associated with kernel methods.

6 Conclusion

In this paper, we presented two algorithms for comparing labeled and unlabeled graphs. These algorithms represent each graph as a set of vectors. The vectors correspond to embeddings of the graph's vertices generated by eigenvalue decomposition of the adjacency matrix. The similarity between two graphs is then determined using the Earth Mover's Distance metric. These similarities do not yield a positive semidefinite matrix. We also present the Pyramid Match graph kernel that finds an approximate correspondence between the sets of vectors of the two graphs. We evaluated the two approaches on six datasets and compared them with several graph kernels. The algorithms showed good performance, while remaining competitive in terms of running time.

References

Borgwardt, K. M., and Kriegel, H. 2005. Shortest-path kernels on graphs. In *Proceedings of the 5th International Conference on Data Mining*, 74–81.

Borgwardt, K. M.; Ong, C. S.and Schönauer, S.; Vishwanathan, S.; Smola, A. J.; and Kriegel, H. 2005. Pro-

tein function prediction via graph kernels. *Bioinformatics* 21(Suppl. 1):i47–i56.

Chang, C., and Lin, C. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology* 2(3):27.

Debnath, A. K.; Lopez de Compadre, R. L.; Debnath, G.; Shusterman, A. J.; and Hansch, C. 1991. Structure-Activity Relationship of Mutagenic Aromatic and Heteroaromatic Nitro Compounds. Correlation with Molecular Orbital Energies and Hydrophobicity. *Journal of Medicinal Chemistry* 34(2):786–797.

Dobson, P. D., and Doig, A. J. 2003. Distinguishing Enzyme Structures from Non-enzymes Without Alignments. *Journal of molecular biology* 330(4):771–783.

Feragen, A.; Kasenburg, N.; Petersen, J.; de Bruijne, M.; and Borgwardt, K. 2013. Scalable kernels for graphs with continuous attributes. In *Advances in Neural Information Processing Systems*, 216–224.

Gärtner, T.; Flach, P.; and Wrobel, S. 2003. On Graph Kernels: Hardness Results and Efficient Alternatives. In *Learning Theory and Kernel Machines*. 129–143.

Gower, J. C., and Legendre, P. 1986. Metric and euclidean properties of dissimilarity coefficients. *Journal of classification* 3(1):5–48.

Gower, J. C. 1982. Euclidean Distance Geometry. *Mathematical Scientist* 7(1):1–14.

Grauman, K., and Darrell, T. 2007. The Pyramid Match Kernel: Efficient Learning with Sets of Features. *The Journal of Machine Learning Research* 8:725–760.

Hermansson, L.; Kerola, T.; Johansson, F.; Jethava, V.; and Dubhashi, D. 2013. Entity Disambiguation in Anonymized Graphs using Graph Kernels. In *Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management*, 1037–1046.

Horváth, T.; Gärtner, T.; and Wrobel, S. 2004. Cyclic Pattern Kernels for Predictive Graph Mining. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 158–167.

Jackson, M. O. 2010. *Social and Economic Networks*. Princeton University Press.

Johansson, F. D., and Dubhashi, D. 2015. Learning with Similarity Functions on Graphs using Matchings of Geometric Embeddings. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 467–476.

Johansson, F.; Jethava, V.; Dubhashi, D.; and Bhattacharyya, C. 2014. Global graph kernels using geometric embeddings. In *Proceedings of the 31st International Conference on Machine Learning*, 694–702.

Kashima, H.; Tsuda, K.; and Inokuchi, A. 2003. Marginalized Kernels Between Labeled Graphs. In *Proceedings of the 20th Conference in Machine Learning*, 321–328.

Kusner, M.; Sun, Y.; Kolkin, N.; and Weinberger, K. Q. 2015. From Word Embeddings To Document Distances. In *Proceedings of the 32nd International Conference on Machine Learning*, 957–966.

Lazebnik, S.; Schmid, C.; and Ponce, J. 2006. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition*, volume 2, 2169–2178.

Luss, R., and d'Aspremont, A. 2008. Support Vector Machine Classification with Indefinite Kernels. In *Advances in Neural Information Processing Systems*, 953–960.

Mahé, P., and Vert, J. 2009. Graph kernels based on tree patterns for molecules. *Machine learning* 75(1):3–35.

Naor, A., and Schechtman, G. 2007. Planar Earthmover is not in L_1. *SIAM Journal on Computing* 37(3):804–826.

Pele, O., and Werman, M. 2009. Fast and robust earth mover's distances. In *Proceedings of the 12th International Conference on Computer Vision*, 460–467.

Rubner, Y.; Tomasi, C.; and Guibas, L. J. 2000. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision* 40(2):99–121.

Schoenberg, I. J. 1935. Remarks to Maurice Frechet's Article "Sur La Definition Axiomatique D'Une Classe D'Espace Distances Vectoriellement Applicable Sur L'Espace De Hilbert". *Annals of Mathematics* 724–732.

Schölkopf, B.; Tsuda, K.; and Vert, J.-P. 2004. *Kernel Methods in Computational Biology*. MIT press.

Shervashidze, N.; Petri, T.; Mehlhorn, K.; Borgwardt, K. M.; and Vishwanathan, S. 2009. Efficient Graphlet Kernels for Large Graph Comparison. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 488–495.

Shervashidze, N.; Schweitzer, P.; Van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-Lehman Graph Kernels. *The Journal of Machine Learning Research* 12:2539–2561.

Su, Y.; Han, F.; Harang, R. E.; and Yan, X. 2016. A Fast Kernel for Attributed Graphs. In *Proceedings of the 2016 SIAM Conference on Data Mining*.

Toivonen, H.; Srinivasan, A.; King, R. D.; Kramer, S.; and Helma, C. 2003. Statistical evaluation of the Predictive Toxicology Challenge 2000–2001. *Bioinformatics* 19(10):1183–1193.

Vert, J.-P. 2008. The optimal assignment kernel is not positive definite. *CoRR, abs/0801.4061*.

Vishwanathan, S. V. N.; Schraudolph, N. N.; Kondor, R.; and Borgwardt, K. M. 2010. Graph Kernels. *The Journal of Machine Learning Research* 11:1201–1242.

Wale, N.; Watson, I. A.; and Karypis, G. 2008. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems* 14(3):347–375.

Yanardag, P., and Vishwanathan, S. 2015a. A Structural Smoothing Framework For Robust Graph Comparison. In *Advances in Neural Information Processing Systems*, 2125–2133.

Yanardag, P., and Vishwanathan, S. 2015b. Deep Graph Kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1365–1374.