
Article:
Rossi, Luca, Torsello, Andrea and Hancock, Edwin R orcid.org/0000-0003-4496-2028 (2015) Unfolding Kernel Embeddings of Graphs:Enhancing Class Separation through Manifold Learning. Pattern Recognition. pp. 3357-3370. ISSN: 0031-3203

https://doi.org/10.1016/j.patcog.2015.03.018

Unfolding Kernel Embeddings of Graphs: Enhancing Class Separation through Manifold Learning

Luca Rossi [1*], Andrea Torsello [2], Edwin R. Hancock [3]

[1] *School of Computer Science, University of Birmingham, UK*
[2] *Department of Environmental Science, Informatics, and Statistics, Ca' Foscari University of Venice, Italy*
[3] *Department of Computer Science, The University of York, York YO10 5DD, UK*

Abstract

In this paper, we investigate the use of manifold learning techniques to enhance the separation properties of standard graph kernels. The idea stems from the observation that when we perform multidimensional scaling on the distance matrices extracted from the kernels, the resulting data tends to be clustered along a curve that wraps around the embedding space, a behaviour that suggests that long range distances are not estimated accurately, resulting in an increased curvature of the embedding space. Hence, we propose to use a number of manifold learning techniques to compute a low-dimensional embedding of the graphs in an attempt to unfold the embedding manifold, and increase the class separation. We perform an extensive experimental evaluation on a number of standard graph datasets using the shortest-path [1], graphlet [2], random walk [3] and Weisfeiler-Lehman [4] kernels. We observe the most significant improvement in the case of the graphlet kernel, which fits with the observation that neglecting the locational information of the substructures leads to a stronger curvature of the embedding manifold. On the other hand, the Weisfeiler-Lehman kernel partially mitigates the locality problem by using the node labels information, and thus does not clearly benefit from the manifold learning. Interestingly, our experiments also show that the unfolding of the space seems to reduce the performance gap between the examined kernels.

Keywords: Kernel Learning, Kernel Unfolding, Graph Kernels, Manifold Learning
PACS:

1. Introduction

Graph-based representations have become increasingly popular due to their ability to characterize in a natural way a large number of systems which are best described in terms parts and binary relations. Concrete examples include the use of graphs to represent shapes [5], metabolic networks [6], protein structure [7], and road maps [8]. However, the rich expressiveness and versatility of graphs comes at the cost of added complexity and a reduced toolset of available pattern analysis algorithms. In fact, our ability to analyse data abstracted in terms of graphs is severely limited by the restrictions posed by standard pattern recognition techniques, which require data to be representable in a vectorial form. There are two reasons why graphs are not easily reduced to a vectorial form: First, there is no canonical ordering for the nodes in a graph, unlike the components of a vector. Hence, correspondences to a reference structure must be established as a prerequisite. Second, the variation in the graphs of a particular class may manifest itself as subtle changes in structure. Hence, even if the nodes or the edges of a graph could be encoded in a vectorial manner, the vectors would be of variable length, thus residing in different spaces.

The first 30 years of research in structural pattern recognition have been mostly concerned with the solution of the correspondence problem as the fundamental means of assessing structural similarity [9]. With the similarity at hand, similarity-based pattern recognition techniques such as the nearest neighbour rule can be used to perform recognition and classification tasks, or graphs may be embedded in a low-dimensional pattern space using either multidimensional scaling or alternative non-linear manifold leaning techniques.

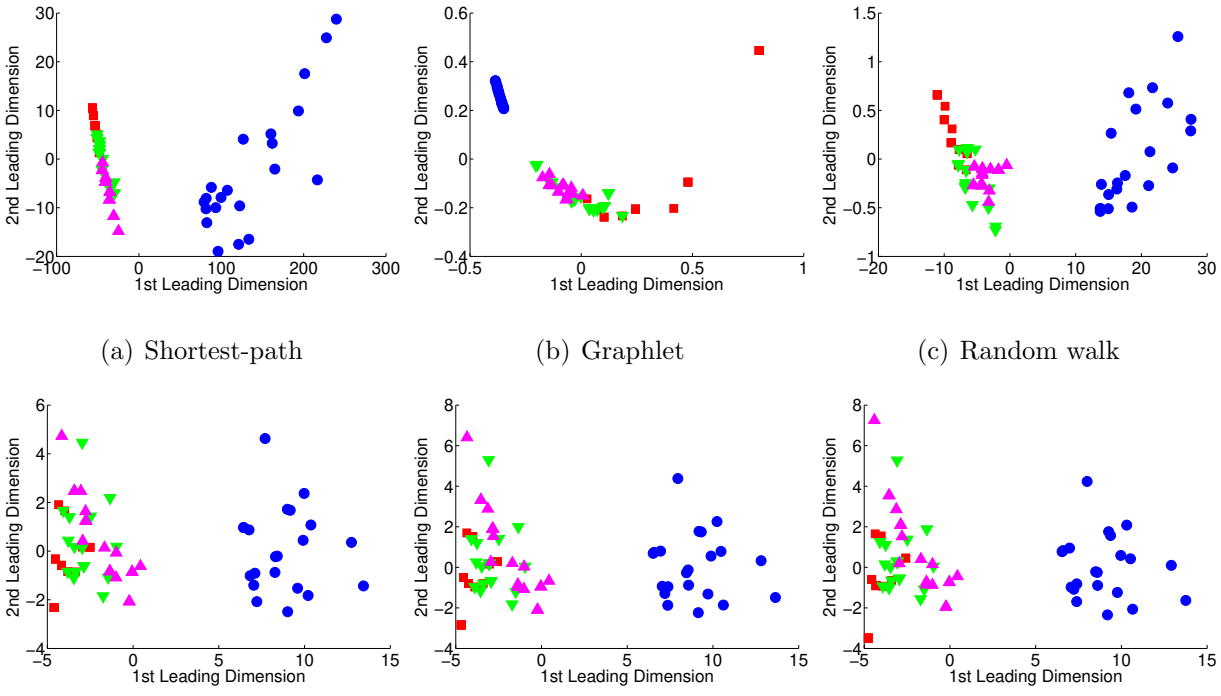
Another alternative is to extract feature vectors from the graphs providing a pattern-space representation. There are a number of ways in which this can be done: One approach is to extract structural or topological features from the graphs under study. Graph spectral features extracted from the eigenvalues and eigenvectors of the adjacency or Laplacian matrices have been shown to be effective here [10, 11]. Again, manifold learning techniques have been used in the literature to provide a way to unfold the pattern space and map the

*Corresponding author. Email address: l.rossi@cs.bham.ac.uk.

data onto low-dimensional spaces where the structural classes are well separated.

The famous kernel trick [12] has shifted the problem from the vectorial representation of data, which now becomes implicit, to a similarity representation. This has allowed standard learning techniques to be applied to data for which no easy vectorial representation exists. More formally, once we define a positive semi-definite kernel $k : X \times X \to \mathbb{R}$ on a set X, there exists a map $\phi : X \to \mathcal{H}$ into a Hilbert space $\mathcal{H}$, such that $k(x, y) = \phi(x)^\top \phi(y)$ for all $x, y \in X$. Also, given the kernel value between $\phi(x)$ and $\phi(y)$ one can easily compute the distance between them by noting that $||\phi(x), \phi(y)||^2 = \phi(x)^\top \phi(x) + \phi(y)^\top \phi(y) - 2\phi(x)^\top \phi(y)$. Thus, any algorithm that can be formulated in terms of dot products between the input vectors can be applied to the implicitly mapped data points through the direct substitution of the kernel for the dot product. For this reason, in recent years pattern recognition has witnessed an increasing interest in structural learning using graph kernels. However, due to the rich expressiveness of graphs, this task has also proven to be difficult, with the problem of defining *complete* kernels, i.e., ones where the implicit map ϕ is injective, sharing the same computational complexity of the graph isomorphism problem [13].

While the graph kernels proposed in the literature provide effective ways to generate implicit embeddings, there is no guarantee that the data in the Hilbert space will exhibit better class separation. This is of course a consequence of the complexity of the structural embedding problem and the limits for efficient kernel computations already analysed by Gärtner et al. [13]. One evidence of this is the fact that the multidimensional scaling embeddings of several graph kernels show the so-called *horseshoe effect* [14] (see Figure 1), i.e., the data tends to cluster tightly along a curve that wraps around the embedding space. This particular behaviour is typically produced by a consistent underestimation of the real distances of the problem, i.e., the geodesic distances on the manifold, and it implies that the data gets placed onto a highly non-linear manifold embedded in the Hilbert space. The horseshoe is in fact the locus of intersection between the manifold and the plane used to visualise the data, and the high curvature is a result of the dimensionality compression on the data, which reduces the degrees of freedom of the points and forces them to cluster along the observed curve. We should also stress that this behaviour can be a consequence



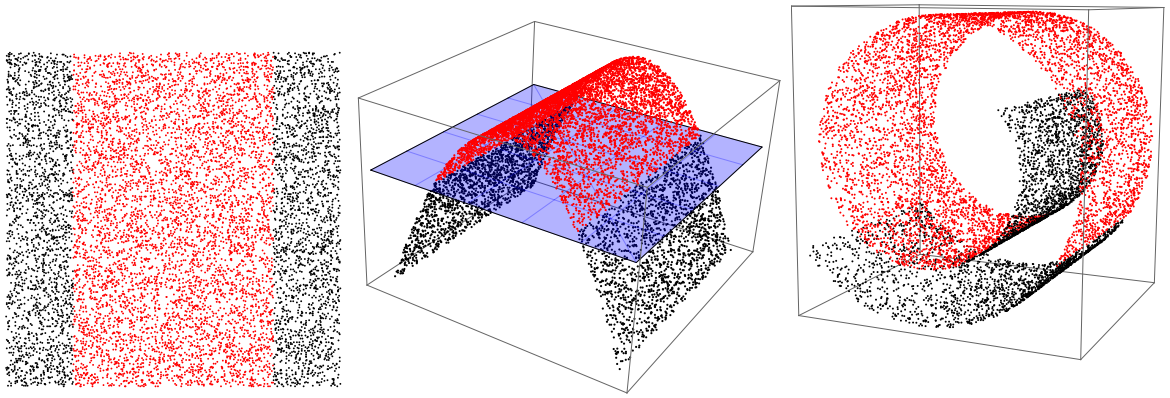
(a) Shortest-path (b) Graphlet (c) Random walk

(d) WL_1 (e) WL_2 (f) WL_3

Figure 1: The MDS embeddings of the shortest-path [1], graphlet [2], random walk [3] and Weisfeiler-Lehman [4] kernels (with $h = 1, 2, 3$) on the COIL dataset.

of kernel normalization, a common procedure through which the data points are projected from the Hilbert space onto the unit sphere. This in turn creates an artificial curvature of the space that can create or exacerbate the observed horseshoe effect. Note, however, that while in general the non-linearity of the mapping is used to improve local class separability, a large global curvature can result in a folding of the manifold that can reduce long range separability.

For this reason, it is natural to investigate the impact of the locality of distance information on the performance of these kinds of kernels. To this end, given a set of graphs, we investigate the use of several manifold learning techniques to embed the graphs onto a low-dimensional vectorial space, in an attempt to unfold the embedding manifold, and increase class separation. More specifically, we investigate the use of four popular non-linear manifold learning techniques, namely Isomap [15], Laplacian Eigenmaps [16], Diffusion Maps [17] and Local Linear Embedding [18]. The selected techniques approach the manifold learn-



(a) Linear embedding (b) Low-curvature embedding (c) High-curvature embedding

Figure 2: Example of reduced linear separability due to high curvature of the embedding. In the linear embedding case the data is not separable, introducing a non-linear mapping to a low-curvature manifold renders the data linearly separable. However, mapping to a high global curvature manifold (e.g. the famous swissroll manifold) results in a reduced linear separability of the data. Furthermore, the higher the curvature the less separable the data is.

ing problem from radically different angles, with Isomap attempting to preserve the global distances and LLE trying to maintain the local neighbourhood geometry.

Experiments on several standard datasets demonstrate that, as expected, the improvement is kernel and dataset dependent, with some kernels and datasets generally gaining very significant improvements in performance, while other not exhibiting significant variation or even modest reduction. Most importantly, the unfolding of the manifold invariably reduces the performance gap between the kernels examined. This suggests that the instances of kernels in the literature do not differ by any intrinsic difference in descriptive power, but in the level of warping of the embedding space. See Figure 2 for an example where the non-linear mapping to a high-curvature manifold reduces the linear separability of data.

The remainder of this paper is organized as follows: Section 2 introduces some related work in graph kernels and manifold learning, while Section 3 illustrates the unwrapping idea and provide a more in-depth description of the kernels and manifold learning techniques used in this study. Section 4 illustrates the experimental results, while Section 5 discusses the main findings and the conclusions are presented in Section 6.

2. Related Works

Many different graph kernels have been proposed in the literature, and they are generally instances of the family of R-convolution kernels first introduced by Haussler [22]. The fundamental idea is that of decomposing two discrete objects and comparing some simpler substructures. Generally speaking, most R-convolution kernels simply count the number of isomorphic substructures in the two graphs being compared, and differ mainly by the type of substructures used in the deconvolution and the algorithms used to count them efficiently.

For example, Kashima et al. [3] propose to count the number of common random walks between two graphs. This kernel, however, is subject to *tottering*. That is, a walk is allowed to go back and forth along a single edge multiple times, thus reducing the discriminative power of the kernel. A solution to this is proposed by Mahe et al. [47], who modify the standard random walk model by preventing the walk from returning to a vertex that has just been visited. Borgwardt and Kriegel [1] measure the similarity by counting the numbers of pairs of shortest paths of the same length in the graphs. Shervashidze et al. [2], on the other hand, count the number of graphlets, i.e. subgraphs with $k = \{13, 4, 5\}$ nodes. More recently, Shervashidze et al. [4] have developed a subtree kernel on subtrees of limited size. They compute the number of subtrees shared between two graphs using the Weisfeiler-Lehman graph invariant. Another kernel based on subtrees enumeration is that of Gaüzere et al.[48]. More specifically, Gaüzere et al. define the similarity between two graphs in terms by decomposing them in bags of treelets, i.e., acyclic and unlabeled trees whose maximal size is equal to 6. Ralaivola et al. [49], on the other hand, develop a series of kernels by exploiting specific structural properties of graphs arising in the domain of organic chemistry. Aziz et al. [23] have defined a backtrackless kernel on cycles of limited length. They compute the kernel value by counting the numbers of pairs of cycles of the same length in a pair of graphs. Costa and Grave [24] have defined a so-called neighbourhood subgraph pairwise distance kernel by counting the number of pairs of isomorphic neighbourhood subgraphs. Recently, Kriege et al [25] counted the number of isomorphisms between pairs of subgraphs, while Neumann et al. [26] have introduced the concept of propagation kernels to handle

partially labelled graphs through the use of continuous-valued node attributes.

One drawback of these kernels is that they neglect locational information for the substructures in a graph. In other words, the similarity does not depend on the relationship between substructures. As a consequence, these kernels cannot establish reliable structural correspondences between the substructures in a pair of graphs. This limits the precision of the resulting similarity measure. To overcome this problem, Fröhlich et al. [27] introduced alternative optimal assignment kernels. Here each pair of structures is aligned before comparison. However, the introduction of the alignment step results in a kernel that is not positive definite in general [28]. The problem results from the fact that alignments are not in general transitive. In other words, if σ is the node-alignment between graph A and graph B, and π is the alignment between graph B and graph C, in general we cannot guarantee that the alignment between graph A and graph C is $\pi \circ \sigma$. On the other hand, when the alignments are transitive, there is a common simultaneous alignment of all the graphs. Under this alignment, the optimal assignment kernel is simply the sum over all the node/edge kernels, and this is positive definite since it is the sum of separated positive definite kernels. While lacking positive definiteness the optimal assignment kernels cannot be guaranteed to represent an implicit embedding into a Hilbert space, they have nonetheless been proven to be very effective in classifying structures. Another example of alignment-based kernels are the edit-distance-based kernels introduced by Neuhaus and Bunke [29]. Here the alignments obtained from graph-edit distance are used to guide random walks on the structures. More recently, a number of quantum walk graph kernels have been proposed in the literature, where the locational information problem is solved by pre-aligning the graph structures [30] or by letting the quantum walk evolve on a suitably defined composite graph structure [31].

Manifold learning is an approach to non-linear dimensionality reduction based on the idea that the dimensionality of many data sets is only artificially high as a consequence of the choice of observables or features. In fact, the data actually resides in a low-dimensional manifold embedded in the high-dimensional feature space, and the manifold may fold or wrap in the feature space so much that the natural feature-space parametrization does not capture the underlying structure of the problem. Manifold learning algorithms attempt to

uncover a non-linear parametrization for the data manifold in order to find a low-dimensional representation of the data that effectively unfolds the manifold and reveals the underlying data structure.

Multidimensional scaling (MDS) [32] is a standard approach for linear dimensionality reduction which aims at findings the rank m projection that best preserves the interpoint distances given by a distance matrix D. To this end, the matrix $M = -\frac{1}{2}HDH$ is first computed, where $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ is the centering matrix, $\mathbf{1}$ denotes the m-dimensional vector of all ones and n is the number of points to embed. Let $M = \Phi\Lambda\Phi^T$, Φ, where Φ is the $n \times n$ matrix $\Phi = (\phi_1|\phi_2|...|\phi_n)$ with the ordered eigenvectors as columns and $\Lambda = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_n)$ is the $n \times n$ diagonal matrix with the ordered eigenvalues as elements. The coordinates of the newly embedded points in m dimensions are then given by the columns of the matrix $X = \Phi_m(\Lambda_m)^{\frac{1}{2}}$, where Φ_m (Λ_m) denotes the submatrix of Φ (Λ) corresponding to the first m columns.

Isomap [15], short for Isometric Feature Mapping, has been one of the first algorithms introduced for manifold learning and has gained significant popularity due to its conceptual simplicity and efficient implementation. In its essence Isomap can be seen as a direct generalization of Multidimensional Scaling (MDS) where the distances are taken to be geodesic distances on the manifold, thus preserving global isometries. The geodesic distances are approximated as the length of the minimal path on a neighbourhood graph. Despite its popularity, the use of geodesic distances make Isomap strongly dependent to the topology of the neighbourhood graph, suffering from short-cutting and other distortion effects. Diffusion Maps [33] attempt to mitigate this problem by substituting geodesic distances with a more robust distance based on the heat diffusion on the graph. The Laplacian Eigenmap draws ideas from spectral graph theory to perform a low-dimensional embedding, and uses the eigenfunctions of the LaplaceBeltrami operator on the manifold as the embedding dimensions.

Locally Linear Embedding (LLE) [18] was introduced at about the same time as Isomap, but is based on a substantially different intuition. Here the manifold is seen as a collection of overlapping coordinate patches. With small neighbourhood sizes and a sufficiently smooth

manifold, the local local geometry of the patches can be considered approximately linear. The idea, then, is to characterize the local geometry of each neighbourhood as a linear function and to find a mapping to a lower dimensional Euclidean space that preserves the linear relationship between a point and its neighbours. Though LLE is based on a different intuition than Laplacian Eigenmap, Belkin and Niyogi [34] were able to show that that the algorithms were equivalent under some conditions.

Maximum Variance Unfolding (MVU) [35], also referred as Semi-Definite Embedding (SDE), attempts to explicitly unfold the manifold by maximizing the variance of the embedded points while keeping fixed the local distances, and thus the manifold topology. The resulting optimization problem is solved through semi-definite programming, which makes it computationally demanding when applied to large sets of points.

Generally speaking, the above techniques can be classified as either global or local approaches, according to the nature of the geometrical information that they attempt to preserve. As an example, Isomap attempts to minimise the embedding distortion, and as such it tries to maintain the global distances between the points, eventually causing local distortions. LLE and Laplacian Eigenmap, on the other hand, attempt to preserve the local geometry of the data, essentially trying to compute a conformal map between the original manifold and the embedding space. Finally, MVU takes into account both local and global geometry preservation, by maximizing the distances between the non-neighbouring points while keeping the distances between neighbours fixed.

The most closely related works to the present paper are [19, 21]. In [19] Ham et al. study the relationship between Isomap, Laplacian Eigenmaps and Local Linear Embedding and their link to kernel PCA [20]. However, while Ham et al. point out a connection between manifold learning and kernel methods, our work is significantly different from theirs as our inputs are (dis)similarities, i.e., kernel values, as opposed to vectorial data. Moreover, our aim is that of using manifold learning to smooth the implicit underlying space to increase the performance of the original kernels. In Riesen et al. [21] the authors apply Isomap on a set of graphs given the edit distance between them, and a graph kernel is constructed by transforming the distance information into kernel values. Our work is similar in the sense

that we use manifold learning to smooth the graph embedding space. However, our work differs from that of Riesen et al. [21] since we focus on the distances induced by standard graph kernels, rather than the graph edit distance. Additionally, we propose a simple yet effective framework to select the optimal embedding parameters for the manifold learning techniques.

3. Graph Kernels and Manifold Learning

As stated in the Introduction, the application of multidimensional scaling to the distances in the implicit Hilbert space, obtained from the kernel matrix of the most widely used R-convolution type graph kernels in the literature, often exhibits the so-called horseshoe effect, i.e., a data distribution concentrated along a highly curved line or manifold. This is an indication of a consistent underestimation of the long-range distances consistent with the properties of these kernels. In fact, R-convolution kernels typically count the number of isomorphic substructures in the decomposition of the two graphs, without taking into consideration locational information for the substructures in a graph. In other words, substructures are similar regardless of their relative position in the graphs. When the graphs are very similar, the similar substructures that are locationally consistent dominate the convolution. However, when the graphs are very dissimilar, several similar small substructures can appear all over the two graphs simply due to the statistics of random graphs, and the smaller and simpler the substructures in the decomposition, the more likely we are to find them in various locations of the two structures. This is a typical aperture problem, where the smaller the probing size, the more uniform the world appears, and the more likely we are to find similarities due to random fluctuations.

Note that this does not mean that there are more common small substructures than common large substructures. Indeed the combinatorial expansion of the possible correspondences will result in an increase of the number of matches with the size of the probing structure. Instead, it is the proportion of correct matches over the total number of possible correspondences of the same size that reduces as the size increases. In other words, the comparison is always among substructures of the same size, and in this case the specificity

of the structure does indeed increase with size. As an example, a single node can have n possible correspondences, but 100% of them will be a match. On the other hand there are 3 possible graphs with 3 nodes; one with zero arcs, one with one arc, one with two arcs and the complete graph. There are 6 possible permutations from any of these structures to any group of three nodes in a graph. In all cases a permutation can be a match only if it maps to a substructure with the same number of arcs. Each of the permutations mapping the structures with zero or 3 arcs to a similar structure will be an exact match. On the other hand, in the case of the structures with one or two arcs, only two of the six possible permutations will result in an exact match. In other words, only 2/3 of the possible permutations give an exact match. Hence, the structure with 3 nodes gives fewer ambiguous matches than a single node, and also results in a smaller proportion of incorrect matches.

Further, the lack of a locality condition and the consequent summation over the entire structure amplifies the effect of these random similarities. This gives rise to a lower bound on the kernel value that is a function only of the random graph statistics. The effect of this non-negligible lower bound due to random fluctuations is a consistent reduction in the estimated distances for dissimilar graphs. This does not affect the metric properties of the kernel distances, as the triangle inequality is not violated by a reduction of the length of the sides of a triangle, nor does it affect embeddability, since it does not affect the positive definiteness of the kernel, but it does add a strong curvature to the embedding manifold, which can effectively fold on itself, and it does increase the effective dimensionality of the embedding.

Note that the added locational constraints in correspondence kernels [27, 29] can effectively reduce this phenomenon since the contribution of random similarities is not summed over all the structures, but it comes at the cost of positive definiteness and it still relies on correspondences that are not reliable for very dissimilar graphs, resulting in distortions that are not guaranteed to be only contractions, and that can affect also the metric properties by breaking the triangle inequality. To this effect, it is worth noting that the Weisfeiler-Lehman kernels make use of node labels, thus improving node localization in the evaluation of the kernel. As a consequence, we can see in Figure 1 that the horseshoe effect is reduced, even

though we have used a very non-specific descriptor such as the node degree. More generally, we should stress that taking node labels into account can help to reduce the locational problem. This in turn has the effect of increasing the discriminative power of the kernel and of reducing the observed horseshoe effect. However, node labels may not always be available. Finally, kernel normalization, which is a common kernel preprocessing technique where the data is projected from the Hilbert space onto the unit sphere, can artificially increase the curvature of the space, thus leading to the observation of a stronger horseshoe effect.

In this paper, we intend to study the use of a number of manifold learning techniques on R-convolution type graph kernels. Note that while graphs can indeed reside in highly non-smooth spaces, here we assume that it is the kernel embeddings that reside in a manifold. Thus, given this setting, we try to increase class separation by embedding the graphs onto a low-dimensional vectorial space, in an attempt to unfold the embedding manifold. To this end, in the next section we illustrate an approach to select the embedding parameters that optimally unfold the kernel which is based on a cross-validation approach maximizing linear separability of data, then we introduce the graph kernels and manifold learning techniques used in our investigation.

3.1. Enhancing Graph Kernels through Manifold Learning

We propose to unfold the manifold and increase the class separation by applying an optimal manifold learning process to the distance matrix. Given a set of n graphs $\mathcal{G} = \{G_1, \cdots, G_n\}$ and their kernel matrix $K = (k_{ij})$ we compute the distance matrix $D = (d_{ij})$ with $d_{ij} = \sqrt{k_{ii} + k_{jj} - 2k_{ij}}$. Then we apply the selected manifold learning algorithm and learn a C-SVM classifier with a linear kernel. Finally we select the optimal set of parameters by selecting in cross validation for the ν-tuple of parameters which maximises

$$\arg\max_{p_1,\ldots,p_{\nu-1}} \max_{C} \alpha \tag{1}$$

where α is the 10-fold cross validation accuracy of the C-SVM, C is the regularizer constant, and $p_1, ..., p_{\nu-1}$ are the parameters of the chosen manifold learning technique. Multi-classification tasks are solved using majority voting over the one-vs-one C-SVM classifiers.

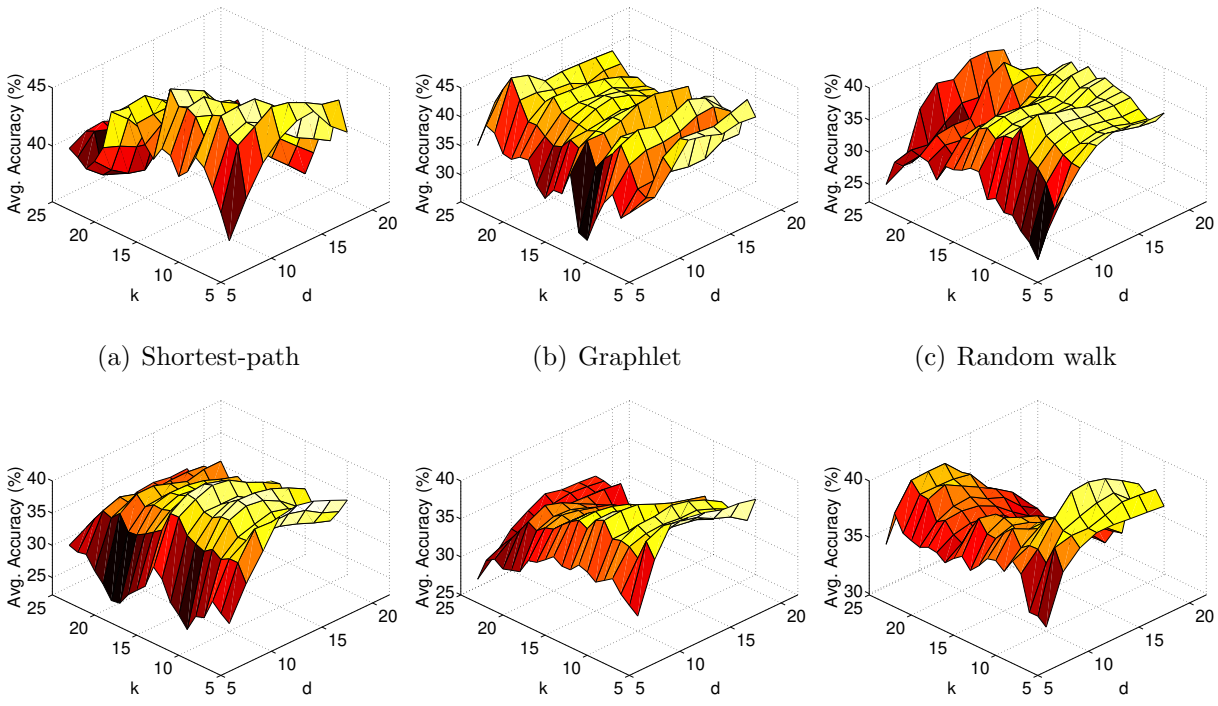
The parameters $p_1, \ldots, p_{\nu-1}$ are dependent on the manifold learning techniques, however there is a recurring parameter shared by every approach that requires a discrete neighbourhood specification: In these cases we construct a k-nearest neighbourhood graph from the distances d_{ij}, then we render the neighbourhood graph connected through a minimum spanning tree among the connected components. More precisely, let C_i and C_j denote two connected components, whose elements encode the graphs in $\mathcal{G}$. Then, the length of the edge between C_i and C_j is the minimum length over the edges connecting elements of C_i and C_j, i.e., $l_{C_iC_j} = \min_{i \in C_i, j \in C_j} l_{ij}$. Finally, where the algorithm requires an undirected neighbourhood graph, the neighbourhood relation is symmetrized by placing an undirected edge every time there is a directed one in either direction, i.e., $(i,j) \in \tilde{N} \iff (i,j) \in N \vee (j,i) \in N$, where N is the directed neighbourhood relation and $\tilde{N}$ is the undirected one. With this construction the only neighbourhood parameter is k.

Finally, given the set of optimal parameters $\{p_1^{opt}, ..., p_{\nu-1}^{opt}, C^{opt}\}$, we compute the embedding of the test graphs using the selected manifold learning technique and we evaluate the class separation on the embedded data using a binary C-SVM with a linear kernel and $C = C^{opt}$.

Figure 3 shows a sample of the 10-fold cross validation accuracy (with $C = C^{opt}$) for the Laplacian Eigenmap embeddings of the graph kernels on the Shock dataset (see Section 4 for a description of the dataset), as the number of the nearest neighbours k and the embedding dimension d vary. While the value of k seems to have a limited impact on the average classification accuracy, we observe that the optimal performance is achieved for values of d between 10 and 15. Moreover, note that the optimal value of d is mostly independent from both the neighborhood construction and the kernel used. Finally, we should stress that the selection of an optimal neighborhood size k is a central issue in manifold learning [37]. As a consequence, the limited impact of k in our framework can be seen as an advantage.

3.2. Graph Kernels

We focus our analysis on four standard graph kernels, namely the shortest-path kernel [1], the graphlet kernel [2], the random walk kernel [3] and the Weisfeiler-Lehman ker-



(a) Shortest-path (b) Graphlet (c) Random walk

(d) WL_1 (e) WL_2 (f) WL_3

Figure 3: 3D plot of the 10-fold cross validation accuracy (with $C = C^{opt}$) of the graph kernels on the Shock dataset as the number of the nearest neighbours k and the embedding dimension d vary. The embedding was computed using Laplacian Eigenmap.

nel [4]. These are all instances of R-convolution kernels, where the common idea is that of decomposing two discrete objects and to compare some simpler substructures.

3.2.1. Shortest-path Kernel

Let $G = (V, E)$ be an undirected graph and let S be its shortest-path graph, i.e., a graph defined over the same set of nodes V where there exist an edge between two nodes if these are connected by a walk in G. The shortest-path kernel counts the numbers of common shortest-path in two graphs G_1 and G_2. To this end, the two original graphs are first transformed into the corresponding shortest-path graphs $S_1 = (V_1, E_1)$ and $S_2 = (V_2, E_2)$. Then, the shortest-path kernel is defined as

$$k_{sp}(S_1, S_2) = \sum_{e_1 \in E_1} \sum_{e_2 \in E_2} k(e_1, e_2) \tag{2}$$

where k is a positive definite kernel that measures the similarity between the shortest paths corresponding to the edges e_1 and e_2. Moreover, the kernel k can be modified to handle node labels. In that case, two shortest-path are matched if the sequence of labels along the two paths are the same.

In their original implementation Borgwardt et al. [1] propose to use the Floyd-Warshall [36] to compute the shortest-path, yielding a $O(|V|^4)$ computational complexity.

3.2.2. Graphlet Kernel

The graphlet kernel measures the similarity between a pair of unlabelled graphs by counting the number of common graphlets of size $k = \{13, 4, 5\}$ in the two graphs. Let N_k denote the number of graphlets of size k. Given a graph $G = (V, E)$, let f_G be the vector of length N_k where the ith component is the frequency of occurrence of the graphlet g_i in G. For a pair of graphs G_1 and G_2, the k-graphlet kernel is defined as

$$k_{gr}(G_1, G_2) = f_{G_1}^{\top} f_{G_2} \tag{3}$$

Note that since there are $\binom{|V|}{k}$ subgraphs of size k in a graph with n nodes, for each graph the of size $|V|$ the computation of f_G has complexity $O(|V|^k)$. However, in the case of graphs with bounded degree d, i.e., graphs where no vertex has more than d neighbours, the complexity can be shown to be $O(|V|d^{k-1})$. Finally, unlike the shortest-path kernel, the graphlet kernel does not take node labels into account.

3.2.3. Random Walk Kernel

The random walk kernel is based on the idea of counting the number of matching walks between a pair of graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Similarly to the shortest-path kernel, the random walk kernel can handle node labels. In this work we use the kernel as defined by Gärtner et al. [13]. To this end, we first introduce the direct product graph $G_\times = (V_\times, E_\times)$, where $V_\times = \{(v_1, v_2) : v_1 \in V_1, v_2 \in V_2\}$ and $E_\times = \{((v_1, v_2), (w_1, w_2)) : (v_1, w_1) \in E_1, (v_2, w_2) \in E_2\}$. Let $A_\times$ denote the adjacency matrix of $G_\times$. Then, the random

walk kernel between G_1 and G_2 is defined as

$$k_{rw}(G_1, G_2) = \sum_{i=1}^{|V_1|} \sum_{j=1}^{|V_2|} \sum_{k=0}^{\infty} \mu^k \left[A_\times^k\right]_{ij} \tag{4}$$

where μ is a fixed decay factor that has the effect of reducing the contribution of longer walks, in addition to allowing the summation to converge. Note that performing a walk on $G_\times$ is equivalent to performing simultaneous walks on G_1 and G_2 [38]. Therefore, the entries of $A_\times^k$ can be seen as the enumerating the simultaneous random walks of length k on G_1 and G_2. In its original formulation the random walk kernel has computational complexity $O(\max(|V_1|, |V_2|)^6)$, however in [39] a more efficient implementation is proposed which reduces the complexity to $O(\max(|V_1|, |V_2|)^3)$.

One drawback of the random walk kernel is that it is known to be subject to *tottering* [47]. That is, the walker can follow a redundant path $h = v_1, \cdots, v_n$ with $v_i = v_{i+2}$, which can decrease the ability of the random walk kernel to characterise the graph structure and can result in an artificially high similarity. For example, the existence of a walk of length 2 can indicate the presence of a path connecting 3 vertices as well as a walk back and forth along a single edge. A number of works have proposed to solve the tottering problem by forbidding repeated visits of the same two nodes [47, 23].

3.2.4. Weisfeiler-Lehman Kernel

The Weisfeiler-Lehman kernel is a state-of-the-art graph kernel introduced by Shervashidze et al. [4]. The kernel enumerates the common subtrees between two graphs by using the Weisfeiler-Lehman test of graph isomorphism. Given two attributed graphs $G_1 = (V_1, E_1, l_1)$ and $G_2 = (V_2, E_2, l_2)$, where l_i denote the set of labels of V_i, the idea is that of associating with each vertex v a multiset label based on the labels of the neighbours of v. This procedure is iterated a fixed number of times, and at each step the resulting multisets are sorted and compressed to generate new vertex labels. Let G_1^i and G_2^i denote the graphs G_1 and G_2 after h iterations of vertices relabeling procedure. The Weisfeiler-Lehman is then defined as

$$k_{wl}^h(G_1, G_2) = \sum_{i=0}^{h} k_\delta(G_1^i, G_2^i) \tag{5}$$

where $k_\delta(G_1^i, G_2^i)$ is a positive definite kernel which enumerates the pairs of vertices in G_1^i and G_2^i that share the same label. The computation of the Weisfeiler-Lehman with h iterations has complexity $O(hm)$, where m is the number of edges of the graph.

3.3. Manifold Learning

In this paper we investigate the use of four standard non-linear manifold learning techniques, namely Isomap [15], Laplacian Eigenmap [16], Diffusion Maps [17] and Local Linear Embedding [18]. In the following subsection we assume that we want to embed n points $\mathbf{x_i}$ onto an m-dimensional space, with $m < n$. Note that these techniques can be easily extended to the case where the input is a pairwise distance matrix rather than a vectorial representation of the data points. Thus, although graphs are not easily reduced to a vectorial form, it is possible to apply manifold learning to a set of graphs by first computing a kernel (distance) matrix over the dataset.

3.3.1. Isomap

Isomap can be viewed as an extension of MDS where the original distance between the points is replaced with their geodesic distance [15]. Isomap assumes that only the pairwise distances between neighbouring point are known, and uses the shortest-path distance to estimate the distances between all the other point. More specifically, Isomap starts by looking for the k-nearest neighbours of each point in the dataset. Then, the k-neigbhours graph is constructed, where each point is connected with its k-nearest neigbhours. The geodesic distance between two points is estimated as the shortest-path distance between the corresponding nodes in the graph. Finally, a low-dimensional projection is generated by applying MDS on the matrix of geodesic distances.

Despite its popularity, Isomap suffers from a number of drawbacks. In fact, the use of geodesic distances makes Isomap strongly dependent on the topology of the neighbourhood graph, and it can lead to problems such as shortcutting or other distortion effects.

3.3.2. Diffusion Maps

Diffusion Maps attempt to mitigate the shortcomings of Isomap by replacing the geodesic distances with a more robust distance based on the heat diffusion on the graph [17]. The first stage requires computing the distance matrix

$$M_{ij} = e^{-\frac{||x_i - x_j||}{2\sigma^2}} \tag{6}$$

The matrix M is then normalised to obtain the transition matrix P, such that P_{ij} is the probability of transitioning from $\mathbf{x_i}$ to $\mathbf{x_j}$ in one time step. The coordinates of the ith point can then be obtained from the eigendecomposition of P as $\tilde{\mathbf{x}}_\mathbf{i} = \{\lambda_1\phi_{i1}, \lambda_2\phi_{i2}, ..., \lambda_m\phi_{im}\}$, where $P = \Phi\Lambda\Phi^T$, Φ is the $n \times n$ matrix $\Phi = (\phi_1|\phi_2|...|\phi_n)$ with the ordered eigenvectors as columns and $\Lambda = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_n)$ is the $n \times n$ diagonal matrix with the ordered eigenvalues as elements.

Note that the matrix P depends on the choice of the time parameter t, which until now we assumed to be equal to 1. However, the diffusion process can be stopped at any a generic time $t \geq 1$. In this case, P^t_{ij} gives the probability of transitioning from i to j in t time steps. In a sense, the Diffusion Map at time t takes into account all the paths of length t between each pair of points, and for this reason it is extremely robust to noise, as opposed to the geodesic distance of Isomap.

3.3.3. Laplacian Eigenmap

Similarly to Isomap, the Laplacian Eigenmap method starts by constructing a weighted graph over the points, where the set of edges connects neighbouring nodes. However, while Isomap tries to preserve the structure by preserving the geodesic distance, i.e., it tries to preserve the global structure of the data, Laplacian Eigenmap ensures that points which are close to each other on the manifold are mapped close to each other in the low-dimensional embedding space, i.e., it tries to preserve local distances.

Let M denote the $n \times n$ neighborhood (similarity) matrix for the original points. In particular, in [16] M is defined as either the Gaussian kernel or the matrix whose element (i, j) is 1 if $\mathbf{x_i}$ and $\mathbf{x_j}$ are k-nearest neighbours. It can be shown that finding the embedding which

preserves the distances between neighbouring points is equivalent to solving the generalised eigenproblem

$$(\Delta - M)\phi_i = \lambda_i S\phi_i \tag{7}$$

with eigenvalues λ_i, eigenvectors ϕ_i and Δ a diagonal matrix whose ith diagonal entry is $\Delta_{ii} = \sum_{j=1}^{n} M_{ij}$. Then, the m dimensional embedding is given by the first m eigenvectors ϕ_i, where the eigenvector corresponding to the zero eigenvalues is left out.

3.3.4. Local Linear Embedding

Like Laplacian Eigenmap, Local Linear Embedding is also a local approach that attempts to preserve the local geometry of each point. The underlying assumption of LLE is that a point and its neighbours in the original space should lie on or close to a locally linear patch of the manifold. Hence, it is possible to reconstruct each point as a linear combination of its neighbours. In particular, the optimal coefficients w_{ij} are found by minimising the reconstruction error

$$\mathcal{E}(W) = \sum_{i=1}^{n} \left(\mathbf{x_i} - \sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{x_j} \right)^2 \tag{8}$$

where $\mathcal{N}(i)$ denotes the neighbourhood of $\mathbf{x_i}$.

For each point $\mathbf{x_i}$ a low-dimensional embedding $\tilde{\mathbf{x}}_\mathbf{i}$ is then computed such that the coordinates of each point are still described by the same linear combination of its neighbours. That is, given the matrix of optimal coefficients W, we look for the set of points $\tilde{\mathbf{x}}_\mathbf{i}$ that minimise the cost function

$$\mathcal{C}(W) = \sum_{i=1}^{n} \left(\tilde{\mathbf{x}}_\mathbf{i} - \sum_{j \in \mathcal{N}(i)} w_{ij} \tilde{\mathbf{x}}_\mathbf{j} \right)^2 \tag{9}$$

The optimal m-dimensional embedding is then obtained from the m eigenvectors of $M = (I - W)^\top (I - W)$ corresponding to the m lowest eigenvalues, where I denotes the identity matrix of size n, and the eigenvalues $\lambda_0 = 0$ and its corresponding eigenvector ϕ_0 are not used.

Datasets	MUTAG	PPIs	COIL	Shock	Reeb
Max # vertices	28	232	35	33	220
Min # vertices	10	3	3	4	41
Avg # vertices	17.93	109.60	12.43	13.16	95.43
# graphs	188	86	84	150	300
# classes	2	2	4	10	15

Table 1: Information on the graph datasets

4. Experimental Results

We perform extensive experiments on 5 different graph datasets (see Table. 1). For each of the kernels of Subsection 3.2 and the manifold learning techniques reviewed in Subsection 3.3, we evaluate the class separation before and after computing the embedding as described in Subection 3.1. More specifically, we first divide the data into a training and test set. In particular, we assign 80% of the data to the training set and the remaining 20% to the test set. We repeat the whole procedure 100 times and we report the results in terms of average classification accuracy, $\pm$ standard error.

Note that in these experiments we do not include the node labels. In other words, the kernels consider only the structural information of the graphs. In the case of the Weisfeiler-Lehman, however, each node is labelled with its degree. Finally, the implementation of the graphlet kernel used in this paper counts instances of graphlets of size 3.

4.1. Datasets

MUTAG: The MUTAG is a dataset of 188 mutagenic aromatic and heteroaromatic compounds labelled according to whether or not they have a mutagenic effect on the Gram-negative bacterium *Salmonella typhimurium* [40].

PPIs: The PPI dataset consists of protein-protein interaction (PPIs) networks related to histidine kinase [41]. Histidine kinase is a key protein in the development of signal transduction. If two proteins have direct (physical) or indirect (functional) association, they are connected by an edge. Here we consider the PPIs from two different groups: 40 PPIs from *Acidovorax avenae* and 46 PPIs from *Acidobacteria*.

COIL: The COIL dataset consists of 4 objects from the COIL-20 [44] dataset. For each object, we consider 21 views obtained from equally spaced viewing directions over 105°. For each image, a graph is obtained as the Delaunay triangulation of the Harris corner points. Each vertex is used as the seed of a Voronoi region, which expands radially with a constant speed. The linear collision fronts of the regions delineate the image plane into polygons, and the Delaunay graph is the region adjacency graph for the Voronoi polygons.

Shock: The Shock dataset consists of graphs from a database of 2D shapes [42]. Each graph is a skeletal-based representation of the differential structure of the boundary of a 2D shape. There are 150 graphs divided into 10 classes, each containing 15 graphs.

Reeb: The Reeb dataset consists of a set of adjacency matrices associated to the computation of reeb graphs of 3D shapes [43]. The original 3D shapes dataset comes from the SHREC database, and consists of 15 classes with 20 shapes each, for a total of 300 objects.

4.2. Experimental Results

Table 2 and Fig. 4 show the average classification accuracy for our approach in the case where the embedding parameters are chosen so as to maximise the 10-fold cross validation accuracy on the training set, as explained in Section 3.1. In Fig. 4 the data is displayed as a series of histograms of the average accuracy. Each individual histogram shows the average accuracy for the different embedding methods used. The columns of the figure repeat the histograms for the different kernels studied, and the different rows repeat the histograms for the different datasets. In Table 2 SP is the shortest-path kernel [1], RW is the random walk kernel [13], GR is the graphlet kernel [2], while WL^h denotes the Weisfeiler-Lehman kernel with h iterations [4]. For each kernel and dataset, the best performing manifold learning technique is highlighted in bold, and the best performing method over the dataset is underlined. Moreover, for each dataset and each kernel combination, whenever manifold learning leads to a statistically significant increase or decrease of the performance with respect to the original kernel the result is highlighted with a $*$ symbol.

Our first observation is that the accuracy of the graphlet kernel is consistently higher when we apply manifold learning. In fact, in 16 out of 20 cases the accuracy of the graphlet

Kernel	MUTAG	PPI	COIL	Shock	Reeb
SP	$\mathbf{86.68 \pm 0.51}^*$	73.35 ± 1.03	$\underline{\mathbf{86.94 \pm 0.74}}^*$	$\mathbf{43.50 \pm 0.79}$	55.37 ± 0.53
SP_{ISO}	85.59 ± 0.54	71.65 ± 1.04	84.00 ± 0.84	38.73 ± 0.83	47.15 ± 0.55
SP_{DIF}	84.16 ± 0.59	$\underline{\mathbf{81.65 \pm 1.08}}^*$	85.75 ± 0.80	41.53 ± 0.75	$\mathbf{56.05 \pm 0.51}^*$
SP_{LAP}	83.97 ± 0.58	65.00 ± 1.37	80.69 ± 0.78	43.30 ± 0.78	44.03 ± 0.50
SP_{LLE}	84.97 ± 0.61	78.71 ± 1.18	86.00 ± 0.82	42.90 ± 0.74	54.48 ± 0.52
GR	81.27 ± 0.60	50.71 ± 0.60	82.25 ± 0.74	28.53 ± 0.61	28.13 ± 0.38
GR_{ISO}	82.19 ± 0.61	71.12 ± 1.10	79.38 ± 0.89	38.27 ± 0.72	$\mathbf{39.53 \pm 0.55}^*$
GR_{DIF}	79.51 ± 0.64	$\mathbf{76.53 \pm 1.11}^*$	80.88 ± 0.79	43.40 ± 0.84	38.38 ± 0.50
GR_{LAP}	$\mathbf{83.00 \pm 0.63}^*$	60.35 ± 1.28	79.25 ± 0.94	38.83 ± 0.81	30.25 ± 0.42
GR_{LLE}	81.73 ± 0.60	69.29 ± 1.10	$\mathbf{83.88 \pm 0.78}^*$	$\mathbf{46.00 \pm 0.86}^*$	34.18 ± 0.46
RW	$\mathbf{87.76 \pm 0.52}^*$	61.94 ± 1.07	82.50 ± 0.82	36.90 ± 0.68	54.23 ± 0.50
RW_{ISO}	86.62 ± 0.57	66.29 ± 1.15	$\mathbf{84.75 \pm 0.89}^*$	31.93 ± 0.77	31.12 ± 0.42
RW_{DIF}	84.16 ± 0.59	$\mathbf{71.71 \pm 1.02}^*$	80.06 ± 0.78	41.17 ± 0.83	$\mathbf{54.73 \pm 0.55}$
RW_{LAP}	86.59 ± 0.50	63.76 ± 1.17	80.75 ± 0.90	39.23 ± 0.85	31.90 ± 0.47
RW_{LLE}	85.24 ± 0.52	70.35 ± 1.02	81.44 ± 0.99	$\mathbf{44.63 \pm 0.87}^*$	53.32 ± 0.52
WL^1	$\mathbf{87.05 \pm 0.52}$	72.18 ± 0.99	$\mathbf{80.81 \pm 0.85}$	36.60 ± 0.70	$\mathbf{51.78 \pm 0.57}$
WL^1_{ISO}	85.76 ± 0.53	71.29 ± 1.06	77.12 ± 0.97	37.53 ± 0.72	49.60 ± 0.51
WL^1_{DIF}	81.27 ± 0.58	$\mathbf{78.88 \pm 1.07}^*$	80.69 ± 0.83	32.93 ± 0.76	51.57 ± 0.52
WL^1_{LAP}	86.68 ± 0.48	67.65 ± 1.18	74.62 ± 1.08	$\underline{\mathbf{50.90 \pm 0.77}}^*$	42.93 ± 0.54
WL^1_{LLE}	84.95 ± 0.57	73.76 ± 1.16	78.25 ± 0.93	47.80 ± 0.97	50.72 ± 0.55
WL^2	$\underline{\mathbf{88.89 \pm 0.47}}^*$	79.12 ± 0.99	$\mathbf{80.31 \pm 0.74}$	44.17 ± 0.72	$\mathbf{60.07 \pm 0.60}^*$
WL^2_{ISO}	86.84 ± 0.57	75.76 ± 1.06	77.88 ± 0.95	40.77 ± 0.80	51.03 ± 0.56
WL^2_{DIF}	85.54 ± 0.59	$\mathbf{79.94 \pm 0.97}$	79.88 ± 0.92	36.53 ± 0.68	51.23 ± 0.56
WL^2_{LAP}	86.27 ± 0.52	70.82 ± 1.06	74.06 ± 0.98	$\mathbf{46.03 \pm 0.78}^*$	49.15 ± 0.55
WL^2_{LLE}	87.11 ± 0.51	75.59 ± 1.00	80.31 ± 0.87	44.13 ± 0.76	54.52 ± 0.58
WL^3	$\mathbf{88.16 \pm 0.49}$	79.94 ± 1.03	$\mathbf{81.06 \pm 0.83}^*$	43.83 ± 0.70	$\underline{\mathbf{60.15 \pm 0.58}}^*$
WL^3_{ISO}	88.00 ± 0.53	74.18 ± 1.00	77.06 ± 0.99	40.40 ± 0.73	53.70 ± 0.53
WL^3_{DIF}	87.03 ± 0.58	$\mathbf{80.41 \pm 0.86}$	78.69 ± 0.87	36.17 ± 0.65	52.03 ± 0.59
WL^3_{LAP}	88.08 ± 0.55	69.65 ± 1.17	75.06 ± 1.10	$\mathbf{44.67 \pm 0.84}$	52.25 ± 0.50
WL^3_{LLE}	86.35 ± 0.57	78.06 ± 0.96	79.44 ± 0.91	43.47 ± 0.73	55.87 ± 0.58

Table 2: Maximum classification accuracy ($\pm$ standard error) for the kernels on the graph datasets. For each kernel and dataset, the best performing manifold learning technique is highlighted in bold, the best performing method over the dataset is underlined and statistically significant results are marked by a $*$.

kernel is significantly higher or equal to that of the original kernel after manifold learning. Over all the 5 datasets the highest accuracy for this kernel is achieved with the application of manifold learning. Moreover, the increase in classification accuracy outweighs the loss in performance that we observe in those situations where our approach performs worse than

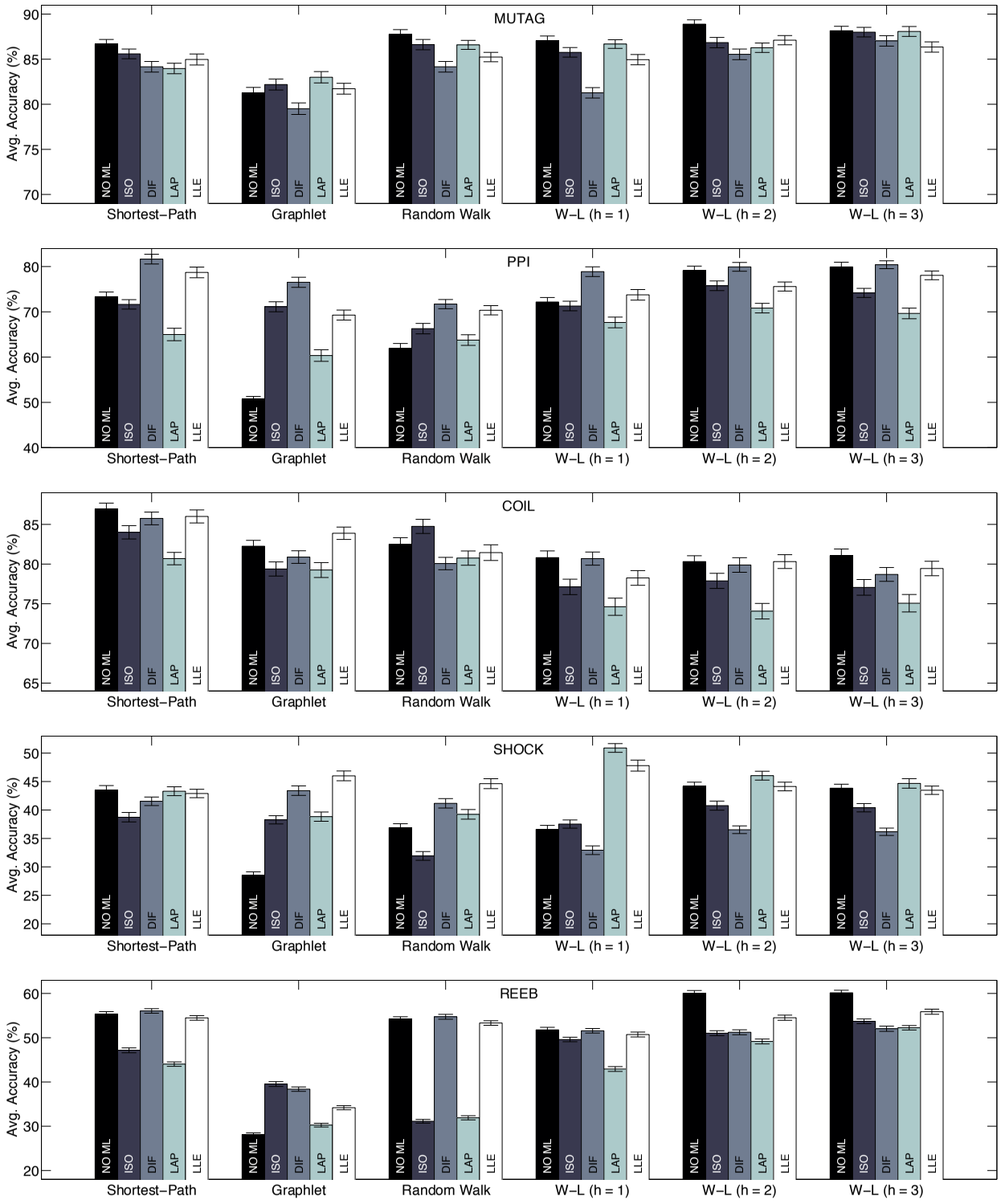
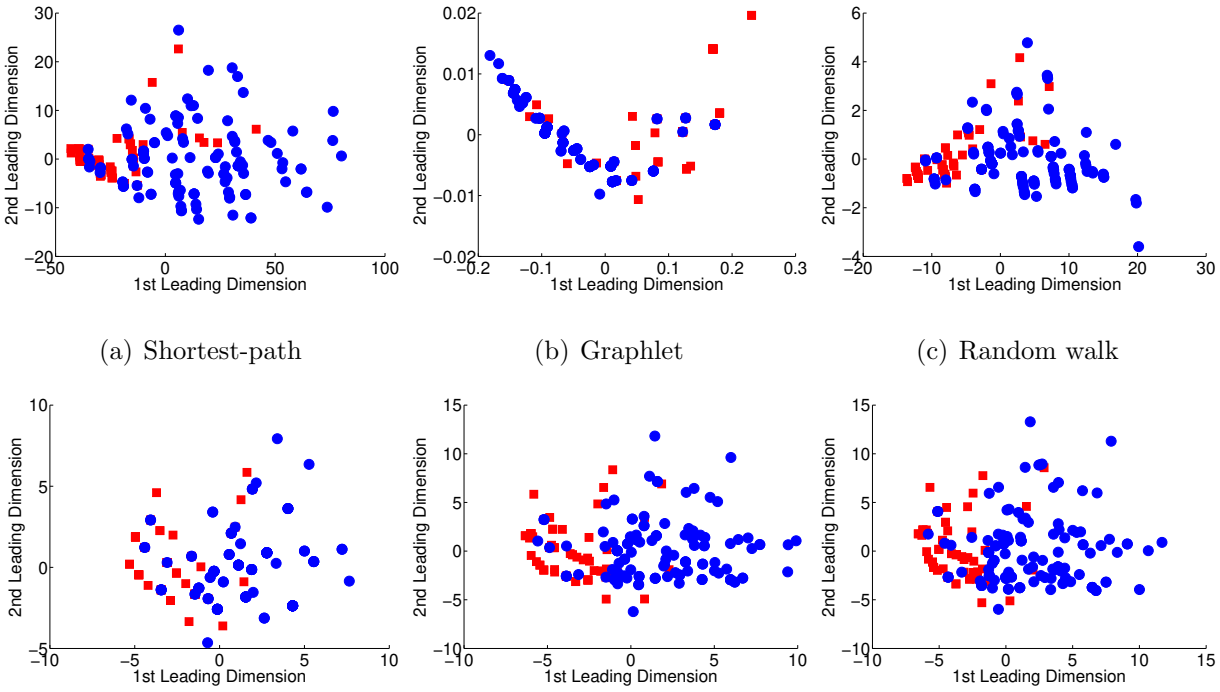


Figure 4: Bar plot showing the classification accuracy (± standard error) reported in Table 2.



(a) Shortest-path (b) Graphlet (c) Random walk

(d) WL_1 (e) WL_2 (f) WL_3

Figure 5: The MDS embeddings of the shortest-path [1], graphlet [2], random walk [3] and Weisfeiler-Lehman [4] kernels (with $h = 1, 2, 3$) on the MUTAG dataset.

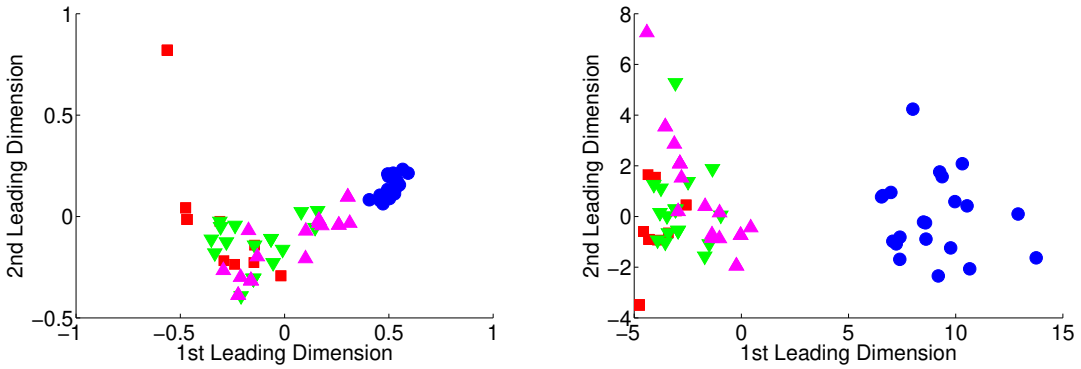
the original kernel. Thus, although it is not clear which manifold learning technique should be preferred, i.e., the optimal technique varies across the 5 datasets, the experimental results suggest that it is generally a good idea to apply our approach when using a graphlet kernel.

The random walk kernel also seems to benefit from manifold learning, albeit to a lesser extent. In particular, we observe the best improvement on the PPI and Shock datasets. However, in contrast to the graphlet kernel, our approach can sometimes lead to a marked decrease in accuracy, as in the case of the Reeb dataset. The shortest-path kernel and the Weisfeiler-Lehman kernels show a similar trend, with limited but sometimes significant improvements. This is the case for the shortest-path kernel on the PPI dataset and the Weisfeiler-Lehman kernel ($h = 1$) on the Shock dataset. In general, none of these kernels shows consistent results in terms of which manifold learning technique to use to enhance class separation. Interestingly, however, we see that on the PPI datasets each of the kernels considered in this study achieve the highest performance when Diffusion Maps are applied.

The above observations seem to support the intuition that those kernels which are based on the enumeration of small and simple substructures, such as graphlets, are more likely to suffer from locality problems. The shortest-path and random walk kernels, on the other hand, utilise larger and more complex structures. The Weisfeiler-Lehman kernel, instead, partially mitigates the problem by making use of the node labels, which improves node localization and disambiguation in the evaluation of the kernel. Moreover, to further support our claims, note that as we increase the value of h the benefit of manifold learning becomes less evident. In fact, by increasing h we are considering subtrees of increasing depth h, thus reducing locality problems.

Table 2 also shows that the improvement is dataset dependent. We have already noted that the on the PPI dataset all the kernels benefit from manifold learning, although to different extents. Similarly, we observe statistically significant improvements for the graphlet, random walk, WL ($h = 1, 2$) kernels on the Shock dataset. By contrast, datasets such as COIL and MUTAG show little or no improvement. In the case of the COIL dataset, it is important to note that the structure of the graph is that of a Delaunay graph. Here the topological information is essentially given by very local structures, i.e., the Delaunay triangles. As a consequence, the graph structure can be efficiently captured by small and simple substructures. Hence, we do not observe a substantial improvement or decrease in performance when manifold learning is applied. More generally, whether or not our framework leads to an increased accuracy ultimately depends on the manifold nature of the data. Finally, we note that the only kernel that benefits from manifold learning on the MUTAG dataset is again the graphlet kernel. In fact, as shown in Fig.5, this is the only kernel for which the MDS embedding shows a clear horseshoe shape.

Table 2 also shows that by unfolding the space through manifold learning we are levelling to some extent the performances of the examined kernels. This in turn suggests that the difference between the kernels mainly depends on the level of warping of the embedding space. This is particularly evident in the PPI and the Shock datasets, where we observe a marked decrease of the performance gap between the different kernels when manifold learning is applied.



(a) Normalized (b) Unnormalized

Figure 6: The projection of the data from the Hilbert space onto the unit sphere (kernel normalization), can lead to the formation of horseshoe shaped artifacts.

Finally, we should stress that the gain in accuracy that can be achieved with the proposed method comes at the cost of an increased computational time. This is due to the need of exploring the parameter space of Eq. 1 to determine the optimal embedding of the input graphs. However, note that in general the learning phase is performed only once. Given the optimal embedding parameters, the classification of the test graphs is straightforward, and it requires mapping the test graphs to the embedding space, where the classification itself is performed using a linear kernel.

4.3. Kernel Normalization

A common practice in machine learning is that of normalizing the kernels before the classification process takes place. More specifically, the kernels are normalized by projecting the data from the Hilbert space onto the unit sphere. However, note that this can potentially lead to the formation of horseshoe shaped artifacts, as Fig. 6 shows.

Table 3 and Fig. 7 show the average classification accuracy for our approach in the case where the kernels are normalized, where in Fig. 7 the data is displayed as a series of histograms of the average accuracy, for eac dataset. If we compare the results in Table 2 with those in Table 3, we see that the normalized kernels seem to derive more benefits from manifold learning, when compared to the unnormalized kernels. Note, however, that the difference in accuracy is not always statistically significant. Interestingly, in the case of

Kernel	MUTAG	PPI	COIL	Shock	Reeb
SP	$\mathbf{85.08 \pm 0.49}$	69.18 ± 1.00	85.25 ± 0.74	41.03 ± 0.78	55.70 ± 0.50
SP_{ISO}	81.95 ± 0.59	75.29 ± 1.15	84.18 ± 0.83	41.80 ± 083	49.30 ± 0.53
SP_{DIF}	84.22 ± 0.57	$\mathbf{81.53 \pm 1.08}^*$	85.87 ± 0.80	41.53 ± 0.75	$\underline{\mathbf{56.05 \pm 0.54}}$
SP_{LAP}	80.35 ± 0.59	69.18 ± 1.12	$\underline{\mathbf{86.06 \pm 0.83}}$	$\mathbf{44.40 \pm 0.75}^*$	47.93 ± 0.82
SP_{LLE}	85.05 ± 0.51	80.00 ± 1.14	83.81 ± 0.85	44.30 ± 0.76	53.50 ± 0.56
GR	82.05 ± 0.56	52.35 ± 0.77	$\mathbf{83.81 \pm 0.62}$	31.53 ± 0.62	27.23 ± 0.42
GR_{ISO}	82.65 ± 0.57	70.82 ± 1.13	79.63 ± 0.90	40.53 ± 0.74	38.40 ± 0.54
GR_{DIF}	80.51 ± 0.63	$\mathbf{76.82 \pm 1.12}^*$	80.87 ± 0.79	45.03 ± 0.87	$\mathbf{38.98 \pm 0.50}^*$
GR_{LAP}	$\mathbf{83.81 \pm 0.55}^*$	64.47 ± 1.17	79.25 ± 0.90	41.07 ± 0.85	29.98 ± 0.52
GR_{LLE}	81.46 ± 0.56	71.52 ± 1.07	83.31 ± 0.78	$\underline{\mathbf{45.13 \pm 0.87}}^*$	34.62 ± 0.48
RW	83.86 ± 0.52	64.41 ± 1.00	78.66 ± 0.85	33.90 ± 0.71	25.92 ± 0.41
RW_{ISO}	82.76 ± 0.57	70.47 ± 1.08	80.88 ± 0.86	37.80 ± 0.73	29.32 ± 1.79
RW_{DIF}	$\mathbf{84.30 \pm 0.58}$	72.29 ± 1.04	80.19 ± 0.77	41.13 ± 0.83	$\mathbf{54.75 \pm 0.55}^*$
RW_{LAP}	82.35 ± 0.59	72.05 ± 1.01	75.88 ± 0.96	39.00 ± 0.85	39.47 ± 0.70
RW_{LLE}	82.65 ± 0.57	$\mathbf{76.94 \pm 1.05}^*$	$\mathbf{81.12 \pm 0.86}^*$	$\mathbf{43.17 \pm 0.84}^*$	46.80 ± 0.65
WL^1	81.27 ± 0.60	76.76 ± 0.98	79.75 ± 0.91	$34.97 \pm 0.0.68$	33.92 ± 0.42
WL^1_{ISO}	82.51 ± 0.62	71.2 ± 1.03	77.16 ± 1.01	34.76 ± 0.65	44.12 ± 0.54
WL^1_{DIF}	80.84 ± 0.61	$\mathbf{78.88 \pm 1.07}^*$	$\mathbf{80.62 \pm 0.84}$	32.93 ± 0.76	$\mathbf{51.55 \pm 0.52}^*$
WL^1_{LAP}	83.00 ± 0.61	66.41 ± 1.22	67.56 ± 1.06	$\mathbf{44.06 \pm 0.86}^*$	40.72 ± 0.72
WL^1_{LLE}	$\mathbf{84.00 \pm 0.58}^*$	75.65 ± 1.02	80.56 ± 0.85	43.57 ± 0.77	51.30 ± 0.47
WL^2	$\underline{\mathbf{87.84 \pm 0.51}}^*$	$\mathbf{83.53 \pm 0.83}^*$	79.12 ± 0.80	39.10 ± 0.72	50.60 ± 0.57
WL^2_{ISO}	84.46 ± 0.51	78.71 ± 1.14	76.25 ± 0.97	34.16 ± 0.78	44.85 ± 0.57
WL^2_{DIF}	85.46 ± 0.59	79.94 ± 0.96	$\mathbf{79.88 \pm 0.92}$	36.53 ± 0.68	51.25 ± 0.56
WL^2_{LAP}	82.81 ± 0.74	71.59 ± 1.08	73.36 ± 0.97	$\mathbf{39.30 \pm 0.78}$	43.03 ± 0.77
WL^2_{LLE}	86.24 ± 0.53	78.35 ± 0.95	77.69 ± 0.89	39.17 ± 0.86	$\mathbf{51.83 \pm 0.59}^*$
WL^3	86.62 ± 0.55	$\underline{\mathbf{84.06 \pm 0.93}}^*$	79.01 ± 0.83	$\mathbf{40.17 \pm 0.70}$	$\mathbf{55.32 \pm 0.58}^*$
WL^3_{ISO}	85.70 ± 0.56	77.47 ± 1.00	74.00 ± 1.00	36.03 ± 0.71	46.5 ± 0.61
WL^3_{DIF}	$\mathbf{87.16 \pm 0.57}^*$	80.41 ± 0.86	78.69 ± 0.87	36.13 ± 0.65	52.03 ± 0.59
WL^3_{LAP}	82.81 ± 0.65	76.59 ± 1.05	71.00 ± 0.98	38.30 ± 0.78	48.87 ± 0.78
WL^3_{LLE}	85.40 ± 0.63	82.30 ± 0.88	$\mathbf{79.25 \pm 0.96}$	40.07 ± 0.74	50.57 ± 0.53

Table 3: Maximum classification accuracy ($\pm$ standard error) for the **normalized** kernels on the graph datasets. For each kernel and dataset, the best performing manifold learning technique is highlighted in bold, the best performing method over the dataset is underlined and statistically significant results are marked by a $*$.

the random walk kernel on the PPI dataset, the accuracy for the normalized kernel with LLE is significantly higher than any other combination of methods for the unnormalized kernel. We also observe that while the performance of the kernels tends to decrease after

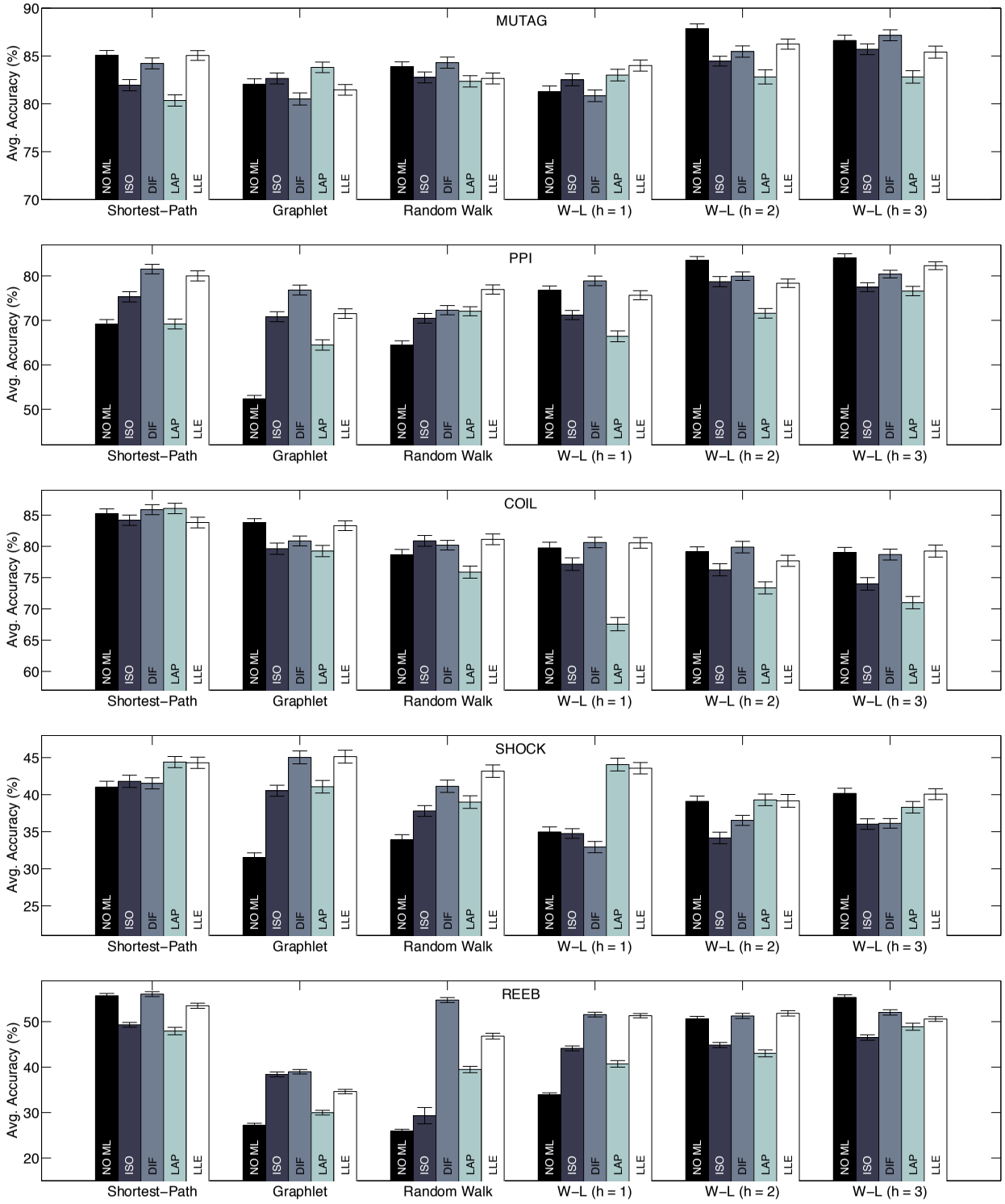


Figure 7: Bar plot showing the classification accuracy (± standard error) reported in Table 3.

the normalization, in some cases applying manifold learning on the normalized kernels compensates for this loss in accuracy. A striking example is that of the random walk and the Weisfeiler-Lehman ($h = 1$) kernels on the Reeb dataset, and also to a lesser extent for the shortest path kernel on PPI. By contrast, in some cases the normalization of the kernel leads to a better performance, but the accuracy of the kernel remains unaltered when manifold learning is applied. For example, this is the case of the Weisfeiler-Lehman ($h = 1$) kernel, where regardless of the normalization the best accuracy is achieved with diffusion maps.

5. Discussion

Although our experimental evaluation shows that applying manifold learning on the kernels can enhance class separation, it is not clear which combination of kernel and manifold learning algorithm leads to the best improvement on a given dataset. In this sense, the only exception is that of the graphlet kernel, which shows a consistent improvement of the classification accuracy regardless of the chosen manifold learning technique. In fact, this is in line with the observation that R-convolution kernels based on the enumeration of small and simple substructures are more likely to suffer from the locality or ambiguity problems. This in turn leads to the underestimation of the distances for dissimilar graphs. The result is to induce a strong curvature on the embedding manifold, which can lead to the manifold folding over on itself, thus increasing the effective dimensionality of the embedding.

This, in turn, suggests a potential heuristic that can be used to decide when to apply the proposed approach. Specifically, one may decide to apply manifold learning to a given dataset and kernel combination after estimating the curvature of the embedding space. However, this is not a trivial task, since the embedding induced by the kernel is not explicit. Moreover, one should estimate the local rather than the global curvature. In fact, in our case measures based on the negative eigenvalues of the distance or dissimilarity matrix such as the negative eigenratio or the negative eigenfraction [45] commonly used for analysing the structure of the manifold resulting from the embedding of similarity data fail to detect any curvature. This is a consequence of the fact that kernels are guaranteed to be positive semidefinite and do not have negative eigenvalues. On the other hand, the presence of

points of high local curvature would be a potential indicator of the type of degeneracies of the manifold that would benefit from the proposed approach.

Wu, Wilson and Hancock [46] have taken some steps in this direction and have recently proposed a way to estimate the curvature of a local patch of the manifold, using pairwise distances between points. Unfortunately, this method works under the assumption that the neighbours of the point where the curvature is being estimated are uniformly distributed. This requirement that may not hold on real data, potentially leading to inaccurate estimations. An alternative approach may be to consider the neighborhood of a point and construct a local kernel. A rapid decay of the eigenvalues of this local kernel would suggest a flat and low-dimensional embedding space. Conversely, a slow decay of the eigenvalues may suggest that the point and its neighborhood lie on a subspace of high curvature.

6. Conclusions

In this paper, we have investigated the use of manifold learning techniques on graph kernels to compute a low-dimensional embedding where the separation of the different classes is enhanced. We observed that when we perform multidimensional scaling on the distance matrices extracted from the kernels, the resulting data tends to be clustered along a curve that wraps around the embedding space, a behaviour that is known to arise when the data lies on a non-linear manifold. Hence, we proposed to unfold the embedding manifold using non-linear dimensionality reduction techniques, in an attempt to increase the separation of the classes. Interestingly, we observed that the unfolding of the space seems level the performances of the different kernels, suggesting that the existing kernels in the literature differ mostly in the level of warping of the embedding space. More in general, we observed that in most of the cases the separation of the data was increased after applying manifold learning on the original kernels.

Acknowledgments

Edwin Hancock was supported by a Royal Society Wolfson Research Merit Award.

References

[1] K. Borgwardt, H. Kriegel, Shortest-path kernels on graphs, in: Proceedings of the Fifth IEEE International Conference on Data Mining, IEEE Computer Society, 2005, pp. 74–81.

[2] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, K. Borgwardt, Efficient graphlet kernels for large graph comparison, in: Proceedings of the International Workshop on Artificial Intelligence and Statistics, 2009, pp. 488–495.

[3] H. Kashima, K. Tsuda, A. Inokuchi, in: Proceedings of the 20th International Conference on Machine Learning, in: ICML, 2003, pp. 321–328.

[4] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, K. M. Borgwardt, Weisfeiler-lehman graph kernels, Journal of Machine Learning Research 12 (2011), pp. 2539–2561.

[5] K. Siddiqi, A. Shokoufandeh, S. Dickinson, S. Zucker, Shock graphs and shape matching, International Journal of Computer Vision 35 (1) (1999), pp. 13–32.

[6] H. Jeong, B. Tombor, R. Albert, Z. Oltvai, A. Barabási, The large-scale organization of metabolic networks, Nature 407 (6804) (2000), pp. 651–654.

[7] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, Y. Sakaki, A comprehensive two-hybrid analysis to explore the yeast protein interactome, Proceedings of the National Academy of Sciences 98 (8) (2001), pp. 4569.

[8] V. Kalapala, V. Sanwalani, A. Clauset, C. Moore, Scale invariance in road networks. Physical Review E 73(2) (2006), 026130.

[9] D. Conte, P. Foggia, C. Sansone, M. Vento, Thirty years of graph matching in pattern recognition, International Journal of Pattern Recognition and Artificial Intelligence 18 (3) (2004), pp. 265–298.

[10] B. Luo, R. C. Wilson, E. R. Hancock, Spectral embedding of graphs, Pattern Recognition 36 (10) (2003), pp. 2213–2230.

[11] R. C. Wilson, E. R. Hancock, B. Luo, Pattern vectors from algebraic graph theory, IEEE Trans. Pattern Anal. Mach. Intell. 27 (7) (2005), pp. 1112–1124.

[12] B. Scholkopf, A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, USA, 2001.

[13] T. Gaertner, P. Flach, S. Wrobel, On graph kernels: Hardness results and efficient alternatives, in: Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, Springer-Verlag, 2003, pp. 129–143.

[14] D. G. Kendall, Abundance matrices and seriation in archaeology, Probability Theory and Related Fields 17 (2) (1971), pp. 104–112.

[15] J. B. Tenenbaum, V. De Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000), pp. 2319–2323.

[16] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: Advances in Neural Information Processing Systems, Vol. 14, 2001, pp. 585–591.

[17] R. R. Coifman, S. Lafon, Diffusion maps, Applied and computational harmonic analysis 21 (1) (2006), pp. 5–30.

[18] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000), pp. 2323–2326.

[19] J. Ham, D. D. Lee, S. Mika, B. Schölkopf, A kernel view of the dimensionality reduction of manifolds, 2004, in: Proceedings of the Twenty-first International Conference on Machine Learning, 2004, pp. 47.

[20] S. Mika, B. Schölkopf, A. J. Smola, K. R. Müller, M. Scholz, G. Rätsch, Kernel PCA and De-Noising in Feature Spaces, in: Advances in Neural Information Processing Systems, 1998, pp. 536–542.

[21] K. Riesen, V. Frinken, H. Bunke, Improving Graph Classification by Isomap, in: Graph-Based Representations in Pattern Recognition, 2009, pp. 205–214.

[22] D. Haussler, Convolution kernels on discrete structures, Technical report, UC Santa Cruz (1999).

[23] F. Aziz, R. C. Wilson, E. R. Hancock, Backtrackless walks on a graph, IEEE Trans. Neural Netw. Learning Syst. 24 (6) (2013), pp. 977–989.

[24] F. Costa, K. D. Grave, Fast neighborhood subgraph pairwise distance kernel, in: Proceedings of the 27th International Conference on Machine Learning, 2010, pp. 255–262.

[25] N. Kriege, P. Mutzel, Subgraph matching kernels for attributed graphs, in: Proceedings of the 29th International Conference on Machine Learning, 2012.

[26] M. Neumann, N. Patricia, R. Garnett, K. Kersting, Efficient graph kernels by randomization, in: ECML/PKDD (1), 2012, pp. 378–393.

[27] H. Fröhlich, J. K. Wegner, F. Sieker, A. Zell, Optimal assignment kernels for attributed molecular graphs, in: Proceedings of the 22nd International Conference on Machine Learning, 2005, pp. 225–232.

[28] J.-P. Vert, The optimal assignment kernel is not positive definite, CoRR abs/0801.4061.

[29] M. Neuhaus, H. Bunke, Edit distance-based kernel functions for structural pattern classification, Pattern Recognition 39 (10) (2006), pp. 1852–1863.

[30] L. Bai, L. Rossi, A. Torsello, E. R. Hancock, A quantum JensenShannon graph kernel for unattributed graphs. Pattern Recognition 48(2) (2015), pp. 344–355.

[31] L. Rossi, A. Torsello, E. R. Hancock, Measuring graph similarity through continuous-time quantum walks and the quantum Jensen-Shannon divergence, Physical Review E 91(2) (2015), 022815.

[32] T. F. Cox, M. A. Cox, Multidimensional scaling, CRC Press, 2000.

[33] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, S. W. Zucker, Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps, Proceedings of the National Academy of Sciences of the United States of America 102 (21) (2005), pp. 7426–7431.

[34] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Computation 15 (6) (2003), pp. 1373–1396.

[35] K. Q. Weinberger, L. K. Saul, Unsupervised learning of image manifolds by semidefinite programming, International Journal of Computer Vision 70 (1) (2006), pp. 77–90.

[36] R. W. Floyd, Algorithm 97: shortest path, Communications of the ACM 5 (6) (1962), pp. 345.

[37] Z. Zhang, J. Wang, H. Zha, Adaptive manifold learning, Pattern Analysis and Machine Intelligence, IEEE Transactions on 34, no. 2 (2012), pp. 253–265.

[38] W. Imrich, S. Klavžar, B. Gorenec, Product graphs: Structure and recognition, Wiley New York, 2000.

[39] S. Vishwanathan, N. N. Schraudolph, R. Kondor, K. M. Borgwardt, Graph kernels, The Journal of Machine Learning Research 99 (2010), pp. 1201–1242.

[40] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, C. Hansch, Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity, Journal of medicinal chemistry 34 (2) (1991), pp. 786–797.

[41] L. J. Jensen, M. Kuhn, M. Stark, S. Chaffron, C. Creevey, J. Muller, T. Doerks, P. Julien, A. Roth, M. Simonovic, et al., String 8a global view on proteins and their functional interactions in 630 organisms, Nucleic acids research 37 (suppl 1) (2009), pp. D412–D416.

[42] A. Torsello, E. R. Hancock, Correcting curvature-density effects in the hamilton–jacobi skeleton, Image Processing, IEEE Transactions on 15 (4) (2006), pp. 877–891.

[43] S. Biasotti, S. Marini, M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno, 3d shape matching through topological structures, in: Discrete Geometry for Computer Imagery, 2003, pp. 194–203.

[44] S. A. Nene, S. K. Nayar, H. Murase, Columbia object image library (coil-20), Dept. Comput. Sci., Columbia Univ., New York. [Online] http://www. cs. columbia. edu/CAVE/coil-20. html 62.

[45] E. Pekalska, A. Harol, R. P. Duin, B. Spillmann, H. Bunke, Non-Euclidean or non-metric measures can be informative, in: Structural, Syntactic, and Statistical Pattern Recognition, 2006, pp. 871–880.

[46] W. Xu, R. C. Wilson, E. R. Hancock, Curvature Estimation for Ricci Flow Embedding, in: Proceedings of the 2014 22nd International Conference on Pattern Recognition, 2014.

[47] P. Mahé, N. Ueda, T. Akutsu, J. L. Perret, J. P. Vert, Extensions of marginalized graph kernels, in Proceedings of the 21st international conference on Machine learning, 2004, pp. 70

[48] B. Gaüzere, L. Brun, D. Villemin, Extensions of marginalized graph kernels, Pattern Recognition Letters, 33(15) (2012), pp. 2038–2047

[49] L. Ralaivola, S. J. Swamidass, H. Saigo, P. Baldi, Graph kernels for chemical informatics, Neural Networks, 18(8) (2005), pp. 1093–1110