Project acronym	SIMBAD
Project full title	Beyond Features: Similarity-Based Pattern Analysis and Recognition
Deliverable Responsible	UNIVERSITY OF YORK Department of Computer Science Heslington Hall, United Kingdom, YO10 5DD
Project web site	http://simbad-fp7.eu
EC project officer	Teresa De Martino
Document title	Spectral and geometric embeddings
Deliverable n.	D4.1
Document type	Report
Dissemination level	Public
Contractual date of delivery	M 18
Project reference number	213250
Status & version	Definitive
Work package	4
Deliverable responsible (SHORT NAME)	UNIYORK
Contributing Partners (SHORT NAME)	DELFT
Author(s)	Edwin Hancock, Bob Duin, Richard Wilson,Peng Ren, Eliza Xu and Wan-Jui Lee
Additional contributor(s)	Andrea Torsello, Mario Figueredo, Ana Fred

WP4.1: Imposing geometricity on non-geometric similarities (embedding)

Edwin Hancock, Bob Duin, Richard Wilson,
Peng Ren, Eliza Xu and Wan-Jui Lee

October 20, 2009

Chapter 1

Overview

The overall goal in this workpackage is that given similarity data possibly in the form of a weighted graph, we aim at developing algorithms for transforming them into instance-specific vectorial representations (embedding) that are suitable for traditional geometric learning algorithms. The work package is divided into the following tasks:

- WP4.1 Spectral and geometric manifold embedding
- WP4.2 Structure-preserving embeddings

The workpackage involves York as lead partner, Delft and Zurich.
This report is the deliverable for WP4.1
Our main achievements have been as follows

- A new vectorisation of graphs (including weighted graphs and hypergraphs) based on the Ihara zeta function.
- A characterisation of embedded graphs based on Gauss curvature and derived from the embedding using the Gauss-Bonnet theorem.
- Progress towards the embedding of indefinite similarities into elliptic and hyperbolic spaces.
- Progress towards the reduction of non-metricity based on applying Ricci flow to the embedded graphs.
- The development of graph-spectral methods for matching labeled graphs.

The report is organised as follows. We commence with an overview of the main technical achievements under the project. The details of the technical developments are confined to the appendices of the report where we collect together published papers and technical notes describing the individual developments.

1.1 Progress and Technical Achievements:

We have made the following progress during the 18 month period covered by the report:

1.1.1 Ihara Zeta Functions and Graph Vectorisation

We have explored a new structural characterisation of graphs based on the Ihara zeta function. Our motivation for embarking on this study was prior work at York, where we demonstrated that the zeta function is the moment generating function of the heat kernel trace and can be used to cluster graphs. Our study is based on the Ihara zeta function which is an extension of Riemann zeta function from prime numbers to prime cycles in a graph. The Ihara zeta function is constacuted by first transforming a graph into its directed line-graph. It is the reciprocal of the characteristic polynomial of the transition matrix for the directed line-graph. Our characterisation of graphs is based on co-efficients of the polynomial. The co-efficients of the polynomial are related to the cycle frequencies in the graph, and are easily computed using the eigenvalues of the directed line-graph transition matrix (the so-called Froebenius operator).

We have made a number of original developments to the study of the Ihara zeta function

- We have shown how the Ihara co-efficients can be used to cluster graph data-sets [14].
- We have shown how to generalise the Ihara zeta function to weighted graphs [16].
- We have started to explore how the Ihara zeta function can be generalised to hyper-graphs, and have obtained some initial experimental results on data-sets derived from image data [13].

- We have noted an interesting relationship between the Ihara zeta function and the discrete time quantum walk on a graph. The relationship is based on the observation that both rely on a representation based on the directed line-graph. In recent work, we have shown that the support matrix for the discrete time quantum walk can lift problems of co-spectrality in trees and strongly regular graphs. We are currently investigating whether the Ihara zeta function also possesses this property [15].

The overall contribution is therefore a vectorial characterisation of graphs that reflects their cycle frequency distribution and which can be computed using the spectrum of the transition matrix of the oriented line-graph. This representation may be applied to simple graphs, weighted graphs and hypergraphs.

Our future plans revolve around investigating whether the Ihara zeta function can be used to define a cycle kernel for graphs. This work is being conducted in collaboration with IST Lisbon. A plan to kernelise the Ihara zeta function using the Shannon-Jensen divergence was formulated when Edwin Hancock and Richard Wilson visited Lisbon in July 2009. Peng Ren is currently exploring how to use the cycle density theorem to effect the kernelisation.

1.1.2 Heat Kernel Embeddings and Spectral Geometry

In prior work, we have shown how to associate curvatures with edges of a graph under the heat kernel embedding. The curvature is determined by the difference between the geodesic distance between nodes and the Euclidean distance between the embedded locations of the nodes [19, 20]. Our aim under SIMBAD has been to develop a deeper understanding of the geometric interpretation of the embedding, and to use this to develop a means of restoring metricity.

We have confined our attention to the embedding of triangulated weighted graphs, where the edge weights exponentially weight the similarity or affinity between nodes. We have made two steps in the towards the objectives of the workpackage.

- Our first step has been to show how the embedded triangles can be used to estimate Gaussian curvature using the Gauss-Bonnet theorem.

The sets of curvatures extracted from a graph have been demonstrated to offer a means of computing graph similarity using the Hausdorff distance between sets, and clustering graphs [4].

- Our second advance has been to show how the Gaussian curvatures can be used to control the regularisation of graphs. The effect of applying this smoothing to the graph data has been to improve the cohesion of clusters that can be obtained [5].

Ongoing work is exploring alternatives to the heat-kernel embedding, which may provide a means for accounting for the indefiniteness of the distance matrix. This is based on the wave equation, and allows to complex embeddings

Our future plans are to develop a more sophisticated regularisation process which will improve the metricity of the embedding. The idea is to evolve the embedded triangles using a Ricci flow controlled by the estimated Guassian curvatures. This can be reformulated as evolving the radii of hyperspheres on which the triangles are embedded.

1.1.3 Embedding non-Euclidean Data

Usually, objects to be classified are represented by features. However, information such as arrangement structure and relations can not be captured by featured-based representation. To overcome this problem dissimilarity representations are used in which a pairwise dissimilarity (or proximity) measure describes the structural relationships between objects in terms of their feature differences. However, when the disimilarity measure is not Euclidean then there are obstacles to the development of valid machine learning techniques. Here we consider two possible causes of this problem. The first is that the distances are indefinite, i.e. there are negative eigenvalues for the dissimilarity matrix. The second is that the data resides on a curved manifold.

Embedding Indefinite Similarities

We have studied the possibility of using constant curvature manifolds for the embedding of indefinite data. These are the elliptic manifolds (constant positive curvature) and the hyperbolic manifolds (constant negative curvature). Geodesic distances on these manifolds are metric but not Euclidean

and so are suitable for representing indefinite similarities. We have developed methods for determining the optimal curvature of the embedding manifold and finding the embedding coordinates. We have also developed projection techniques for finding the optimal projection of points when they do not lie precisely on the constant-curvature manifold. The analysis of indefinite data-sets shows that the elliptic embedding can account for a significant amount of the negative eigenfraction, and the hyperbolic embedding nearly all of it. Results with the 1NN classifier show that these embeddings preserve the class information very well.

We have also provided a new indefinite data-set based on graphs from the COIL dataset. The similarities are based on the matching costs between four classes of graphs derived from objects in the data-set, and show the characteristics of indefinite similarities.

This work is being done in collaboration with TU-Delft. We had a preliminary meeting in York in August 2008 to discus potential avenues of investigation, and have conducted exchanges both at collaboration meetings (York November 2008, Zurich June 2009) and by email.

Ricci Flows

To deal with data residing on a manifold, we commence from our picture of graph embedding based on spectral geometry. Here each edge of the graph is characterised by a curvature, and from these curvatures we are able to compute a charactersisation based on Gauss curvature using the Gauss-Bonnet theorem. Our work develops the idea of Ricci flow in the constant curvature Riemannian by evolving the distance measure through updating Gaussian curvatures on the edges of the graph, so that geometricity can be imposed on non-Euclidean distance measures.

We adopt a representation in which first order cycles of a graph are represented by triangles on a sphere. The radius of the sphere is determined from the curvatures of the edges of a geodesic triangle using the Gauss-Bonnet theorem. Under this representation, a graph is represented by a set of spheres. Our idea is to evolve the radii of the spheres under a Ricci flow, so as to locally flatten and to impose geometricity (i.e. to make the disimilarities on the edges Euclidean). Based on Ricci flow for a constant curvature manifold, the curved space can be transformed into a smooth flattened space. Our method transforms the pairwise distances into a Euclidean space by updating the Gaussian curvatures. We have applied our method to the Chicken

pieces dataset. The experimental results show our method can transformed non-Euclidean distances into a Euclidean space. However, the resulting Euclidean distances have poor discriminating power from 1NN classifier. The loss of local structure maybe caused by the evolution process on individual edges independently. The heat kernel regularization fails to preserve the ranking of the distance measures. Hence, our future working is to find a way to keep the local structure during Ricci flow smoothing process.

This work is being conducted in collaboration with the University of Venice. The technical plan for the work was conducted when Edwin Hancock and Richard Wilson visited Venice in April 2009.

1.1.4 Joint Eigenspaces for spectral graph and subgraph embeddings

This work was done by the Delft University of Technology group. The starting point of this research is to construct dissimilarity measurements on structure data and thus the property of non-geometricity on structure data dissimilarity can be further studied. So far, graph is the most general form of structure data, and therefore we focus on defining dissimilarity for graphs in this work. For constructing dissimilarity measurements for graphs, the first thing to consider is the form of graphs. Roughly speaking, there are two kinds of graphs, i.e. graphs with labels/attributes and graphs with no labels/attributes. In the literature of graph comparison, graphs with and without labels/attributes are usually handled with different techniques. Often spectral methods [21, 8, 12, 3, 17] are used to compare graphs without labels/attributes, but for labeled/attributed graphs, usually graph edit distance based techniques [10, 11, 2] are used simply because spectral methods can not handle such complex data structure.

For graphs with labels/attributes, search-based techniques are the main solutions so far, such as graph edit distance [10, 11, 2] and random walk path [1, 18]. Comparing two graphs with labels/attributes is unfortunately a NP-complete problem, and therefore search-based methods are inevitably time-consuming. That is why, when it comes to comparing graphs with no labels/attributes, one usually turns to spectral methods. For a graph with no labels/attributes, it can be easily represented in the matrix form. The most commonly used matrix for representing such graphs are the adjacency matrix and the Laplacian matrix. With such matrices, instead of searching

for similar or identical parts of the graphs, one can directly perform spectral analysis on the matrices. Not only because it can be performed in polynomial time but also the mathematical background provides such methods good properties to capture the graphs in the way that search-based methods cannot perform.

In this work, we first proposed a spectral method for embedding graphs with no labels/attributes into a joint eigenspace, and then compared such graphs pairwise to obtain the dissimilarity between all pairs of graphs. The embedding of graphs is performed by projecting the eigenvalues from a pair of graphs into a joint eigenspace which is expanded by the eigenvectors of both graphs. The detail of this reseach can be found in [6].

As we have mentioned, spectral embeddings in the literature are only suitable for graphs with no labels/attributes, and it is one of the most significant drawback of spectral embeddings. To overcome this limitation of spectral embeddings, we proposed a solution [7] for embedding labeled graphs with spectral methods. The basic idea is to decompose labeled graphs into unlabeled subgraphs with respect to the labels. First, we have to find out the set of labels from all the graphs (both the training and testing objects). For each label, we can subtract a subgraph from one graph. This subgraph might be empty if none of the nodes in the graph has such a label. The advantage of decomposing graphs by labels is that a subgraph is only composed of nodes with the same label. Therefore, the label information is not important anymore in the subgraph. As a result, we can embed the subgraphs with the spectral method proposed in [6].

In [7], we only consider graphs with labels. To further extend the spectral embeddings to graphs with attributes, we categorize the attributes with continuous values into discrete labels. Then we only need to consider discrete label problems. To categorize the continuous attributes, we adopt the most well-known clustering approach, K-means [9]. The K-means clustering approach is an iterative process to find a certain number of stable cluster centers where the total distance of each object to its closest center reaches the minimum. A node of a graph will be taken as an object, and each attribute on the node will correspond to a feature of an object. Therefore, all the nodes of all the graphs will form the dataset for clustering. Then the K-means clustering method will separate the objects into the desired number of clusters. In the end, each cluster will be given a unique label, and the nodes related to the objects in one cluster will be assigned to the label of the cluster. After categorizing the attributes into labels, we can perform the

subgraph embedding described in [7].

To have better understanding about geodesic graph embedding and stimulate the international cooperation, the post-doctor researcher Wan-Jui Lee visited Prof. Horst Bunke and worked with him and other researchers in the computer vision and artificial intelligence group in University of Bern for a week in August, 2009.

Bibliography

[1] E. Farhi A. M. Childs and S. Gutmann. An example of the difference between quantum and classical randomwalks. *Quantum Information Processing*, 1(1-2):35–43, 2002.

[2] H. Bunke. Error correcting graph matching: On the influence of the underlying cost function. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(9):917–922, 1999.

[3] T. Caelli and S. Kosinov. An eigenspace projection clustering method for inexact graph matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(4):515–519, 2004.

[4] Hewayda ElGhawalby and Edwin R. Hancock. Geometric characterizations of graphs using heat kernel embeddings. In *IMA Conference on the Mathematics of Surfaces*, pages 124–142, 2009.

[5] Hewayda ElGhawalby and Edwin R. Hancock. Graph regularisation using gaussian curvature. In *GbRPR*, pages 233–242, 2009.

[6] Wan-Jui Lee and Robert P.W. Duin. An inexact graph comparison approach in joint eigenspace. In *Structural, Syntactic, and Statistical Pattern Recognition, Proc. SSSPR2008, Lecture Notes in Computer Science*, number 5342, pages 35–44, Orlando, Florida, USA, Dec. 2008.

[7] Wan-Jui Lee and Robert P.W. Duin. A labelled graph based multiple classifier system. In *8th International Workshop on Multiple Classifier Systems, Lecture Notes on Computer Science*, number 5519, pages 201–210, Reykjavik,Iceland, June 2009.

[8] B. Luo and E. R. Hancock. Structural graph matching using the em algorithm and singular value decomposition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(10):1120–1136, 2001.

[9] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[10] M. Neuhaus and H. Bunke. Edit distance-based kernel functions for structural pattern classification. *Pattern Recognition*, 39:1852–1863, 2006.

[11] M. Neuhaus and H. Bunke. Automatic learning of cost functions for graph edit distance. *Information Sciences*, 177(1):239–247, 2007.

[12] E. R. Hancock R. C. Wilson and B. Luo. Pattern vectors from algebraic graph theory. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(7):1–13, 2005.

[13] Peng Ren, Tatjana Aleksic, Richard C. Wilson, and Edwin R. Hancock. Hypergraphs, characteristic polynomials and the ihara zeta function. In *CAIP*, pages 369–376, 2009.

[14] Peng Ren, Richard C. Wilson, and Edwin R. Hancock. Graph characteristics from the ihara zeta function. In *SSPR/SPR*, pages 257–266, 2008.

[15] Peng Ren, Richard C. Wilson, and Edwin R. Hancock. Characteristic polynomial analysis on matrix representations of graphs. In *GbRPR*, pages 243–252, 2009.

[16] Peng Ren, Richard C. Wilson, and Edwin R. Hancock. Weighted graph characteristics from oriented line graph polynomials. In *ICCV*, 2009.

[17] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10(5):695–703, 1988.

[18] W. Woess. *Random Walks on Infinite Graphs and Groups*. Cambridge University Press, 2000.

[19] Bai Xiao, Edwin R. Hancock, and Richard C. Wilson. Geometric characterization and clustering of graphs using heat kernel embeddings. *Image and Vision Computing*, 2009.

[20] Bai Xiao, Edwin R. Hancock, and Richard C. Wilson. Graph characteristics from the heat kernel trace. *Pattern Recognition*, 42(11):2589–2606, 2009.

[21] F. Zhang and E. R. Hancock. Graph spectral image smoothing using the heat kernel. *Pattern Recognition*, 41:3328–3342, 2008.

Appendix A

Ihara Zeta Functions

Weighted Graph Characteristics from Oriented Line Graph Polynomials

Peng Ren Richard C. Wilson Edwin R. Hancock
Department of Computer Science
University of York, York, YO10 5DD, UK
{pengren, wilson, erh}@cs.york.ac.uk

Abstract

We develop a novel method for extracting graph characteristics from edge-weighted graphs, based on an extension of the Ihara zeta function from unweighted to edge-weighted graphs. This is effected by generalizing the determinant form of the Ihara zeta function. We use the set of the reciprocal polynomial coefficients of the resulting Ihara zeta function, i.e. the Ihara coefficients, to construct our characterization. We also present a spectral analysis of the edge-weighted graph Ihara coefficients and indicate their advantages over graph spectral methods. Experimental results reveal that the Ihara coefficients are effective for the purpose of clustering edge-weighted graphs.

1. Introduction

Graph-based methods were one of the earliest paradigms proposed for scene understanding [10][2], and have provided powerful tools for the analysis of shape, object and scene structures in computer vision. Examples include the analysis of shock-graphs [19], articulated shape representation [14] and object tracking [20]. Recently, the appearance based methods based on the arrangement of SIFT features have provided impressive practical results for object recognition and video analysis [15]. However, here too, the need for structural representations has been demonstrated with the SIFT features being augmented by constellations [9] and k-fans [7] to represent the arrangement of feature points.

When dealing with graph-based representations of image structures there are two ways in which the analysis of structure can be conducted. The first of these is to perform a detailed correspondence analysis and to seek matches between the nodes of the structures under consideration. There are many examples of node correspondence algorithms in computer vision, especially for feature point matching [6][13][23]. These methods exploit local connectivity information together with node and edge attributes to match relational structures by optimizing a criterion function. However, correspondence matching can become intractable when the sizes of the structures become large and when feature points tend to be less discriminative. It is for these reasons that a second approach based on comparing permutation invariant characteristics extracted from graphs is adopted as an alternative.

There are a number of alternative node permutation invariant characterizations that can be used. Perhaps the simplest of these are topological properties such as the size, edge density, the perimeter or diameter and the number of cycles of different degrees. A more sophisticated alternative is to use features extracted from a matrix characterization of a graph. Here the initial matrix representation $\boldsymbol{M}$ can be based either on the adjacency matrix, the Laplacian matrix or the signless Laplacian. The matrix can be characterized using either its eigenvalues $\mathrm{sp}(\boldsymbol{M})$ and eigenvectors (i.e. using spectral graph theory) or by the coefficients of its characteristic polynomial $\det(\lambda\boldsymbol{I}-\boldsymbol{M})$ (i.e. using algebraic graph theory). The two approaches are essentially equivalent and have led to a number of practical characterizations that can be used for both object recognition and shape clustering [21][13][22].

An alternative characterization that has received relatively little attention in the computer vision and pattern recognition community is provided by the zeta function. In number theory, the Riemann zeta function is determined by the locations of the prime numbers. There is a natural extension of the Riemann zeta function from prime numbers to graphs. For instance, the Ihara zeta function is determined by the set of prime cycles on a graph, and is detailed in [11]. Bass [4] has generalized the explicit factorizations to all finite graphs. It is interesting to note that the Ihara zeta function is computed by first transforming the graph in-hand into an oriented line graph, and then computing the characteristic polynomial of the adjacency matrix of the oriented line graph. The zeta function is determined by the reciprocal of the characteristic polynomial, and the prime cycles determine the poles of the zeta function in a manner analogous to the prime numbers. There have been a number of recent applications of zeta functions in computer vision and pattern recognition. For instance, Ren *et al.* [17]

have shown how to use the coefficients of the characteristic polynomial of the oriented line graph to cluster unweighted graphs. Zhao *et al.* [24] have used Savchenko's formulation of the zeta function, expressed in terms of cycles, to generate merge weights for clustering over a graph-based representation of pairwise similarity data [24]. Bai *et al.* [1] have shown that the Riemann zeta function is the moment generating function of the heat-kernel trace and have used the moments to cluster graphs.

Although the the zeta function draws on the characteristic polynomial of a graph and is hence akin to methods from algebraic graph theory, it first relies on a graph transformation. This is an interesting observation since the quest for improved alternatives to the adjacency and Laplacian matrices has been a quest in spectral graph theory. Recently, the signless Laplacian (i.e. the degree matrix plus the adjacency matrix) has been suggested. Additionally, Emms *et al.* [8] have recently shown that a unitary matrix characterization of the oriented line graph can be used to reduce or even completely lift the cospectrality of certain classes of graphs, including trees and strongly regular graphs. This points to the fact that one potentially profitable route to improving methods from spectral graph theory may reside in graph transformation.

Unfortunately, the existing definition of the Ihara zeta function applies only to unweighted graphs. The task of utilizing the Ihara zeta function as a characterization of edge-weighted graphs has yet to be investigated. The determinant (i.e. characteristic polynomial) expression of the Ihara zeta function is not available as a characterization of edge-weighted graphs due to its binary representation. In this paper, we address this shortcoming and derive a determinant expression applicable to edge-weighted graphs. This is effected with the assistance of Bartholdi zeta function. The Ihara coefficients for edge-weighted graphs are computed and the resulting pattern vectors are used to cluster edge-weighted graphs extracted from visual objects.

2. The Ihara Zeta Function

The Ihara zeta function of unweighted graphs is a generalization of the Riemann zeta function from number theory. In the definition of the Ihara zeta function, the 'prime number' in the Euler product expansion of the Riemann zeta function is replaced by a 'prime cycle', i.e. cycles with no backtracking in a graph. As a result, the Ihara zeta function is generally an infinite product. However, one of its elegant features is that it can be collapsed down into a rational function, which renders it of practical utility.

2.1. Rational Expression

For a graph $G(V,E)$ with the vertex set V of cardinality $|V| = N$ and the edge set E of cardinality $|E| = M$, the rational expression of the Ihara zeta function is [11]:

$$Z_G(u) = \left(1-u^2\right)^{\chi(G)} \det\left(\boldsymbol{I}_N - u\boldsymbol{A} + u^2\boldsymbol{Q}\right)^{-1}. \quad (1)$$

Here, $\chi(G)$ is the Euler number of the graph $G(V,E)$, which is defined as the difference between cardinalities of the vertex set and the edge set of the graph, i.e. $\chi(G) = N-M$, and $\boldsymbol{A}$ is the adjacency matrix of the graph. The degree matrix $\boldsymbol{D}$ is constructed by placing the column sums of the adjacency matrix as diagonal elements, while setting the off-diagonal elements to zero. Finally, with $\boldsymbol{I}_k$ denoting the $k\times k$ identity matrix, $\boldsymbol{Q}$ is the matrix difference of the degree matrix $\boldsymbol{D}$ and the identity matrix $\boldsymbol{I}_N$, i.e. $\boldsymbol{Q} = \boldsymbol{D} - \boldsymbol{I}_N$. From (1) it has been shown that the Ihara zeta function is permutation invariant to vertex label permutations [17]. This is because permutation matrices, which represent vertex label permutations in matrix calculation, have no effect on the determinant in (1).

2.2. Determinant Expression

For md2 graphs, i.e. the graphs with vertex degree at least 2, it is straightforward to show that (1) can be rewritten in the form of the reciprocal of a polynomial. However, it is difficult to compute the coefficients of the reciprocal of the Ihara zeta function from (1) in a uniform way, except by resorting to software for symbolic calculation. To efficiently compute these coefficients, it is more convenient to transform the rational form of the Ihara zeta function in (1) into a concise expression. The Ihara zeta function can also be written in the form of a determinant [12]:

$$Z_G(u) = \det(\boldsymbol{I}_{2M} - u\boldsymbol{T})^{-1} \quad (2)$$

where $\boldsymbol{T}$ is the Perron-Frobenius operator of the oriented line graph, and is an $2M \times 2M$ square matrix.

To obtain the Perron-Frobenius operator $\boldsymbol{T}$, we must construct the oriented line graph of the original graph from the associated symmetric digraph. The symmetric digraph $SDG(V,E_d)$ of a graph $G(V,E)$ is composed of a finite nonempty vertex set V identical to that of $G(V,E)$ and a finite multiset E_d of oriented edges called arcs, which consist of ordered pairs of vertices. For arc $e_d(u,w) \in E_d$ where u and v are elements in V, the origin of $e_d(u,w)$ is defined to be $o(e_d) = u$ and the terminus is $t(e_d) = v$. Its inverse arc, which is formed by switching the origin and terminus of $e_d(u,w)$, is denoted by $e_d(w,u)$. For the graph $G(V,E)$, we can obtain the associated symmetric digraph $SDG(V,E_d)$ by replacing each edge of $G(V,E)$ by the arc pair in which the two arcs are inverse to each other.

The oriented line graph of the original graph can be defined using the symmetric digraph. It is a dual graph of the symmetric digraph since its oriented edge set and vertex set are constructed from the vertex set and the oriented edge

(arc) set of its corresponding symmetric digraph. The construction of the vertex set V_L and oriented edge set E_{dL} of the oriented line graph can be formulated as follows:

$$\begin{cases} V_L &= E_d(SDG) \\ E_{dL} &= \{(e_d(u,v), e_d(v,w)) \\ &\quad \in E_d(SDG) \times E_d(SDG);\ u \neq w\}. \end{cases} \tag{3}$$

The Perron-Frobenius operator $\boldsymbol{T}$ of the original graph is the adjacency matrix of the associated oriented line graph. For the (i,j)th entry of $\boldsymbol{T}$, $\boldsymbol{T}(i,j)$ is 1 if there is one edge directed from the vertex with label i to the vertex with label j in the oriented line graph, and is 0 otherwise.

Unlike the adjacency matrix of an undirected graph, the Perron-Frobenius operator is not a symmetric matrix. This is because of a constraint that arises in the construction of oriented edges. Specifically, the arc pair with two arcs that are the reverse of each other in the symmetric digraph are not allowed to establish an oriented edge in the oriented line graph. This constraint arises from the second requirement in the edge definition appearing in (3).

3. Ihara Coefficients for Weighted Graphs

Although (2) characterizes unweighted graphs in a compact way using the determinant, when it comes to edge-weighted graphs, i.e. the edges not only record the vertex connections but also have attributes attached on them, then the determinant scheme introduced in Section 2.2 is not applicable. This is because the Perron-Frobenius operator for an unweighted graph is the adjacency matrix of the associated oriented line graph, which only bears relational information and defaults the edge weights to binary values. This hinders the generalization of the determinant expression of the Ihara zeta function to edge-weighted graphs.

To characterize edge-weighted graphs, we generate the Perron-Frobenius operator for edge-weighted graphs from a simplified version of the Bartholdi zeta function, which is a more sophisticated zeta function with two independent variables. We then introduce a scheme to characterize the edge-weighted graphs using the polynomial coefficients of the reciprocal Ihara zeta function.

3.1. Bartholdi Zeta Function

The Bartholdi zeta function of a graph aims to generalize the zeta function using the concept of 'prime circle' where backtracking is allowable. It was first introduced and developed by Barhtholdi in [3]. For a graph $G(V,E)$, the rational expression of the Bartholdi zeta function is:

$$\begin{aligned} Z_{GB}(u,t) = &\left(1-(1-t)^2u^2\right)^{\chi(G)} \times \\ &\det\left(\boldsymbol{I}_N - u\boldsymbol{A} + (1-t)u^2(\boldsymbol{D}-(1-t)\boldsymbol{I}_N)\right)^{-1}. \end{aligned} \tag{4}$$

Compared with the Ihara zeta function in its rational form (1), the rational expression for the Bartholdi zeta function involves an additional variable t, which is closely related to the cyclic bump count of a circle in a graph. One noteworthy property of the Bartholdi zeta function is that when $t = 0$, it reduces to the Ihara zeta function. For more details of the Bartholdi zeta function and its relationship with the Ihara zeta function, we refer readers to [3] and [16].

3.2. Ihara Polynomial for Edge-weighted Graphs

Based on Bartholdi's work, Mizuno *et al.* [16] have developed the following determinant expression for the Bartholdi zeta function:

$$Z_{GB}(u,t) = \det\left(\boldsymbol{I}_{2M} - (\boldsymbol{B} - (1-t)\boldsymbol{J})u)\right)^{-1}. \tag{5}$$

In this form the zeta function is equivalent to (4) and is suitable for dealing with both unweighted graphs and edge-weighted graphs. There are two $2M \times 2M$ operators $\boldsymbol{B}$ and $\boldsymbol{J}$ in (5), which are both closely related to the symmetric digraph of the original graph. For a graph $G_w(V,E)$ with weighted edges, the associated symmetric digraph $SDG_w(V,E)$ can be constructed by replacing each edge of $G_w(V,E)$ by the arc pair in which the two arcs are the reverses of each other. The edge weights are then attached to each of its generating arcs. This is similar to that adopted in the case of unweighted graphs introduced in Section 2.2, except that there is the additional step of weight attachment.

Based on the symmetric digraph, the elements of the operators $\boldsymbol{J}$ and $\boldsymbol{B}$ can be computed as follows:

$$\boldsymbol{J}_{ij} = \begin{cases} 1 & \text{if } e_{di} = \bar{e}_{dj} \\ 0 & \text{otherwise,} \end{cases} \quad \boldsymbol{B}_{ij} = \begin{cases} w_{dj} & \text{if } t(e_{di}) = o(e_{dj}) \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Here, $t(e_{di})$ and $o(e_{dj})$ denote the the origin and terminus of the arc e_{di} in the symmetric digraph, respectively. $\bar{e}_{dj}$ denotes the inverse arc of e_{di}. w_{dj} is the weight of the oriented edge e_{dj} in the symmetric graph. It is equal to the weight of the edge, from which the arc e_{dj} is derived, in the original graph.

In the situation of edge-weighted graphs, when $t = 0$ in (5), the determinant expression of the Bartholdi zeta function reduces to the corresponding Ihara form denoted as:

$$\begin{aligned} Z_G(u) &= \det(\boldsymbol{I}_{2M} - u(\boldsymbol{B}-\boldsymbol{J}))^{-1} \\ &= \det(\boldsymbol{I}_{2M} - u\boldsymbol{T}_w)^{-1} \end{aligned} \tag{7}$$

where we define $\boldsymbol{T}_w = \boldsymbol{B} - \boldsymbol{J}$ to be the Perron-Frobenius operator for the edge-weighted graphs. $\boldsymbol{T}_w$ can also reasonably be regarded as the adjacency matrix of the oriented line graph of the original edge-weighted graph.

There are several notes needing to be made here. First, unlike in (3), there is no requirement on the exclusion of

reversed arcs in the computation of the operator $\boldsymbol{B}$ in (6). Second, the operator $\boldsymbol{J}$ literally records all the reversed arc relations. Third, when it comes to unweighted graph, $\boldsymbol{T}_w$ reduces to $\boldsymbol{T}$ in (2). This is because in this case $\boldsymbol{B}$ reduces to a binary matrix representing orientations and connections only, and the operator $\boldsymbol{T}_w$, i.e. the difference of $\boldsymbol{B}$ and $\boldsymbol{J}$, naturally satisfies the all the constraints in (3). Above all, our proposed scheme establishes a generalized version of the Perron-Frobenius operator, which are both available to unweighted graphs and edge-weighted graphs.

3.3. Pattern Vectors from Ihara Coefficients

To establish pattern vectors from the Ihara zeta function for the purpose of characterizing edge-weighted graphs in machine learning, one approach is to consider taking function samples as elements. However, if this strategy is adopted, there will be dangers of sampling at poles, and these give rise to infinities. Hence, pattern vectors consisting of function samples are potentially unstable since the distribution of poles is unknown beforehand.

To overcome this problem, we note that the coefficients of the reciprocal of the Ihara zeta function, which we refer to as the Ihara coefficients, do not give rise to infinities.

According to (7), the reciprocal of the Ihara zeta function for edge-weighted graphs can be rewritten as follows:

$$\begin{aligned} Z_G^{-1}(u) &= \det(\boldsymbol{I} - u\boldsymbol{T}_w) = (u)^{2M}\det\left(\frac{1}{u}\boldsymbol{I} - \boldsymbol{T}_w\right) \\ &= c_0 + c_1 u + \cdots + c_{2M-1}u^{2M-1} + c_{2M}u^{2M}. \end{aligned} \tag{8}$$

From (8), the Ihara coefficients $c_0,\ c_1,\ \ldots\ c_{2M-1}$ and c_{2M} are actually the coefficients of the characteristic polynomial of the matrix $\boldsymbol{T}_w$. The pattern vectors characterizing graphs are then established with Ihara coefficients as elements. This is to be contrasted with the work of Bai *et al.* [1], who sample the zeta function values. The Ihara coefficients not only avoid the hazards of infinities that are encountered if function samples are used, but also convey direct information concerning graph structure and topology.

3.4. Spectral Interpretation and Computation

For unweighted graphs, the Ihara coefficients are essentially graph structural descriptors on the circle frequencies and vertex degrees [17][18]. For edge-weighted graphs, the Ihara coefficients have no direct links with the graph structure. Here we study the Ihara coefficients for edge-weighted md2 graphs from a spectral standpoint. We then perform a comprehensive analysis on the effectiveness of the Ihara coefficients for clustering edge-weighted md2 graphs.

As stated in Section 3.2, there is always one weighted oriented line graph associated with any weighted md2 graph. For an md2 graph, the cardinality of its vertex set is not greater than that of its edge set, subject to the least vertex degree constraint. Therefore, in practice the cardinality of the vertex set of the oriented line graph is usually much greater than, or at least equal to, that of the original graph. The construction of the oriented line graph is thus a process of transforming the original graph into a version with adjacency matrix $\boldsymbol{T}_w$ in a higher dimensional space than that of the original graph. Furthermore, the Ihara coefficients have a strong relationship with the spectrum of the Perron-Frobenius operator such that each coefficient can be derived from the polynomial of the eigenvalue set $\{\lambda_1, \lambda_2\ \ldots\ \lambda_n\}$ of $\boldsymbol{T}_w$ as follows:

$$c_r = (-1)^r \sum_{k_1<k_2<\ \ldots\ <k_r} \lambda_{k_1}\lambda_{k_2}\ \ldots\ \lambda_{k_r}. \tag{9}$$

The subscript number k in (9) runs over all possible combination of the coefficient labels. The difference between (9) and the elementary symmetric polynomials adopted in [22] is that the Ihara coefficient c_r is in fact the product of the elementary symmetric polynomial and the factor $(-1)^r$, where the subscript r indicates c_r the coefficient of the variable to the power r. Equation (9) provides an efficient way to compute the Ihara coefficients by enumerating the eigenvalues of a $2M \times 2M$ matrix. This is close in spirit to the first method for computing the polynomial coefficients suggested by Brooks [5]. Moreover, it is more efficient than the third method suggested by Brooks, which is based on computing the determinant of $2M \times 2M$ matrix $\binom{2M}{2M-r}$ times to obtain the coefficient c_r.

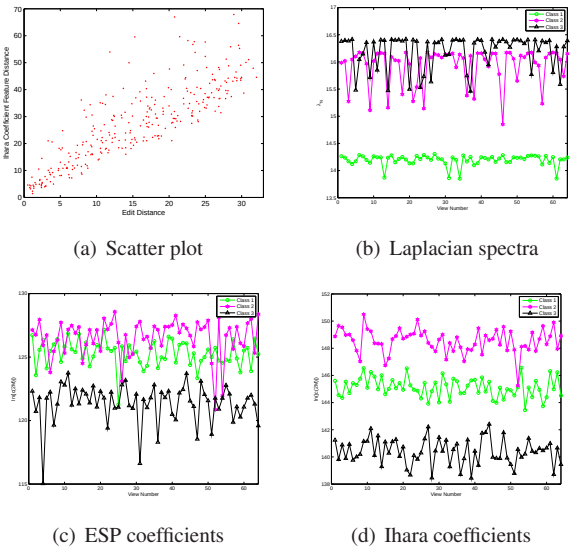
The advantages of the Ihara coefficients over the Laplacian spectral method are twofold. First, the graph is transformed to a higher dimensional feature space. Thus the characteristics from the Perron-Frobenius operator generally reflect more about the graph structure than graph matrix representations in the original domain. Second, the Ihara coefficients make use of the complete set of eigenvalues of the Perron-Frobenius operator and they do not suffer from spectral truncation, as does the pattern vectors consisting of a fixed number of leading nonzero Laplacian eigenvalues.

4. Experimental Evaluation

We evaluate our proposed scheme in two ways. We first evaluate the ability of the Ihara coefficients to distinguish between randomly generated edge-weighted graphs under controlled structural errors. Second, we focus on real-world data and assess the effectiveness of the Ihara coefficient pattern vectors in detecting object clusters.

4.1. Synthetic Data

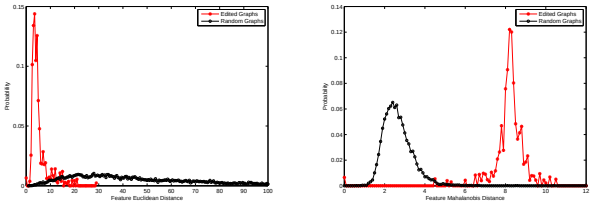
We first investigate the relationship between graph edit distance and the feature distance between Ihara coefficient pattern vectors. The edit distance of two graphs G_1 and



(a) Scatter plot (b) Laplacian spectra

(c) ESP coefficients (d) Ihara coefficients

Figure 1. Feature plots



(a) Euclidean distance (b) Mahalanobis distance

Figure 2. Distance distribution

G_2 is the minimum edit cost taken over all sequences of edit operations that transform G_1 into G_2. Therefore, if we establish a new graph by deleting a certain number of edges from the seed graph and if we assign each deletion an edit cost of the deleted edge weight, the edit distance between the two graphs is equal to the overall edit cost.

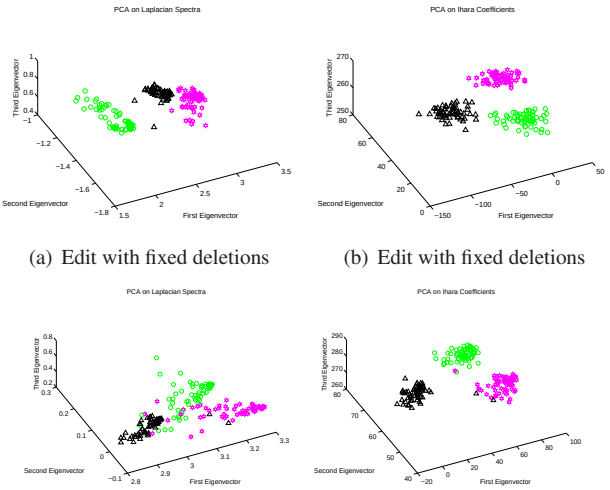
We commence with a single randomly generated md2 graph as the seed graph with 100 vertices and 300 weighted edges. In this subsection, the edge weights are always generated so as to have a uniform distribution over the interval [0.5, 1.5]. We obtain edited versions of the seed graph by randomly deleting edges, with the number of deleted edges varying from 1 to 30. For each number of edge deletions, we perform 10 randomized edge deletion trials subject to the md2 constraint. We compute the Ihara coefficients using (9) and construct the pattern vector in the form of $\boldsymbol{v}_{G1} = [c_3,\ c_4,\ \ln(|c_{2M-3}|),\ \ln(|c_{2M-2}|),\ \ln(|c_{2M-1}|),\ \ln(|c_{2M}|)]^T$. The final four components of the pattern vector are scaled in a logarithmic manner to avoid unbalanced variance. The feature distance between pattern vector $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ is $d_{i,j} = \sqrt{(\boldsymbol{v}_i - \boldsymbol{v}_j)^T(\boldsymbol{v}_i - \boldsymbol{v}_j)}$, which measures the distinction between two samples in the feature space. Figure 1(a) plots the feature distances obtained using the pattern vectors composed of the Ihara coefficients, versus the corresponding graph edit distances between the seed graph and its modified variants. The main feature to note from the plot is that the Ihara coefficient distance generally follows the edit distance. Moreover, for small distances the variation of Ihara coefficient distance is approximately linear with edit distance. For large edit distance the Ihara coefficient distance becomes more scattered.

To take this study on synthetic data one step further, we study the distribution of Ihara coefficient feature distance. We investigate two sets of graphs. The first set consists of graphs which are obtained by randomly deleting one edge from a seed graph with 100 vertices and 301 weighted edges, subject to the md2 constraint. The second set are md2 graphs randomly generated with 100 vertices and 300 weighted edges. Figure 2(a) shows the distribution of the Euclidean feature distance between the pattern vectors of Ihara coefficients for the first set of graphs (red asterisk curve) and that of the second set (black circle curve). We can see that the modal distance between pattern vectors for graphs with random edge edits is much smaller than that for the structurally distinct graphs. For comparison, Figure 2(b) shows the Mahalanobis distances for the two sets of graphs. We can see that the two sets are almost totally non-overlapping. From Figures 2(a) and 2(b), the distance between pattern vectors appears to provide a scope for distinguishing between distinct graphs when there are variations in edge structure due to noise.

To provide an illustration and make a more comprehensive comparison with the graph spectral methods, we create two graph sets, which are established according to different types of graph edits separately, for experiments. Both sets are three classes of graphs separately derived from three seed graphs, which are again randomly generated with 100 vertices and 300 weighted edges. However, we perform two different types of edit operations on the seed graphs to establish the two graph sets separately. The first is to randomly delete eight edges at each time, and the second is with a random number of edge deletions from one to eight in each trial. We first perform the first type of edit operations on three seed graphs. Sixty four random trials of the edits are performed on each of the three seed graphs separately. Figures 1(b), 1(c) and 1(d) show the largest Laplacian eigenvalue, the final coefficient of the elementary symmetric polynomial [22] of Laplacian spectrum (ESP's) and the final Ihara coefficient as a function of trial number respectively. The main feature to note is that in the case of the Ihara coefficients, the variance is smallest and there is little overlap. The other two methods are overlapped to a more severe degree.

We then embed the pattern vectors for the two sets of edited graphs separately into a three-dimensional space us-

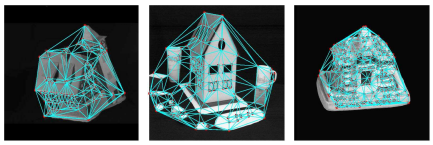
ing principal component analysis(PCA) to evaluate their clustering performances. We applied this procedure both to the Laplacian spectral pattern vector consisting of the second through to the seventh Laplacian eigenvalues, which is one of the most effective representations for graphs, and to the Ihara coefficient vector v_{G1}. Figure 3 shows the experimental results. Figures 3(a) and 3(b) show the clusters generated by the Laplacian spectra and the Ihara coefficients respectively, subject to the first type of graph edits. Although the spectral method appears to produce 'good' clusters in Figure 3(a), there are some deficiencies. First, the right two clusters are so close that they almost merge into one. Second, the left cluster has a non-compact ring shape and there are no graphs near the cluster center. However, the clusters in Figure 3(b) produced by the Ihara coefficient method are both more compact and more separable. Figures 3(c) and 3(d) show the results for the second type of graph edits. In this more complex situation, the Laplacian spectral method yields clusters with considerable scattering, as illustrated in Figure 3(c). However, for the Ihara coefficients, although there are a small number of outlier samples (two black triangles in the pink star cluster and one pink star in the green circle cluster), the overall performance is much better and provides the basis for a usable clustering technique.



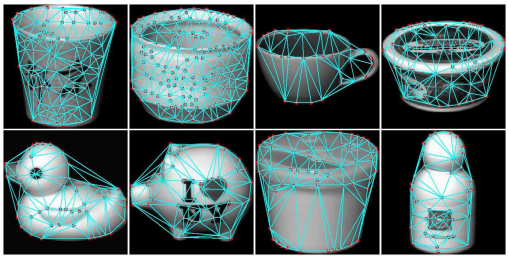
(a) Edit with fixed deletions (b) Edit with fixed deletions

(c) Edit with random deletions (d) Edit with random deletions

Figure 3. Clusters for three classes of graphs

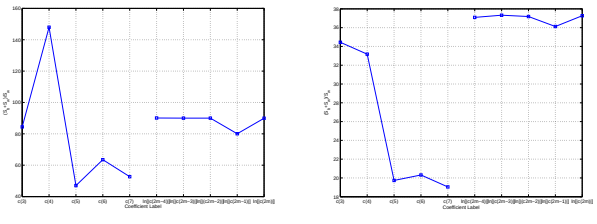


(a) House sequences



(b) COIL datasets

Figure 4. Datasets for Experiments



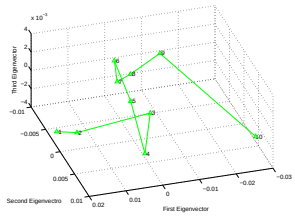
(a) Houses (b) COIL

Figure 5. Criterion function value

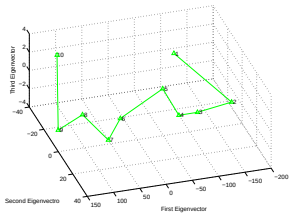
4.2. View-based Object Recognition

We apply the pattern vectors composed of Ihara coefficients to two graph datasets. The first set of graphs are extracted from three sequences of images of model houses, referred to as the CMU, MOVI and Chalet sequences (samples shown in Figure 4(a)). The second set of graphs are extracted from images of objects in the COIL database (samples shown in Figure 4(b)). To establish graphs we first extract corner points using the Harris detector. Then we establish Delaunay graphs based on these corner points as vertices. The established Delaunay graphs are by construction md2 graphs. The edges are weighted with the exponential of the negative distance between two connected vertices, i.e. $w_{ij} = \exp[-k\,||\boldsymbol{x}_i - \boldsymbol{x}_j||]$ where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are coordinates of corner points i and j in an image and k is a scalar scaling factor. The graphs extracted from sample objects are superimposed upon the sample images in Figure 4.

We first explore which coefficients provide the strongest discrimination between the graphs for the different object classes. Although the full set of coefficients associated with a graph can be used to construct a pattern vector, only a subset of the coefficients contribute significantly. Some coefficients may be redundant. Some others may reduce the effectiveness of the clustering algorithm. We thus need to select the subset of salient coefficients, i.e. those that take on distinct values for different classes and exhibit small within class variance. To do this, we compute the between-class scatter $S_b = \sum_{i=1}^{M} N_i(\bar{c}_{k,i} - \bar{c}_k)^2$ and the within-class scatter $S_w = \sum_{i=1}^{M} \sum_{c_{k,i,j} \in C_i} (c_{k,i,j} - \bar{c}_{k,i})^2$ of the individual coefficients, where $\bar{c}_k$ is the mean of the c_k samples, $\bar{c}_{k,i}$ is the mean of the c_k samples in class C_i, N_i is the number of the c_k samples in class C_i and M is the total number of classes. We then use the criterion function $J = (S_b + S_w)/S_w$ to evaluate the performance of individ-

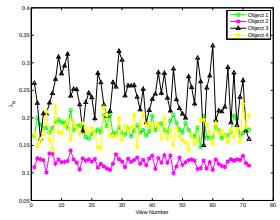


(a) Laplacian spectra

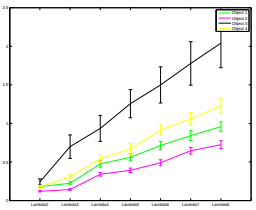


(b) Ihara coefficients

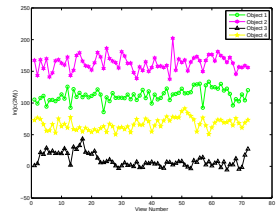
Figure 6. Eigenprojection of graphs from Chalet houses



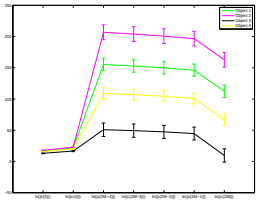
(a) Laplacian spectra



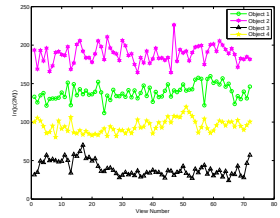
(b) Statistics of Laplacian spectra



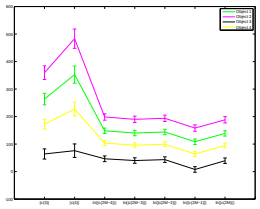
(c) ESP coefficients



(d) Statistics of ESP coefficients



(e) Ihara coefficients

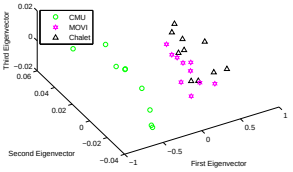


(f) Statistics of Ihara coefficients

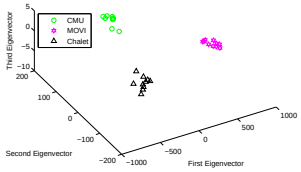
Figure 7. Coefficient data for the COIL dataset

ual coefficients. We select the coefficients according to the criterion that the individual coefficients making the largest contributions to the criterion function are the most significant ones. For each of three selected objects, ten sample images are used as training data to compute the criterion function value. Figures 5(a) and 5(b) show the criterion function values for the coefficients extracted from the house dataset and the COIL dataset respectively. It is clear that the leading few and trailing coefficients contribute more to distinguishing the objects than the intermediate coefficients.

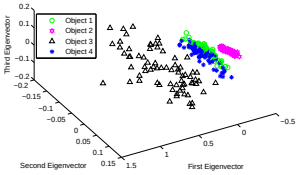
To investigate whether the proposed scheme can learn the structural variation within a graph class, we project the pattern vectors onto the leading three eigenvectors of the



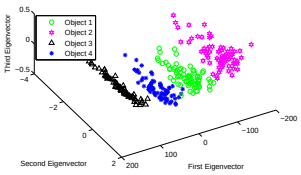
(a) Laplacian spectra on houses



(b) Ihara coefficients on houses



(c) Laplacian spectra on COIL



(d) Ihara coefficients on COIL

Figure 8. Clusters for real-world object classes

class covariance matrix and thus embed the graphs in a pattern space. We use $\boldsymbol{v}_{G2} = [c_3,\ c_4,\ \ln(|c_{2M}|)]^T$ as the pattern vector characterizing the edge-weighted graphs. Figures 6(a) and 6(b) show the projections of the graphs from the Chalet images, based on the Laplacian spectra and $\boldsymbol{v}_{G2}$ respectively. Each point in the pattern space is marked with a view number which corresponds to the camera angle. We can see that the spectral method yields a trajectory hardly tractable. However, the Ihara coefficients produce a clear trajectory and the neighboring images in the sequence are generally close together in the eigenspace.

Next we evaluate the performance of the pattern vectors in distinguishing real-world graph classes. Figure 7 provides some details of the variation of the Laplacian eigenvalues, the ESP's and the Ihara coefficients for graphs extracted from the first four COIL objects in Figure 4(b). In these plots different colored lines correspond to different COIL objects. In the left hand column of Figure 7, we show coefficient values plotted as a function of view numbers for the four objects. In the right hand column of Figure 7, we show the coefficient mean values and standard errors for the four objects over different views as a function of coefficient indices. The main features to note from these plots are that a) both the Ihara coefficients and the ESP's are better separated than the Laplacian eigenvalues, b) there is now little difference between the Ihara coefficients and the ESP's. This latter point is attributable to the highly regular nature of the Delaunay triangulation.

We then use $\boldsymbol{v}_{G2} = [c_3,\ c_4,\ \ln(|c_{2M}|)]^T$ and $\boldsymbol{v}_{G3} = [\ln(|c_{2M-2}|),\ \ln(|c_{2M-1}|),\ \ln(|c_{2M}|)]^T$ as the pattern vectors characterizing the edge-weighted graphs extracted from house sequences and COIL dataset respectively, and embed them into a three-dimensional space using PCA. For

Pattern Vector	Number of Object Classes				
	4	5	6	7	8
Laplacian Spectra	0.94	0.87	0.87	0.86	0.87
Ihara Coefficients	0.99	0.95	0.90	0.88	0.89

Table 1. Rand Indices

comparison, we use the leading three non-zero Laplacian eigenvalues as the spectral pattern vector for the two dataset. Figures 8(a) and 8(b) show the clusters of graphs extracted from the house sequences, produced by the Laplacian spectral method and the Ihara coefficients respectively. Figures 8(c) and 8(d) show the clusters of graphs extracted from the first four COIL objects in Figure 4(b), produced by the Laplacian spectral method and the Ihara coefficients respectively. From Figure 8 we can see that the Ihara coefficients outperform the Laplacian spectral method in producing good clusters.

To take the quantitative evaluation of the pattern vectors, we concentrate our attention on the COIL dataset, and evaluate the clustering performance obtained with different numbers of object classes. After performing PCA on the pattern vectors, we locate the clusters using the K-means method and calculate the Rand index for the resulting clusters. The Rand index is defined as $R_I = Z/(Z+Y)$, where Z is the number of agreements and Y is the number of disagreements in cluster assignment. It takes a value in the interval [0,1], where 1 corresponds to a perfect clustering. The Rand indices for the Laplacian spectral method and for the Ihara coefficients are listed in Table 1. From Table 1 it is clear that the Ihara coefficients outperform Laplacian spectra for all numbers of object classes studied.

5. Conclusions

We have studied how to extract characteristics from edge-weighted graphs using the Ihara zeta function, and have exploited the resulting characterization for the purposes of clustering edge-weighted graphs. We use Ihara coefficients to construct pattern vectors, rather than sampling the zeta function which associates the potential pitfall of encountering infinities. The main contribution in this paper is that we provide a route that allows the Ihara coefficients to be extended from unweighted to edge-weighted graphs. This is achieved by establishing the Perron-Frobenius operator for edge-weighted graphs with the assistance of the Bartholdi zeta function. We perform a spectral analysis that explains why the Ihara coefficients are more effective in distinguishing graph classes than the graph spectral methods. Experiments were conducted on both synthetic and real-world data, and reveal not only that the Ihara coefficients are effective for graph clustering but that they also outperform the Laplacian spectra in graph characterization.

Acknowledgments

We acknowledge the financial support from the FET programme within the EU FP7, under the SIMBAD project (contract 213250).

References

[1] X. Bai, E. R. Hancock, and R. C. Wilson. Graph characteristics from the heat kernel trace. *Pattern Recognition*, 42(11):2589–2606, 2009. 2, 4

[2] H. G. Barrow and R. M. Burstall. Subgraph isomorphism, matching relational structures and maximal cliques. *Information Processing Letters*, 4:83–84, 1976. 1

[3] L. Bartholdi. Counting paths in graphs. *L' Enseignement Mathmatique*, 45:83–131, 1999. 3

[4] H. Bass. The ihara-selberg zeta function of a tree lattice. *International Journal of Mathematics*, 6:717–797, 1992. 1

[5] B. P. Brooks. The coefficients of the characteristic polynomial in terms of the eigenvalues and the elements of an n×n matrix. *Applied Mathematics Letters*, 19:511–515, 2006. 4

[6] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *IJPRAI*, 18(3):265–298, 2004. 1

[7] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. Spatial priors for part-based recognition using statistical models. *In CVPR*, 2005. 1

[8] D. Emms, S. Severini, R. C. Wilson, and E. R. Hancock. Coined quantum walks lift the cospectrality of graphs and trees. *Pattern Recognition*, 42(9):1988–2002, 2009. 2

[9] R. Fergus, P. Perona, and A. Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *IJCV*, 71(3):273–303, 2007. 1

[10] M. Fischler and R. Elschlager. The representation and matching of pictorical structures. *IEEE Trans. Computer*, 22(1):67–92, 1973. 1

[11] Y. Ihara. On discrete subgroups of the two by two projective linear group over p-adic fields. *Journal of Mathematics Society Japan*, 18:219–235, 1966. 1, 2

[12] M. Kotani and T. Sunada. Zeta functions of finite graphs. *Journal of Mathematics University of Tokyo*, 7(1):7–25, 2000. 2

[13] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. *In ICCV*, 2005. 1

[14] T. Liu and D. Geiger. Approximate tree matching and shape similarity. *In ICCV*, 1999. 1

[15] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1

[16] H. Mizuno and I. Sato. Bartholdi zeta function of graph coverings. *Journal of Combinatorial Theory, Series B*, 89:27–41, 2003. 3

[17] P. Ren, R. C. Wilson, and E. R. Hancock. Pattern vectors from the ihara zeta function. *In ICPR*, 2008. 1, 2, 4

[18] G. Scott and C. K. Storm. The coefficients of the ihara zeta function. *Involve - A Journal of Mathematics*, 1(2):217–233, 2008. 4

[19] T. S. Sebastian, P. N. Klein, and B. B. Kimia. Recognition of shapes by editing shock graphs. *In ICCV*, 2001. 1

[20] L. S. Shapiro and J. M. Brady. Feature-based correspondence: an eigenvector approach. *Image and Vision Computing*, 10:283–288, 1992. 1

[21] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE TPAMI*, 22(8):888–905, 2000. 1

[22] R. C. Wilson, B. Luo, and E. R. Hancock. Pattern vectors from algebraic graph theory. *IEEE TPAMI*, 27(7):1112–1124, 2005. 1, 4, 5

[23] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. *In CVPR*, 2008. 1

[24] D. Zhao and X. Tang. Cyclizing clusters via zeta function of a graph. *In NIPS*, 2008. 2

Graph Characteristics from the Ihara Zeta Function

Peng Ren, Richard C. Wilson, and Edwin R. Hancock

Department of Computer Science, The University of York, York, YO10 5DD, UK
{pengren,wilson,erh}@cs.york.ac.uk

Abstract. This paper shows how to extract permutation invariant graph characteristics from the Ihara zeta function. In a previous paper, we have shown that the Ihara zeta function leads to a polynomial characterization of graph structure, and we have shown empirically that the coefficients of the polynomial can be used as to cluster graphs. The aim in this paper is to take this study further by showing how to select the most significant coefficients and how these can be used to gauge graph similarity. Experiments on real-world datasets reveal that the selected coefficients give results that are significantly better than those obtained with the Laplacian spectrum.

1 Introduction

One of the bottlenecks in comparing graphs using graph edit distance[13] is that its computation requires correspondence matches. This problem can be overcome if pattern vectors composed of graph characteristics are used instead, and similarity is measured using the distance between vectors. There are a number of alternative characterizations available. For instance, one could use classical properties of the graph such as diameter, perimeter length or Cheeger number[4] or normalized quantities such as the edge or triangle densities. Several possibilities are offered by spectral graph theory, and these include the Laplacian spectrum[10] or symmetric polynomials computed from the spectral matrix[16].

Recently, however Bai and Hancock have shown that the zeta function of a graph offers an interesting means of characterizing its structure[1]. They have demonstrated that the Rosenberg zeta function[11] is related to the Mellin moments of the heat kernel trace, and have used the moments generated by sampling the Rosenberg zeta function to generate graph feature vectors. The zeta function computed by Bai and Hancock is determined by the Laplacian spectrum. However, in general the zeta function of a graph can be thought of as an analogue of the Riemann zeta function, with prime numbers replaced by prime paths[2]. In fact, the zeta function is a compact representation of information concerning the distribution of paths and path lengths on a graph.

In is this paper, we aim to explore this relationship in a greater depth. We turn to the Ihara zeta function. The Ihara zeta function was first detailed in [7] and [8]. Hashimoto subsequently deduced explicit factorizations for bi-regular bipartite graphs[6]. Bass has generalized Hashimoto's factorization to all finite

N. da Vitora Lobo et al. (Eds.): SSPR&SPR 2008, LNCS 5342, pp. 257–266, 2008.

graphs[2]. In a recent paper we have performed a preliminary study of the Ihara zeta function, and have shown how it can be sampled to give a permutation invariant means of characterizing graphs[12]. However, to be rendered tractable for real world problems the issue of how to avoid sampling the infinities at poles and how to generate stable pattern vectors from the Ihara zeta function must be addressed. We have shown how to overcome this problem by establishing pattern vectors using the polynomial coefficients of the reciprocal Ihara zeta function.

The aim in this paper is to take this work one step further. In particular we aim to study in depth the information conveyed by the individual polynomial coefficients and their relationship to the path length structure of a graph. We show how a reduced set of selected coefficients can be used to efficiently characterize graphs, and that the Euclidean distance between vectors composed of these coefficients can be used to approximate the edit distance between graphs.

2 The Ihara Zeta Function

In this section we review the theory of the Ihara zeta function used in our earlier work. The Ihara zeta function of a graph can be denoted in the form of a rational function[2]:

$$Z_G(u) = \left(1-u^2\right)^{\chi(G)} \det\left(\boldsymbol{I} - u\boldsymbol{A} + u^2\boldsymbol{Q}\right)^{-1} \tag{1}$$

Here, $\chi(G)$is the Euler Number of the graph, which is defined as the difference between the vertex number $|V(G)|$ and the edge number $|E(G)|$ of the graph, i.e. $\chi(G) = |V(G)| - |E(G)|$, and $\boldsymbol{A}$ is the adjacency matrix of the graph. The degree matrix $\boldsymbol{D}$ can be generated by placing the column sums as the diagonal elements, while setting the off-diagonal elements zeros. Finally, $\boldsymbol{Q}$ is the difference of the degree matrix $\boldsymbol{D}$ and the identity matrix $\boldsymbol{I}$, i.e. $\boldsymbol{Q} = \boldsymbol{D} - \boldsymbol{I}$.

3 Polynomial Expression

For md2 graphs, i.e. the graphs with vertex degree at least 2, it is clear-cut that (1) can be rewritten in the form of the reciprocal of a polynomial. However, it is difficult to compute the coefficients of the reciprocal of the Ihara zeta function from (1) in a uniform way, except by resorting to software for symbolic calculation. To efficiently compute these coefficients, it is more convenient to transform the form of the Ihara zeta function in (1) into a concise expression. The Ihara zeta function can also be written in the form of determinant expression[9]

$$Z_G(u) = \frac{1}{\det(\boldsymbol{I} - u\,\boldsymbol{T})} \tag{2}$$

where $\boldsymbol{T}$ is the Perron-Frobenius Operator[15] on the oriented line graph of the original graph, and is an $2m \times 2m$ square matrix with dimensionality m the number of the edges of the original graph. According to (2), the reciprocal of the Ihara zeta function can be rewritten as:

$$\begin{aligned} Z_G^{-1}(u) &= \det(\boldsymbol{I} - u\,\boldsymbol{T}) \\ &= (u)^{2m}\det\left(\frac{1}{u}\boldsymbol{I} - \boldsymbol{T}\right) \\ &= u^{2m}[c_0\left(\frac{1}{u}\right)^{2m} + c_1\left(\frac{1}{u}\right)^{2m-1} + \cdots + c_{2m-1}\left(\frac{1}{u}\right) + c_{2m}] \\ &= c_0 + c_1 u + \cdots + c_{2m-1}u^{2m-1} + c_{2m}u^{2m} \end{aligned} \tag{3}$$

From (3), the coefficients of the reciprocal of the Ihara zeta function can be derived from the coefficients of the characteristic polynomial of the matrix $\boldsymbol{T}$. The calculation of the above coefficients can be converted to a summation of a series of determinants[3]:

$$c_k = \sum_{\binom{2m}{2m-k}} \begin{vmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,2m} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,2m} \\ \cdots & \cdots & \cdots & \cdots \\ b_{2m,1} & b_{2m,2} & \cdots & b_{2m,2m} \end{vmatrix} \tag{4}$$

There are in total $\binom{2m}{2m-k}$ determinants in the sum. The relevant matrix in each determinant is created by replacing $(2m-k)$ of the $2m$ diagonal elements of the matrix $\boldsymbol{T}$ with -1 and the remaining elements in those corresponding rows and columns with 0.

4 Pattern Vectors Using Coefficients as Feature Components

To establish pattern vectors from the Ihara zeta function for the purposes of machine learning, one approach is to consider taking function samples as elements. However, if this strategy is adopted then there is the danger of sampling at poles, and these give rise to infinities. Hence, pattern vectors consisting of function samples are potentially unstable since the distribution of poles is unknown beforehand.

To overcome this problem, we note that the coefficients of the reciprocal of the Ihara zeta function do not give rise to infinities. These coefficients are essentially descriptors of graph structures. As long as G is a simple graph, a) the coefficients c_3, c_4, and c_5 are respectively the negative of twice the number of triangles, squares, and pentagons in G, b) c_6 is the negative of twice the number of hexagons in G plus 4 times the number of pairs of edge disjoint triangles plus twice the number of pairs of triangles with a common edge, and c) c_7 is the negative of twice the number of heptagons in G plus 4 times the number of edge disjoint pairs of one triangle and one square plus twice the number of pairs of one triangle and one square that share a common edge[14]. The highest order coefficient is associated with the number of edges incident to vertex v_i, i.e. the node degree $d(v_i)$:

$$c_{2m} = (-1)^{\chi(G)} \prod_{v_i \in V} (d(v_i) - 1) \tag{5}$$

The set of coefficients can play the role of pattern vectors for clustering, not only because of their immunity to the distribution of the poles, but also due to their ability to characterize the graph structures.

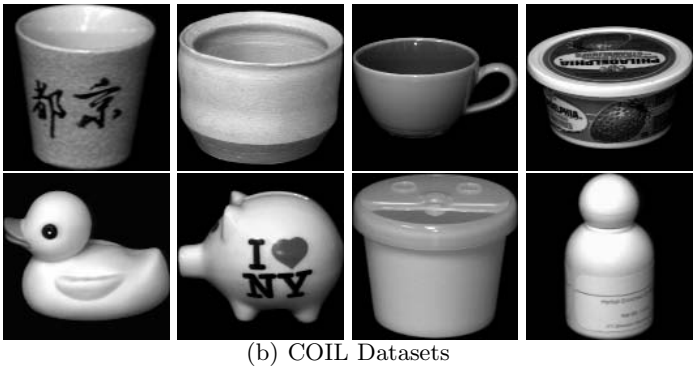
Although the full set of coefficients associated with a graph can be used to construct a pattern vector, only a subset of the coefficients contribute significantly. Some coefficients may be redundant and others may reduce the effectiveness of machine learning algorithms. We thus need to select the subset of salient coefficients, i.e. those that take on distinct values for different classes and exhibit small distinctiveness within class variation. To do this, we compute the between-class scatter $S_b = \sum_{i=1}^{M} N_i(\bar{c}_{k,i} - \bar{c}_k)^2$ and the within-class scatter $S_w = \sum_{i=1}^{M} \sum_{c_{k,i,j} \in C_i} (c_{k,i,j} - \bar{c}_{k,i})^2$ of the individual coefficients, where $\bar{c}_k$ is the mean of all the c_k samples, $\bar{c}_{k,i}$ is the mean of the c_k samples in class C_i, N_i is the number of the c_k samples in class C_i and M is the total number of classes. We then use the criterion function $J = (S_b + S_w)/S_w$ to evaluate the performance of individual coefficients. Individual coefficients which give a large contribution to the criterion function are the most significant.

5 Experiment

5.1 Synthetic Graphs

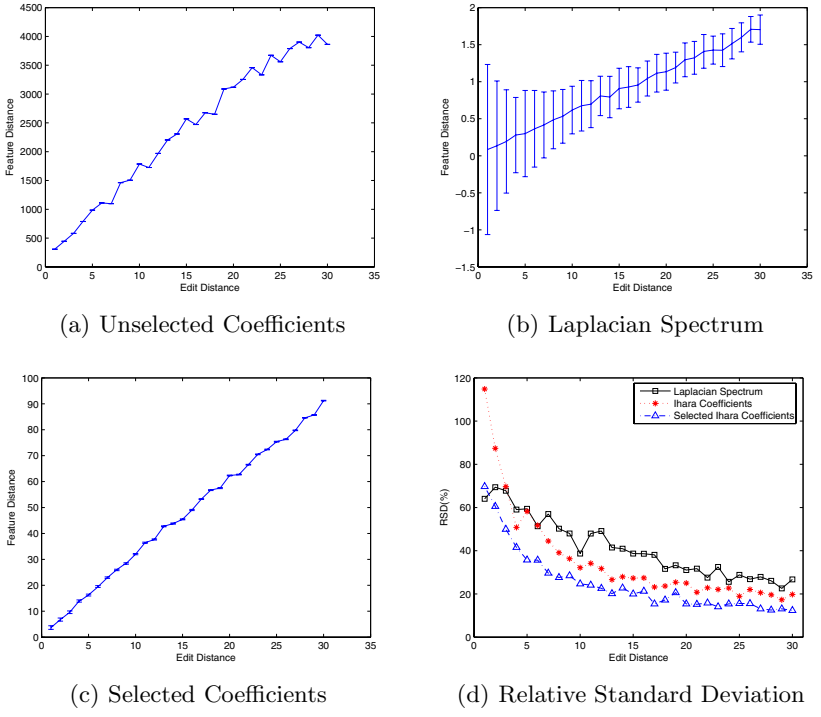
We commence by investigating the relationship between the edit distance and the Euclidean distance between vectors of Ihara coefficients. We begin with a set of randomly generated md2 graphs. The seed graph for the set has 100 vertices

(a) House Sequences



(b) COIL Datasets

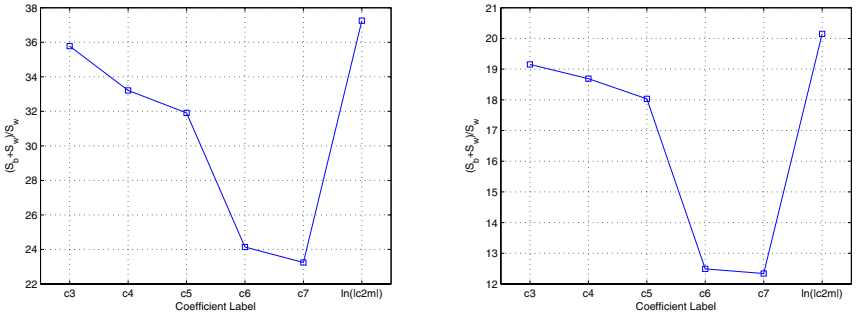
Fig. 1. Datasets for Experiments



(a) Unselected Coefficients (b) Laplacian Spectrum

(c) Selected Coefficients (d) Relative Standard Deviation

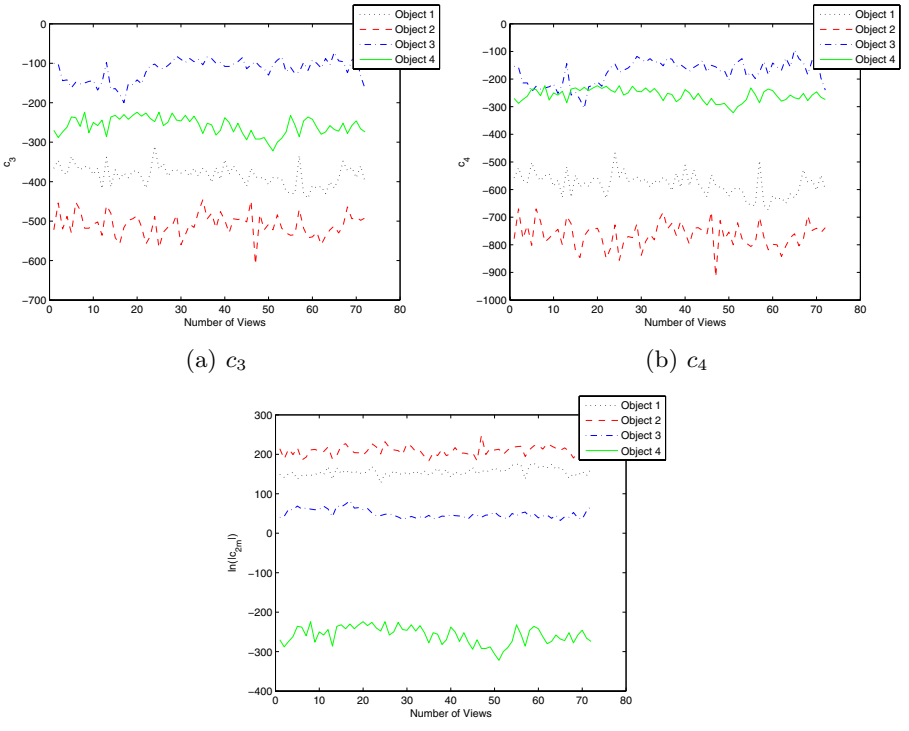
Fig. 2. Feature Distance



(a) COIL (b) Houses

Fig. 3. Criterion Function Value

and 300 edges. The remainder of the graphs in the set are obtained by deleting the edges of the seed graph (indexed from 1 to 30). At each level of editing, 100 trials are performed and the edges deleted are chosen randomly, subject to preserving the md2 constraint. We compare the Euclidean distance between the Ihara coefficients with the both the edit distance and the distance between Laplacian spectra. In our experiments, we compute the Ihara coefficients using (4) and

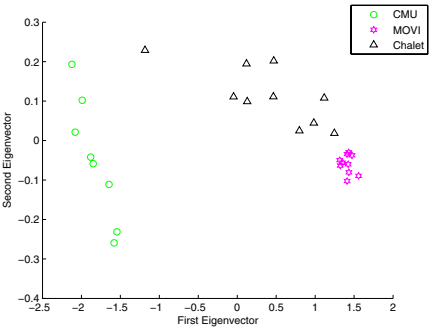


(a) c_3 (b) c_4

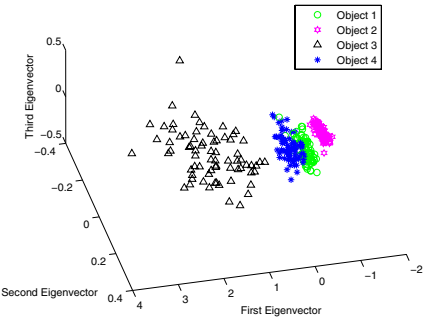
(c) $\ln(c_{2m})$

Fig. 4. Coefficients from COIL

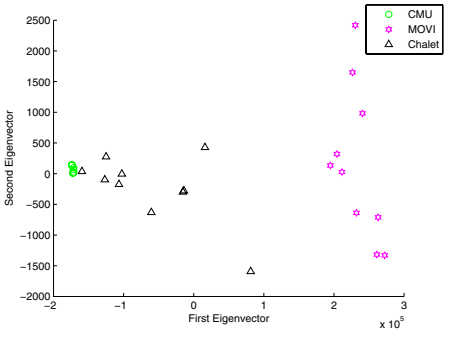
(5), constructing a pattern vector in the form $v_G = [c_3\ c_4\ c_5\ c_6\ c_7\ \ln(|c_{2m}|)]^T$. The final component of the pattern vector is scaled in a logarithmic manner for the purpose of avoiding high order oscillations. The Lapalacian spectral pattern vectors are formed by taking the second smallest through to the seventh smallest eigenvalues of graph Laplacian as components. The experimental results are shown in Fig.2. Fig.2(a) shows the distance between coefficient pattern vectors from the seed graph and the edited graph as a function of the edit distance, i.e. the number of edges deleted. The relative standard deviation (RSD) is also shown as an error bar. The coefficient distance generally follows the edit distance. Fig.2(b) shows the Laplacian spectral distance as well as the corresponding RSD as a function of the edit distance (but scaled differently to Fig.2(a)). From Fig.2(a) and Fig.2(b) it is clear that the dynamic range of the coefficient feature distance is much larger than that of the spectral distance for corresponding edit distance. Thus the coefficients are more sensitive to graph edits than the Laplacian spectra. To evaluate how reliable the coefficient distance and the spectral distance predict the edit distance, we plot their RSD as a function of edit distance in Fig.2(d). The RSD of the coefficient distance is larger than that of the spectral distance. This means that the Laplacian spectra are more stable than the coefficients under graph edits.



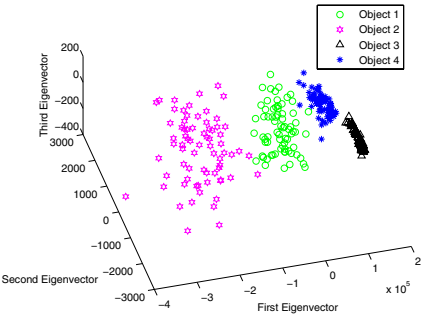
(a) Laplacian Spectrum on Houses



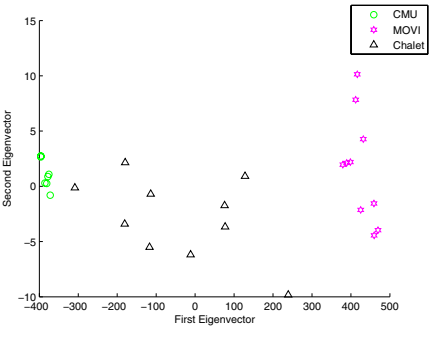
(b) Laplacian Spectrum on COIL



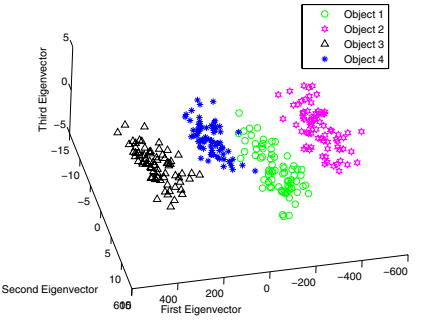
(c) Unselected Coefficients on Houses



(d) Unselected Coefficients on COIL



(e) Selected Coefficients on Houses



(f) Selected Coefficients on COIL

Fig. 5. Clustering Performance

5.2 Visual Clustering

We apply the pattern vectors composed of Ihara coefficients to two graph datasets used previously in the work of Bai and Hancock. The first set of graphs are extracted from three sequences of images of model houses (referred to as the CMU, MOVI and chalet sequences in Fig.1(a)). The second set of graphs are extracted

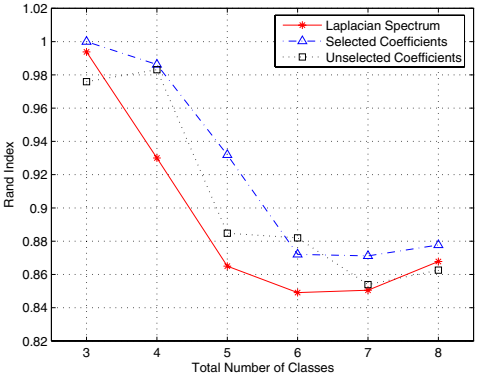


Fig. 6. Rand Index

from images of the objects in COIL database (Fig.1(b)). We first extract corner points using the Harris detector. Then we establish Delaunay graphs based on these corner points as nodes. The pattern vectors are formed as explained in Section 5.1. We perform PCA on the pattern vectors to embed them into a 3-dimensional space. We then locate the clusters using K-means method.

Table 1. Rand Indices

	(a)	(b)	(c)	(d)	(e)	(f)	(g)
House	0.82	0.71	0.83	0.73	0.73	0.83	0.40
COlL	0.99	0.84	0.98	0.79	0.79	0.80	0.66

Table 1 gives the Rand indices obtained when clustering is attempted using different features. The methods are (a) the Laplacian spectrum (second smallest eigenvalue through to the seventh samllest eignevalues), (b) the Rosenberg zeta function used by Bai and Hancock sampled at the integer values 1 to 6, (c) the Ihara zeta function sampled over the range from 0.001 to 0.006 with interval 0.001, (d) the Ihara zeta function sampled over the range from 0.01 to 0.06 with interval 0.01, (e) the Ihara zeta function sampled over the range from 0.11 to 0.16 with interval 0.01 and (f) the Ihara zeta function sampled over the range from 0.1 to 0.6 with interval 0.1. The clusters are located using the K-means method. In Table 1 we can see that the proposed method outperforms the Rosenberg zeta function, and is comparable with the Laplacian spectrum. The final three columns in Table 1 indicate the performance of pattern vectors constructed by sampling the Ihara zeta function. If the samples are appropriately chosen, then this method is comparable with the use of the coefficients. However, when function is sampled in the proximity of poles, the clustering performance deteriorates.

Finally, we explore which combination of coefficients gives the best performance. To do this, we select the coefficients according to the criterion introduced in Section 4.2. We select 3 objects for each of which 10 sample images are used as

training data to compute the criterion function value. Fig.3 shows the criterion function values for the coefficients extracted from the two datasets. The first and last coefficients offer more discrimination than the middle ones. This is because the remaining coefficients provide no significantly increase in information over c_3, c_4 and c_{2m} since they are determined by the number of triangles and squares in the graph and the node degrees. This information can be fully represented by c_3, c_4 and c_{2m}. Based on this feature selection analysis we work with the pattern vector $v_{Gs} = [c_3 \; c_4 \; \ln(|c_{2m}|)]^T$. The three components of v_{Gs} extracted from the first 4 objects in the COIL dataset are shown in Fig.4 as a function of view numbers. Each line in a plot represents the coefficient extracted from one object. The lines in each plot are well separated thus indicating that the three coefficients are sufficient to distinguish different object classes. Fig.2(c) shows the feature distance computed using the three selected coefficients as a function of the edit distance. Compared with Fig.2(a), the selected coefficients are more linear with the edit distance than the unselected coefficients. From Fig.2(d), it is clear that although the unselected coefficients are less stable than the graph spectra, the selected coefficients offer best stability. We apply the pattern vectors of a) Laplacian spectra, b) unselected Ihara coefficients and c) selected Ihara coefficients to both the house sequences and the first four objects in COIL dataset. Fig.4 shows the clustering results obtained by performing PCA on the pattern vectors. The selected coefficients outperform both the Laplacian spectra and the unselected coefficients,and give clusters with better separation.

We then confine our attention to the COIL dataset, and evaluate the clustering performance obtained with different numbers of object classes. After performing PCA on the pattern vectors, we locate the clusters using the K-means method and calculate the Rand index. The Rand index for each pattern vector is plotted as a function of class number in Fig.4. The selected coefficients gives the best performance and the Laplacian spectra the poorest performance.

6 Conclusion

In this paper, we show how to construct pattern vectors for graph characterization using the Ihara zeta function. We compute polynomial coefficients using the reciprocal of the Ihara zeta function. We construct pattern vectors by performing feature selection on the coefficients. Unlike the samples of the Ihara zeta function, the coefficients are not prone to singularities due to poles. The method outperforms the Laplacian spectrum both in terms of stability and in clustering performance on md2 graphs.

References

1. Bai, X., Hancock, E.R.: Recent results on heat kernel embedding of graphs. In: Brun, L., Vento, M. (eds.) GbRPR 2005. LNCS, vol. 3434, pp. 373–382. Springer, Heidelberg (2005)
2. Bass, H.: The Ihara-Selberg zeta function of a tree lattice. International Journal of Mathematics 6(3), 717–797 (1992)

3. Brooks, B.P.: The coefficients of the characteristic polynomial in terms of the eigenvalues and the elements of an n×n matrix. Applied Mathematics letters 19(6), 511–515 (2006)
4. Chung, F.K.: Spectral graph theory. American Mathematical Society (1997)
5. Haemers, W.H., Spence, E.: Enumeration of cospectral graphs. European Journal of Combinatorics 25(2), 199–211 (2004)
6. Hashimoto, K.: Zeta functions of finite graphs and representations of p-adic groups. Advanced Study of Pure Mathematics 15, 211–280 (1989)
7. Ihara, Y.: Discrete subgroups of PL(2, k_φ). In: Proceeding Symposium of Pure Mathematics, pp. 272–278 (1965)
8. Ihara, Y.: On discrete subgroups of the two by two projective linear group over p-adic fields. Journal of Mathematics Society Japan 18, 219–235 (1996)
9. Kotani, M., Sunada, T.: Zeta functions of finite graphs. Journal of Mathematics University of Tokyo 7(1), 7–25 (2000)
10. Luo, B., Wilson, R.C., Hancock, E.R.: Spectral embedding of graphs. Pattern Recognition 36(10), 2213–2223 (2003)
11. Rosenberg, S.: The Laplacian on a Riemannian manifold. Cambridge University Press, Cambridge (2002)
12. Ren, P., Wilson, R.C., Hancock, E.R.: Pattern vectors from the Ihara zeta function. In: IEEE International Conference on Pattern Recognition (submitted, 2008)
13. Sanfeliu, A., Fu, K.S.: A distance measure between attributed relational graphs for pattern recognition. IEEE Transanctions on Systems, Man, and Cybernetics 13, 353–362 (1983)
14. Scott, G., Storm, C.K.: The coefficients of the Ihara zeta function. Involve - A Journal of Mathematics 1(2), 217–233 (2008)
15. Storm, C.K.: The zeta function of a hypergraph. Electronic Journal of Combinatorics 13(1) (2006)
16. Wilson, R.C., Luo, B., Hancock, E.R.: Pattern vectors from algebraic graph theory. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(7), 1112–1124 (2005)

Characteristic Polynomial Analysis on Matrix Representations of Graphs

Peng Ren, Richard C. Wilson, and Edwin R. Hancock

Department of Computer Science, The University of York, York, YO10 5DD, UK
{pengren, wilson, erh}@cs.york.ac.uk

Abstract. Matrix representations for graphs play an important role in combinatorics. In this paper, we investigate four matrix representations for graphs and carry out an characteristic polynomial analysis upon them. The first two graph matrices are the adjacency matrix and Laplacian matrix. These two matrices can be obtained straightforwardly from graphs. The second two matrix representations, which are newly introduced [9][3], are closely related with the Ihara zeta function and the discrete time quantum walk. They have a similar form and are established from a transformed graph, i.e. the oriented line graph of the original graph. We make use of the characteristic polynomial coefficients of the four matrices to characterize graphs and construct pattern spaces with a fixed dimensionality. Experimental results indicate that the two matrices in the transformed domain perform better than the two matrices in the original graph domain whereas the matrix associated with the Ihara zeta function is more efficient and effective than the matrix associated with the discrete time quantum walk and the remaining methods.

1 Introduction

Pattern analysis using graph structures has proved to be a challenging and sometimes elusive problem. The main reason for this is that graphs are not vectorial in nature, and hence they are not amenable to classical statistical methods from pattern recognition or machine learning [7]. One way to overcome this problem is to extract feature vectors from graphs which succinctly capture their structure in a manner that is permutation invariant. There are a number of ways in which this may be accomplished. One approach is to use simple features such as the numbers of edges and nodes, edge density or diameters. A more sophisticated approach is to count the numbers of cycles of different order. Alternatively graph-spectra can be used [7][10].

However, one elegant way in which to capture graph structure is to compute the characteristic polynomial. To do so requires a matrix characterization $\boldsymbol{M}$ of the graph, and the characteristic polynomial is the determinant $\det(\lambda \boldsymbol{I} - \boldsymbol{M})$ where $\boldsymbol{I}$ is the identity matrix and λ the variable of the polynomial. The simplest way to exploit the characteristic polynomial is to use its coefficients. With an appropriate choice of matrix $\boldsymbol{M}$, these coefficients are determined by the cycle frequencies in the graph. They are also easily computed from the spectrum of $\boldsymbol{M}$. Moreover, since it is determined by the numbers of cycles in a graph, the characteristic polynomial may also be used to define an analogue of the Riemann zeta function from number theory for a graph. Here the zeta function is

A. Torsello, F. Escolano, and L. Brun (Eds.): GbRPR 2009, LNCS 5534, pp. 243–252, 2009.

determined by the reciprocal of the characteristic polynomial, and prime cycles determine the poles of the zeta function in a manner analogous to the prime numbers. The recent work of Bai *et al*[2] and Ren *et al*[8][9] has shown that practical characterizations can be extracted from different forms of the zeta function and used for the purposes of graph-based object recognition. Finally, it is interesting to note that if the matrix $\boldsymbol{M}$ is chosen to be the adjacency matrix $\boldsymbol{T}$ of the oriented line graph derived from a graph, which is also called the Perron-Frobenius operator, then the characteristic polynomial is linked to the Ihara zeta function of the original graph.

As noted above, the characteristic polynomial is determined by the choice of the matrix $\boldsymbol{M}$. Here there are a number of alternatives including the adjacency matrix $\boldsymbol{A}$, the Laplacian matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ where $\boldsymbol{D}$ is the node degree matrix, and the Perron-Frobenius operator $\boldsymbol{T}$ where the graph is transformed prior to the computation of the characteristic polynomial. To compute the Iharra zeta function the oriented line graph is first constructed and then the characteristic polynomial is computed from its adjacency matrix. This is similar to the approach taken by Emms *et al* [3] in their study of discrete time quantum walks. However, rather than characterizing the oriented line graph using the adjacency matrix, they construct a unitary matrix $\boldsymbol{U}$ which captures the transitions of a quantum walk controlled by a Grover coin. The resulting unitary matrix proves to be a powerful tool for analyzing graphs since the spectrum of the positive support of its third power (denoted by $\mathrm{sp}(S^{+}(\boldsymbol{U}^3))$ can be used to resolve structural ambiguities due to the cospectrality of strongly regular graphs.

The aim in this paper is to explore the roles of matrix graph representation in the construction of characteristic polynomials. In particular we are interested in which combination is most informative in terms of graph-structure and which gives the best empirical performance when graph clustering is attempted using the characteristic polynomial coefficients. We study both the original graph and its oriented line graph. The matrix characterizations used are the adjacency matrix $\boldsymbol{A}$, the Laplacian matrix $\boldsymbol{L}$, the transition matrix $\boldsymbol{T}$ and the unitary characterization $\boldsymbol{U}$.

2 Classical Graph Matrix Representations

To commence, suppose that the graph under study is denoted by $G = (V, E)$ where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. Since we wish to adopt a graph spectral approach we introduce the **adjacency matrix** $\boldsymbol{A}$ for the graph where the elements are

$$\boldsymbol{A}(u,v) = \begin{cases} 1 \text{ if } u,v \in E \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

We also construct the diagonal degree matrix $\boldsymbol{D}$, whose elements are given by $\boldsymbol{D}(u,u) = d_u = \sum_{u \in V} A(u,v)$. From the degree matrix and the adjacency matrix we construct the **Laplacian matrix** $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, i.e. the degree matrix minus the adjacency matrix

$$\boldsymbol{L}(u,v) = \begin{cases} d_u \text{ if } u = v \\ 1 \text{ if } u,v \in E \text{ but } u \neq v \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

3 The Ihara Zeta Function

The Ihara zeta function for a graph is a generalization of the Riemann zeta function from number theory. In the definition of the Ihara zeta function, the 'prime number' in the Euler product expansion of the Riemann zeta function is replaced by a 'prime cycle', i.e. cycles with no backtracking in a graph. The definition of the Ihara zeta function of a graph $G(V, E)$ is a product form which runs over all the possible prime cycles

$$Z_G(u) = \prod_{[p]} \left(1 - u^{L(p)}\right)^{-1} \tag{3}$$

Here, p denotes a prime cycle and $L(p)$ denotes the length of p. As shown in (3), the Ihara zeta function is generally an infinite product. However, one of its elegant features is that it can be collapsed down into a rational function, which renders it of practical utility.

3.1 Rational Expression

For a graph $G(V, E)$ with the vertex set V of cardinality $|V| = N$ and the edge set E of cardinality $|E| = M$, the rational expression of the Ihara zeta function is [4][5]:

$$Z_G(u) = \left(1 - u^2\right)^{\chi(G)} \det\left(\boldsymbol{I}_N - u\boldsymbol{A} + u^2\boldsymbol{Q}\right)^{-1} \tag{4}$$

Here, $\chi(G)$is the Euler number of the graph $G(V, E)$, which is defined as the difference between the vertex number and the edge number of the graph, i.e. $\chi(G) = N - M$, $\boldsymbol{A}$ is the adjacency matrix of the graph, $\boldsymbol{I}_k$ denotes the $k \times k$ identity matrix, and $\boldsymbol{Q}$ is the matrix difference of the degree matrix $\boldsymbol{D}$ and the identity matrix $\boldsymbol{I}_N$, i.e. $\boldsymbol{Q} = \boldsymbol{D} - \boldsymbol{I}_N$. From (4) it has been shown that the Ihara zeta function is permutation invariant to vertex label permutations [9]. This is because permutation matrices, which represent vertex label permutations in matrix calculation, have no effect on the determinant in (4).

3.2 Determinant Expression

For md2 graphs, i.e. the graphs with vertex degree at least 2, it is straightforward to show that (4) can be rewritten in the form of the reciprocal of a polynomial. However, it is difficult to compute the coefficients of the reciprocal of the Ihara zeta function from (4) in a uniform way, except by resorting to software for symbolic calculation. To efficiently compute these coefficients, it is more convenient to transform the rational form of the Ihara zeta function in (4) into a concise expression. The Ihara zeta function can also be written in the form of a determinant[6]:

$$Z_G(u) = \det(\boldsymbol{I}_{2M} - u\boldsymbol{T})^{-1} \tag{5}$$

where $\boldsymbol{T}$ is the Perron-Frobenius operator on the oriented line graph of the original graph, and is an $2M \times 2M$ square matrix.

To obtain the Perron-Frobenius operator $\boldsymbol{T}$, we must construct the oriented line graph of the original graph from the associated symmetric digraph. The symmetric digraph

$DG(V, E_d)$ of a graph $G(V, E)$ is composed of a finite nonempty vertex set V identical to that of $G(V, E)$ and a finite multiset E_d of oriented edges called arcs, which consist of ordered pairs of vertices. For arc $e_d(u, w) \in E_d$ where u and v are elements in V, the origin of $e_d(u, w)$ is defined to be $o(e_d) = u$ and the terminus is $t(e_d) = v$. Its inverse arc, which is formed by switching the origin and terminus of $e_d(u, w)$, is denoted by $e_d(w, u)$. For the graph $G(V, E)$, we can obtain the associated symmetric digraph $SDG(V, E_d)$ by replacing each edge of $G(V, E)$ with the arc pair in which the two arcs are inverse to each other.

The oriented line graph associated with the original graph can be defined using the symmetric digraph. It is a dual graph of the symmetric digraph since its oriented edge set and vertex set are constructed from the vertex set and the oriented edge (arc) set of its corresponding symmetric digraph. The construction of the oriented edge set and the vertex set of the oriented line graph can be formulated as follows:

$$\begin{cases} V_L &= E_d(SDG) \\ E_{dL} &= \{(e_d(u, v), e_d(v, w)) \\ &\qquad \in E_d(SDG) \times E_d(SDG);\ u \neq w\} \end{cases} \tag{6}$$

The Perron-Frobenius operator $\boldsymbol{T}$ of the original graph is the adjacency matrix of the associated oriented line graph. For the $(i, j)th$ entry of $\boldsymbol{T}$, $\boldsymbol{T}(i, j)$ is 1 if there is one edge directed from the vertex with label i to the vertex with label j in the oriented line graph, and is 0 otherwise.

Unlike the adjacency matrix of an undirected graph, the Perron-Frobenius operator is not a symmetric matrix. This is because of a constraint that arises in the construction of oriented edges. Specifically, the arc pair with two arcs that are the reverse of one another in the symmetric digraph are not allowed to establish an oriented edge in the oriented line graph. This constraint arises from the second requirement in the edge definition appearing in (6).

The Perron-Frobenius operator $\boldsymbol{T}$ is matrix representation which can convey the information contained in the Ihara zeta function for a graph. It is the adjacency matrix of the oriented line graph associated with the original graph. As $\boldsymbol{T}$ is not symmetric, the Laplacian form of the Perron-Frobenius operator can not be uniformly defined because the relevant vertex degree can be calculated from either incoming or outgoing edges in the oriented line graph which is a directed graph. In this study, we consider three types of Laplacian matrices of the Perron-Frobenius operator. They are defined as the incoming degree matrix minus the $\boldsymbol{T}$, the outgoing degree matrix minus $\boldsymbol{T}$ and the sum of both incoming degree matrix and the outgoing degree matrix minus $\boldsymbol{T}$, respectively.

4 The Discrete Time Quantum Walks

The discrete-time quantum walk is the quantum counterpart of the discrete-time classical random walk and has been used in the design of new quantum algorithms on graphs [1]. Quantum processes are reversible, and in order to make the discrete-time quantum walk reversible a particular state must specify both the current and previous location of the walk. To this end each edge of the graph $G(V, E)$, $e(u, v) \subset E$, is replaced by a pair of arcs $e_d(u, v)$ and $e_d(v, u)$, and the set of these arcs is denoted by E_d. This is the same

as the intermediate step of constructing the digraph to compute the determinant of the Ihara zeta function. The state space for the discrete-time quantum walk is the set of arcs E_d. If the walk is at vertex v having previously been at vertex u with probability 1, then the state is written as $|\psi\rangle = |uv\rangle$. Transitions are possible form one arc $e_d(w,x)$ to another arc $e_d(u,v)$, i.e. from a state $|wx\rangle$ to $|uv\rangle$, and only if $x = u$ and x is adjacent to v. Note that this corresponds to only permitting transitions between adjacent vertices. The state vector for the walk is a quantum superposition of states on single arcs of the graph, and can be written as

$$|\psi\rangle = \sum_{e_d(u,v)\in E_d} \alpha_{uv}|uv\rangle \tag{7}$$

where the quantum amplitudes are complex, i.e. $\alpha \in \mathbb{C}$. Using (7), the probability that the walk is in the state $|uv\rangle$ is given by $\Pr(|uv\rangle) = \alpha_{uv}\alpha^*_{uv}$. As with the classical walk,the evolution of the state vector is determined by a matrix, in this case denoted $\boldsymbol{U}$, according to $|\psi_{t+1}\rangle = \boldsymbol{U}|\psi_t\rangle$. Since the evolution of the walk is linear and conserves probability the matrix $\boldsymbol{U}$ must be unitary. That is, the inverse is equal to the complex conjugate of the matrix transposed, i.e. $\boldsymbol{U}^{-1} = \boldsymbol{U}^{\dagger}$. The entries of $\boldsymbol{U}$ determine the probabilities for transitions between states. Thus, there are constraints on these entries and there are therefore constraints on the permissible amplitudes for the transitions. The sum of the squares of the amplitudes for all the transitions from a particular state must be unity. Consider a state $|\psi\rangle = |u_1 v\rangle$ where the neighborhood of v, $\mathcal{N}(v) = u_1, u_2, \cdots, u_r$. A single step of the walk should only assign non-zero quantum amplitudes to transitions between adjacent states, i.e.the states $|vu_i\rangle$ where $u_i \in \mathcal{N}$. However, since $\boldsymbol{U}$ must be unitary these amplitudes cannot all be the same. Recall that the walk does not rely on any labeling of the edges or vertices. Thus, the most general form of transition will be one that assigns the same amplitudes to all transitions $|u_1 v\rangle \rightarrow |vu_i\rangle$, $u_i \in \mathcal{N} \setminus u_1$, and a different amplitude to the transition $|u_1 v\rangle \rightarrow |vu_1\rangle$. The second of these two transitions corresponds to the walk returning along the same edge to which it came. Thus, the transition will be of the form

$$|u_1 v\rangle \rightarrow |vu_1\rangle + b\sum_{i=2}^{r}|vu_i\rangle, \quad a, b \in \mathbb{C} \tag{8}$$

It is usual to use the Grover diffusion matrices which assign quantum amplitudes of $a = 2/d_v - 1$ when the walk returns a long the same edge and $b = 2/d_v$ for all other transitions. Such matrices are used as they are the matrices furthest from the identity which are unitary and are not dependent on any labeling of the vertices.

Using the Grover diffusion matrices, the matrix, $\boldsymbol{U}$, that governs the evolution of the walk has entries

$$\boldsymbol{U}_{(u,v),(w,x)} = \begin{cases} \frac{2}{d_x} - \delta_{vw} & \text{if } u = x \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Note that the state transition matrix $\boldsymbol{U}$ in discrete time quantum walk and the Perron-Frobenius operator $\boldsymbol{T}$ in the Ihara zeta function have has similar form. They are of the same dimensionality for a graph. Specifically, all the non-zero entries of $\boldsymbol{T}$ are 1 while the same entries in $\boldsymbol{U}$ are weighted twice of the reciprocal of the connecting

vertex degree in the original graph. Additionally, the entries representing reverse arcs in $\boldsymbol{T}$ generally have a non-zero value $2/d_x - 1$ while the same entries in $\boldsymbol{T}$ are always set zero.

In [3], the spectrum of the positive support of third power of $\boldsymbol{U}$ (denoted by $\mathrm{sp}(S^+(\boldsymbol{U}^3))$ is shown to distinguish cospectral graphs. Thus, $\mathrm{sp}(S^+(\boldsymbol{U}^3))$ proves an effective graph representation matrix.

5 Characteristic Polynomials

Once the graph representation matrices are to hand, our task is how to characterize graphs using different matrix representations and thus distinguish graphs from different classes. One simple but effective way to embed graphs into a pattern space is furnished by spectral methods [7]. The eigenvalues of the representation matrices are used as the elements of graph feature vectors. However, graphs with different sizes have different numbers of eigenvalues. There are generally two ways to overcome this problem. The first is to establish the patten space with a dimensionality the same as the cardinality of the vertex set of the graph with the greatest size. Feature vectors of the smaller graphs are adjusted to the same length by padding zeros before the non-zero eigenvalues up to the dimension of the pattern space. One drawback of this method is that the upper bound on the dimension, i.e. the size of the largest graph, should be known beforehand. Furthermore, for a pattern space with a high dimensionality, there would be too many unnecessary zeros in the feature vectors of small graphs. The second method for dealing with size difference of graphs is spectral truncation. In this case, a fix-sized subset of eigenvalues for the different graphs are used to establish feature vectors. For example, a fixed number of the leading non-zero eigenvalues are chosen as the elements of a feature vector. This method does not require prior knowledge of the size of the largest graph. Nevertheless, it only takes advantage of a fraction of the spectral information available and thus induces varying degrees information loss. To overcome these drawbacks associated with traditional spectral methods, we take advantage of the characteristic polynomial of the representation matrices. The characteristic polynomial $p(\lambda)$ of a matrix $\boldsymbol{M}$ with size N is defined as follows

$$p(\lambda) = \det(\lambda \boldsymbol{I} - \boldsymbol{M}) = c_0\lambda^N + c_1\lambda^{N-1} + \cdots + c_{N-1}\lambda + c_N \tag{10}$$

From (10), the characteristic polynomial of a matrix $\boldsymbol{M}$ is a function with variable λ. The roots $\{\lambda_1, \lambda_2 \cdots \lambda_N\}$ of the equation $p(\lambda) = 0$ is the set of eigenvalues of the matrix $\boldsymbol{M}$, i.e. the spectrum of $\boldsymbol{M}$. The key point here is that there is a close relationship between the roots and the polynomial coefficients as follows

$$c_r = (-1)^r \sum_{k_1 < k_2 < \ \ldots \ < k_r} \lambda_{k_1}\lambda_{k_2} \ \ldots \ \lambda_{k_r} \tag{11}$$

Since these coefficients are closely related to the spectrum, they can be regarded as a possible way to characterize graphs. Here we propose to use the characteristic polynomial coefficients as the elements of the feature vector for a graph, instead of the eigenvalues. In this way we embed the graphs into a pattern space using the feature vectors

based on the characteristic polynomial coefficients. The merit of using the characteristic polynomial coefficients over spectral embedding methods is that the coefficients naturally take advantage of the complete spectrum and thus do not induce spectral truncation. Hence the dimensionality of the pattern space can be determined without taking into account the graph size differences.

6 Experimental Results

We experiment with the proposed feature vectors consisting of characteristic polynomial coefficients on graphs extracted from the COIL database (samples shown in Figure 1). We first extract corner points using the Harris detector. Then we establish Delaunay graphs based on these corner points as nodes. The graphs extracted from sample objects are superimposed upon the sample images in Figure 1.

We choose to work with the coefficient subset $\{c_3,\ c_4,\ c_{N-3},\ c_{N-2},\ c_{N-1},\ c_N\}$ because these coefficients tend to be the most salient ones in the relevant matrix representations [8]. We establish the feature vector at two levels of scales, i.e. scaling the last four coefficients by natural logarithm $v_1 = \{c_3,\ c_4, \ln(c_{N-3}), \ln(c_{N-2}), \ln(c_{N-1}), \ln(c_N)\}^T$ and scaling all the coefficients by natural logarithm $v_2 = \{\ln(c_3),\ \ln(c_4), \ln(c_{N-3}), \ln(c_{N-2}), \ln(c_{N-1}), \ln(c_N)\}^T$. We conduct tests on the feature vectors consisting of the characteristic polynomial coefficients on the following alternative matrices:

(a) Adjacency matrix of the original graph;
(b) Laplacian matrix of the original graph;
(c) Adjacency matrix of the oriented lined graph (i.e. the Perron-Frobenius operator $\boldsymbol{T}$ in the Ihara zeta function);
(d) Laplacian matrix associated with incoming vertex degree of the oriented lined graph;
(e) Laplacian matrix associated with outgoing vertex degree of the oriented lined graph;
(f) Laplacian matrix associated with the sum of incoming and outgoing vertex degree of the oriented lined graph;
(g) The positive support of third power of the state transition matrix $\boldsymbol{U}$ of discrete random walks (i.e. $\mathrm{sp}(S^+(\boldsymbol{U}^3))$

We perform PCA on the pattern vectors to embed them into a 3-dimensional space. We then locate the clusters using K-means method and calculate the Rand index for the resulting clusters. The Rand index is defined as $R_I = Z/(Z+Y)$, where Z is the number of agreements and Y is the number of disagreements in cluster assignment. It takes a value in the interval [0,1], where 1 corresponds to a perfect clustering.

There are 72 view images of each object in COIL database. The original image size is 128×128. For the extracted delaunay graphs with more than 120 vertices and with average vertex degree 5.6, the intermediate and higher coefficients of the characteristic polynomial of (d)(e)(f) tend to be larger than 1.79×10^{308} and are not suitable for matlab computation. Therefore, for the first set of experiments we resize the images in to COIL database to a resolution 70×70 to reduce the number of detected corners and hence to ensure that the computations do not overflow memory.

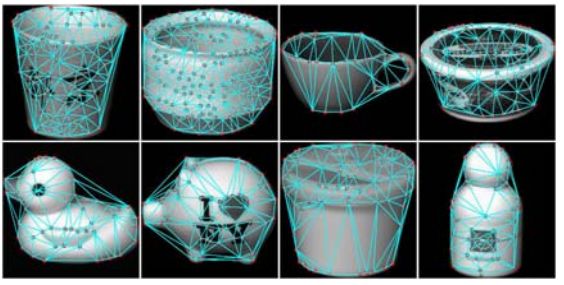


Fig. 1. Datasets for Experiments

Table 1. Rand Indices for v_1 on 70×70 images

Pattern Vector	Number of Object Classes				
	4	5	6	7	8
(a)	0.8595	0.8522	0.8269	0.8233	0.8348
(b)	0.7185	0.7343	0.7302	0.7436	0.7792
(c)	0.8942	0.8319	0.8233	0.8291	0.8450
(d)	0.7076	0.6969	0.7292	0.7747	0.7717
(e)	0.7076	0.6969	0.7292	0.7747	0.7717
(f)	0.7076	0.6969	0.7292	0.7747	0.7717
(g)	0.7048	0.6637	0.6897	0.6979	0.7567

Table 2. Rand Indices for v_2 on 70×70 images

Pattern Vector	Number of Object Classes				
	4	5	6	7	8
(a)	0.8764	0.8321	0.8090	0.8067	0.8165
(b)	0.9301	0.8501	0.8177	0.8171	0.8459
(c)	0.9300	0.8408	0.8246	0.8182	0.8429
(d)	0.9018	0.8509	0.8111	0.8073	0.8405
(e)	0.9101	0.8327	0.8173	0.8214	0.8481
(f)	0.9234	0.8318	0.8208	0.8250	0.8491
(g)	0.9296	0.8434	0.8237	0.8259	0.8514

Table 3. Rand Indices for v_1 on 128×128 images

Pattern Vector	Number of Object Classes				
	4	5	6	7	8
(a)	0.9864	0.8522	0.8269	0.8233	0.8348
(b)	0.8382	0.8351	0.7923	0.7953	0.7797
(c)	0.9897	0.9319	0.8877	0.8757	0.8865

Table 4. Rand Indices for v_2 on 128×128 images

Pattern Vector	Number of Object Classes				
	4	5	6	7	8
(a)	0.9794	0.9403	0.8854	0.8663	0.8744
(b)	0.9897	0.9277	0.8885	0.8801	0.8733
(c)	0.9897	0.9334	0.9000	0.8845	0.8921

Table 5. Rand Indices for traditional spectral methods on 128×128 images

Pattern Vector	Number of Object Classes				
	4	5	6	7	8
Laplacian Spectra	0.9245	0.8658	0.8534	0.8496	0.8601
Quantum Spectra	0.9897	0.9263	0.8920	0.8779	0.8789
Heat Contents	0.9897	0.9251	0.8995	0.8776	0.8891

The experimental results for the two types of scaled feature vectors on the resized images are shown in Table 1 and Table 2. From Table 1 and Table 2 we can see that although the within-class variation of c_3 and c_4 is reasonably small, the scheme in which coefficients are scaled by the natural logarithm behaves slightly better. For feature vector v_1, the adjacency matrix of the original graph and the Perron-Frobenius operator $\boldsymbol{T}$ perform better than the alternatives. For feature vector v_2 each of the matrix representations has a similar performance in distinguishing graph classes.

Furthermore, we test our methods on the images with original size. In this case, the characteristic polynomial coefficients of the Laplacian matrices for oriented line graphs and that of quantum walk $\mathrm{sp}(S^{+}(\boldsymbol{U}^3))$ do not work well due to their computation inefficiency. Table 3 gives the results using v_1 on the adjacency matrix together with Laplacian matrix for original graphs and the adjacency matrix for the oriented line graph. Here we can see that the matrix representation in the transformed domain (i.e. oriented line graph) performs much better than that in original domain. As far as v_2 is concerned, the Perron-Frobenius operator is also generally better than the traditional matrix representations. To compare the proposed polynomial methods with the traditional methods, we list the results for traditional spectral methods in Table 5. Among these three method, the first two only take advantage of the eigenvalues while the heat contents involve the information contained both in the eigenvalues and the eigenvectors. We can see that the results obtained using characteristic polynomial coefficients of the oriented line graph are better than those obtained using the traditional methods.

7 Conclusion

We have performed a characteristic polynomial analysis on four matrix representations for graphs. We argue that the polynomial coefficients perform better than graph spectra in distinguishing graph classes. For graphs of large size, the characteristic polynomial coefficients of the Laplacian of the oriented line graphs and those of the quantum walk

represented $\mathrm{sp}(S^+(\boldsymbol{U}^3))$ are less efficient to compute due to their extremely large dynamic range. On the other hand, the coefficients for the Perron-Frobenius operator do not have this problem. For reasonably large graphs (given size), the coefficients of the Perron-Frobenius operator perform better than the alternative methods described in this paper. Above all, the Perron-Frobenius operator is superior in computational efficiency and is effective in characterizing graphs among the four matrix representations, from the characteristic polynomial point of view.

Acknowledgments

We acknowledge the financial support from the FET programme within the EU FP7, under the SIMBAD project (contract 213250).

References

1. Aharonov, D., Ambainis, A., Kempe, J., Vazirani, U.: Quantum walks on graphs. In: Proceedings of ACM Theory of Computing (2001)
2. Bai, X., Hancock, E.R., Wilson, R.C.: Graph characteristics from the heat kernel trace. In: Pattern Recognition (2009) (to appear)
3. Emms, D., Severini, S., Wilson, R.C., Hancock, E.R.: Coined quantum walks lift the cospectrality of graphs and trees. In: Proceedings of SSPR (2008)
4. Ihara, Y.: Discrete subgroups of pl(2, k_φ). In: Proceeding Symposium of Pure Mathematics, pp. 272–278 (1965)
5. Ihara, Y.: On discrete subgroups of the two by two projective linear group over p-adic fields. Journal of Mathematics Society Japan 18, 219–235 (1966)
6. Kotani, M., Sunada, T.: Zeta functions of finite graphs. Journal of Mathematics University of Tokyo 7(1), 7–25 (2000)
7. Luo, B., Wilson, R.C., Hancock, E.R.: Spectral embedding of graphs. Pattern Recognition 36(10), 2213–2223 (2003)
8. Ren, P., Wilson, R.C., Hancock, E.R.: Graph characteristics from the ihara zeta function. In: Proceedings of SSPR (2008)
9. Ren, P., Wilson, R.C., Hancock, E.R.: Pattern vectors from the ihara zeta function. In: Proceedings of The 19th International Conference of Pattern Recognition (2008)
10. Wilson, R.C., Luo, B., Hancock, E.R.: Pattern vectors from algebraic graph theory. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(7), 1112–1124 (2005)

Hypergraphs, Characteristic Polynomials and the Ihara Zeta Function

Peng Ren[1], Tatjana Aleksić[2], Richard C. Wilson[1], and Edwin R. Hancock[1]

[1] Department of Computer Science, The University of York, York, YO10 5DD, UK
{pengren,wilson,erh}@cs.york.ac.uk
[2] University of Kragujevac, Faculty of Science, 34000 Kragujevac, Serbia
taleksic@kg.ac.rs

Abstract. In this paper we make a characteristic polynomial analysis on hypergraphs for the purpose of clustering. Our starting point is the Ihara zeta function [8] which captures the cycle structure for hypergraphs. The Ihara zeta function for a hypergraph can be expressed in a determinant form as the reciprocal of the characteristic polynomial of the adjacency matrix for a transformed graph representation. Our hypergraph characterization is based on the coefficients of the characteristic polynomial, and can be used to construct feature vectors for hypergraphs. In the experimental evaluation, we demonstrate the effectiveness of the proposed characterization for clustering hypergraphs.

1 Introduction

There has recently been an increasing interest in hypergraph-based methods for representing and processing visual information extracted from images. The main reason for this is that hypergraph representations allow nodes to be multiply connected by edges, and can hence capture multiple relationships between features. To the best of our knowledge, the first attempt for representing visual objects using hypergraphs dates back to Wong *et al.*'s [9] framework for 3D object recognition. Here a 3D object model based on a hypergraph representation is constructed, and this encodes the geometric and shape information of polyhedrons as vertices and hyperedges. Object synthesis and recognition tasks are performed by merging and partitioning the hyperedge and vertex set. The method is realized by using set operations and the hypergraphs are not characterized in a mathematically consistent way. Later Bretto *et al.* [2] have introduced a hypergraph model for image representation, where they successfully solved the problems of image segmentation, noise reduction and edge detection. However, their method also relies on a crude form of set manipulation. Recently Bunke *et al.* [3] have developed a hypergraph matching algorithm for object recognition, where consistency checks are conducted on hyperedges. The computational paradigm underlying their method is based on tree search operations. Hypergraphs have also been represented using matrices. For instance, Zass *et al.* [10] have presented a matrix representation for regular hypergraphs and used them for correspondence matching. However, the method has not been investigated for irregular hypergraphs.

One common feature of existing hypergraph-based methods is that they exploit domain specific and goal directed representations, and do not lend themselves to generalization. The reason for this lies in the difficulty in formulating a hypergraph in a

X. Jiang and N. Petkov (Eds.): CAIP 2009, LNCS 5702, pp. 369–376, 2009.

mathematically uniform way for computation. There has yet to be a widely accepted and uniform way for representing and characterizing hypergraphs, and this remains an open problem with exploiting hypergraphs for machine learning. Moreover, to be easily manipulated, hypergraphs must be represented in an mathematically consistent way, using structures such as matrices or vectors.

Since Chung's [4] definition of the Laplacian matrix for k-uniform hypergraghs, there have been several attempts to develop matrix representations of hypergraphs [5][7] [11]. These hypergraph representations have found widespread applications in categorical data analysis. Recently, we have shown that matrix representation are also suitable for image processing [6], and have proposed an improved hypergraph Laplacian based on the developments of Zhou *et al.*'s method [11]. However, this method is based on a relatively impoverished spectral characterization and overlooks much of the detail of hypergraph-structure.

In this paper, we make a first attempt to characterize hypergraphs using characteristic polynomials. Specifically, we use the Ihara coefficients, which are the polynomial coefficients of the reciprocal Ihara zeta function for a hypergraph. The Ihara zeta function for a hypergraph has been detailed by Storm [8]. Based on this work, we establish feature vectors using the Ihara coefficients. We apply our feature vectors to clustering hypergraphs extracted from images of different object views and demonstrate their effectiveness in hypergraph characterization.

2 Hypergraph Laplacian Spectrum

A hypergraph is normally defined as a pair $H(V, E_H)$ where V is a set of elements, called nodes or vertices, and E is a set of non-empty subsets of V called hyperedges. The incidence matrix $\boldsymbol{H}$ of $H(V, E_H)$ is a $|E_H(H)| \times |V(H)|$ matrix with the (i, j)th entry 1 if the vertex $v_j \in V(H)$ is contained in the hyperedge $e_i \in E_H$ and 0 otherwise. For $H(V, E_H)$, one of the alternative definitions of the adjacency matrix and the corresponding Laplacian matrix is $\boldsymbol{A}_H = \boldsymbol{H}\boldsymbol{H}^T - \boldsymbol{D}_H$ and $\boldsymbol{L}_H = \boldsymbol{D}_H - \boldsymbol{A}_H = 2\boldsymbol{D}_H - \boldsymbol{H}\boldsymbol{H}^T$ respectively, where $\boldsymbol{D}_H$ is the diagonal vertex degree matrix whose diagonal element $d(v_i)$ is the summation of the ith row of $\boldsymbol{H}$ [6]. The eigenvalues of $\boldsymbol{L}_H$ is referred to as the hypergraph Laplacian spectrum and are straightforward characteristics from $H(V, E_H)$. We will use this spectral characterization for experimental comparison in Section 5.

3 Ihara Zeta Function for a Hypergraph

The definition of the Ihara zeta function for a hypergraph $H(V, E_H)$ is as follows [8]:

$$\zeta_H(u) = \prod_{p \in P_H} \left(1 - u^{|p|}\right)^{-1}. \tag{1}$$

Here P_H is the set of the equivalence classes of prime cycles in the hypergraph $H(V, E_H)$. A prime cycle in a hypergraph is a closed path with no backtracking, that is, no hyperedge is traversed twice in the prime cycle.

The Ihara zeta function for a hypergraph in the form of (1) is generally an infinite product. However, one of its elegant features is that it can be collapsed down into a rational function, which renders it of practical utility. To recast the hypergraph Ihara zeta function as a rational function, the graph representation of hypergraph is needed. There are several ways in which a hypergraph can be transformed into a graph representation. One of the most useful representations is the clique expansion, to which we turn our attention in more detail in the following section. Agarwal *et al.* [1] has reviewed the alternative graph representations of hypergraphs and detailed their relationships with each other with a particular emphasis on machine learning. To obtain the rational expression for the hypergraph Ihara zeta function, we make use of the bipartite graph representation of hypergraphs. To establish the associated bipartite graph, we use a dual representation in which each hyperedge is represented by a new vertex. The new vertex is incident to every original vertex encompassed by the corresponding hyperedge. The new vertex set and together with the original vertex set constitute the associated bipartite graph; the new vertices corresponding to hyperedges in one partition and the original vertices to the second partition. To provide an example, the bipartite graph associated with the hypergraph in Fig. 1 is shown in Fig. 2.

The Ihara zeta function for the hypergraph $H(V, E_H)$ can be equivalently expressed in a rational form as follow [8]:

$$\zeta_H(u) = (1-u)^{\chi(BG)} \det\left(\boldsymbol{I}_{|V(H)|+|E_H(H)|} - \sqrt{u}\boldsymbol{A}_{BG} + u\boldsymbol{Q}_{BG}\right)^{-1} \tag{2}$$

where $\chi(BG) = |V|$ is the Euler Number which equals the difference between the cardinalities of vertex set and edge set of the associated bipartite graph, $\boldsymbol{A}_{BG}$ its adjacency matrix and $\boldsymbol{Q}_{BG} = \boldsymbol{D}_{BG} - \boldsymbol{I}_{|V(H)|+|E_H(H)|}$.

4 Perron-Frobenius Operator for Hypergraphs

From (1) it is clear that the Ihara zeta function for a hypergraph can be rewritten in the form of the reciprocal of a polynomial. Although the Ihara zeta function can be evaluated efficiently using (2), the task of enumerating the coefficients of the polynomial appearing in the denominator of the Ihara zeta function (2) is difficult, except by resorting to software for symbolic calculation. To efficiently compute these coefficients, we adopt the determinant expression of the Ihara zeta function for a hypergraph $H(V, E_H)$ [8]:

$$\zeta_H(u) = \det(\boldsymbol{I}_H - u\boldsymbol{T}_H)^{-1}. \tag{3}$$

Here $\boldsymbol{T}_H$ is a square matrix which is referred to as the Perron-Frobenius operator of the hypergraph $H(V, E_H)$. It is the adjacency matrix of the oriented line graph associated with $H(V, E_H)$.

The establishment of the oriented line graph associated with $H(V, E_H)$ commences by constructing an $|e_i|$-clique by connecting each pair of vertices in the e_i through an edge for each hyperedge $e_i \in E_H$. The resulting graph is denoted by $GH(V, E_G)$. For the example hypergraph in Fig. 1, the associated $GH(V, E_G)$ is shown in Fig. 3. In this example, the oriented edges derived from the same hyperedge are colored the same while from different hyperedges are colored differently.

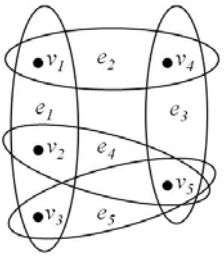


Fig. 1. Hypergraph

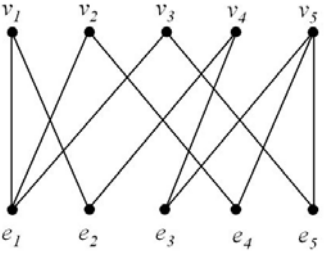


Fig. 2. Bipartite Graph

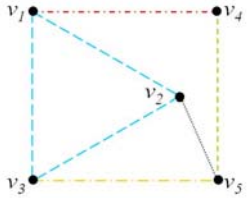


Fig. 3. Clique

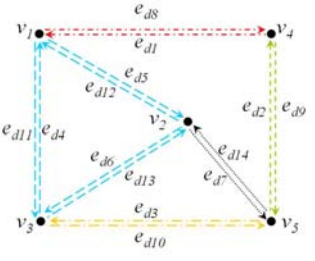


Fig. 4. Di-clique

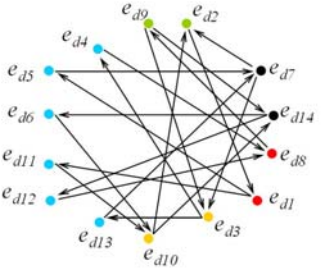


Fig. 5. Oriented Line Graph

For the graph $GH(V, E_G)$, the associated symmetric digraph $DGH(V, E_d)$ can be obtained by replacing each edge of $GH(V, E_G)$ by an arc (oriented edge) pair in which the two arcs are inverse to each other. For $GH(V, E_G)$ in Fig. 3, the associated $DGH(V, E_d)$ is shown in Fig. 4. Finally, we can establish the oriented line graph of the hypergraph based on the symmetric digraph. The vertex set and edge set of the the oriented line graph are defined as follows:

$$\begin{cases} V_{ol} & = E_d(DGH) \\ E_{ol} & = \{(e_d(u,v), e_d(v,w)) \in E_d \times E_d \ ; \ u \cup w \not\subset E_H\}. \end{cases} \tag{4}$$

The oriented line graph of the hypergraph in Fig. 1 is shown in Fig. 5. Here what we should note is that the oriented edges in the same clique of DGH can not establish an oriented edge in the oriented line graph. For instance, in Fig. 4 the terminus of the arc e_{d5} points to the origin of the arc e_{d6}. However, there is no oriented edge between vertices e_{d5} and e_{d6} in Fig. 5 because they are derived from the same hyperedge e_1 in Fig. 1. Therefore, this constraint prevents connections between any nodes with the same color in Fig. 5.

To establish feature vectors from the hypergraph Ihara zeta function for the purposes of characterizing hypergraphs in machine learning, it is natural to consider taking function samples as the vector elements. Although the function values at most of the sampling points will perform well in distinguishing hypergraphs, there is the possibility

of sampling at poles giving rise to meaningless infinities. Hence, the feature vectors consisting of function samples are potentially unstable representations of hypergraphs, since the distribution of poles is unknown beforehand.

On the other hand, from (3) it is clear that the reciprocal of the hypergraph Ihara zeta function is the characteristic polynomial of the Perron-Frobenius operator $\boldsymbol{T}_H$ and it can be deployed as:

$$\zeta_H^{-1}(u) = c_0 + c_1 u + \cdots + c_{M-1}u^{M-1} + c_M u^M \tag{5}$$

where M is the dimensionality of the square matrix $\boldsymbol{T}_H$. Each coefficient can be computed from the elementary symmetric polynomials of the eigenvalue set $\{\tilde{\lambda}_1, \tilde{\lambda}_2 \ ... \ \tilde{\lambda}_n\}$ of $\boldsymbol{T}_H$ as follows:

$$c_r = (-1)^r \sum_{k_1 < k_2 < \ ... \ < k_r} \tilde{\lambda}_{k_1} \tilde{\lambda}_{k_2} \ ... \ \tilde{\lambda}_{k_r}. \tag{6}$$

The characteristic polynomial coefficients in (5) do not give rise to infinities. Furthermore, these coefficients highly relate to the hypergraph-structure since the Ihara zeta function records the information about prime cycles in the hypergraphs. We refer to the set of characteristic polynomial coefficients as Ihara coefficients. We use the Ihara coefficients as the elements of the feature vector for a hypergraph and then apply them to clustering hypergraphs.

5 Experimental Evaluation

We apply the proposed feature vectors to two hypergraph datasets extracted from images of different object views. The first set of hypergraphs are extracted from house images in the CMU, MOVI and Chalet sequences (samples are shown in Fig. 6(a)) and the second set are extracted from images of eight objects in COIL dataset (samples are shown in Fig. 6(b)). To establish hypergraphs on the objects, we first extract feature points using the Harris detector as the vertices of hypergraphs. Let $\boldsymbol{c}(v_i)$ denote the spatial coordinate of the feature point v_i in an image, $I(v_i)$ denote the intensity of v_i. For each image, we construct the hypergraph using the method introduced in [6], where an element of incidence matrix is denoted as:

$$H(i,j) = \begin{cases} 1 \text{ if } \|\boldsymbol{c}(v_i) - \boldsymbol{c}(v_j)\| \leq Th_{j1} \text{ and if } \mid I(v_i) - I(v_j) \mid \leq Th_{j2} \\ 0 \text{ otherwise.} \end{cases} \tag{7}$$

Here Th_{j1} is a fixed value which represents the distance threshold for neighborhood and is set to 1/4 the size of the image, and Th_{j2} is the similarity threshold, which is determined by the standard deviation of the intensities of the feature points in the neighborhood of v_j.

We compute the Ihara coefficients as introduced in Section 4, generating the feature vector in the form of $\boldsymbol{v}_I = [c_3, \ c_4, \ \ln(|c_{M-3}|), \ \ln(|c_{M-2}|), \ \ln(|c_{M-1}|), \ \ln(|c_M|)]^T$. The last four components of the feature vector are manipulated in a logarithmic way to avoid problems of dynamic range. Fig. 7 shows the PCA projection of the hypergraphs from the Chalet images based on Ihara coefficients. Each point in the pattern space is marked with a view number which corresponds to the camera angle. The coefficients

(a) Houses

(b) COIL

Fig. 6. Dataset

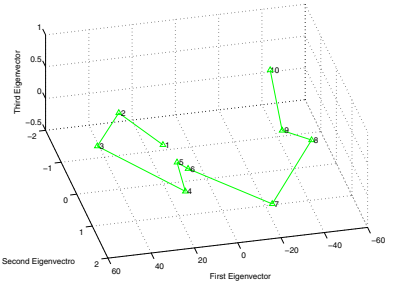


Fig. 7. Within-class Trajectory

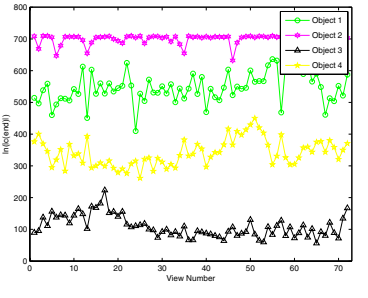


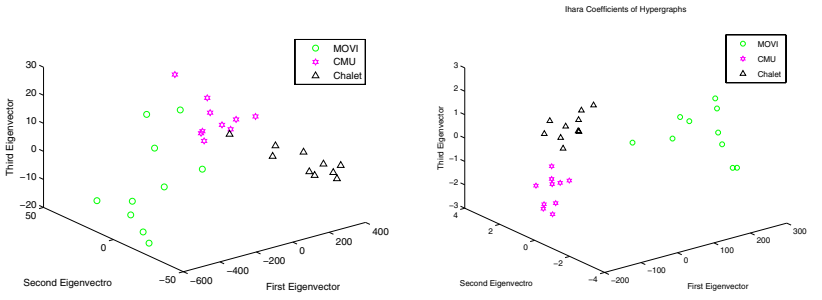
Fig. 8. Ihara Coefficient Plot

Table 1. Rand Indices

Feature Vector	Number of Object Classes			
	5	6	7	8
Spectra	0.8574	0.8564	0.8454	0.8449
Ihara Coefficients	0.9355	0.8859	0.8716	0.8812

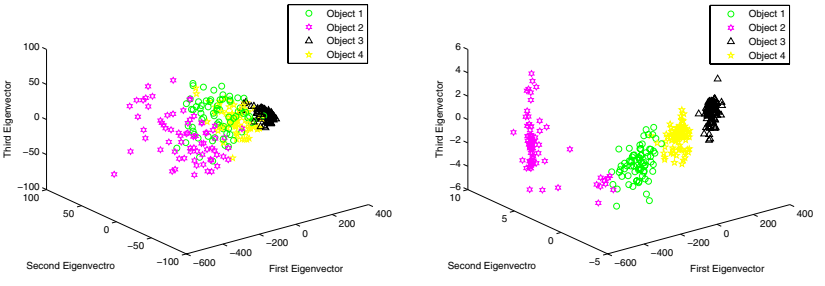
produce a clear trajectory and the neighboring images in the sequence are generally close together in the eigenspace.

Fig. 8 illustrates the behavior of the coefficient $\ln(|c_M|)$ of the first four objects in the COIL dataset. The four dotted lines represent the coefficient $\ln(|c_M|)$ of the four



(a) Truncated Laplacian Spectra (b) Ihara Coefficients

Fig. 9. Clusters for Three Classes of Houses



(a) Truncated Laplacian Spectra (b) Ihara Coefficients

Fig. 10. Clusters for Four Objects in COIL Dataset

objects separately. The coefficients of the different objects are well separated, thus indicating that the objects are well clustered. We then embed the feature vectors into a three-dimensional space using PCA for visualization. Figs. 9(a) and 9(b) indicate the results of the three classes of houses in Fig. 6(a) by using the feature vector consisting of the first through to sixth nonzero Laplacian eigenvalues and the proposed feature vector consisting of the Ihara coefficients respectively. Figs. 10(a) and 10(b) indicate the results for the first four objects in COIL dataset by using the feature vector consisting of the first through to sixth nonzero Laplacian eigenvalues and the proposed feature vector consisting of the Ihara coefficients respectively. From Fig. 9 and 10 we can see the proposed method is superior to the truncated Laplacian spectra in clustering hypergraphs.

To take the quantitative evaluation of the feature vectors one step further, we concentrate our attention on the COIL dataset, and evaluate the clustering performance obtained with different numbers of object classes. After performing PCA on the feature vectors, we locate the clusters using the K-means method and calculate the Rand index for the resulting clusters. The Rand indices for the truncated Laplacian spectra and for the Ihara coefficients are listed in Table 1. It is clear that the Ihara coefficients outperform the truncated Laplacian spectra for all numbers of object classes studied.

6 Conclusion and Future Work

We have performed a characteristic polynomial analysis on hypergraphs and characterize hypergraphs based on the Ihara zeta function. We have used the Ihara coefficients as the elements in the feature vector. Experimental results show the effectiveness of the proposed method.

However, the reason why the Ihara coefficients are superior to the Laplacian spectra in representing hypergraphs still needs to be further investigated. Moreover, the manipulations on the Perron-Frobenius operator for a hypergraph are computationally expensive and a more efficient computing method is needed for obtaining the Ihara coefficients. Therefore, our future research focuses on theoretically explaining the effectiveness of the Ihara coefficients and seeking a more efficient computation method.

Acknowledgments

We acknowledge the financial support from the FET programme within the EU FP7, under the SIMBAD project (contract 213250). Tatjana Aleksić is supported by Grant 144015G of the Serbian Ministry for Science and The British Scholarship Trust.

References

1. Agarwal, S., Branson, K., Belongie, S.: Higher-order learning with graphs. In: ICML (2006)
2. Bretto, A., Cherifi, H., Aboutajdine, D.: Hypergraph imaging: an overview. Pattern Recognition 35(3), 651–658 (2002)
3. Bunke, H., Dickinson, P., Neuhaus, M., Stettler, M.: Matching of hypergraphs — algorithms, applications, and experiments. Studies in Computational Intelligence 91, 131–154 (2008)
4. Chung, F.: The laplacian of a hypergraph. AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science 10, 21–36 (1993)
5. Li, W., Sole, P.: Spectra of regular graphs and hypergraphs and orthogonal polynomials. European Journal of Combinatorics 17, 461–477 (1996)
6. Ren, P., Wilson, R.C., Hancock, E.R.: Spectral embedding of feature hypergraphs. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) S+SSPR 2008. LNCS, vol. 5342, pp. 308–317. Springer, Heidelberg (2008)
7. Rodriguez, J.A.: On the laplacian eigenvalues and metric parameters of hypergraphs. Linear and Multilinear Algebra 51, 285–297 (2003)
8. Storm, C.K.: The zeta function of a hypergraph. Electronic Journal of Combinatorics 13 (2006)
9. Wong, A.K.C., Lu, S.W., Rioux, M.: Recognition and shape synthesis of 3d objects based on attributed hypergraphs. IEEE Transactions on Pattern Analysis and Machine Intelligence 11(3), 279–290 (1989)
10. Zass, R., Shashua, A.: Probabilistic graph and hypergraph matching. In: CVPR (2008)
11. Zhou, D., Huang, J., Scholkopf, B.: Learning with hypergraphs: Clustering, classification, and embedding. In: NIPS (2007)

Appendix B

Gauss-Bonnet Theorem anf Graph Embedding

Geometric Characterizations of Graphs Using Heat Kernel Embeddings

Hewayda El-Ghawalby[1,2] and Edwin R. Hancock[1,⋆]

[1] Department of Computer Science, University of York, UK
[2] Faculty of Engineering, Suez Canal University, Egypt
{howaida,erh}@cs.york.ac.uk

Abstract. In this paper, we investigate the heat kernel embedding as a route to computing geometric characterisations of graphs. The reason for turning to the heat kernel is that it encapsulates information concerning the distribution of path lengths and hence node affinities on the graph. The heat kernel of the graph is found by exponentiating the Laplacian eigensystem over time. The matrix of embedding co-ordinates for the nodes of the graph is obtained by performing a Young-Householder decomposition on the heat kernel. Once the embedding of its nodes is to hand we proceed to characterise a graph in a geometric manner. To obtain this characterisation, we focus on the edges of the graph under the embedding. Here we use the difference between geodesic and Euclidean distances between nodes to associate a sectional curvature with edges. Once the section curvatures are to hand then the Gauss-Bonnet theorem allows us to compute Gaussian curvatures at nodes on the graph. We explore how the attributes furnished by this analysis can be used to match and cluster graphs.

Keywords: Graph spectra, kernel methods, graph embedding, differential geometry, graph clustering.

1 Introduction

Graphs are used pervasively in computer science as representations of data with a network or relational structure. However, the analysis of data in this form has proved an elusive problem. The reason for this is that most of the available pattern analysis tools are couched in terms of vectorial rather than graph representations. One way to circumvent this problem is to is to embed the nodes of a graph in a vector space and to study the properties of the point distribution that results from the embedding. This is a problem that arises in a number of areas including manifold learning theory and graph-drawing. In the mathematics literature, there is a considerable body of work aimed at understanding how graphs can be embedded on manifolds [14]. In the pattern analysis community,

⋆ The authors acknowledge the financial support from the FET programme within the EU FP7, under the SIMBAD project (contract 213250).

E.R. Hancock et al. (Eds.): Mathematics of Surfaces 2009, LNCS 5654, pp. 124–142, 2009.

there has recently been renewed interest in the use of embedding methods motivated by graph theory. One of the best known of these is ISOMAP [23]. Here a neighborhood ball is used to convert data-points into a graph, and Dijkstra's algorithm is used to compute the shortest(geodesic) distances between nodes. By applying multidimensional scaling (MDS) to the matrix of geodesic distances the manifold is reconstructed. The resulting algorithm has been demonstrated to locate well-formed manifolds for a number of complex data-sets. Related algorithms include locally linear embedding which is a variant of PCA that restricts the complexity of the input data using a nearest neighbor graph [17], and the Laplacian eigenmap that constructs an adjacency weight matrix for the data-points and projects the data onto the principal eigenvectors of the associated Laplacian matrix (the degree matrix minus the weight matrix) [3]. Collectively, these methods are sometimes referred to as manifold learning theory.

The spectrum of the Laplacian matrix has been widely studied in spectral graph theory [4] and has proved to be a versatile mathematical tool that can be put to many practical uses including routing [1], indexing [19], clustering [18] and graph-matching [24,15].

One of the most important properties of the Laplacian spectrum is its close relationship with the heat equation. The heat equation can be used to specify the flow of information with time across a network or a manifold [26]. According to the heat-equation the time derivative of the kernel is determined by the graph Laplacian. The solution to the heat equation is obtained by exponentiating the Laplacian eigensystem over time. Because the heat kernel encapsulates the way in which information flows through the edges of the graph over time, it is closely related to the path length distribution on the graph. Recently, Lebanon and Lafferty [12] have shown how the heat kernel to construct statistical manifolds that can be used for inference and learning tasks. Moreover, in related work we have explored how a number of different invariants that can be computed from the heat kernel can be used for graph clustering [25]. Colin de Verdiere has shown how to compute geodesic invariants from the Laplacian spectrum [6].

In fact, a graph can be viewed as residing on a manifold whose pattern of geodesic distances is characterised by the heat kernel. Differential invariants can be computed from the heat kernel, and these in turn are related to the Laplacian eigensystem. This field of study is sometimes referred to as spectral geometry [8,26]. One of the most interesting recent developments in this area has been to establish a link between graph-spectra and the geometry of the underlying manifold [9,28,2,20]. Here considerable insight can be achieved through the analysis of the heat kernel of the graph [9,2]. There are a number of different invariants that can be computed from the heat-kernel. Asymptotically for small time, the trace of the heat kernel [4] (or the sum of the Laplacian eigenvalues exponentiated with time) can be expanded as a rational polynomial in time, and the co-efficiants of the leading terms in the series are directly related to the geometry of the manifold. For instance, the leading co-efficient is the volume of the manifold, the second co-efficient is related to the Euler characteristic, and the third co-efficient to the Ricci curvature.

The aim in this paper is to investigate whether the heat kernel can be used to provide a geometric characterisation of graphs that can be used for the purposes of graph-clustering. This is of course a problem that can be addressed directly by using the spectral geometry of the combinatorial Laplacian. However, there are two major obstacles. First, the results delivered by spectral geometry are interesting, they apply under the assumption that the graph Laplacian converges to the corresponding continuous Laplace operator provided that the graph is sufficiently large. Second, the calculations involved are intricate and the resulting expressions are not very elegant. Hence, we adopt a more pragmatic approach in this paper where we aim to characterise the geometry of point distribution based on embeddings derived from the heat-kernel.

The method involves performing a Young-Householder decomposition of the heat-kernel to recover the matrix of embedding co-ordinates. In other words, we perform kernel principal components analysis on the heat kernel to map nodes of the graph to points in the vector space. We provide an analysis which shows how the eigenvalues and eigenvectors of the covariance matrix for the point distribution resulting from the kernel mapping are related to those of the Laplacian.

With the embeddings to hand, we develop a graph characterisation based on differential geometry. To do this we compute the sectional curvatures associated with the edges of the graph, making use of the fact that the sectional curvature is determined by the difference between the geodesic and Euclidean distances. We take this analysis one step further, and use the Gauss-Bonnet theorem to compute the Gaussian curvatures associated with triangular faces of the graph. We characterise graphs using sets of curvatures, defined either on the edges or the faces. We explore whether these characterisations can be used for the purposes of graph matching and graph clustering. To this end, we compute the similarities of the sets using robust variants of the Hausdorff distance. This allows us to compute the similarity of different graphs without knowing the correspondences between edges or faces.

The outline of this paper is as follows. In Section 2 we provide some background on the heat-kernel and its relationship with the Laplacian spectrum. Section 3 describes how the Young-Householder factorisation of the heat kernel leads to a co-ordinate embedding of the nodes of a graph. Section 4 shows how the Euclidean and geodesic distances between nodes under the embedding can be used to estimate sectional curvatures for the edges and Gaussian curvatures for the triangular faces, using two different embeddings. In Section 5 we experiment with the method on the houses database. Finally, Section 7 offers some conclusions and directions for future research.

2 Heat Kernels on Graphs

In this section, we give a brief introduction on the graph heat kernel. To commence, suppose that the graph under study is denoted by $G = (V, E)$ where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. Since we wish to adopt

a graph spectral approach we introduce the adjacency matrix A for the graph where the elements are

$$A(u,v) = \begin{cases} 1 \text{ if } & (u,v) \in E \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

We also construct the diagonal degree matrix D, whose elements are given by the degree of the nodes, i.e. $D(u,u) = deg(u) = \sum_{v \in V} A(u,v)$. From the degree matrix and the adjacency matrix we construct the Laplacian matrix $L = D - A$, i.e. the degree matrix minus the adjacency matrix,

$$L(u,v) = \begin{cases} deg(v) \text{ if } & u = v \\ -1 & \text{if } u \text{ and } v \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The normalized Laplacian $\hat{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ has elements

$$\hat{L}(u,v) = \begin{cases} 1 & \text{if } u = v \text{ and } d_v \neq 0 \\ -\frac{1}{\sqrt{deg(u)deg(v)}} & \text{if } u \text{ and } v \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The spectral decomposition of the normalized Laplacian matrix is $\hat{L} = \Phi \Lambda \Phi^T$, where $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_{|V|})(\lambda_1 < \lambda_2 < ... < \lambda_{|V|})$is the diagonal matrix with the ordered eigenvalues as elements and $\Phi = (\phi_1|\phi_2|....|\phi_{|V|})$ is the matrix with the ordered eigenvectors as columns. Since $\hat{L}$ is symmetric and positive semi-definite, the eigenvalues of the normalized Laplacian are all non-negative. The multiplicity of the zero eigenvalue is the number of isolated cliques in the graph. For a connected graph, the multiplicity of the zero eigenvalue is one. The eigenvector associated with the smallest non-zero eigenvector is referred to as the Fiedler-vector [4].

2.1 Heat Equation

We are interested in the heat equation associated with the Laplacian, and this is given by.

$$\frac{\partial h_t}{\partial t} = -\hat{L} h_t \tag{4}$$

where h_t is the heat kernel and t is time. The heat kernel is the fundamental solution of the heat equation. It can be viewed as describing the flow of information across the edges of the graph with time. The rate of flow is determined by the Laplacian of the graph. The solution to the heat equation is

$$h_t = e^{-t\hat{L}} \tag{5}$$

From [4] we can proceed to compute the heat kernel on a graph by exponentiating the Laplacian eigenspectrum, i.e.

$$h_t = \Phi \exp[-\Lambda t] \Phi^T = \sum_{i=1}^{|V|} \exp[-\lambda_i t] \phi_i \phi_i^T \tag{6}$$

The heat kernel is a $|V| \times |V|$ matrix. For the nodes u and v of the graph G the heat kernel element is

$$h_t(u,v) = \sum_{i=1}^{|V|} \exp[-\lambda_i t]\phi_i(u)\phi_i(v) \tag{7}$$

When t tends to zero, then $h_t \simeq I - \hat{L}t$, i.e. the kernel depends on the local connectivity structure or topology of the graph. If, on the other hand, t is large, then $h_t \simeq I - \exp[-\lambda_2 t]\phi_2\phi_2^T$, where λ_2 is the smallest non-zero eigenvalue and ϕ_2 is the associated eigenvector, i.e. the Fiedler vector. Hence, the large time behavior is governed by the global structure of the graph.

2.2 Geodesic Distance from the Heat Kernel

It is interesting to note that the heat kernel is also related to the path length distribution on the graph. To show this, consider the matrix $P = I - \hat{L}$, where I is the identity matrix. The heat kernel can be rewritten as $h_t = e^{-t(I-P)}$. We can perform the MacLaurin expansion on the heat kernel to re-express it as a polynomial in t. The result of this expansion is

$$h_t = e^{-t}(I + tP + \frac{(tP)^2}{2!} + \frac{(tP)^3}{3!} + ...) = e^{-t}\sum_{k=0}^{\infty} P^k \frac{t^k}{k!} \tag{8}$$

For a connected graph, the matrix P has elements

$$P(u,v) = \begin{cases} 0 & \text{if} \quad u = v \\ \frac{1}{\sqrt{deg(u)deg(v)}} & \text{if} \quad u \neq v \quad \text{and} \quad (u,v) \in E \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

As a result, we have that

$$P^k(u,v) = \sum_{S_k} \prod_{i=1}^{k} \frac{1}{\sqrt{deg(u_i)deg(u_{i+1})}} \tag{10}$$

where the walk S_k is a sequence of vertices $u_0, ..., u_k$ of length k such that $(u_i, u_{i+1}) \in E$. Hence, $P^k(u,v)$ is the sum of weights of all walks of length k joining nodes u and v. In terms of this quantity, the elements of the heat kernel are given by

$$h_t(u,v) = \exp[-t]\sum_{k=0}^{|V|^2} P^k(u,v)\frac{t^k}{k!} \tag{11}$$

We can find a spectral expression for the matrix P^k using the eigendecomposition of the normalized Laplacian. Writing $P^k = (I - \hat{L})^k$ it follows that $P^k = \Phi(I - \Lambda)^k\Phi^T$. The element associated with the nodes u and v is

$$P^k(u,v) = \sum_{i=1}^{|V|}(1 - \lambda_i)^k\phi_i(u)\phi_i(v) \tag{12}$$

The geodesic distance between nodes, i.e. the length of the walk on the graph with the smallest number of connecting edges, can be found by searching for the smallest value of k for which $P^k(u,v)$ is non zero, i.e. $d_G(u,v) = floor_k P_k(u,v)$.

3 Heat Kernel Embedding

In this section we first show how the heat kernel can be used to embed the nodes of a graph in a vector space using the Young-Householder decomposition. Second, we provide an analysis revealing the relationship between the eigenvalues and eigenvectors of the heat-kernel and those of the covariance matrix for the point distribution resulting from the embedding.

3.1 Co-ordinate Embedding

We use the heat kernel to map the nodes of the graph into a vector space. Let $Y = (y_1|...|y_u|...|Y_{|V|})$ be the $|V| \times |V|$ matrix with the vectors of co-ordinates as columns. The vector of co-ordinates y_u for the node index u is hence the u^{th} column of Y. The co-ordinate matrix is found by performing the Young-Householder decomposition $h_t = Y^T Y$ on the heat-kernel. Since $h_t = \Phi \exp[-\Lambda t]\Phi^T$, $Y = \exp[-\frac{1}{2}\Lambda t]\Phi^T$. Hence, the co-ordinate vector for the node indexed u is

$$y_u = (\exp[-\frac{1}{2}\lambda_1 t]\phi_1(u), \exp[-\frac{1}{2}\lambda_2 t]\phi_2(u), ..., \exp[-\frac{1}{2}\lambda_{|V|} t]\phi_{|V|}(|V|))^T \quad (13)$$

The kernel mapping $\mathcal{M} : V \rightarrow \mathcal{R}^{|V|}$, embeds each node on the graph in a vector space $\mathcal{R}^{|V|}$. The heat kernel $h_t = Y^T Y$ can also be viewed as a Gram matrix, i.e. its elements are scalar products of the embedding co-ordinates. Consequently, the kernel mapping of the nodes of the graph is an isometry. The squared Euclidean distance between nodes u and v is given by

$$d_E(u,v)^2 = (y_u - y_v)^T (y_u - y_v) = \sum_{i=1}^{|V|} \exp[-\lambda_i t] \Big\{ \phi_i(u) - \phi_i(v) \Big\}^2 \quad (14)$$

Figure 1 shows the steps to embed the graph into a manifold.

3.2 Point Distribution Statistics

One very simple way to characterize the embedded point-set is to study the properties of the covariance matrix of the point-set generated by the embedding methods. To construct the covariance matrix, we commence by computing the mean coordinate vector. The mean co-ordinate vector for the heat kernel embedding is

$$\hat{y} = \frac{1}{|V|} Y e = \frac{1}{|V|} \exp[-\frac{1}{2}\Lambda t]\Phi^T e \quad (15)$$

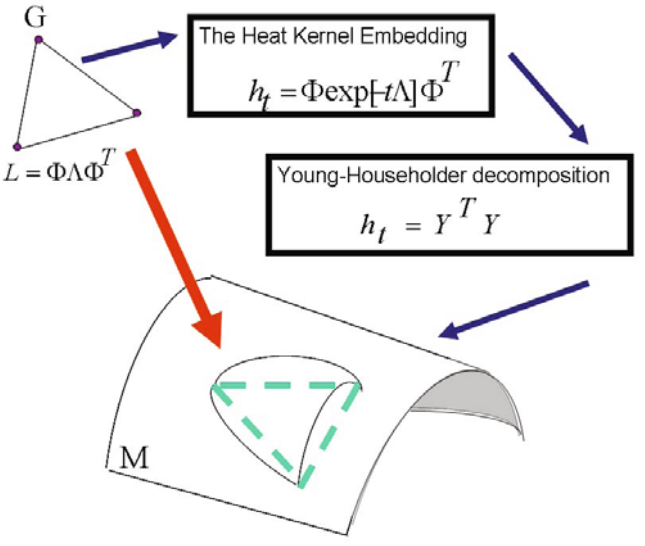


Fig. 1. Illustration of the geometric embedding of the graph into a manifold

where $e = (1, 1, ..., 1)^T$ is the all ones vector of length $|V|$. The matrix of centred co-ordinates is found by subtracting the mean position vector from each of the co-ordinate vectors and is given by

$$Y_C = Y - \frac{1}{|V|} Y e e^T = \exp[-\frac{1}{2}\Lambda t]\Phi^T (I - \frac{1}{|V|} e e^T) = \exp[-\frac{1}{2}\Lambda t]\Phi^T M^T \quad (16)$$

where $M^T = (I - \frac{1}{|V|} e e^T)$. The covariance matrix for the embedded point-positions is

$$S_Y = Y_C Y_C^T = \exp[-\frac{1}{2}\Lambda t]\Phi^T M^T M \Phi \exp[-\frac{1}{2}\Lambda t] \quad (17)$$

Hence, we can write

$$S_Y = \frac{1}{|V|} C^T C$$

where $C = M\Phi \exp[-\frac{1}{2}\Lambda t]$. To compute the eigenvectors of S_Y we first construct the matrix,

$$CC^T = M\Phi \exp[-\Lambda t]\Phi^T M^T = M h_t M^T$$

i.e. CC^T has eigenvalue matrix $\Lambda_h = \exp[-\Lambda t]$ and un-normalised eigenvector matrix $U = M\Phi$. As a result the matrix $C^T C$ has normalised eigenvector matrix $\hat{U} = C^T U \Lambda_h^{-\frac{1}{2}}$ and eigenvalue matrix Λ_h.

To see this note that

$$(C^T U \Lambda_h^{-\frac{1}{2}})\Lambda_h (C^T U \Lambda_h^{-\frac{1}{2}})^T = C^T U U^T C = C^T C \quad (18)$$

Hence C^TC has eigenvector matrix $\Lambda_h = \exp[-\Lambda t]$ and normalised eigenvector matrix

$$\hat{U} = (M\Phi \exp[-\frac{1}{2}\Lambda t])^T M\Phi(\exp[-\Lambda t])^{-\frac{1}{2}} = \exp[-\frac{1}{2}\Lambda t]\Phi^T M^T M\Phi \exp[\frac{1}{2}\Lambda t] \tag{19}$$

Finally, it is interesting to note that the projection of the centred co-ordinates onto the eigenvectors of the covariance matrix S_Y is

$$Y_P = \hat{U}^T Y_C = \exp[-\frac{1}{2}\Lambda t]\Phi^T M^T = Y_C$$

4 Geometric Characterisation

In this section we develop our differential characterisation of graphs. We commence by showing how the geodesic and Euclidean distances estimated from the spectrum of the Laplacian and the heat kernel embedding can be used to associate a sectional curvature with the edges of a graph. Next, we turn our attention to geodesic triangles formed by the embedding of first order cycles, i.e. triangles, of the graph. From the turning angles of the geodesic triangles, we estimate Gaussian curvature.

4.1 Sectional Curvature

In this section we show how the Euclidean distance and geodesic distances computed for embedding can be used to compute the sectional curvature associated with edges of the graph. The sectional curvature is determined by the degree to which the geodesic bends away from the Euclidean chord. Hence for a torsionless geodesic on the manifold, the sectional curvature can be estimated easily if the Euclidean and geodesic distances are known. Suppose that the geodesic can be locally approximated by a circle. Let the geodesic distance between the pair of points u and v be $d_G(u,v)$ and the corresponding Euclidean distance be $d_E(u,v)$. Further let the radius of curvature of the approximating circle be $r_s(u,v)$ and suppose that the tangent vector to the manifold undergoes a change in direction of $2\theta_{u,v}$ as we move along a connecting circle between the two points. We show an illustration of the above in Figure 2.

In terms of the angle $\theta_{u,v}$, the geodesic distance, i.e. the distance traversed along the circular arc, is $d_G(u,v) = 2r_s(u,v)\theta_{u,v}$, and as a result we find that $\theta_{u,v} = d_G(u,v)/2r_s(u,v)$. The Euclidean distance, on the other hand, is given by $d_E(u,v) = 2r_s(u,v)\sin\theta_{u,v}$, and can be approximated using the MacLaurin series

$$d_E(u,v) = 2r_s(u,v)\{\theta_{u,v} - \frac{1}{6}\theta_{u,v}^3 + ...\} \tag{20}$$

Substituting for $\theta_{u,v}$ obtained from the geodesic distance, we have

$$d_E(u,v) = d_G(u,v) - \frac{d_G(u,v)^3}{24r_s^2(u,v)} \tag{21}$$

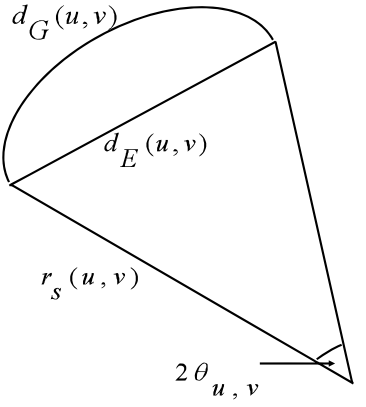


Fig. 2. Illustration of relationship between the geodesic distance, Euclidean distances and the sectional curvature

Solving the above equation for the radius of curvature, the sectional curvature of the geodesic connecting the nodes u and v is approximately

$$k_s(u,v) = \frac{1}{r_s(u,v)} = \frac{2\sqrt{6}(d_G(u,v) - d_E(u,v))^{\frac{1}{2}}}{d_G(u,v)^{\frac{3}{2}}} \tag{22}$$

4.2 The Gauss-Bonnet Theorem

The Gauss-Bonnet Theorem links the topology and geometry of a surface in an elegant and compact manner. Spivak [21] and Stillwell [22] give accounts of the early history of its development and application.

For a smooth compact oriented Riemannian 2-manifold M, let $\triangle_g$ be a triangle on M whose sides are geodesics, i.e. paths of shortest length on the manifold. Further, let α_1, α_2 and α_3 denote the interior angles of the triangle. According to Gauss's theorem, if the Gaussian curvature K (i.e. the product of the maximum and minimum curvatures at a point on the manifold) is integrated over $\triangle_g$, then

$$\int_{\triangle_g} K dM = \sum_{i=1}^{3} \alpha_i - \pi \tag{23}$$

where dM is the Riemannian volume element.

4.3 Gaussian Curvature

To estimate the Gaussian curvature from the above, we must determine the interior angles α_i of the geodesic triangle. To this end we assume that T is a triangulation of a smooth manifold M, $\triangle_g$ be a geodesic triangle on M with

angles $\{\alpha_i\}_{i=1}^3$ and geodesic edge lengths $\{d_{gi}\}_{i=1}^3$. Further suppose that $\triangle_e$ be the corresponding Euclidean triangle with edge lengths $\{d_{ei}\}_{i=1}^3$ and interior angles $\{\varphi_i\}_{i=1}^3$. We assume that the geodesic index i is a great arc on a sphere with radius r_i, $i = 1, 2, 3$. By averaging over the constituent geodesic edges, we treat the geodesic triangles as residing on a hyper−sphere with radius $r = \frac{1}{3}\sum_{i=1}^3 r_i$.

To commence, we compute the area of the geodesic triangle. Here we'll make use of the geometry of the sphere, the area of the spherical triangle is given by

$$A_g = (\sum_{i=1}^{3} \alpha_i - \pi)r^2 \tag{24}$$

From (24) we can see that

$$\sum_{i=1}^{3} \alpha_i - \pi = \frac{A_g}{r^2} \tag{25}$$

Now, considering a small area element on the sphere given in spherical coordinates by $dA = R^2 \sin\theta d\theta d\varphi$, the integration of dA bounded by θ gives us another formula for computing the area of the geodesic triangle

$$A = \frac{1}{2r} d_e^2 \tag{26}$$

where d_e^2 is computed from the embedding using (14).

From (23), (25) and (26), we get the following formula for the Gaussian curvature residing over the geodesic triangle:

$$\int_{\triangle_g} K dM = \frac{1}{2r^3} d_e^2 \tag{27}$$

5 Graph Similarity

We represent the graphs using sets of curvatures defined either over the edges (i.e. sectional curvatures) or triangular faces (i.e. Gaussian curvatures) of the graphs under consideration. The sets of curvatures are unordered, i.e. we do not know the correspondences between edges or faces in different graphs, We hence require a set-based similarity measure to compare graphs in the absence of correspondences. One route is provided by the Hausdorff distance. However, this is known to be sensitive to noise, so we explore median and probabilistic variants of the Hausdorff distance.

5.1 Hausdorff Distance

The Hausdorff distance provides a means of computing the distance between sets of unordered observations when the correspondences between the individual

items are unknown. In its most general setting, the Hausdorff distance is defined between compact sets in a metric space. Given two such sets, we consider, for each point in one set, the closest point in the second set. Hausdorff distance is the maximum over all these values. More formally, the classical Hausdorff distance(HD)[11] between two finite point sets A and B is given by $H(A,B) = \max(h(A,B), h(B,A))$ where the directed Hausdorff distance from A to B is defined to be

$$h(A,B) = \max_{a \in A} \min_{b \in B} \|a - b\| \tag{28}$$

and $\|.\|$ is some underlying norm on the points of A and B (e.g., the L2 or Euclidean norm). Dubuisson and Jain [7] proposed a robust modified Hausdorff distance (MHD) based on the average distance value instead of the maximum value, in this sense they defined the directed distance of the MHD as

$$h(A,B) = \frac{1}{N_A} \sum_{a \in A} \min_{b \in B} \|a - b\| \tag{29}$$

Using these ingredients we can describe how Hausdorff distances can be extended to graph-based representations. To commence let us consider two graphs $G_1 = (V_1, E_1, k_1)$ and $G_2 = (V_2, E_2, k_2)$, where V_1,V_2 are the sets of nodes, E_1,E_2 the sets of edges and k_1,k_2 the matrices whose elements are the curvature defined in the previous section. We can now write the distances between two graphs as follows:

1) The classical Hausdorff distance (HD) is

$$h_{HD}(G_1,G_2) = \max_{i \in V_1} \max_{j \in V_1} \min_{I \in V_2} \min_{J \in V_2} \|k_2(I,J) - k_1(i,j)\| \tag{30}$$

2) The modified Hausdorff distance (MHD) is

$$h_{MHD}(G_1,G_2) = \frac{1}{|V_1|} \sum_{i \in V_1} (\frac{1}{|V_1|} \sum_{i \in V_1} \min_{I \in V_2} \min_{J \in V_2} \|k_2(I,J) - k_1(i,j)\|) \tag{31}$$

5.2 A Probabilistic Similarity Measure (PSM)

One of the well documented problems with both the Hausdorff and modified Hausdorff distances, is lack of robustness. In order to overcome this problem, Huet and Hancock [10] have recently developed a probabilistic variant of the Hausdorff distance. This measures the similarity of the sets of attributes rather than using defined set based distance measures. For the graphs G_1 and G_2, the set of all nodes connected to the node $I \in G_2$ by an edge is defined as $C_I^2 = \{J|(I,J) \in E_2\}$, and the corresponding set of nodes connected to the node $i \in G_1$ by an edge is $C_i^1 = \{j|(i,j) \in E_2\}$. For the match of the graph G_2 onto G_1 Huet and Hancock's similarity measure is

$$S(G_1,G_2) = \frac{1}{|V_2| \times |V_1|} \sum_{i \in V_1} \max_{I \in V_2} \sum_{j \in C_i^1} \max_{J \in C_I^2} P((i,j) \rightarrow (I,J)|k_{(I,J)}^2, k_{(i,j)}^1) \tag{32}$$

In this formula the *a posteriori* probability $P((i,j) \rightarrow (I,J)|k^2_{(I,J)}, k^1_{(i,j)})$ represents the value for the match of the G_2 edge (I,J) onto the G_1 edge (i,j) provided by the corresponding pair of attribute structures $k^2_{(I,J)}$ and $k^1_{(i,j)}$.

The similarity measure commences by finding the maximum probability over the nodes in C^2_I then averaging the edge-compatibilities over the nodes in C^1_i. Similarly, we consider the maximum probability over the nodes in the graph G_2 followed by averaging over the nodes in G_1. It is worth mentioning that unlike the Hausdorff distance, this similarity measure does not satisfy the distance axioms. Moreover, while the Hausdorff distance is saliency-based (i.e. it measures the maximum distance between two sets of observations) our measure here returns the maximum similarity. back to the formula where we still need to compute the probability $P((i,j) \rightarrow (I,J)|k^2_{I,J}, k^1_{i,j})$ For that purpose we will use a robust weighting function

$$P((i,j) \rightarrow (I,J)|k^2_{(I,J)}, k^1_{(i,j)}) = \frac{\Gamma_\sigma(\|k^2_{(I,J)}, k^1_{(i,j)}\|)}{\sum_{(I,J)\in E_2} \Gamma_\sigma(\|k^2_{(I,J)}, k^1_{(i,j)}\|)}) \tag{33}$$

where $\Gamma_\sigma(.)$ is a distance weighting function. There are several alternative robust weighting functions. Here we work with a Gaussian of the form $\Gamma_\sigma(\rho) = \exp(-\frac{\rho^2}{2\sigma^2})$.

5.3 Multidimensional Scaling

With the graph distances in hand, we require a means of visualizing the distribution of graphs. We choose to use the classical Multidimensional Scaling (MDS) method [5] to embed the data specified in the matrix in Euclidean space. Let H be the distance matrix with row r and column c entry H_{rc}. The first step of MDS is to calculate a matrix T whose element with row r and column c is given by $T_{rc} = -\frac{1}{2}[H^2_{rc} - \widehat{H}^2_{r.} - \widehat{H}^2_{.c} + \widehat{H}^2_{..}]$ where $\widehat{H}_{r.} = \frac{1}{N}\sum_{c=1}^{N} H_{rc}$ is the average value over the rth row in the distance matrix, $H_{.}c$ is the similarly defined average value over the cth column and $\widehat{H}_{..} = \frac{1}{N^2}\sum_{r=1}^{N}\sum_{c=1}^{N} H_{rc}$ is the average value over all rows and columns of the distance matrix. Then, we subject the matrix T to an eigenvector analysis to obtain a matrix of embedding coordinates X. If the rank of T is k; $k \leq N$, then we will have $k$non-zero eigenvalues. We arrange these k non-zero eigenvalues in descending order, i.e., $l_1 \geq l_2 \geq ... \geq l_k \geq 0$. The corresponding ordered eigenvectors are denoted by u_i where l_i is the ith eigenvalue. The embedding coordinate system for the graphs is $X = [\sqrt{l_1}u_1, \sqrt{l_2}u_2, ..., \sqrt{l_k}u_k]$ for the graph indexed i, the embedded vector of the coordinates is $x_i = (X_{i,1}, X_{i,2}, ..., X_{i,k})^T$.

6 Experiments

In this section we experiment with the curvature-based attributes extracted using the heat-kernel embedding, and explore whether they can be used for the purposes of graph-clustering. In our experiments the graphs extracted from images

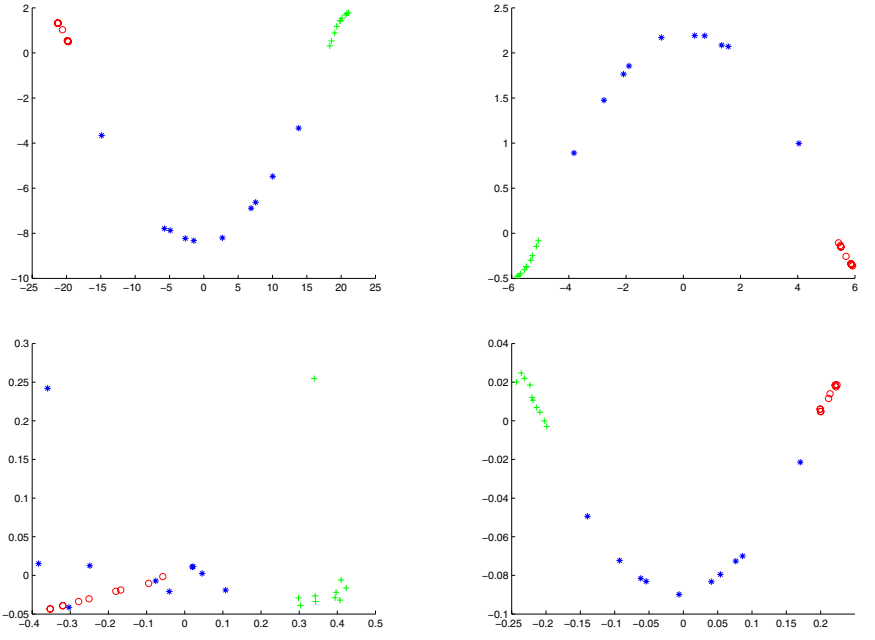


Fig. 3. MDS embedding obtained using MHD for house data represented by the sectional curvatures residing on the edges

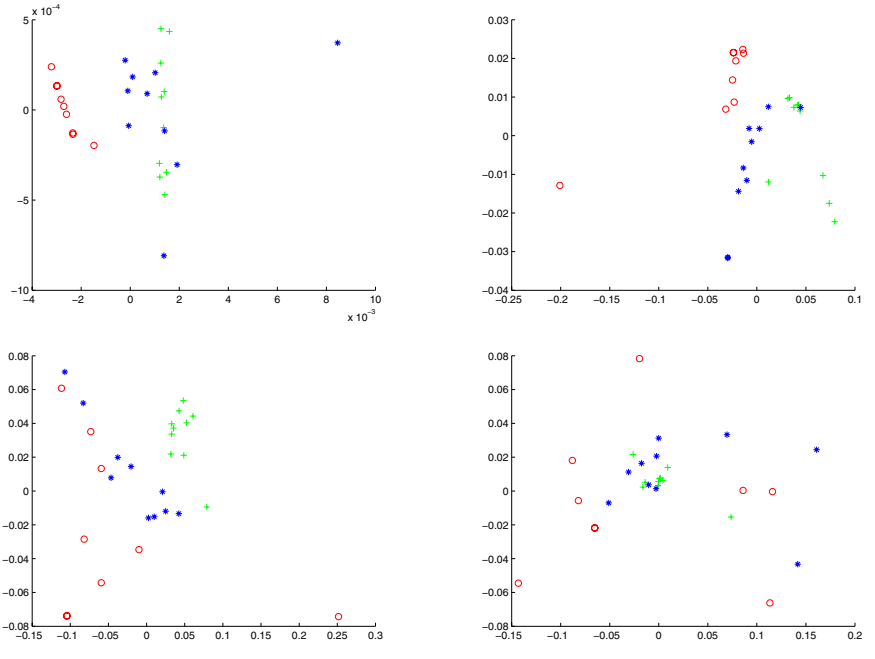


Fig. 4. MDS embedding obtained using MHD for the houses data represented by the Gaussian curvature associated with the geodesic triangles

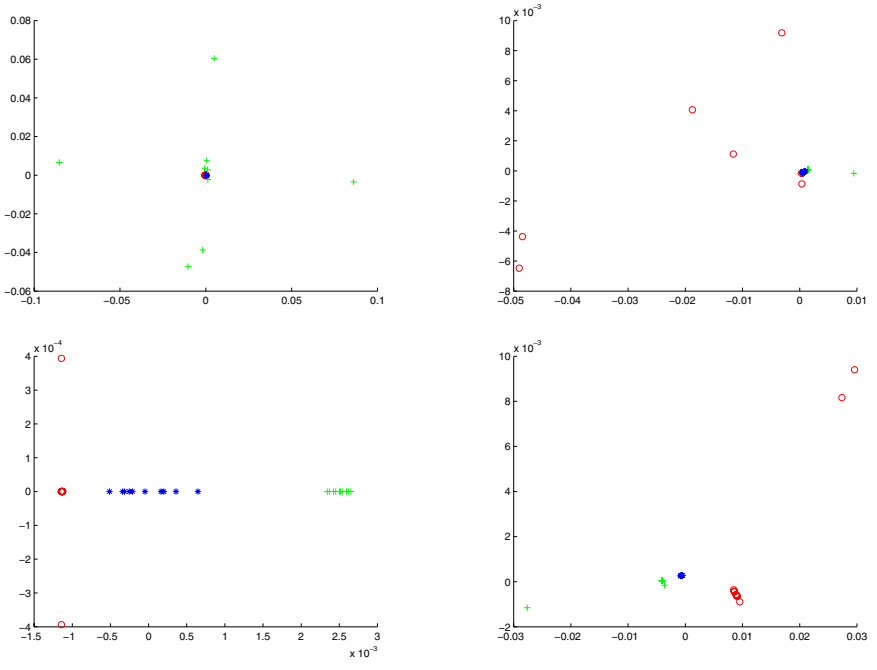


Fig. 5. MDS embedding obtained using the probabilistic similarity measure for the houses data set represented by the sectional curvature residing on the edges

of toy houses in the standard CMU, MOVI and chalet house image sequences [16]. These data sets contain different views of model houses from equally spaced viewing directions. From the house images, corner features are extracted, and Delaunay graphs representing the arrangement of feature points are constructed. Our data consists of ten graphs for each of the three houses. Each node in a Delaunay graph belongs to a first order cycle, and as a result the graph is a triangulation. To commence, we obtain the embedding for each of the thirty graphs following the steps outlined in Section 3.1. We compute the Euclidean distances between the nodes in each graph based on the heat kernel embedding obtained with the values of $t = 10.0, 1.0, 0.1$ and 0.01. We work with two representations of the graphs. The first is the sectional curvature associated with the edges, outlined in Section 4.1. The second is the Gaussian curvature on the triangles of the Delaunay triangulations extracted from the graphs, as outlined in Section 4.3. We use both the sectional and gaussian curvature as features for the purposes of gauging the similarity of graphs using both the modified Hausdorff distance and the probabilistic similarity measure. We subject the distance matrices to the Multidimensional Scaling (MDS) procedure to embed the graphs into a low dimensional space. Figures 3, 4, 5, 6 show the results obtained, where the subfigures are ordered from left to right and from top to bottom, using the heat kernel embedding with the values $t = 10.0, 1.0, 0.1$ and 0.01.

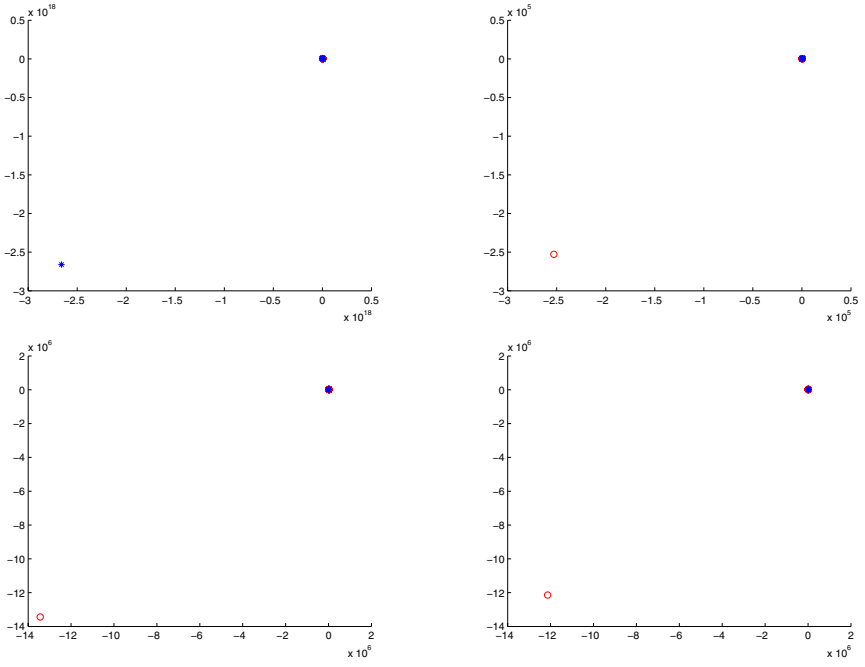


Fig. 6. MDS embedding obtained using the probabilistic similarity measure for the houses data set represented by the Gaussian curvature associated with the geodesic triangles

Table 1. A rand index vs. t

		t=10	t=1.0	t=0.1	t=0.01
MHD	Sectional curvature	0.1333	0.2333	0.1333	0.0333
MHD	Gaussian curvature	0.1667	0.0333	0.1333	0.4000
PSM	Sectional curvature	0.0000	0.0333	0.2667	0.3667
PSM	Gaussian curvature	0.3000	0.3000	0.3000	0.3000

To investigate the data in more detail Table 1 shows the rand index for the data as a function of t. This index is computed as follows: a) We commence by computing the mean for each cluster, b) We then compute the distance from each point to each mean. c) If the distance from the correct mean is smaller than those to remaining means, then the classification is correct, if not then classification is incorrect. d) The rand index is R= (♯ incorrect) / (♯ incorrect + ♯ correct).

A comparison shows that the curvature attributes associated with the edges give slightly better clusters than those obtained using the attributes derived from the triangles.

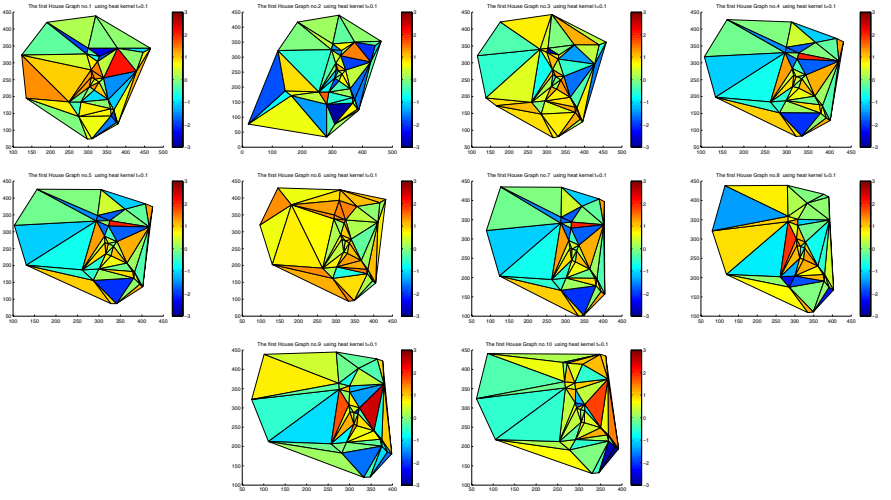


Fig. 7. The distribution of the Gaussian curvatures of the geodesic triangles for the ten graphs of the 1st house

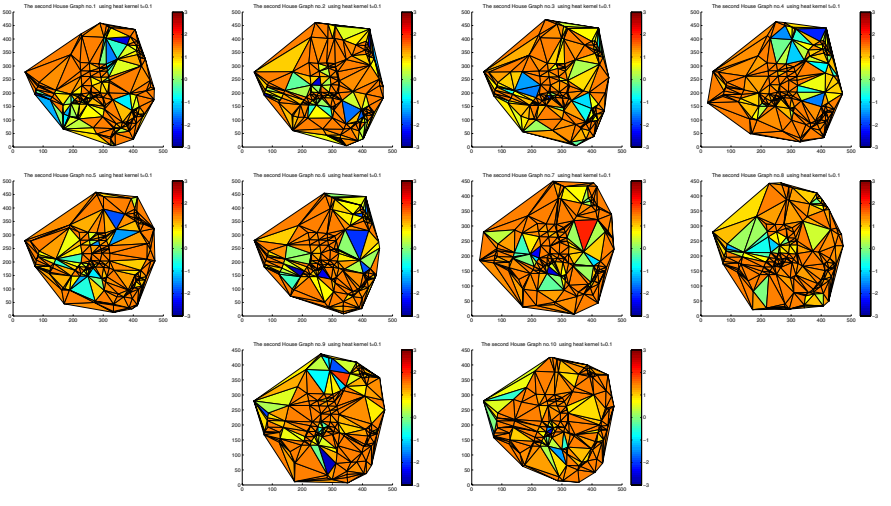


Fig. 8. The distribution of the Gaussian curvatures of the geodesic triangles for the ten graphs of the 2nd house

Finally, we investigate how the Gaussian curvatures of the geodesic triangles are distributed over the Delaunay graph. Figures 7, 8 and 9 show distribution for sample embeddings computed from the heat kernel at $t = 0.1$.

From the sequence it is clear that the Gaussian curvature distribution over the different views of each house is stable, moreover it moves smoothly from

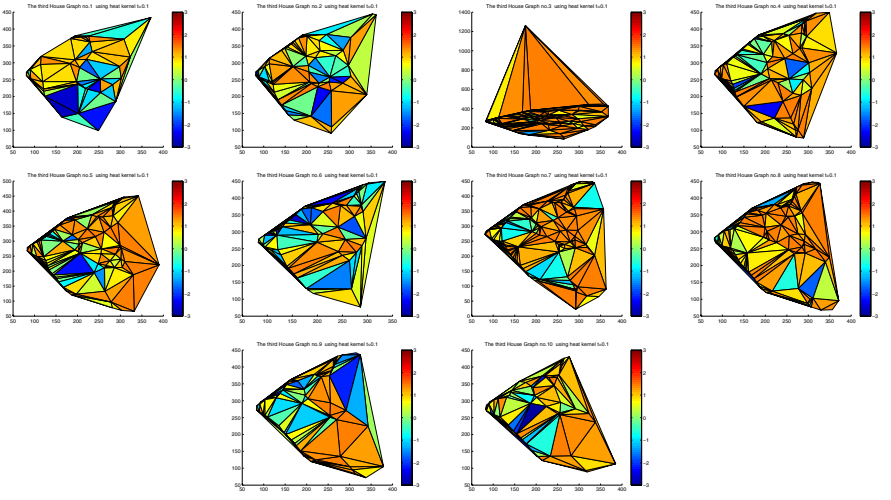


Fig. 9. The distribution of the Gaussian curvatures of the geodesic triangles for the ten graphs of the 3rd house

positive (elliptical) to negative (hyperbolic) regions. This suggests that the arrangement of triangles and their Gaussian curvatures could be used as the basis of a matching algorithm.

7 Conclusion

In this paper we have explored how to use the heat kernel to characterise graphs in a geometric manner. We commence by performing a Young-Householder decomposition on the heat kernel to recover the matrix of embedding co-ordinates. Once embedded we explore a number of alternative ways of characterising the graphs in a geometric manner. These include the sectional curvatures of edges estimated from the geodesic and Euclidean distances between nodes, and the Gaussian curvature of first order cycles. The conclusions of our experimental study is that both characterisations are effective as a means of characterising graphs for the purposes of clustering.

Our future plans revolve around developing ways of controlling noise in the representation. Here we aim to exploit graph-spectral regularisation [27] and curvature-based diffusion methods. We also aim to explore whether the pattern of sectional and Gaussian curvatures can be used to construct pattern spaces for graphs.

There are clearly a number of ways in which the work reported in this paper can be extended. First, it would be interesting to explore the use of the sectional curvature as a means of directly embedding the nodes of the graphs on a manifold. One of the possibilities that exists here is the variant of MDS reported by

Lindman and Caelli [13]. A second line of investigation would be the Euclidean distances or sectional curvature associated with the edges as attributes for the purposes of matching. Finally, it would be interesting to investigate if the distances and curvatures could be used to aid the process of visualising or drawing graphs.

References

1. Atkins, J.E., Boman, E.G., Hendrickson, B.: A spectral algorithm for seriation and the consecutive ones problem. SIAM J. Comput. 28(1), 297–310 (1998)
2. Barlow, M.T.: Diffusions on fractals. Lecture Notes Math., vol. 1690, pp. 1–121. Springer, Heidelberg (1998)
3. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems, vol. 14 (2002)
4. Chung, F.R.K.: Spectral graph theory. CBMS 92 (1997)
5. Cox, T., Cox, M.: Multidimensional Scaling. Chapman-Hall, Boca Raton (1994)
6. de Verdi‘ere, Y.C.: Spectres de graphes. Societe Mathematique De France (1998)
7. Dubuisson, M., Jain, A.: A modified Hausdorff distance for object matching, pp. 566–568 (1994)
8. Gilkey, P.B.: Invariance theory, heat equation, and the index theorem. Mathematics Lecture Series (1984)
9. Grigor'yan, A.: Heat kernels on manifolds, graphs and fractals. European Congress of Mathematics I, 393–406 (2001)
10. Heut, B., Hancock, E.R.: Relational object recognition from large structural libraries. Pattern Recognition 32, 1895–1915 (2002)
11. Huttenlocher, D., Klanderman, G., Rucklidge, W.: Comparing images using the Hausdorff distance. IEEE. Trans. Pattern Anal. Mach. Intell. 15, 850–863 (1993)
12. Lebanon, G., Lafferty, J.D.: Hyperplane margin classifiers on the multinomial manifold. In: ICML (2004)
13. Lindman, H., Caelli, T.: Constant curvature Riemannian scaling. Journal of Mathematical Psychology 17, 89–109 (1978)
14. Linial, N., London, E., Rabinovich, Y.: The geometry of graphs and some of its algorithmic applications. Combinatorica 15, 215–245 (1995)
15. Luo, B., Hancock, E.R.: Structural graph matching using the EM algorithm and singular value decomposition. IEEE Trans. Pattern Anal. Mach. Intell. 23(10), 1120–1136 (2001)
16. Luo, B., Wilson, R.C., Hancock, E.R.: Spectral embedding of graphs. Pattern Recogintion 36, 2213–2230 (2003)
17. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science 290 (5500), 2323–2326 (2000)
18. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE PAMI 22, 888–905 (2000)
19. Shokoufandeh, A., Dickinson, S.J., Siddiqi, K., Zucker, S.W.: Indexing using a spectral encoding of topological structure. In: CVPR, pp. 2491–2497 (1999)
20. Smola, E.J., Kondor, R.: Kernels and regularization on graphs (2004)
21. Spivak, M.: A Comprehensive Introduction to Differential Geometry, 2nd edn., vol. 1-5. Publish or Parish, Houston (1979)
22. Stillwell, J.: Mathematics and its History. Springer, New York (1974)

23. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319 (2000)
24. Umeyama, S.: An eigendecomposition approach to weighted graph matching problems. IEEE Trans. Patt. Anal. Mach. Intell. 10, 695–703 (1988)
25. Xiao, B., Hancock, E.R.: Trace formula analysis of graphs. In: Yeung, D.-Y., Kwok, J.T., Fred, A., Roli, F., de Ridder, D. (eds.) SSPR 2006 and SPR 2006. LNCS, vol. 4109, pp. 306–313. Springer, Heidelberg (2006)
26. Yau, S.T., Schoen, R.M.: Differential Geometry. Science Publication Co. (1988) (in Chinese)
27. Zhou, D., Schölkopf, B.: A regularization framework for learning from graph data. In: ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields, pp. 132–137 (2004)
28. Zhu, X., Kandola, J.S., Ghahramani, Z., Lafferty, J.D.: Nonparametric transforms of graph kernels for semi-supervised learning. In: NIPS (2004)

Graph Regularisation Using Gaussian Curvature

Hewayda ElGhawalby[1,2] and Edwin R. Hancock[1,⋆]

[1] Department of Computer Science, University of York, YO10 5DD, UK
[2] Faculty of Engineering, Suez Canal university, Egypt
{howaida,erh}@cs.york.ac.uk

Abstract. This paper describes a new approach for regularising triangulated graphs. We commence by embedding the graph onto a manifold using the heat-kernel embedding. Under the embedding, each first-order cycle of the graph becomes a triangle. Our aim is to use curvature information associated with the edges of the graph to effect regularisation. Using the difference in Euclidean and geodesic distances between nodes under the embedding, we compute sectional curvatures associated with the edges of the graph. Using the Gauss Bonnet Theorem we compute the Gaussian curvature associated with each node from the sectional curvatures and through the angular excess of the geodesic triangles. Using the curvature information we perform regularisation with the advantage of not requiring the solution of a partial differential equation. We experiment with the resulting regularization process, and explore its effect on both graph matching and graph clustering.

Keywords: Manifold regularization, Heat kernel, Hausdorff distance, Gaussian curvature, Graph matching.

1 Introduction

In computer vision, image processing and graphics the data under consideration frequently exists in the form of a graph or a mesh. The fundamental problems that arise in the processing of such data are how to smooth, denoise, restore and simplify data samples over a graph. The Principal difficulty of this task is how to preserve the geometrical structures existing in the initial data. Many methods have been proposed to solve this problem. Among existing methods, variational techniques based on regularization, provide a general framework for designing efficient filtering processes. Solutions to the variational models can be obtained by minimizing an appropriate energy function. The minimization is usually performed by designing a continuous partial differential equation, whose solutions are discretized in order to fit with the data domain. A complete overview of these methods in image processing can be found in ([1,2,3,4]). One of the problems associated with variational methods is that of distretisation, which for some types

⋆ The authors acknowledge the financial support from the FET programme within the EU FP7, under the SIMBAD project (contract 213250).

A. Torsello, F. Escolano, and L. Brun (Eds.): GbRPR 2009, LNCS 5534, pp. 233–242, 2009.

of data can prove to be intractable. An alternative to the variational approach is to make direct use of differential geometry and the calculus of variation to regularize data on manifolds. There are two principal ways in which this may be effected. The first approach is to use an intrinsic-parametric description of the manifold and an explicit form of the metric, referred to as the Polyakov action [12,22,23,24,25]. The second approach is to use an implicit representation of the manifold, referred to as the harmonic map [1,5,6,16,18]. In [19,20,21], the relation between these two approaches was explained and a new approach regularization on manifolds refered to as the Beltrami flow was introduced. An implementation for the case of a manifold represented by a level set surface was introduced in [19]. A method to compute the Beltrami flow for scalar functions defined on triangulated manifolds using a local approximation of the operator was proposed in [14].

The Laplace-Beltrami operator on a Riemannian manifold has been extensively studied in the mathematics literature. Recently, there has been intense interest in the spectral theory of the operator, and this has lead to the field of study referred to as spectral geometry. This work has established relations between its first eigenvalue of the Beltrami operator and the geometrical properties of the manifold including curvatures, diameter, injectivity radius and volume. Recently, an alternative operator referred to as the p-Laplacian has attracted considerable interest, and has proved a powerful means of solving geometric nonlinear partial differential equations arising in non-Newtonian fluids and nonlinear elasticity.

In prior work [10], we have explored the problem of how to characterise graphs in a geometric manner. The idea has been to embed graphs in a vector-space. Under this embedding nodes become points on a manifold, and edges become geodesics on the manifold. We use the differences between the geodesic and Euclidean distances between points (i.e. nodes) connected by an edge to associate sectional curvatures with edges. Using the Gauss-Bonnet theorem, we can extend this characterisation to include the Gauss curvatures associated with nodes (i.e. points on a manifold). Unfortunately, the approximations required to compute these curvature characterisations can lead to unstable values. For this reason in this paper, we turn to regularisation as a means of smoothing the Guassian curvature estimates. To do this we investigate two cases of the p-Laplacian, the Laplace and Curvature operators, for the purpose of regularisation, and use the Gaussian curvature associated with the heat-kernel embedding of nodes as regularisation function on the manifold. The idea of using functionals on graphs, in a regularization process, has also been proposed in other contexts, such as semi-supervised data learning [28,29] and image segmentation [2].

2 Functions and Operators on Graphs

In this section, we recall some basic prerequisites concerning graphs, and define nonlocal operators which can be considered as discrete versions of continuous differential operators.

2.1 Preliminaries

An undirected unweighted graph denoted by $G = (V, E)$ consists of a finite set of nodes V and a finite set of edges $E \subseteq V \times V$. The elements of the adjacency matrix A of the graph G is defined by:

$$A(u,v) = \begin{cases} 1 & if(u,v) \in E \\ 0 & otherwise \end{cases} \tag{1}$$

To construct the Laplacian matrix we first establish a diagonal degree matrix D with elements $D(u,u) = \sum_{v \in V} A(u,v) = d_u$. From the degree and the adjacency matrices we can construct the Laplacian matrix $L = D - A$, that is the degree matrix minus the adjacency matrix.

$$L(u,v) = \begin{cases} d_u & if u = v \\ -1 & if(u,v) \in E \\ 0 & otherwise \end{cases} \tag{2}$$

The normalized Laplacian is given by $\widehat{L} = D^{-1/2}LD^{-1/2}$. The spectral decomposition of the normalized Laplacian matrix is $\widehat{L} = \Phi L \Phi^T = \sum_{i=1}^{|V|} \lambda_i \phi_i \phi_i^T$ where $|V|$ is the number of nodes and $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_{|V|}), (0 < \lambda_1 < \lambda_2 < ... < \lambda_{|V|})$ is the diagonal matrix with the ordered eigenvalues as elements and $\Phi = (\phi_1 | \phi_2 | ... | \phi_{|V|})$ is the matrix with the eigenvectors as columns.

2.2 Embedding Graphs onto Manifolds

We follow Bai and Hancock [26] and make use of the heat kernel embedding. The heat kernel plays an important role in spectral graph theory. It encapsulates the way in which information flows through the edges of graph over time under the heat equation, and is the solution of the partial differential equation

$$\frac{\partial h_t}{\partial t} = -\widehat{L} h_t \tag{3}$$

where h_t is the heat kernel and t is the time. The solution is found by exponentiating the Laplacian eigenspectrum as follows

$$h_t = \exp[-\widehat{L}t] = \Phi \exp[-t\Lambda]\Phi^T \tag{4}$$

For the heat kernel, the matrix of embedding coordinates Y (i.e. the matrix whose columns are the vectors of node coordinates) is found by performing the Young-Householder [27] decomposition $h_t = Y^T Y$ as a result the matrix of node embedding coordinates is

$$Y = (y_1 | y_2 | ... | y_{|V|}) = \exp[-\frac{1}{2} t\Lambda]\Phi^T \tag{5}$$

where y_u is the coordinate vector for the node u. In the vector space, the Euclidean distance between the nodes u and v of the graph is

$$d_e^2(u,v) = (y_u - y_v)^T (y_u - y_v) = \sum_{i=1}^{|V|} \exp[-\lambda_i t](\phi_i(u) - \phi_i(v))^2 \tag{6}$$

2.3 Functions on Graphs

For the purpose of representing the data we use a discrete real-valued function $f : V \rightarrow \Re$, which assigns a real value $f(u)$ to each vertex $u \in V$. Functions of this type form a discrete N-dimensional space. They can be represented by vectors of $\Re^N$, $f = f(u)_{u \in V}$, and interpreted as the intensity of a discrete signal defined on the vertices of the graph. By analogy with continuous functional spaces, the discrete integral of a function $f : V \rightarrow \Re$, on the graph G, is defined by $\int_G f = \sum_{u \in V} f(u)$. Let $H(V)$ denote the Hilbert space of the real-valued functions on the vertices of G and endowed with the usual inner product:

$$\langle f,h \rangle_{H(V)} = \sum_{u \in V} f(u)h(u) \quad , \quad f,h : V \rightarrow \Re \tag{7}$$

with the induced L_2- norm: $\|f\|_2 = \langle f,h \rangle^{1/2}_{H(V)}$.

2.4 The *p*-Laplacian Operator

For a smooth Riemannian manifold M and a real number $p \in (1, +\infty)$, the p-Laplacian operator of a function $f \in H(V)$, denoted $L^p : H(V) \rightarrow H(V)$ is defined by

$$L^p f(u) = \frac{1}{2} \sum_{v \sim u} \left(\frac{f(u) - f(v)}{(\sqrt{\sum_{u \sim v} (f(u) - f(v))^2})^{p-2}} + \frac{f(u) - f(v)}{(\sqrt{\sum_{v \sim u} (f(v) - f(u))^2})^{p-2}} \right) \tag{8}$$

This operator arises naturally from the variational problem associated to the energy function [13]. The p-Laplace operator is nonlinear, with the exception of p=2, where it corresponds to the combinatorial graph Laplacian, which is one of the classical second order operators defined in the context of spectral graph theory [7]

$$Lf(u) = \sum_{v \sim u} (f(u) - f(v)) \tag{9}$$

Another particular case of the p-Laplace operator is obtained with p=1 . In this case, it is the curvature of the function f on the graph

$$\kappa f(u) = \frac{1}{2} \sum_{v \sim u} \left(\frac{f(u) - f(v)}{\sqrt{\sum_{u \sim v} (f(u) - f(v))^2}} + \frac{f(u) - f(v)}{\sqrt{\sum_{v \sim u} (f(v) - f(u))^2}} \right) \tag{10}$$

κ corresponds to the curvature operator proposed in [17] and [4] in the context of image restoration. More generally, κ is the discrete analogue of the mean curvature of the level curve of a function defined on a continuous domain of $\Re^N$.

3 The Gaussian Curvature

Curvature is a local measure of geometry and can be used to represent local shape information. We choose the function f to be the Gaussian curvature defined

over the vertices. Gaussian curvature is one of the fundamental second order geometric properties of a surface, and it is an intrinsic property of a surface independent of the coordinate system used to describe it. As stated by Gauss's theorema egregium [11], it depends only on how distance is measured on the surface, not on the way it is embedded on the space.

3.1 Geometric Preliminaries

Let T be the embedding of a triangulated graph onto a smooth surface M in $\Re^3$, A_g be the area of a geodesic triangle on M with angles $\{\alpha_i\}_{i=1}^3$ and geodesic edge lengths $\{d_{gi}\}_{i=1}^3$, and A_e be the area of the corresponding Euclidean triangle with edge lengths $\{d_{ei}\}_{i=1}^3$ and angles $\{\varphi_i\}_{i=1}^3$. Assuming that each geodesic is a great arc on a sphere with radius R_i, $i = 1, 2, 3$ corresponding to a central angle 2θ, and that the geodesic triangle is a triangle on the surface of a sphere with radius $R = \frac{1}{3}\sum_{i=1}^3 R_i$, with the Euclidean distance between the pair of nodes to be $d_e = \frac{1}{3}\sum_{i=1}^3 d_{ei}$. Considering a small area element on the sphere given in spherical coordinates by $dA = R^2 \sin\theta d\theta d\varphi$, the integration of dA bounded by 2θ gives us the following formula for computing the area of the geodesic triangle

$$A_g = \frac{1}{2R} d_e^2 \tag{11}$$

where d_e^2 is computed from the embedding using (6).

3.2 Gaussian Curvature from Gauss Bonnet Theorem

For a smooth compact oriented Riemannian 2-manifold M, let $\triangle_g$ be a triangle on M whose sides are geodesics, i.e. paths of shortest length on the manifold. Further, let α_1, α_2 and α_3 denote the interior angles of the triangle. According to Gauss's theorem, if the Gaussian curvature K (i.e. the product of the maximum and minimum curvatures at a point on the manifold) is integrated over $\triangle_g$, then

$$\int_{\triangle_g} K dM = \sum_{i=1}^{3} \alpha_i - \pi \tag{12}$$

where dM is the Riemannian volume element. Since all the points, except for the vertices, of a piecewise linear surface have a neighborhood isometric to a planar Euclidean domain with zero curvature, the Gauss curvature is concentrated at the isolated vertices. Hence, to estimate the Gaussian curvature of a smooth surface from its triangulation, we need to normalize by the surface area, which here is the area of the triangle. Consequently, we will assign one third of the triangle area to each vertex. Hence, the Gaussian curvature associated with each vertex will be

$$\kappa_g = \frac{\int_{\triangle_g} K dM}{\frac{1}{3}A} \tag{13}$$

from (12) we get

$$\kappa_g = \frac{\sum_{i=1}^{3} \alpha_i - \pi}{\frac{1}{3}A} \tag{14}$$

Substituting by the area from (12), eventually we get

$$\kappa_g = \frac{3}{R^2} \tag{15}$$

Recalling that the Gaussian curvature is the product of the two principle curvatures, and that the curvature of a point on a sphere is the reciprocal of the radius of the sphere gives an explanation for the result in (15). As we assumed earlier that the geodesic is a great arc of a circle of radius R, in [10] we deduced that

$$\frac{1}{R^2} = d_g - \frac{24(d_g - d_e)}{d_g^3} \tag{16}$$

and since for an edge of the graph $d_g = 1$, we have

$$\frac{1}{R^2} = 24(1 - d_e) \tag{17}$$

From (15) and (17) the Gaussian curvature associated with the embedded node can be found from the following formula

$$\kappa_g = 72(1 - d_e) \tag{18}$$

4 Hausdorff Distance

We experiment with the Gaussian curvatures as node-based attributes for the purposes of graph-matching. We represent the graphs under study using sets of curvatures, and compute the similarity of sets resulting from different graphs using the robust modified Hausdorff distance. The Hausdorff distance provides a means of computing the distance between sets of unordered observations when the correspondences between the individual items are unknown. In its most general setting, the Hausdorff distance is defined between compact sets in a metric space. Given two such sets, we consider for each point in one set is the closest point in the second set. The modified Hausdorff distance is the average over all these values. More formally, the modified Hausdorff distance (HD) [9] between two finite point sets A and B is given by

$$H(A, B) = \max(h(A, B), h(B, A)) \tag{19}$$

where the directed modified Hausdorff distance from A to B is defined to be

$$h(A, B) = \frac{1}{N_A} \sum_{a \in A} \min_{b \in B} \|a - b\| \tag{20}$$

and $\|.\|$ is some underlying norm on the points of A and B (e.g., the L2 or Euclidean norm). Using these ingredients we can describe how the modified Hausdorff distances can be extended to graph-based representations. To commence let us consider two graphs $G_1 = (V_1, E_1, T_1, \kappa_1)$ and $G_2 = (V_2, E_2, T_1, \kappa_2)$, where V_1,V_2 are the sets of nodes, E_1, E_2 the sets of edges, T_1,T_2 are the sets of triangles, and κ_1,κ_2 the sets of the Gaussian curvature associated with each node defined in §3.2. We can now write the distances between two graphs as follows:

$$h_{MHD}(G_1, G_2) = \frac{1}{|V_1|} \sum_{i \in V_1} \min_{j \in V_2} \|\kappa_2(j) - \kappa_1(i)\|) \tag{21}$$

5 Multidimensional Scaling

For the purpose of visualization, the classical Multidimensional Scaling (MDS) [8] is a commonly used technique to embed the data specified in the matrix in Euclidean space. Given that H is the distance matrix with row r and column c entry H_{rc}. The first step of MDS is to calculate a matrix T whose element with row r and column c is given by $T_{rc} = -\frac{1}{2}[H^2_{rc} - \widehat{H}^2_{r.} - \widehat{H}^2_{.c} + \widehat{H}^2_{..}]$ where $\widehat{H}_{r.} = \frac{1}{N}\sum_{c=1}^{N} H_{rc}$ is the average value over the rth row in the distance matrix, $H_{.c}$ is the similarly defined average value over the cth column and $\widehat{H}_{..} = \frac{1}{N^2}\sum_{r=1}^{N}\sum_{c=1}^{N} H_{rc}$ is the average value over all rows and columns of the distance matrix. Then, we subject the matrix T to an eigenvector analysis to obtain a matrix of embedding coordinates X. If the rank of T is k; $k \leq N$, then we will have $k$non-zero eigenvalues. We arrange these k non-zero eigenvalues in descending order, i.e., $l_1 \geq l_2 \geq ... \geq l_k \geq 0$. The corresponding ordered eigenvectors are denoted by u_i where l_i is the ith eigenvalue. The embedding coordinate system for the graphs is $X = [\sqrt{l_1}u_1, \sqrt{l_2}u_2, ..., \sqrt{l_k}u_k]$ for the graph indexed i, the embedded vector of the coordinates is $x_i = (X_{i,1}, X_{i,2}, ..., X_{i,k})^T$.

6 Experiments

For the purposes of experimentation we use the standard CMU, MOVI and chalet house image sequences as our data set [15]. These data sets contain different views of model houses from equally spaced viewing directions. From the house images, corner features are extracted, and Delaunay graphs representing the arrangement of feature points are constructed. Our data consists of ten graphs for each of the three houses. To commence, we compute the Euclidean distances between the nodes in each graph based on the Laplacian and then on the heat kernel with the values of $t = 10.0, 1.0, 0.1$ and 0.01. Then we compute the Gaussian curvature associated with each node using the formula given in §.

Commencing with each node attributed with the the Gaussian curvatures (as the value of a real function f acting on the nodes of the graph), we can regularise each graph by applying the the p-Laplacian operator to the Gaussian curvatures. For each graph we construct a set of regularised Gaussian curvatures using both the Laplace operator and the curvature operator, as a special cases

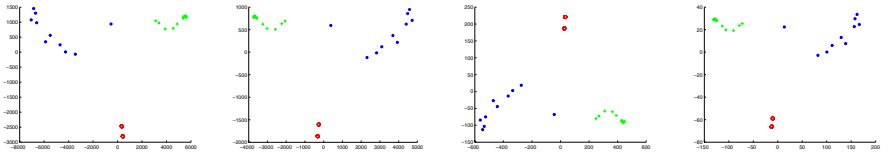


Fig. 1. MDS embedding obtained using Laplace operator to regularize the houses data resulting from the heat kernel embedding

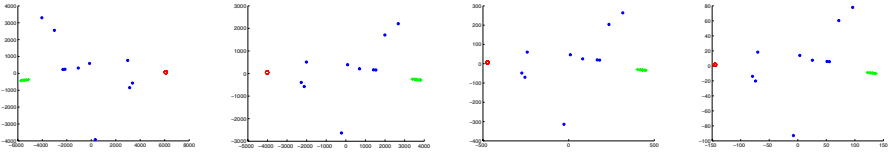


Fig. 2. MDS embedding obtained using Curvature operator to regularize houses data resulting from the heat kernel embedding

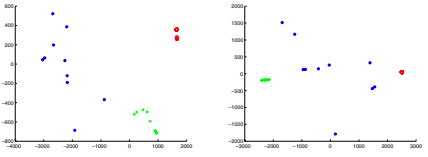


Fig. 3. MDS embedding obtained using Laplace operator(left) and the Curvature oprator (right) to regularize the houses data resulting from the Laplacian embedding

of the p-Laplacian operator. The next step is to compute the distances between the sets for the thirty different graphs using the modified Hausdorff distance. Finally, we subject the distance matrices to the Multidimensional Scaling (MDS) procedure to embed them into a 2D space. Here each graph is represented by a single point. Figure 1 shows the results obtained using the Laplace operator. The subfigures are ordered from left to right, using the heat kernel embedding with the values $t = 10.0, 1.0, 0.1$ and 0.01. Figure 2 shows the corresponding results obtained when the Curvature operator is used. Figure 3 shows the results obtained whenusing the Laplacian embedding, from the Laplace operator (left) and the Curvature operator (right).

To investigate the results in more detail table 1 shows the rand index for the distance as a function of t. This index is computed as follows: 1) compute the mean for each cluster; 2) compute the distance from each point to each mean; 3) if the distance from correct mean is smaller than those to remaining means, then classification is correct, if not then classification is incorrect; 4) compute the Rand-index (incorrect/(incorrect+correct)).

Table 1. A rand index vs. t

	lap	t=10	t=1.0	t=0.1	t=0.01
Lalplace operator	0.0000	0.0000	0.0000	0.0000	0.0000
Curvature operator	0.1667	0.0000	0.0000	0.0000	0.0000

From this experimental study, we conclude that the proposed regularization procedure, using two special cases of the p-Laplacian operator (Laplace and Curvature operators) improves the processes of graph matching and clustering.

7 Conclusion and Future Work

In this paper, a process for regularizing the curvature attributes associated with the geometric embedding of graphs was presented. Experiments show that it is an efficient procedure for the purpose of gauging the similarity of pairs of graphs. The regularisation procedure improves the results obtained with graph clustering. Our future plans are twofold. First, we aim to explore if geodesic flows along the edges of the graphs can be used to implement a more effective regularisation process. Second, we aim to apply our methods to problems of image and mesh smoothing.

References

1. Bertalmio, M., Cheng, L.T., Osher, S., Sapiro, G.: Variational problems and partial differential equations on implicit surfaces. Journal of Computational Physics 174, 759–780 (2001)
2. Bougleux, S., Elmoataz, A.: Image smoothing and segmentation by graph regularization. LNCS, vol. 3656, pp. 745–752. Springer, Heidelberg (2005)
3. Boykov, Y., Huttenlocher, D.: A new bayesian framework for object recognition. In: Proceeding of IEEE Computer Society Conference on CVPR, vol. 2, pp. 517–523 (1999)
4. Chan, T., Osher, S., Shen, J.: The digital tv filter and nonlinear denoising. IEEE Trans. Image Process 10(2), 231–241 (2001)
5. Chan, T., Shen, J.: Variational restoration of non-flat image features: Models and algorithms. SIAM J. Appl. Math. 61, 1338–1361 (2000)
6. Cheng, L., Burchard, P., Merriman, B., Osher, S.: Motion of curves constrained on surfaces using a level set approach. Technical report, UCLA CAM Technical Report (00-32) (September 2000)
7. Chung, F.R.: Spectral graph theory. In: Proc. CBMS Regional Conf. Ser. Math., vol. 92, pp. 1–212 (1997)
8. Cox, T., Cox, M.: Multidimensional Scaling. Chapman-Hall, Boca Raton (1994)
9. Dubuisson, M., Jain, A.: A modified hausdorff distance for object matching, pp. 566–568 (1994)
10. ElGhawalby, H., Hancock, E.R.: Measuring graph similarity using spectral geometry. In: Campilho, A., Kamel, M.S. (eds.) ICIAR 2008. LNCS, vol. 5112, pp. 517–526. Springer, Heidelberg (2008)

11. Gauss, C.F.: Allgemeine Flächentheorie (Translated from Latin). W. Engelmann (1900)
12. Kimmel, R., Malladi, R., Sochen, N.: Images as embedding maps and minimal surfaces: Movies, color, texture, and volumetric medical images. International Journal of Computer Vision 39(2), 111–129 (2000)
13. Lim, B.P., Montenegro, J.F., Santos, N.L.: Eigenvalues estimates for the p-laplace operator on manifolds. arXiv:0808.2028v1 [math.DG], August 14 (2008)
14. Lopez-Perez, L., Deriche, R., Sochen, N.: The beltrami flow over triangulated manifolds. In: Sonka, M., Kakadiaris, I.A., Kybic, J. (eds.) CVAMIA/MMBIA 2004. LNCS, vol. 3117, pp. 135–144. Springer, Heidelberg (2004)
15. Luo, B., Wilson, R.C., Hancock, E.R.: Spectral embedding of graphs. Pattern Recogintion 36, 2213–2230 (2003)
16. Memoli, F., Sapiro, G., Osher, S.: Solving variational problems and partial differential equations, mapping into general target manifolds. Technical report, UCLA CAM Technical Report (02-04) (January 2002)
17. Osher, S., Shen, J.: Digitized pde method for data restoration. In: Anastassiou, E.G.A. (ed.) Analytical-Computational methods in Applied Mathematics, pp. 751–771. Chapman & Hall/CRC, New York (2000)
18. Sapiro, G.: Geometric Partial Differential Equations and Image Analysis. Cambridge University Press, Cambridge (2001)
19. Sochen, N., Deriche, R., Lopez-Perez, L.: The beltrami flow over implicit manifolds. In: ICCV (2003)
20. Sochen, N., Deriche, R., Lopez-Perez, L.: Variational beltrami flows over manifolds. In: IEEE ICIP 2003, Barcelone (2003)
21. Sochen, N., Deriche, R., Lopez-Perez, L.: Variational beltrami flows over manifolds. Technical report, INRIA Resarch Report 4897 (June 2003)
22. Sochen, N., Kimmel, R.: Stereographic orientation diffusion. In: Proceedings of the 4th Int. Conf. on Scale-Space, Vancouver, Canada (October 2001)
23. Sochen, N., Kimmel, R., Malladi, R.: From high energy physics to low level vision. Report, LBNL, UC Berkeley, LBNL 39243, August, Presented in ONR workshop, UCLA, September 5 (1996)
24. Sochen, N., Kimmel, R., Malladi, R.: A general framework for low level vision. IEEE Trans. on Image Processing 7, 310–318 (1998)
25. Sochen, N., Zeevi, Y.: Representation of colored images by manifolds embedded in higher dimensional non-euclidean space. In: Proc. IEEE ICIP 1998, Chicago (1998)
26. Xiao, B., Hancock, E.R.: Heat kernel, riemannian manifolds and graph embedding. In: Fred, A., Caelli, T.M., Duin, R.P.W., Campilho, A.C., de Ridder, D. (eds.) SSPR&SPR 2004. LNCS, vol. 3138, pp. 198–206. Springer, Heidelberg (2004)
27. Young, G., Householder, A.S.: Disscussion of a set of points in terms of their mutual distances. Psychometrika 3, 19–22 (1938)
28. Zhou, D., Schlkopf, B.: Regularization on discrete spaces. In: Kropatsch, W.G., Sablatnig, R., Hanbury, A. (eds.) DAGM 2005. LNCS, vol. 3663, pp. 361–368. Springer, Heidelberg (2005)
29. Zhou, D., Schlkopf, B.: In: Chapelle, O., Schlkopf, B., Zien, A. (eds.) Semi-supervised learning, pp. 221–232 (2006)

Appendix C

Embedding Indeffinite Similarities

Embedding Indefinite Similarities on Curved Surfaces

Richard C. Wilson

September 21, 2009

1 Dissimilarities

In many pattern recognition problems, it is possible to define a distance between patterns which reflects the structure of the problem.

$$D_{ij} = d(x_i, x_j) \tag{1}$$

Alternatively, we can use a similarity, which is related to the distance by

$$S_{ij} = -\frac{1}{2} D_{ij}^2 \tag{2}$$

This the equivalent to a kernel matrix, but in general will not be a valid kernel. The centred similarity matrix is given by

$$\mathbf{K} = (\mathbf{I} - \mathbf{J}/n)\mathbf{S}(\mathbf{I} - \mathbf{J}/n) \tag{3}$$

This is a kernel matrix provided it is positive semi-definite and the objects x_i can then be embedded in a Euclidean space using the kernel embedding

$$\mathbf{X} = \mathbf{U}_K \mathbf{\Lambda}_K^{\frac{1}{2}} \tag{4}$$

where $\mathbf{U}_K$ and $\mathbf{\Lambda}_K$ are the eigenvector and eigenvalue matrices of $\mathbf{K}$, respectively.

If $\mathbf{K}$ is indefinite, which is often the case, then the objects cannot exist in Euclidean space with the given distances. Previous work has shown that they can be embedded in a non-Riemannian pseudo-Euclidean space[2]. This space uses the non-Euclidean inner product

$$< \mathbf{x}, \mathbf{y} >= \mathbf{x}^T \mathbf{M} \mathbf{y} \tag{5}$$

where

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & \dots & & 0 \\ 0 & 1 & & & \vdots \\ \vdots & & \ddots & & \\ & & \dots & -1 & \\ & & & & \ddots \end{pmatrix}$$

The values of -1 correspond to the 'negative' part of the space. This inner-product induces a norm, or distance measure:

$$|\mathbf{x}|^2 =< \mathbf{x}, \mathbf{x} >= \mathbf{x}^T\mathbf{M}\mathbf{x} = \sum_{i_+} x_i^2 - \sum_{i_-} x_i^2 \tag{6}$$

We can then write the kernel as

$$\mathbf{K} = \mathbf{U}_K|\mathbf{\Lambda}_K|^{\frac{1}{2}} M |\mathbf{\Lambda}_K|^{\frac{1}{2}} \mathbf{U}_K^T \tag{7}$$

where the negative part of the space corresponds to the negative eigenvalues of $\mathbf{K}$, and the kernel embedding as

$$\mathbf{X} = \mathbf{U}_K|\mathbf{\Lambda}_K|^{\frac{1}{2}} \tag{8}$$

So the pseudo-Euclidean embedding reproduces precisely the original distance and similarity matrices. But, while the pseudo-Euclidean embedding reproduces the original distance matrix, it introduces a number of other problems. The embedding space is non-metric and points in the space can have negative distances to each other. Locality is not preserved in such a space.

2 Riemannian Embedding

An n-dimensional Riemannian space is defined by its metric tensor g_{ij} in some local coordinate system $\{u_1, u_2 \dots u_n\}$. This can be related to an infinitesimal distance element in the space by

$$ds^2 = \sum_{ij} g_{ij} du_i du_j \tag{9}$$

For the space to be Riemannian, the metric tensor must be positive definite. Distances between points the space are measured by geodesics; the shortest possible path between the two points. A Riemannian space may be given any positive definite metric tensor. We can define a space by giving an embedding for it in an $n+1$ dimensional flat (Euclidean or pseudo-Euclidean) space. The embedding then implies a particular metric for the space. Note that the converse is not true; there may be many different embeddings with the same metric and therefore the same Riemannian space. Finally, the geodesic distance in a Riemannian space is metric distance, but in general it is non-Euclidean. However, finding the geodesic between two points on the manifold is not easy; it involves solving a set of coupled second-order differential equations. There are, however, manifolds which are non-Euclidean but on which it is easy to find the geodesics. We discuss two alternatives in the next section.

2.1 Elliptic geometry

Elliptic geometry is the geometry on the surface of a hypersphere. The hypersphere can be straightforwardly embedded in Euclidean space; for example the

embedding of a sphere in three dimensions is well known:

$$\begin{aligned} x &= r \sin u \sin v \\ y &= r \cos u \sin v \\ z &= r \cos v \end{aligned}$$

As noted above, this embedding implies a particular metric tensor.

$$\begin{aligned} ds^2 &= dx^2 + dy^2 + dz^2 \\ &= r^2 \sin^2 v du^2 + r^2 dv \end{aligned} \tag{10}$$

and so the metric tensor is

$$g = r^2 \begin{pmatrix} \sin^2 v & 0 \\ 0 & 1 \end{pmatrix} \tag{11}$$

The embedding of an $(n-1)$-dimensional sphere in n-dimensional space is an extension of this. We can define the surface implicitly using the constraint

$$\sum_i x_i^2 = r^2 \tag{12}$$

For the hypersphere to be a Riemannian space, we should have a positive definite metric tensor g. This is equivalent to the statement $ds^2 > 0$ for any infinitesimal movement in the surface. We have

$$ds^2 = \sum_i dx_i^2 \tag{13}$$

which clearly must be positive for any values of dx_i. This surface has a constant sectional curvature of $K = 1/r^2$ everywhere.

The geodesic distance between two points in curved space is the length of the shortest curve lying in the space and joining the two points. For an elliptic space, the geodesic is a great circle on the hypersphere. The distance is the length of the arc of the great circle which joins the two points. If the angle subtended by two points at the centre of the hypersphere is θ_{ij}, then the distance between them is

$$d_{ij} = r\theta_{ij} \tag{14}$$

With the coordinate origin at the centre of the hypersphere, we can represent a point by a position vector $\mathbf{x}_i$ of length r. Since the inner product is $< \mathbf{x}_i, \mathbf{x}_j >= r^2 \cos\theta_{ij}$ we can also write

$$d_{ij} = r \cos^{-1} \frac{< \mathbf{x}_i, \mathbf{x}_j >}{r^2} \tag{15}$$

2.2 Hyperbolic geometry

As we previously observed, the pseudo-Euclidean(pE) space has been used to embed points derived from indefinite kernels. The pE space is clearly non-Riemannian as points may have negative distances to each other. However, it is still possible to define a sub-space which is Riemannian. As an example, take the 3D pE space with a single negative dimension (z) and the 'sphere' defined by

$$< \mathbf{x}, \mathbf{x} >= x^2 + y^2 - z^2 = -r^2 \tag{16}$$

This space is called *hyperbolic*.

This surface has a parameterisation given by

$$\begin{aligned} x &= r \sin u \sinh v \\ y &= r \cos u \sinh v \\ z &= r \cosh v \end{aligned} \tag{17}$$

As before, there is a particular metric tensor associated with this embedding:

$$\begin{aligned} ds^2 &= dx^2 + dy^2 - dz^2 \\ &= r^2 \sinh^2 v du^2 + r^2 dv \qquad (18) \\ & \qquad (19) \end{aligned}$$

and so the metric tensor is

$$g = r^2 \begin{pmatrix} \sinh^2 v & 0 \\ 0 & 1 \end{pmatrix} \tag{20}$$

The metric tensor is positive definite, and so the surface is Riemannian and distances measured on the surface are metric, even though they are not in the embedding space.

We can extend this hyperbolic space to more dimensions. Firstly, we take the case when there is just one negative dimension, z in the embedding space.

$$\sum_i x_i^2 - z^2 = -r^2 \tag{21}$$

From this constraint we have $\sum x_i dx_i - z dz = 0$. The metric is then

$$\begin{aligned} ds^2 &= \sum_i dx_i^2 - dz^2 \\ z^2 ds^2 &= z^2 \sum_i dx_i^2 - z^2 dz^2 \\ &= z^2 \sum_i dx_i^2 - (\sum x_i dx_i)^2 \\ &= \sum_i dx_i^2 (z^2 - x_i^2) - 2 \sum_{i \neq j} x_i x_j dx_i dx_j \end{aligned}$$

$$
\begin{aligned}
&= \sum_i dx_i^2(r^2 + \sum_{j,i\neq j} x_i^2) - 2\sum_{ij,i\neq j} x_i x_j dx_i dx_j \\
&= r^2 \sum_i dx_i^2 + [\sum_{ij,i\neq j} (x_i dx_j - x_j dx_i)]^2 \\
&> 0 \qquad (22)
\end{aligned}
$$

so the hyperbolic surface is Riemannian.

In the case of two negative dimensions y and z, we have

$$\sum_i x_i^2 - y^2 - z^2 = -r^2 \qquad (23)$$

and therefore the constraint on an infinitesimal movement in the space $\sum x_i dx_i - ydy - zdz = 0$. The offset $dx_i = 0$, $dy = zdu$, $dz = -ydu$ satisfies this constraint and gives the metric

$$ds^2 = -dy^2 - dz^2 = -(y^2 + z^2)du^2 < 0 \qquad (24)$$

so this space is non-Riemannian. This argument can clearly be extended to more negative dimensions, so the hyperbolic space is metric only with one negative dimension.

Finally, the sectional curvature of this space, as with the hypersphere, is constant everywhere. In this case, the curvature is negative and given by $K = -1/r^2$. For the hyperbolic space, the geodesic is the analogue of a great circle. While the notion of angle in Euclidean space is intuitive, it is less so in pE space. However, we can define such a notion from the inner product. The inner product is defined as

$$
\begin{aligned}
< \mathbf{x}_i, \mathbf{x}_j > &= \sum_k x_{ik} x_{jk} - z_i z_j \qquad (25) \\
&= -|\mathbf{x}_i||\mathbf{x}_j| \cosh\theta_{ij} \qquad (26) \\
& \qquad (27)
\end{aligned}
$$

which in turn defines the notion of hyperbolic angle. From this angle, the distance between two points in the space is

$$d_{ij} = r\theta_{ij} \qquad (28)$$

With the coordinate origin at the centre of the hypersphere, we can represent a point by a position vector $\mathbf{x}_i$ of length r. Since the inner product is $< \mathbf{x}_i, \mathbf{x}_j >= r^2 \cosh\theta_{ij}$ we can also write

$$d_{ij} = r \cosh^{-1} - \frac{< \mathbf{x}_i, \mathbf{x}_j >}{r^2} \qquad (29)$$

3 Points in Elliptic and Hyperbolic spaces

The elliptic and hyperbolic spaces defined in the previous section are metric but clearly not Euclidean. They are therefore candidates for representing points

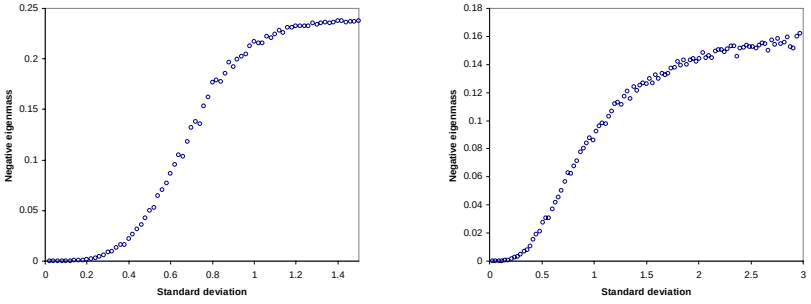


Figure 1: Negative eigenvalue fraction of points on elliptic and hyperbolic surfaces

which produce indefinite kernels. The first question we wish to answer is, to what extent do the points in a curved space produce indefinite kernels. To answer this question, we have constructed the kernel matrices of points in these spaces. The points are generated via a parameterisation (e.g. Eqns 10 and 17) and drawing points from the parameters via a normal distribution.

Figure 1 shows the fraction of negative eigenvalues for the centred kernel matrix of points on elliptic and hyperbolic surfaces. The negative eigenfraction is defined as

$$m_- = \frac{\sum_{\lambda_i<0} |\lambda_i|}{\sum_i |\lambda_i|}$$

These curved surfaces produce significant negative eigenfraction, up to 30% in the case of the elliptic surface, and are useful for modelling indefinite kernels.

3.1 Embedding in Elliptic space

Given a distance matrix $\mathbf{D}$, we wish to find the set of points in an elliptic space which produce the same distance matrix. Since the curvature of the space is unknown, we must additionally find the radius of the hypersphere. We have n objects of interest, and therefore we would normally look for an $n-1$-dimensional Euclidean space. Since we have freedom to set the curvature, we must look for a $n-2$-dimensional elliptic space embedded in the Euclidean space.

We begin by constructing a space with the origin at the centre of the hypersphere. If the point positions are given by $\mathbf{x}_i, i = 1 \dots n$, then we have

$$< \mathbf{x}_i, \mathbf{x}_j >= r^2 \cos\theta_{ij} = r^2 \cos\frac{d_{ij}}{r} \tag{30}$$

Next, we construct the matrix of point positions $\mathbf{X}$, with each position vector as a column. Then we have

$$\mathbf{X}\mathbf{X}^T = \mathbf{Z} \tag{31}$$

where $Z_{ij} = r^2 \cos d_{ij}/r$. Since the embedding space has dimension $n-1$, $\mathbf{X}$ consists of n points of dimension $n-1$ and $\mathbf{Z}$ should then be an n by n matrix

which is positive semi-definite with rank $n-1$. In other words, $\mathbf{Z}$ should have a single zero eigenvalue, with the rest positive. We can use this observation to determine the radius of curvature. Given a radius r and a distance matrix $\mathbf{D}$, we can construct $\mathbf{Z}(r)$ and find the smallest eigenvalue λ_0. By minimising the magnitude of this eigenvalue, we can find the optimal radius.

$$r^* = \arg\min_r |\lambda_0 [\mathbf{Z}(r)]| \tag{32}$$

The smallest eigenvalue can be determined efficiently using the power method without the expense of the full eigendecompositon. Give the optimal radius, the embedding positions are determined via the full eigendecomposition:

$$\mathbf{Z}(r^*) = \mathbf{U}_Z \mathbf{\Lambda}_Z \mathbf{U}_Z \tag{33}$$

$$\mathbf{X} = \mathbf{U}_Z \mathbf{\Lambda}_Z^{\frac{1}{2}} \tag{34}$$

If the points truly lie on a hypersphere, then this is sufficient. However, in general this is not the case, the optimal smallest eigenvalue λ_0 will be less than zero, and there will be residual negative eigenvalues. As in Euclidean embedding, we can drop the negative eigenvalues to get an approximate solution. After dropping eigenvalues, the point positions may no longer lie on a sphere, so we apply an additional length correction to the vectors:

$$\mathbf{Z}^+ = \mathbf{U}_Z \mathbf{\Lambda}_Z^+ \mathbf{U}_Z \tag{35}$$

$$Z'_{ij} = Z^+_{ij} / \sqrt{Z^+_{ii} Z^+_{jj}} \tag{36}$$

3.2 Embedding in Hyperbolic space

In hyperbolic space, we have

$$< \mathbf{x}_i, \mathbf{x}_j >= r^2 \cosh \theta_{ij} = r^2 \cosh \frac{d_{ij}}{r} \tag{37}$$

with the inner product defined by Eqn 5. Constructing $\mathbf{Z}$ as before, we get

$$\mathbf{X}\mathbf{M}\mathbf{X}^T = \mathbf{Z} \tag{38}$$

Again we have an embedding space of dimension $n-1$, but $\mathbf{Z}$ is no longer positive semi-definite. In fact, $\mathbf{Z}$ should have precisely one negative eigenvalue (since the hyperbolic space has just one negative dimension) and again a single zero eigenvalue. We must now minimise the magnitude of the second smallest eigenvalue:

$$r^* = \arg\min_r |\lambda_1 [\mathbf{Z}(r)]| \tag{39}$$

The embedded positions become

$$\mathbf{X} = \mathbf{U}_Z |\mathbf{\Lambda}_Z|^{\frac{1}{2}} \tag{40}$$

As with the elliptic embedding, in general the points do not lie on the embedding space and there will be residual negative eigenvalues. In this case, we drop all but the most negative eigenvalue, as one negative dimension exists in the embedding. We then length-correct the vectors.

4 Results

We have applied this process to the Chicken Pieces data[3, 1] Firstly, the negative eigenfractions are shown below. The elliptic and hyperbolic NEF measures the remaining negative eigenvalues after embedding.

L(cost=45)	NEF	NEF elliptic	NEF hyperbolic
5	0.21604	0.079	0.0075
10	0.25714	0.104	0.0105
15	0.28571	0.144	0.0134
20	0.30657	0.167	0.0149
25	0.31992	0.186	0.0188
30	0.33068	0.213	0.0216
35	0.33942	0.229	0.0223
40	0.34459	0.237	0.0245

Secondly, we used the 1NN classifier to measure the classification accuracy of the data. PPES is the positive Euclidean embedding.

L	Original	Elliptic	Hyperbolic	PPES
5	0.350 ± 0.022	0.438 ± 0.022	0.462 ± 0.025	0.464 ± 0.023
10	0.170 ± 0.016	0.305 ± 0.022	0.352 ± 0.022	0.344 ± 0.021
15	0.079 ± 0.011	0.172 ± 0.017	0.252 ± 0.020	0.243 ± 0.020
20	0.069 ± 0.012	0.120 ± 0.014	0.182 ± 0.019	0.176 ± 0.020
25	0.048 ± 0.010	0.075 ± 0.012	0.142 ± 0.013	0.145 ± 0.015
30	0.048 ± 0.009	0.088 ± 0.011	0.138 ± 0.016	0.133 ± 0.018
35	0.065 ± 0.011	0.092 ± 0.013	0.163 ± 0.016	0.153 ± 0.015
40	0.087 ± 0.014	0.108 ± 0.013	0.160 ± 0.018	0.156 ± 0.015

The elliptic embedding results are encouraging, giving an improvement over the positive Euclidean embedding. Although the negative eigenfraction for the hyperbolic embedding is very low, the classification results are not good. This seems to be principally because of the normalisation step which disrupts the distances. A better approach to this will probably improve the results.

References

[1] G. Andreu, A. Crespo, and J.M. Valiente. Selecting the toroidal self-organizing feature maps (tsofm) best organized to object recognition. In *ICNN'97*, pages 1341–1346, 1997.

[2] Elzbieta Pekalska and Robert P. W. Duin. *The dissimilarity representation for pattern recognition.* World Scientific, 2005.

[3] Elzbieta Pekalska, Artsiom Harol, Robert P. W. Duin, Barbara Spillmann, and Horst Bunke. Non-euclidean or non-metric measures can be informative. In *SSPR/SPR*, pages 871–880, 2006.

Appendix D

Evolving Embeddings using Ricci Flow

Ricci Flow Embedding

Eliza Xu

October 7, 2009

1 Introduction

Usually, objects to be classified are represented by features. Some information such as structure and relation can not be captured by featured-based representation. This leads to the development of dissimilarity representation. In dissimilarity representation, pairwise dissimilarity (or proximity) measure describes the properties of objects in terms of their differences. However the lack of Euclidean property in distance measures prevents the use of well-developed machine learning techniques.

Embedding provides a way to apply feature-based classifiers on dissimilarity data. Embedding produces a vectorial representation by projecting dissimilarity data into a fixed-dimensional vector space. There is considerable body of work aimed at understanding how to construct such an embedding in pattern recognition for several decades. Multidimensional scaling(MDS) [4] is one of the earliest embedding techniques to find vectorial representation in an Euclidean space from dissimilarity data.The classical MDS produces the coordinates to approximate the distance from dissimilarity matrix.

MDS methods are efficient techniques that can detect the linear structure of data. However, MDS methods fail to detect the nonlinear structures [18]. More recent approaches try to reduce dimensionality that minimizes the distortion by inferring a low dimensional manifold in which data resides.Three of the best known of these are ISOMAP [18], locally linear embedding [17] and the Laplacian eigenmap [2]. The above manifold learning methods focus on the edge relation of the graph. Robles-Kelly and Hancock [16] proposes an embedding method using the relation between the matrix representations of the graph and the differential geometry of the manifold.EIGhawalby and Hancock [9, 8] extend the above work by using sectional curvature, Gaussian curvature and the ratio of the geodesic triangle to the corresponding Euclidean triangle as features for computing the similarity of graphs.

The emphasis of the above embedding methods is on finding a low-dimensional representation. In order to apply non-Euclidean dissimilarity data on traditional learning techniques, much work is attempting to transform the non-Euclidean dissimilarity into Euclidean space. Only considering the positive definite subspace [6] is an obvious way. Kondor etc and Pekalska[11, 14] impose geometricity by adding a suitable constant $c > |\lambda_{min}|$ where λ_{min} is the smallest negative eigenvalue to all off-diagonal elements of the dissimilarity matrix. It is equivalent to add the constant to the non-zero eigenvalues and keeps the same eigenvectors.

Pekalska etc [6, 14] demonstrate that the discriminating power of the corrected measure is not that good as the original non-Euclidean distance measures

by testing the above correcting approaches on five dissimilarity dataset with four classifiers. It puts the necessarity of imposing geometricity into doubt and emphasizes the discriminating power of original dissimilarity measures are more important than the Euclidean property. Hence, how to correct the dissimilarity such that the new dissimilarity in Euclidean space is not less discriminative than the non-Euclidean dissimilarity is a problem. Our work develops the idea of Ricci flow in the constant curvature Riemannian by evolving the distance measure through updating Gaussian curvatures on the edges of the graph, so that geometricity can be imposed on non-Euclidean distance measures.

2 Ricci Flow on Embedding

In this section, we make use of the relationship between Ricci flow and the constant curvature Riemannian Space, for the purposes of embedding dissimilarity representations in a Euclidean space. To start this study, we review some of the basics of Ricci flow and constant curvature Riemannian geometry. Using the properties of constant curvature Riemannian geometry, we show how to correct dissimilarity representations to make them more Euclidean(metric) by evolving Gaussian curvatures associated with the Euclidean distances and the geodesic distances on the manifold between graphs. We use the give pairwise distances as the initial geodesic distances and Young-Householder coordinates to compute the Euclidean distances to compute Gaussian curvatures. Fixing the Euclidean distances, we gradually update the geodesic based on the evolving Gaussian curvatures. We use the resulting geodesic distances to embed graphs onto a Riemannian manifold. We illustrate the method on imposing geometry using the Chicken pieces data set.

3 Ricci Flow in Constant Curvature Riemannian Space

In this section, we provide the theoretical basis for our graph embedding method. Our aim is to correct the dissimilarity representations to make them more Euclidean. Curvature describes a geometric property of the ssurface. EIGhawalby and Hancock [9, 8, 7] use the sectional curvature and Gaussian curvature embedded in a Riemannian space as features for measuring the similarity pairs of graphs.

Gaussian curvature is an important parameter of a constant curvature Riemannian space. It is the product of the maximum and minimum curvatures of all geodesics passing through the point [13]. The constant curvature could be positive in elliptic space, negative in hyperbolic space and zero in Euclidean space. The curvature of a curve at a given point is $\pm\frac{1}{r}$ where r is the radius of the circle that best fits the curve at that point. The Gaussian curvature at a constant curvature space is:

$$K = \frac{1}{r^2}$$

Introduced by Richard Hamilton in 1981, Ricci flow [5] is a process evolving a metric according to its Ricci curvature. In two dimensions, that is, a process of

increasing or decreasing the distance between points along directions of constant curvature. The 2-dimensional geometric evolution equation is:

$$\frac{dg_{ij}}{dt} = -2Kg_{ij} \tag{1}$$

where K is the Gaussian curvature of the surface, g_{ij} is metric tensor determined by the curvature and the coordinate system in the Riemannian space. A two-dimensional sphere surface [1] in spherical-polar co-ordinates can be represented as:

$$\Delta s^2 = r^2(\sin\theta)^2(\Delta\phi)^2 + r^2(\Delta\theta)^2$$

where Δs is the length of small curve segment of a curve in the sphere with radius r, θ is the angle measured from the z axis, ϕ is the angle from the x axis in the xy plane. So the metric is:

$$g = \begin{bmatrix} r^2\sin^2\theta & 0 \\ 0 & r^2 \end{bmatrix} \tag{2}$$

$$g_{11} == r^2\sin^2\theta = \frac{\sin^2\theta}{K}$$

Based on the Gaussian curvature at constant curvatue space and the Ricci flow equation, we have

$$\frac{dr}{dt} = -\frac{1}{r}$$

$$\frac{dK}{dt} = 2K^2 \tag{3}$$

The equation (1) applies any higher dimensional elliptic or hyperbolic space, as the metric tensor is a function of radius and various angles measured from the coordinates basis, the angles could be taken away from both sides of equation (3.9). Thus the updating rule for curvature K is:

$$K_{new} = K_{old} + \frac{dK}{dt}\Delta t = K_{old} + 2K_{old}^2\Delta t \tag{4}$$

Lindman and Caelli [13] presents the Euclidean distances in m+1 dimensional space in terms of the distance and curvature in m-dimensional manifold as follows:

$$d_E(u,v) = \begin{cases} \frac{2}{K^{\frac{1}{2}}}\sinh(\frac{K^{\frac{1}{2}}}{2}d_G(u,v)) & \text{elliptic hypersphere} \\ \frac{2}{|K|^{\frac{1}{2}}}\sin(\frac{|K|^{\frac{1}{2}}}{2}d_G(u,v)) & \text{hyperbolic space} \\ d_G(u,v) & \text{Euclidean space} \end{cases} \tag{5}$$

Based on the second equation in hyperbolic space with fixing Euclidean distance, the new geodesic distance under new Gaussian curvature can be represented in terms of the old geodesic distance and both of the old and new Gaussian curvatures as following:

$$d_G^{new}(u,v) = 2|K_{new}|^{\frac{1}{2}} \operatorname{arcsinh}\left(\frac{|K_{new}|^{\frac{1}{2}}}{|K_{old}|^{\frac{1}{2}}} \sinh(\frac{|K_{old}|^{\frac{1}{2}}}{2} d_{G_{old}}(u,v))\right) \quad (6)$$

A way to estimate Gaussian curvature on the edges of graphs is presented in [3, 9, 8, 7] using the Euclidean distance and geodesic distance. It is assumed the geodesic distance is approximately an arc of a circle. The geodesic distance between two nodes u and v is:

$$d_G(u,v) = 2r\theta \quad (7)$$

And the Euclidean distance between two nodes u and v is:

$$d_E(u,v) = 2r\sin\theta = 2r\sin\frac{d_G(u,v)}{2r} \quad (8)$$

The Gaussian curvature is approximately as follows by solving the above equation for the radius of curvature and the MacLaurin series:

$$K(u,v) = \frac{1}{r^2(u,v)} = \frac{24(d_G(u,v) - d_E(u,v))}{d_G^3(u,v)} \quad (9)$$

The above equation is not suitable for large dissimilarity values. The Newton's method is used to get a more accurate approximation for Gaussian curvature:

$$f(x) = d_E - \frac{2}{x}\sinh\frac{x}{2}d_G$$

where

$$x = \frac{1}{r}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

In [3, 9], methods of graph embedding are introduced. Images are represented by Delaunay graphs. They embeds the nodes of a graph onto the manifold and extracts the set of Gaussion curvatures which are regarded as features for each graph. Then a pairwise distance matrix for pairs of graphs is obtained by using the features and is used for clustering the images. In our work, the pairwise distance matrix for graphs is given or obtained by the given edit distance measures in dissimilarity representation. We assume the distance matrix is an edge-weight matrix and edge weight represents the distance between two graphs (that is, each graph is regarded as a node in the set of graph nodes). Our aim is to correct the non-metric distances in a Euclidean space, so that traditional machine learning methods can be used to cluster the image data.

Let's assumed a curve in a sphere surface with a radius, the curve bends less away from the Euclidean chord when the radius is increased. In Euclidean space, the geodesic distance is equal to the Euclidean distance. Based on Ricci flow in constant curvature manifold, the curved space can be transformed into a smooth flattened space. It provides a way to correct the non-metric distances of nodes residing on a elliptic or hyperbolic manifold to be Euclidean distances of nodes residing on a Euclidean manifold with small distortion.

To do this, we develop a method to update edge based Gaussian curvature while keeping the original Euclidean distances. The embedding coordinates are given by the positive eigenvalues and their corresponding eigenvectors of the centered Gram matrix. With the set of embedding coordinates at hand, the pairwise Euclidean distances between nodes are computed. We regard the distances from the given distance matrix as pairwise geodesic distances. Because the positive space embedding for Euclidean distances is used and the centered Gram matrix has negative eigenvalues, the Euclidean distance is smaller than the geodesic distance. Hence, the embedding space is hyperbolic. With the geodesic distance and Euclidean distance at hand, we can start to compute the Gaussian curvature for each edge based on [3, 9]. We can update the Gaussian curvatures with small time steps and compute the new geodesic distances with the new Gaussian curvature and fixing the Euclidean distances from [13]. We start the next round of above process with the new geodesic distances. During the updating process, the geodesic distances evolve as the Gaussian curvature and come closer to the Euclidean distances on positive space embedding. With a number of iterations, the geodesic distances get equal to the Euclidean distances and the space nodes residing are smoothed to be Euclidean space from original non-metric space.

4 Curvature Updating Algorithm

In this section, we give an algorithm for transforming a non-metric dissimilarity representation into a Euclidean dissimilarity representations. The algorithm is based on the 2-dimensioanl Ricci flow and constant curvature Riemmanian manifold embedding, combining them to deal with non-metric pairwise data. We give an algorithm to correct the non-Euclidean dissimilarity representation so that the dissimilarity representation can use machine learning algorithm and show the algorithm bridges the gap between the statistical machine learning methods and the structural pattern recognition.

Given a set $X = \{x_1, \cdots, x_N\}$ of N objects and a dissimilarity measure d, a dissimilarity representation is an $N \times N$ matrix $D(X, X)$with the elements $d(u, v)$ representing the pairwise distance between x_u x_v. Each object is represented by an N element vector of dissimilarities. Only if the corresponding Gram matrix is positive semidefinite, the dissimilarity matrix $D(X, X)$ has Euclidean behaviour. That is, the eigenvalues of the Gram matrix are non-negative. Therefore, for a non-Euclidean distance matrix, the Gram matrix has negative eigenvalues. The following implementation steps shows how to transform the distance matrix from non-Euclidean to Euclidean.

Begin with a pairwise distance matrix $N \times N$ $D(X, X)$,

1. Project objects in positive space and get the coordinates y and the corresponding Euclidean distance
   (a) Compute the centralized Gram matrix $G = -\frac{1}{2}JD_0^2J$, where $J = I - \frac{1}{N}11^T$ so that the embedded coordinates have zero mean.
   (b) Take $P(P < N)$ positive eigenvalues of Gram matrix and project objects on the eigenvectors with positive eigenvalues using Young-Householder decomposition for preserving distance. Y is the $|P| \times |N|$

matrix with the vectors of co-ordinates as columns.

$$G = \Phi\Lambda\Phi^T$$

$$Y = \sqrt{\lambda}\Phi = (y_1|y_2|y_3|\dots|y_N)$$

The Euclidean distance between objects t_u and t_v is:

$$d_E(u,v) = \sqrt[2]{(y_u - y_v)^T(y_u - y_v)} = \sqrt[2]{\sum_{i=1}^{N} \lambda_i(\phi_i(u) - \phi_i(v))^2}$$

It shows the embedded Euclidean distance gets smaller if positive eigenvalue part is taken away, and gets bigger if negative eigenvalues part is taken away. That is why the geodesic distance is smaller than the Euclidean distance, the curvature is negative, the space is assumed to be hyperbolic space.

2. From geodesic distance d_G and Euclidean distance d_E, we can get constant curvature space with curvature K_{old} individual edge is residing on from equation (9).

3. Update Gaussian curvature with small step from equation (4).

4. Obtain new geodesic distance d_G^{new} from previous geodesic distance and curvatures with fixed Euclidean distance based on equation (6).

5. Get the new distance matrix D composed of new geodesic distances between objects, go back to step(1) until D is Euclidean, that is, there is no negative eigenvalues from its centralised Gram matrix.

The above approach transform distance matrix such that co-ordinates in Euclidean space can be found that preserves all pairwise distances. So that new objects can be added to this Euclidean space with the line projects and traditional learning methods can be used to classify objects with dissimilarity representation. However our current approach updates curvatures independently and ignore the relation among connected edges. There is more work we can do to regularize the curvatures.

5 Regularizing curvature

In addition to preserving distances, the Ricci flow embedding is to transform non-Euclidean dissimilarity into Euclidean space by evolving curvatures. The Ricci flow embedding updates Gaussian curvature independently for individual edge. Thus, it ignores the geometric structure and edge relations in graph from pairwise relationships. It does not guarantee that close points are kept close, far points remain far. Graph regularization provides a way to smooth and data samples over graph, which is normally solved by minimizing appropriate energy functions [12, 19]. So we attempt to preserve local structure by regularizing curvature.

We adds additional step to smooth Gaussian curvatures of nearest neighbors before updating all curvatures, with all the other steps remaining the same as

in above section. Laplace operator and curvature operator [19] are used to regularize local curvatures. The Laplace operator [19] of a function f on each graph vertex is:

$$\Delta f(u) = f(u) - \sum_{v \backsim u} \frac{w([v,u])}{\sqrt{g(v)g(u)}} f(v)$$

In our implementation, the function $f(u)$ corresponds to the Gaussian curvature of edge u. In the dissimilarity representation, each node is connected with all the other graph using edges with weight of pairwise distances. In [10], the connection relation among nodes are based on the Delauney triangulation which is for low dimensional data. However our data is high dimensional which is difficult to do triangulation, so we try on nearest neighbor graph.

we built 6 nearest neighbor graph from dissimilarity matrix, and construct dual graph of the nearest neighbor graph so that each edge is corresponding a vertex in dual graph. The Gaussian curvature on each edge is the function on each vertex in the corresponding dual graph. The Laplace operator makes Gaussian curvature influenced by its neighbors, so the initial positive curvature could jump into negative values or the other way around. The Gaussian curvatures are not monotonically smoothly approaching zero which is against the smoothing process. Heat kernel guarantees positive values. So the heat kernel was implemented to smooth Gaussian curvatures. The heat kernel regularization on Gaussian curvature is:

$$\Delta f(u) = exp[-\hat{L}t]f(u)$$

where $\hat{L}$ is normalized Laplacian of the dual graph of nearest neighbor graph.

The Euclidean property of a transformed non-Metric distance measure can be evaluated the Gram matrix, as the Gram matrix of a Euclidean distance measure have non-negative eigenvalues. The distortion degree of distance measures is evaluated by comparing the discriminative power of distance measures through applying different classifiers. The distance measure with minimum distortion have the similar performance to the original distance measure as only smaller information is lost from original distance.

6 Experiments

Chicken pieces data [15, 6] is used in our experiment. The chicken pieces data contains 446 binary image in five classes: breast(96 examples), back(76 examples), thigh and back(61 examples), wing(117 examples) and drumstick(96 examples). It generates different distance matrix with straight line segment of a fixed length L and the angles between the neighboring segments and editing cost C. Our experimental results are from the data with $cost = 45$ and $L = \{5, 10, 15, 20, 25, 30\}$. The originally asymmetric dissimilarities are made symmetric by averaging[15, 6].

There are two indicators to measure the degree of Euclidean behavior on a distance matrix [15]. One is the mass contribution of negative eigenvalues $J_{eigS} = \sum_{\lambda_i<0} |\lambda_i| / \sum_{j=1}^{N} |\lambda_i|$; the other is the ratio of the absolute value of the smallest negative eigenvalue to the largest positive one $J_{eigM} = |\lambda_{min}|/\lambda_{max}$. Duin [15, 6] shows the dissimilarity measure becomes increasingly non-Euclidean as both the J_{eigS} and J_{eigM} grow with increasing L. In our experiment, we show

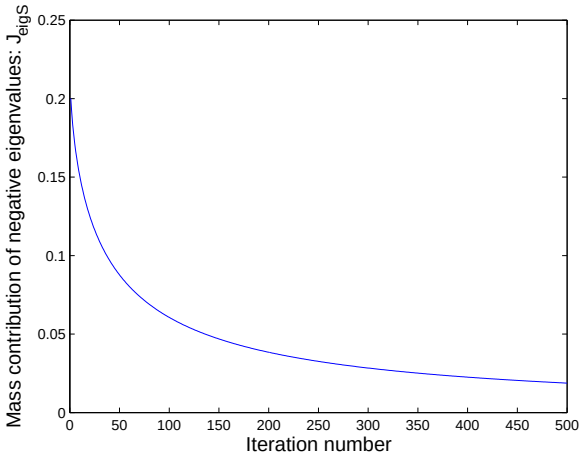


Figure 1: J_{eigS}

the Gaussian curvature, the mass contribution of negative eigenvalues and the number of negative eigenvalues during iteration. After curvature updating, the final dissimilarity measure with values of L from 5 to 30 are obtained.

The mass contribution and the number of negative eigenvalues for distance measures with $L = 5.0$, $C = 120$ during evolving process in first 500 iterations shows in Figure 1. It demonstrates the both the mass contribution and the number of negative eigenvalues decrease as curvatures are updating, indicating the dissimilarity measure becomes decreasingly non-Euclidean and inclined to Euclidean. Figure 2 shows the Gaussian curvature for edge with initial biggest curvature, edge with middle curvature and edge with minimum curvature. All of the curvatures grows towards zero, indicating the evolving process transforms the hyperbolic space(negative Gaussian curvature) to Euclidean space(zero Gaussian curvature).

In our classification experiments, we perform ten runs of 10-fold cross-validation. In each run, all objects are first randomly split into 10 subsamples with original class distribution, the training set T from 9 sub samples and test set S from the left one sub samples. Then 1-NN classifier are trained on $D(T, T)$ and tested on $D(S, T)$. Finally the average errors are determined. This is repeated 10 times and the results are averaged out. Projecting on the positive subspace is one of the most obvious corrections for a non-Euclidean dissimilarity matrix. It takes away information from negative subspace. In order to see the discriminative power of the final dissimilarity measure by updating curvatures, we apply the classification experiments on original distance matrix, distance matrix embed into the positive space and get and our final dissimilarity matrix. The same objects are used for training and testing in identical runs over different values of L, so that we can get reliable differences for the given dissimilarity data in classification performance.

Our classification performance results can be observed from Figure 3. The 1-NN performs identically for the final distance measure and the final distance measures embedded in positive subspace, indicating the final distance measures are Euclidean transformed from the original non-Euclidean property. The re-

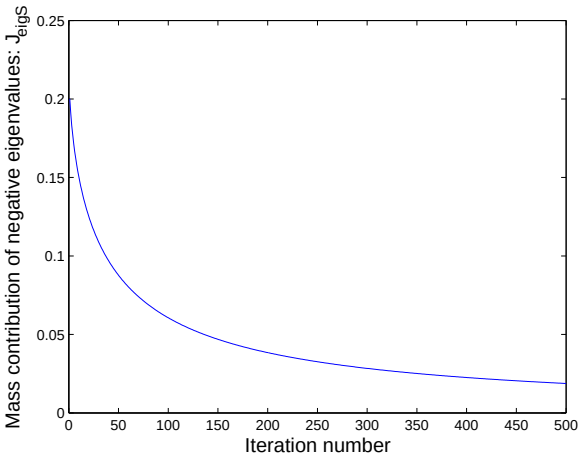
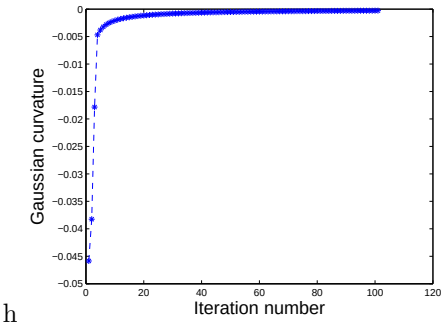
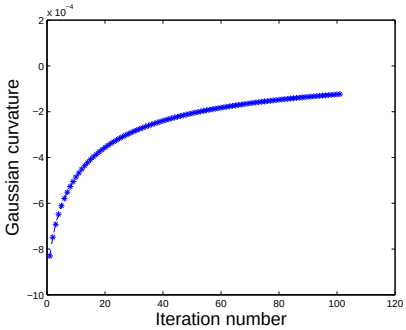


Figure 2: Number of negative eigenvalues

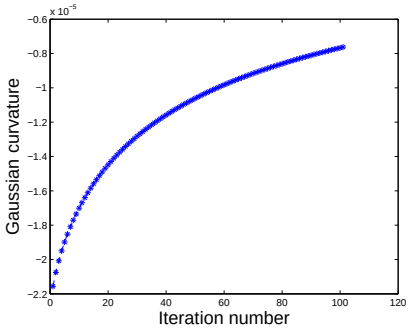


h

(a) maximum curvature



(b) middle curvature



(c) minimum curvature

Figure 3: Curvatures

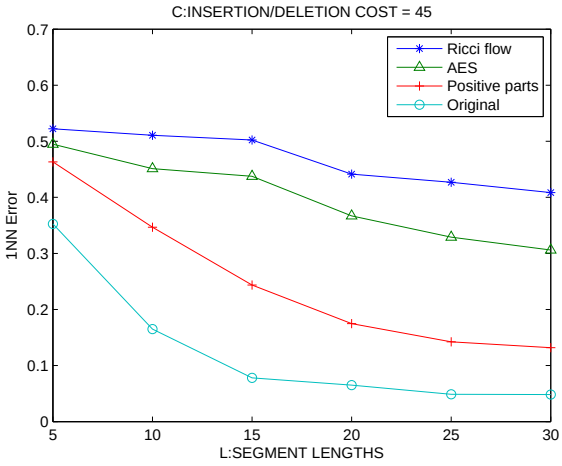
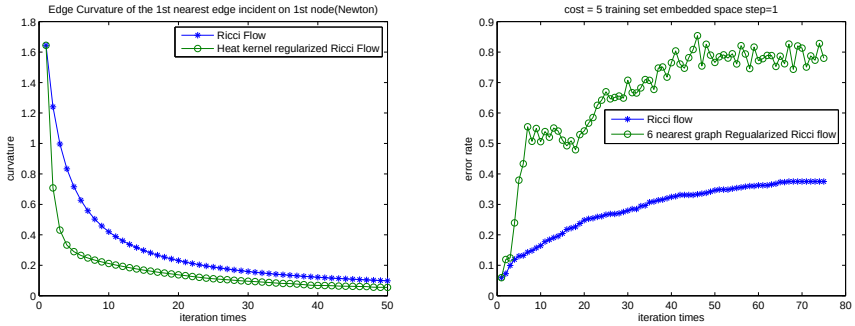


Figure 4: Error rate from 1NN



(a) Curvature

(b) Error rate from 1NN

Figure 5: L=5.0 C=120 N=446

sults show our final dissimilarity measures is not better than the corrected dissimilarity of original distances on positive subspace, and even worse than the associate Euclidean space(AES). The final distance measures does not preserve the ranking of dissimilarity values, as the ricci flow embedding process moves the ranking order. The heat kernel regularization is attempted to keep the ranking of local dissimilarity values by using nearest graph. The heat kernel results is in graph 4. The graph 4(a) shows the curvature change during evolving process and 4(b) shows the the performance on 1NN. The discriminating power of evolved dissimilarity with Ricci flow embedding is worse than that without regularization. It is because the regularization moves the nearest neighbors more violently which cause the neighbor nodes are not stable.

Our approach is a potentially good way to transform the non-Euclidean dissimilarity measure to be Euclidean, but there is room to improve the discriminating power of the evolved Euclidean distance

7 Conclusion

In this section, we presents an approach to impose geometricity on non-Euclidean dissimilarity measures. Our method evolve the distance measure by updating Gaussian curvatures on the edges of the graph, based on the 2-dimensional Ricci flow and constant curvature Riemmannian space. Applying our method to the Chicken pieces dataset, the distance measures can be transformed into Euclidean space but have less discriminative power than the obvious transformation on positive subspace and lose local structure from the original non-Euclidean distance measures.

The loss of information maybe caused by the the evolving process on individual edges independently and ignore the structure in local manifold. The heat kernel regularization fails to preserve the ranking of distance measures. Hence, one direction to improve our current work is to keep the local structure during Ricci flow smoothing process. Maybe we can try local structure preserving embedding like Local linear embedding or stochastic neighbor embedding. Our current work is performed only on 1NN classifier. After we solve this problem, another way to develop our current work is to try traditional feature-based classifiers like the distance based classifier Fisher's Linear Discriminant.

References

[1] *Understanding space and time*. Milton Keynes, The Open University Press, 1979.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 1:585–592, 2002.

[3] X. Biao. Heat kernel analysis on graphs. Technical report, Ph. D. thesis, Computer Science Department, University of York, UK, 2007.

[4] I. Borg and P.J.F. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer, 2005.

[5] B. Chow and F. Luo. Combinatorial Ricci flows on surfaces. *J. Differential Geom*, 63(1):97–129, 2003.

[6] R.P.W. Duin, P. Elzbieta, W.J. Lee, and H. Bunke. On euclidean corrections for non-euclidean dissimilarities. In *Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 551–561. Springer, 2008.

[7] Hewayda ElGhawalby and Edwin R. Hancock. Graph characteristic from the gauss-bonnet theorem. In *SSPR/SPR*, pages 207–216, 2008.

[8] Hewayda ElGhawalby and Edwin R. Hancock. Measuring graph similarity using spectral geometry. In *ICIAR*, pages 517–526, 2008.

[9] Hewayda ElGhawalby and Edwin R. Hancock. Characterizing graphs using spherical triangles. In *IbPRIA*, pages 465–472, 2009.

[10] Hewayda ElGhawalby and Edwin R. Hancock. Graph regularisation using gaussian curvature. In *GbRPR*, pages 233–242, 2009.

[11] R.I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 315–322, 2002.

[12] O. Lezoray, A. Elmoataz, and S. Bougleux. Graph regularization for color image processing. *Computer Vision and Image Understanding*, 107(1-2):38–55, 2007.

[13] Harold Lindman and Terry Caelli. Constant curvature riemannian scaling. *Journal of Mathematical Psychology*, 17:89–109, 1978.

[14] E. Pekalska, R.P.W. Duin, S. Gunter, and H. Bunke. On not making dissimilarities euclidean. *Lecture notes in computer science*, pages 1145–1154, 2004.

[15] Elżbieta Pkalska, Artsiom Harol, Robert Duin, Barbara Spillmann, and Horst Bunke. Non-euclidean or non-metric measures can be informative. pages 871–880. 2006.

[16] A. Robles-Kelly and E.R. Hancock. A Riemannian approach to graph embedding. *Pattern Recognition*, 40(3):1042–1056, 2007.

[17] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding, 2000.

[18] J.B. Tenenbaum, V. Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction, 2000.

[19] D. Zhou and B. Scholkopf. Regularization on discrete spaces. *Lecture notes in computer science*, 3663:361, 2005.

Appendix E

Joint Eigenspaces

An Inexact Graph Comparison Approach in Joint Eigenspace

Wan-Jui Lee and Robert P.W. Duin

Faculty of Electrical Engineering, Mathematics and Computer Sciences,
Delft University of Technology, The Netherlands
W.J.Lee@tudelft.nl, r.duin@ieee.org

Abstract. In graph comparison, the use of (dis)similarity measurements between graphs is an important topic. In this work, we propose an eigendecomposition based approach for measuring dissimilarities between graphs in the *joint* eigenspace (JoEig). We will compare our JoEig approach with two other eigendecomposition based methods that compare graphs in *different* eigenspaces. To calculate the dissimilarity between graphs of different sizes and perform inexact graph comparison, we further develop three different ways to resize the eigenspectra and study their performance in different situations.

1 Introduction

Graphs are a general and powerful data structure for object representation in structural pattern recognition[11,2]. The nodes in a graph can represent different parts of a object and the relations between the parts are represented by edges. Also, labels and attributes for the nodes and edges can further be used to incorporate more information in a graph representation. But for simplicity, we only consider nodes and edges without labels and attributes in this work.

In pattern recognition and other related areas, (dis)similarity among objects is an important issue. If the graphs are used for object representation, the problem turns into determining (dis)similarity between objects, which is usually referred to as graph comparison. In the study of graph comparison, there are two main directions. One is by comparing their spatial structures and the other is by comparing their spectral structures. Early approaches to graph comparison were mainly based on the first. The best known example is maximum common subgraph [2] where the similarity of two graphs is decided by how much they spatially share in common. An alternative is the graph edit distance [8]. In graph edit distance computation, one introduces a set of graph edit operations, such as deletion, insertion or substitution, to apply on nodes as well as edges. The edit distance of two graphs is defined as the shortest sequence of edit operations that transforms one graph into the other. These methods for finding spatial (dis)similarity between graphs are guaranteed to find the optimal solutions but require exponential time and space due to the NP-completeness of the problem.

N. da Vitora Lobo et al. (Eds.): SSPR&SPR 2008, LNCS 5342, pp. 35–44, 2008.

In 1988, Umeyama [6] introduced the spectra of graphs for the exact graph comparison problem. This approach is based on spectral graph theory that is concerned with characterising the structural properties of graphs using the eigenvectors of the adjacency matrix or the closely related Laplacian matrix (the degree matrix minus the adjacency matrix). To compare two different graphs A and B, the idea is to transform graph A into the space of graph B. So there exists an approximated graph $A^{'}$ of graph A in the space of graph B. The difference between A and B is actually calculated by the difference between $A^{'}$ and B. There is another approach for inexact graph comparison based on graph spectra proposed by Caelli [1]. Instead of transforming graph A into the space of graph B, it projects graph A into its own eigenspace and the same is done for graph B. Because graph A and B could have different numbers of nodes, a renormalization step is introduced for eigenspace projections. Therefore, both graphs are projected into a unit hypersphere defining the eigenspaces. In this renormalized space, two graphs are close if their rescaled eigenvectors have similar angles and their eigenvalues are similar. The authors declare that the graph comparison was done in a common eigenspace of these two graphs, but actually it's done in their own eigenspaces but with the common unit hypersphere.

In Umeyama's method, the graphs are compared in the original space of one of the graphs, but this is not unique since there are two graphs. As a matter of fact, we further discover that the projection of graph A in the space of graph B is actually determined by how the eigenvalues of graph A live in the eigenspace of graph B. So if we want to have a symmetric dissimilarity measurement using Umeyama's method, we have to take an average or choose one of the results from two different eigenspaces, i.e. the eigenspace of A or B. In Caelli's approach, the dissimilarities of the graphs are compared by their projections in their own eigenspaces. For a fair comparison of two graphs, we will bring them into the same eigenspace and then compare them in this unique eigenspace. Inspired by Umeyama's approach, we actually can find an unique joint eigenspace (JoEig) for both graphs, which is a space with the eigenvectors of both graphs. However, the sizes of these two sets of eigenvectors could be different and therefore it is essential to adjust these two sets of eigenvectors for sharing the same number of dimensions in order to construct the joint eigenspace. We study three possibilities for setting the number of eigenvectors in this work. The first two choices are setting the number of the eigenvectors according to the size of the larger or the smaller graph, respectively. We study as well for a number smaller than both of the graphs to emphasize the importance of larger eigenvalues. We will compare these three settings and discuss how the dimensionality of eigenvectors effects the performance of classifiers in the experiments.

The rest of the paper is organized as follows. In Section 2, we introduce Umeyama's and Caelli's approaches for comparing graphs, respectively. Our graph dissimilarity measurement approach, JoEig, is described in Section 3. Simulation results are presented in Section 4. Finally, a conclusion is given in Section 5.

2 Previous Eigendecomposition Approaches for Graph Comparison

Before we introduce Umeyama's and Caelli's approaches for graph comparison, some definitions and introduction on graphs are given as in the following.

A graph is a set of nodes connected by edges in its most general form. Consider the undirected graph $G = (V, E, W)$ with the node set $V = \{v_1, v_2, \ldots, v_n\}$, the edge set $E = \{e_1, e_2, \ldots, e_m\} \subset V \times V$, and the weight function $W : E \rightarrow (0, 1]$. If the graph edges are weighted, the adjacency matrix A for the graph G is the $n \times n$ matrix with elements

$$A_{ij} = \begin{cases} W(v_i, v_j), & \text{if } (v_i, v_j) \in E; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Clearly since the graph is undirected, the matrix A is symmetric. The Laplacian [10] of the graph is defined by $L = D - A$, where D is the diagonal node degree matrix whose elements $D_{ii} = \sum_{k=1}^{n} A_{ik}$. The Laplacian matrix of G is positive semidefinite and singular, and it is more often adopted for spectral analysis than the adjacency matrix because of its properties.

2.1 An Eigendecomposition Approach for the Weighted Graph Matching Problem

In 1988, Umeyama [6] developed an eigendecomposition approach for the weighted graph matching problem. Let $G = (V_1, E_1, W_1)$, $H = (V_2, E_2, W_2)$ be weighted graphs with n nodes (these two graphs are with the same size). The weighted graph matching problem is the problem of finding a one-to-one correspondence Φ between $V_1 = \{v_1, v_2, \ldots, v_n\}$ and $V_2 = \{v_1^{'}, v_2^{'}, \ldots, v_n^{'}\}$ which minimizes the difference between G and H. The following criterion is used as a measure of difference:

$$J(P) = \|PL_GP^T - L_H\|^2 \tag{2}$$

where the permutation matrix P presents the node correspondence Φ and $\|\cdot\|$ is the Euclidean norm ($\|L\| = (\sum_{i=1}^{n}\sum_{j=1}^{n} l_{ij}^2)^{\frac{1}{2}}$). Thus the weighted graph matching problem is reduced to the problem of finding the permutation matrix P which minimizes $J(P)$. In general, it is not easy to find the exact solution of the weighted graph matching problem since it is purely combinatorial. Thus, an efficient method which gives a "nearly" optimum solution, i.e., a permutation matrix $P^{'}$ whose criterion value $J(P^{'})$ is very close to the optimum value. Since the optimum criterion value cannot be known in advance, the aim is to determine a permutation matrix of a small criterion value.

Let G and H be weighted undirected graphs and L_G and L_H be their Laplacian matrices, respectively. The eigendecomposition of L_G and L_H are performed as

$$L_G = V_G D_G V_G^T, \; L_H = V_H D_H V_H^T, \tag{3}$$

where V_G and V_H are orthonormal matrices and D_G and D_H are diagonal matrices of the eigenvalues (in ascending order) of G and H, respectively. Now if G and H are isomorphic,

$$PV_G D_G V_G^T P^T = V_H D_H V_H^T. \tag{4}$$

It means there exists a permutation matrix P that has the one-to-one correspondence of nodes between G and H. Also, the eigenvalues of these two graphs should be the same. That is, $D_G = D_H$, and therefore $PV_G = V_H$ which leads to $P = V_H V_G^T$. As a result, the difference between two graphs can be computed as

$$\|V_H V_G^T L_G V_G V_H^T - L_H\|^2, \text{ or } \|V_G V_H^T L_H V_H V_G^T - L_G\|^2. \tag{5}$$

Even though Umeyama's approach is only designed for graphs with the same size, we extend it in this paper to compare it with the other approaches. First of all, the Lapacian matrix is used instead of adjacency matrix. Secondly, the eigenvalue and eigenvector matrices are resized because the graphs might be with different numbers of nodes. We study three different ways for computing the resized eigenvalue diagonal matrix and the eigenvector matrix, and they will be discussed further in the next section. Finally, the difference between two graphs is the difference between two approximations instead of the one between the original graph and its approximation. This is to prevent asymmetricity which can easily be seen in Eq.(5). The difference between two graphs can be different if the original graph is different. Therefore we use a new criterion for Umeyama's approach to make the results symmetric, and that is

$$\|V_H V_G^T L_G V_G V_H^T - V_G V_H^T L_H V_H V_G^T\|^2 = \|V_H D_G V_H^T - V_G D_H V_G^T\|^2. \tag{6}$$

By observing Eq.(6), we can see that the approximation of one graph in Umeyama's approach is by relocating the other graph's eigenvalues in its eigenspace. This suggests that the difference between these two approximations is the difference derived from two different eigenspace.

2.2 An Eigenspace Projection Clustering Method for Inexact Graph Matching

Subspace projection methods are conventionally used to reduce the dimensionality of data by minimizing the number of dimensions and the amount of information loss. To solve the inexact graph matching problem, Caelli [1] proposed an approach to project graphs into their subspaces. Furthermore, to compare graphs in different subspaces, these subspaces should have the same dimensionality. This was done by setting a number k to fix the dimensionalities of these subspaces. In Caelli's approach, the graph Laplacian matrix, L, is first decomposed into the familiar eigenvalue and eigenvector matrix product $L = VDV^T$ as in Eq.(3). The original data is then projected onto a smaller number of k most important (i.e., the k largest eigenvalues) principal components.

To compare the similarities of two vertices from two graphs of different sizes, an additional renormalization step for such projections was introduced. Both eigenvector (V) and eigenvalue (D) matrices are truncated according to the chosen number of projection dimensions (k)-largest eigenvalues and their associated eigenvectors. The renormalized eigenvectors and eigenvalues become: $V_k^{'} = \frac{V_k}{\|V_k\|}$, $D_k^{'} = \frac{D_k}{\|D_k\|}$. As a result, the renormalized subspace projection of each vertex (column of L) was obtained as $L_k^{'} = D_k^{'}(V_k^{'})$. In this renormalized subspace, vertices from different graphs will be close together if their rescaled eigenvectors have similar angles and their rescaled eigenvalues are similar. Such scaling can map initially quite different dimensional vectors into similar positions in the subspace as long as their eigenspectrum components in this lower dimensional subspace share similar amounts of the matrix variance and the rescaled unit eigenvectors have similar orientations. It is easy to see that the similarities are dependent on similar truncated and rescaled eigenspectra as well as similar unit rescaled eigenvectors but they are independent of the relative sizes of the original graphs.

To be more precise, in this work, the difference between two graphs using this method was computed as

$$\|D_G^{'}(V_G^{'})^T - D_H^{'}(V_H^{'})\|^2, \tag{7}$$

where $D^{'}$ is the resized eigenvalue diagonal matrix while $V^{'}$ is the resized eigenvector matrix. The number of the k most important components are set to the size of the smaller graph.

3 JoEig: Graph Comparison in Joint Eigenspace

Unlike the previous methods, which project graphs into different eigenspaces and compare them, we project each pair of two graphs into a joint eigenspace. This joint eigenspace is expanded by both set of eigenvectors. Given Eq.(3), we can further rewrite the equations to

$$D_H = V_H^T L_H V_H,\ D_G = V_G^T L_G V_G. \tag{8}$$

If graph G and H are isomorphic, we will have $D_G = D_H$ and therefore

$$V_G^T L_G V_G = V_H^T L_H V_H. \tag{9}$$

By multiplying V_G on the left side and V_H^T on the right side of Eq.(9), we can further derive

$$L_G V_G V_H^T = V_G V_H^T L_H. \tag{10}$$

Therefore, $V_G V_H^T$ is the joint projection vector for both graphs G and H. By introducing Eq.(3) into Eq.(10),we will have

$$V_G D_G V_G^T V_G V_H^T = V_G V_H^T V_H D_H V_H^T. \tag{11}$$

After removing $V_G^T V_G$ and $V_H^T V_H$ from Eq.(11),

$$V_G D_G V_H^T = V_G D_H V_H^T \tag{12}$$

will be obtained. Therefore, the difference between two graphs using our method can be defined as

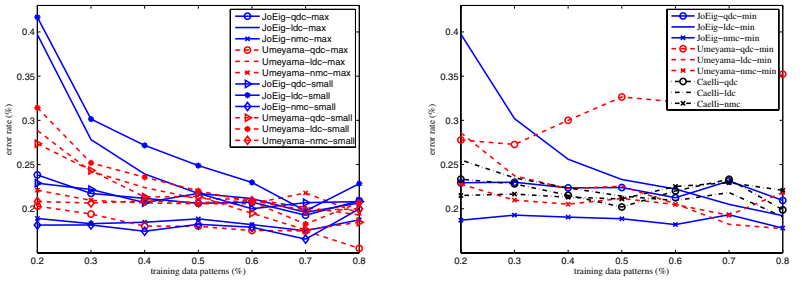
$$\|V_G D_G V_H^T - V_G D_H V_H^T\|^2. \tag{13}$$

Unlike Umeyama's method in which the approximation of one graph is defined by relocating the other graph's eigenvalues in its own eigenspace, our approach approximates the graph by relocating its eigenvalues in the joint eigenspace constructed by the eigenvectors of both graphs as shown in Eq.(13).

However, the sizes of graphs might be different, and therefore it might not be possible to make a matrix product between V_G and D_H or V_H and D_G as in Eq.(13). A feasible solution is to fix the number of eigenvectors for both graphs. In this work, we study three possibilities for setting the number of eigenvectors. First, we make full use of the eigenvectors from the larger graph (by larger graph, we mean graphs with more nodes). In this case, we have to expand the eigenvectors of the smaller graph with zero vectors and also assign zero eigenvalues to them. But this will lower the influence of the smaller graph because the newly added zero eigenvalue and eigenvectors have limit impact on the results. On the other hand, we also try to make full use of the eigenvectors from the smaller graph and keep the same number of eigenvectors and eigenvalues in the larger graph as in the smaller graph by removing less important eigenvalues and eigenvectors from the larger graph. Less important eigenvectors are those with smaller eigenvalues. This, on the other hand, is in favour of the smaller graph. Moreover, we ignore the size of these two graphs and just pick a reasonable fix small number of eigenvectors and eigenvalues for both graphs. In this way, the strength of the relationship inside a graph is more important than the size of the graph, but it is not easy to choose an appropriate number of eigenvalues and eigenvectors to use.

4 Experiments

In this section, we compare the JoEig approach with other two methods, i.e. Umeyama's and Caelli's methods, described in Section 2 in the dissimilarity space [9]. Three classifiers, i.e., linear discriminant (ldc), quadratic discriminant (qdc) and nearest mean classifier (nmc), are adopted to have a more general understanding over the performance in the dissimilarity space. Three real-world datasets, i.e., Mutag [3], House [13] and Coil-20 [7], are used in the experiments. We use 60 representative objects to construct the dissimilarity space for the House and Coil-20 datasets and 30 representative objects for the Mutag dataset. Also, the eigenvalue diagonal and eigenvector matrices are resized in three different ways. The first is to enlarge the matrices of the smaller graph to the size of the larger graph, and it is marked as *max* in the figures. The second is by



(a) (b)

Fig. 1. Mugtag Dataset: Learning curves of (a) JoEig and Umeyama's method with qdc, ldc, and nmc using max and small strategies and (b) all methods with qdc, ldc, and nmc using min resizing strategies

shrinking the matrices of the larger graph to the size of the smaller graph, and it is called *min*. The third one is called *small*, and it is done by reducing the matrices of both graphs to the size of the smaller graph minus 5. Moreover, all the results in the following are the average over 50 repetitions of experiments resulting in a very small standard deviation.

4.1 Experiment 1: Mutag Dataset

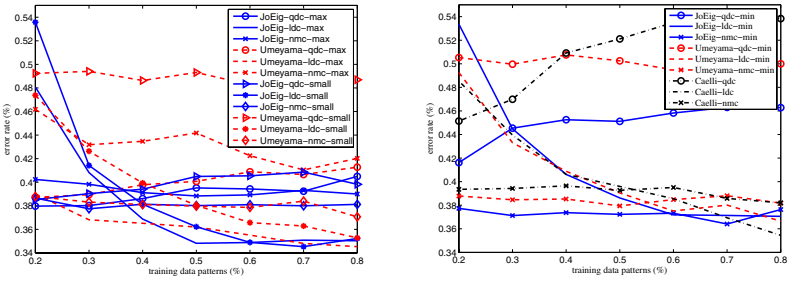
The Mutag dataset consists originally of 230 chemical compounds assayed for mutagenicity in Salmonella typhimurium. Among the 230 compounds, however, only 188 (125 positive ones, 63 negative ones) are considered to be learnable [3] and thus only 188 patterns are used in the simulations.

From Figure 1, it can be observed that under the cases of using a smaller number of eigenvalues and eigenvectors, nmc performs much better than qdc and ldc using JoEig and Umeyama's method, especially with small sample sizes. However, by using *max* to enlarge the smaller graph, Umeyama's method prefers qdc than nmc while nmc still performs better than qdc with JoEig. Which indicates that Umeyama's method using the *max* strategy brings the data points into a dissimilarity space where the global distance is not good enough for separating classes and therefore requires a more complex classifier to fulfill the task.

4.2 Experiment 2: House Dataset

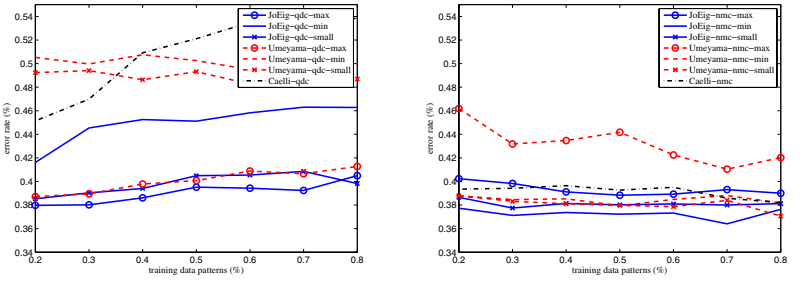
The house dataset contains 101 , 111, and 182 image sequences from three different houses, respectively. We extract the feature points using the scale-invariant feature transform (SIFT) method[5] and then compute the Voronoi tessellations of the feature points to construct the region adjacency graph, i.e., the Delaunay triangulation, of the Voronoi regions.

As shown in Figure 2, all methods prefer ldc and qdc over nmc when the *max* strategy is used, but nmc is more preferable when the *min* strategy is adopted.



(a) (b)

Fig. 2. House Dataset: Learning curves of (a) JoEig and Umeyama's method with qdc, ldc, and nmc using max and small strategies and (b) all methods with qdc, ldc, and nmc using min resizing strategies



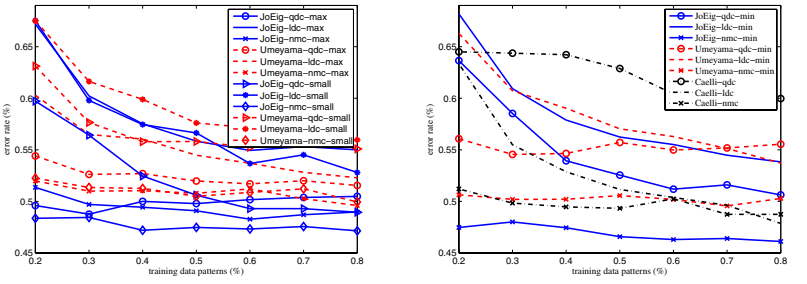
(a) (b)

Fig. 3. House Dataset: Learning curve of (a) quadratic discriminant and (b) nearest mean classifier using JoEig, Umeyama's, and Caelli's methods for graph comparison

Compared to the other two datasets, the graphs in this dataset are relatively more similar and therefore their relative distances are also closer. Which makes the global distance of not so great help in separating these three classes. However, selecting a small number of eigenvalues and eigenvectors to reconstruct the graph distances in the dissimilarity space seems to enhance the distance between graphs and makes it an easier task for nmc to handle. From Figure 3, we can also observe that nmc performs better with *small* and *min* strategies and qdc is better with the *max* strategy.

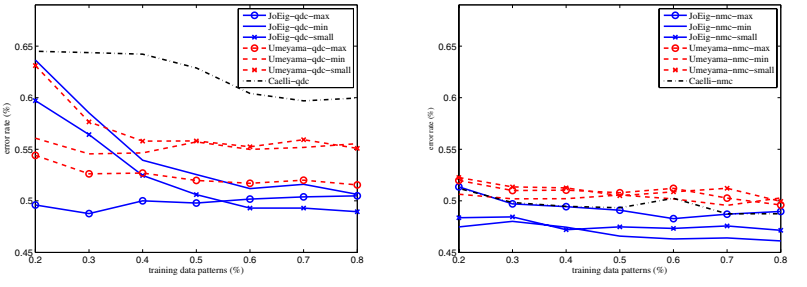
4.3 Experiment 3: Coil-20 Dataset

The Coil-20 contains multiple views of the same object in different poses with respect to the camera. There are originally 20 objects (classes) in the data, but we only use 5 objects and in total 358 images to form the dataset. Graphs are derived with the same method described in Section 4.2.



(a) (b)

Fig. 4. Coil-20 Dataset: Learning curves of (a) JoEig and Umeyama's method with qdc, ldc, and nmc using max and small strategies and (b) all methods with qdc, ldc, and nmc using min resizing strategies



(a) (b)

Fig. 5. Coil-20 Dataset: Learning curve of (a) quadratic discriminant and (b) nearest mean classifier using JoEig, Umeyama's, and Caelli's methods for graph comparison

As given in Figure 4, again we discover that nmc performs better with JoEig and Umeyama's method using different resizing strategies despite the fact that COIL-20 dataset is with a much more complex graph structure than the Mutag dataset. From Figure 5, we derive the same conclusion as before that nmc performs better with *small* and *min* strategies while qdc is better with the *max* strategy.

5 Conclusions

We propose a graph comparison approach named JoEig in the joint eigenspace based on eigendecomposition. The main contribution of this work is introducing the joint eigenspace and comparing graphs in it. We also study three different ways for resizing the eigenvalue diagonal and eigenvector matrices to solve the inexact graph matching problem. From a series of examples presented in the experiments, we conclude that our proposed JoEig approach is in general better

than Umeyama's and Caelli's methods. Furthermore, nmc performs better with *small* and *min* strategies while qdc is better with the *max* strategy. All in all, resizing graphs with *small* or *min* strategies and then using the nmc classifier in JoEig space is probably a better solution despite the original structure of the graph dataset.

References

1. Caelli, T., Kosinov, S.: An Eigenspace Projection Clustering Method for Inexact Graph Matching. IEEE Trans. Pattern Analysis and Machine Intelligence 26(4), 515–519 (2004)
2. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty Years of Graph Matching in Pattern Recognition. International Journal of Pattern Recognition and Artificial Intelligence 18(3), 265–298 (2004)
3. Debnath, A.K., Lopez de Compadre, R.L., Debnath, G., Shusterman, A.J., Hansch, C.: Structure-Activity Relationshop of Mutagenic Aromatic and Heteroaromatic Nitro Compounds. Correlation with Molecular Orbital Energies and Hydrophobicity, Journal of Medicinal Chemistry 34, 786–797 (1991)
4. He, P.R., Zhang, W.J., Li, Q., Wu, F.X.: A New Method for Detection of Graph Isomorphism Based on the Quadratic Form. Journal of Mechanical Design 125, 640–642 (2003)
5. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
6. Umeyama, S.: An Eigendecomposition Approach to Weighted Graph Matching Problems. IEEE Trans. Pattern Analysis and Machine Intelligence 10(5), 695–703 (1988)
7. Nene, S.A., Nayar, S.K., Murase, H.: Columbia University Image Library (COIL-20), `http://www1.cs.columbia.edu/CAVE/software/softlib/coil-20.php`
8. Neuhaus, M., Bunke, H.: Edit Distance-Based Kernel Functions for Structural Pattern Classification. Pattern Recognition 39, 1852–1863 (2006)
9. Pekalska, E., Duin, R.P.W.: Dissimilarity Representations Allow for Building Good Classifiers. Pattern Recognition Letters 23(8), 943–956 (2002)
10. Qiu, H.J., Hancock, E.R.: Spectral Simplication of Graphs. In: Proceedings of the 8th European conference on computer vision, Czech Republic, pp. 114–126 (2004)
11. Wilson, R.C., Hancock, E.R., Luo, B.: Pattern Vectors from Algebraic Graph Theory. IEEE Trans. Pattern Analysis and Machine Intelligence 27(7), 1–13 (2005)
12. Zhao, G.X., Luo, B., Ting, J., Ma, J.X.: Using Eigen-Decomposition Method for Weighted Graph Matching. In: Huang, D.-S., Heutte, L., Loog, M. (eds.) ICIC 2007. LNCS, vol. 4681, pp. 1283–1294. Springer, Heidelberg (2007)
13. CMU Vision and Autonomous System Center's Image Database, `http://www.ius.cs.cmu.edu/idb/`

A Labelled Graph Based Multiple Classifier System*

Wan-Jui Lee and Robert P.W. Duin

Faculty of Electrical Engineering, Mathematics and Computer Sciences,
Delft University of Technology, The Netherlands
W.J.Lee@tudelft.nl, r.duin@ieee.org

Abstract. In general, classifying graphs with labelled nodes (also known as labelled graphs) is a more difficult task than classifying graphs with unlabelled nodes. In this work, we decompose the labelled graphs into unlabelled subgraphs with respect to the labels, and describe these decomposed subgraphs with the travelling matrices. By utilizing the travelling matrices to calculate the dissimilarity for all pairs of subgraphs with the JoEig approach[6], we can build a base classifier in the dissimilarity space for each label. By combining these label base classifiers with the global structure base classifiers built on dissimilarities of graphs considering the full adjacency matrices and the full travelling matrices, respectively, we can solve the labelled graph classification problem with the multiple classifier system.

1 Introduction

Multiple classifier system [5] which is an efficient technique for improving the classification performance grows rapidly in the field of statistical pattern recognition in the last decade. But strikingly, there are very few attempts [1,10] in the literature to create base classifiers in the structural pattern recognition domain [2]. In structural pattern recognition, graphs are a general and powerful data structure for object representation. The nodes in a graph can represent different objects and the relationships between these objects or parts of the objects are represented by edges. Also, labels and attributes for the nodes and edges can further be used to incorporate more information in a graph representation.

One of the few examples for creating structural base classifiers is discussed in [12]. The idea is to generate different graph-based classifiers by randomly removing nodes and their incident edges from the training graphs until a maximum number of nodes is reached for all graphs. Because of the randomness, different graph-based classifiers can be created and each becomes a base classifier in the multiple classifier system. However, with this setting, we still need to compute similarity/dissimilarity for labelled graphs using time-consuming techniques such as the maximum common subgraph [2] or the graph edit distance [7] considering a labelled graph classification problem.

* We acknowledge financial support from the FET programme within the EU FP7, under the SIMBAD project (contract 213250).

J.A. Benediktsson, J. Kittler, and F. Roli (Eds.): MCS 2009, LNCS 5519, pp. 201–210, 2009.

Unlike graphs with unlabelled nodes, graphs with labelled nodes usually need to be processed and described with more complicated algorithms and structures. Also, classifying graphs with labelled nodes is a more difficult task than classifying graphs with unlabelled nodes. In this work, we propose a method to decompose labelled graphs into sets of unlabelled subgraphs, and describe the decomposed subgraphs with the travelling matrices. By using these travelling matrices as the adjacency matrices, we can reduce the problem of classifying labelled graphs into classifying sets of unlabelled graphs.

For each label, we can find out its corresponding nodes in a graph and calculate the travelling distances between all pairs of these nodes using Dijkstra's algorithm [3] for finding the shortest path given a pair of nodes. With the travelling distances, the connectivity information within the subgraph constructed by these corresponding nodes can be described with the travelling matrix. In the travelling matrix, the diagonal elements are always zero (a node is unreachable with itself) and the rest of the elements are the inverse of the travelling distances between nodes. Note that for a fully connected graph, the travelling matrix will reduce to an adjacency matrix. As a result, for a certain label, the travelling matrix of the subgraph for each graph can be found and used to represent the local structure. With this local representation, we can compute the dissimilarity between subgraphs with graph comparison methods. In this work, we adopt the JoEig (Joint Eigenspace) [6] approach to calculate the dissimilarity between pairs of subgraphs. The JoEig is an eigendecomposition based approach for comparing graphs. The main idea is to project a pair of graphs into a joint eigenspace which is expanded by the eigenvectors of both graphs and to compare the projected graphs. After the dissimilarity between all pairs of subgraphs are derived, we can create a base classifier in the dissimilarity space [8] for this label.

However, these label base classifiers only consider local structures of graphs. Obviously, there are also needs for base classifiers considering global structures of graph. Therefore, we also consider base classifiers with two different global structures of graphs. One is with the full adjacency matrix and the other is with the full travelling matrix. By combining the label base classifiers and the global structure base classifiers, we can solve the labelled graph classification problem with unlabelled graph representations.

The rest of the paper is organized as follows. In Section 2, we recap the JoEig approach for comparing unlabelled graphs. A multiple classifier system utilizes the label information of graphs is proposed in Section 3. Simulation results are presented in Section 4. Finally, a conclusion is given in Section 5.

2 Unlabelled Graph Comparison

Before we introduce the JoEig [6] approach for unlabelled graph comparison, some definitions and introduction on graphs are given as in the following.

A graph is a set of nodes connected by edges in its most general form. Consider the undirected graph $G = (V, E, W)$ with the node set $V = \{v_1, v_2, \ldots, v_n\}$, the edge set $E = \{e_1, e_2, \ldots, e_m\} \subset V \times V$, and the weight function $W : E \to (0, 1]$.

If the graph edges are weighted, the adjacency matrix A for the graph G is the $n \times n$ matrix with elements

$$A_{ij} = \begin{cases} W(v_i, v_j), & \text{if } (v_i, v_j) \in E; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Clearly since the graph is undirected, the matrix A is symmetric. The Laplacian [9] of the graph is defined by $L = D - A$, where D is the diagonal node degree matrix whose elements $D_{ii} = \sum_{k=1}^{n} A_{ik}$. The Laplacian matrix of G is positive semidefinite and singular, and it is more often adopted for spectral analysis than the adjacency matrix because of its properties.

2.1 JoEig: Graph Comparison in Joint Eigenspace

JoEig projects each pair of two graphs into a joint eigenspace. This joint eigenspace is expanded by both set of eigenvectors.

Let G and H be weighted undirected graphs and L_G and L_H be their Laplacian matrices, respectively. The eigendecomposition of L_G and L_H are performed as

$$L_G = V_G D_G V_G^T, \; L_H = V_H D_H V_H^T, \tag{2}$$

where V_G and V_H are orthonormal matrices and D_G and D_H are diagonal matrices of the eigenvalues (in ascending order) of G and H, respectively. With the joint projection vector $V_G V_H^T$, both graphs G and H will be projected to their joint eigenspace as $L_G V_G V_H^T$ and $V_G V_H^T L_H$. The difference between two graphs using JoEig is defined as

$$\|V_G D_G V_H^T - V_G D_H V_H^T\|^2. \tag{3}$$

The JoEig approach approximates a graph by relocating its eigenvalues in the joint eigenspace constructed by the eigenvectors of both graphs.

There are also three possibilities for setting the number of eigenvectors to compare graphs with different sizes in JoEig. In this work, we choose to make full use of the eigenvectors from the smaller graph and keep the same number of eigenvectors and eigenvalues in the larger graph as in the smaller graph by removing less important eigenvalues and eigenvectors from the larger graph.

3 A Labelled Graph Based Multiple Classifier System

We use the example graph shown in Figure 1(a) through this section to explain our method. This example graph is with 8 nodes and each node is labelled with one symbol. There are no attributes on the edges and the elements of the adjacency matrix A given in Eq.(4) of this graph are either 1 or 0 to indicate whether there is an edge between two nodes or not.

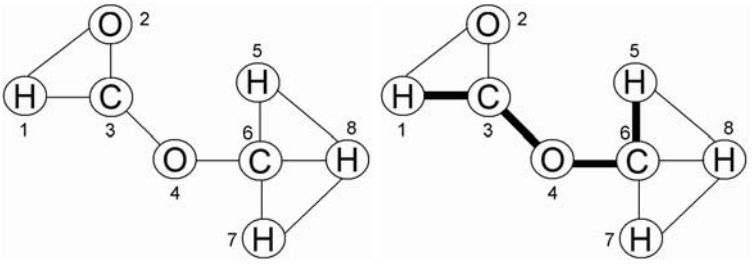


Fig. 1. An example of (a) labelled graph; (b) the shortest path between node 1 and node 5

$$A = \begin{pmatrix} 0\,1\,1\,0\,0\,0\,0\,0 \\ 1\,0\,1\,0\,0\,0\,0\,0 \\ 1\,1\,0\,1\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,1\,0\,1 \\ 0\,0\,0\,1\,1\,0\,1\,1 \\ 0\,0\,0\,0\,0\,1\,0\,1 \\ 0\,0\,0\,0\,1\,1\,1\,0 \end{pmatrix}. \tag{4}$$

3.1 Travelling Matrix

Our goal is to solve the labelled graph classification problem by decomposing labelled graphs into sets of unlabelled subgraphs. In order to represent the decomposed subgraphs, we need an other way than by the adjacency matrix to describe the connectivity information between nodes. The main reason is that, if we only choose nodes with the same label to form a subgraph, there are probably nodes with no neighbors at all and the subgraph might fall into isolated parts if it is represented by the adjacency matrix. To avoid this phenomenon, we propose the travelling matrix to represent the connectivity information of subgraphs. The basic assumption is that the larger the travelling distance between two nodes is, the less connective they are. The travelling distance between a pair of nodes can be easily computed with Dijkstra's shortest path algorithm [3]. An example of the shortest path between node 1 and node 5 is given in Figure 1(b), and the travelling distance between these two nodes is 4 since there are at least 4 edges one node has to travel to reach the other one. Therefore, an element in the travelling matrix is defined as the inverse of the travelling distance between two nodes. Also, the elements on the diagonal are all defined as zero. For the example graph in Figure 1(a), its full travelling matrix T will be

$$T = \begin{pmatrix} 0 & 1 & 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{3} & \frac{1}{4} & \frac{1}{4} \\ 1 & 0 & 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{3} & \frac{1}{4} & \frac{1}{4} \\ 1 & 1 & 0 & 1 & \frac{1}{3} & \frac{1}{2} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & 1 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 0 & 1 & \frac{1}{2} & 1 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{2} & 1 & 1 & 0 & 1 & 1 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 1 & 0 & 1 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 1 & 1 & 1 & 0 \end{pmatrix}. \quad (5)$$

Note that for a fully connected graph, the travelling matrix will reduce to an adjacency matrix. For the example in Figure 1(a), there are three different labels, i.e., C, H and O. For each label, we will extract a subgraph consisting only nodes with this particular label. For example, Figure 2(a), Figure 2(b) and Figure 2(c) are the subgraphs extracted with label H, O, and C, respectively. The solid line means that these two nodes are connected in the original graph, and the dash line means they are not connected in the original graph but now weakly connected by their travelling information. Now that a subgraph only consists of nodes with the same label, it means that we can actually ignore the label within the subgraph and fully describe this subgraph with a connectivity matrix (which is, travelling matrix by our definition). The travelling matrices for these 3 subgraphs are

$$T_H = \begin{pmatrix} 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{2} & 1 \\ \frac{1}{4} & \frac{1}{2} & 0 & 1 \\ \frac{1}{4} & 1 & 1 & 0 \end{pmatrix}, T_O = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix}, \text{ and } T_C = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix}, \text{ respectively.}$$

3.2 Dissimilarity and Base Classifiers

Given m graphs with n distinctive labels among the graphs, we want to create n base classifiers with respect to the labels. So, for a certain label, we extract a subgraph and its travelling matrix from each graph consisting only with this label

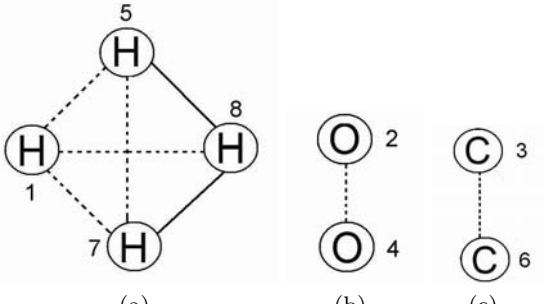


Fig. 2. Examples of subgraphs extracted with label (a) H, (b) O and (c) C, respectively, from the graph in Figure 1(a)

as described above. With these m subgraphs, the dissimilarity are calculated pairwise with the JoEig approach as described in Section 2. Some graphs might have no nodes with such label at all, and therefore the subgraphs of these graphs are empty. In this case, we directly set the dissimilarity to 0 if the two subgraphs are both empty, and set the dissimilarity to 1 if only one of the subgraphs is empty. As a result, we can obtain a $m \times m$ dissimilarity matrix for each label. With this dissimilarity matrix, we can build a base classifier for this label in the dissimilarity space [8]. In the end, we can construct n label base classifiers by doing the same to each label.

However, the label base classifiers only consider subgraphs which describe the local structures of graphs. To increase the diversity of the multiple classifier system, we also create global structure base classifiers. We propose two different global structure base classifiers, one for the full travelling matrix and the other for the full adjacency matrix. So we pairwise compare the original graphs with the JoEig approach to derive the dissimilarity matrix. But these original graphs can be represented with the full travelling matrices or the full adjacency matrices. Similar to the above, we also build global structural base classifiers for these two dissimilarity matrices.

4 Experiments

In this section, we compare the performance of the single base classifiers as described in Section 3 with the classifier combiner. Two classifiers, i.e., linear discriminant (ldc) and nearest mean classifier (nmc), are adopted to build base classifiers in the dissimilarity space [8]. All the base classifiers and the classifier combiner are built with the PRTOOLS [4]. Two real-world datasets, i.e., Mutagenicity and AIDS [11], are used in the experiments. We use 15% of training objects as the representative objects to construct the dissimilarity space for both datasets. Also, the eigenvalue diagonal and eigenvector matrices are resized to the size of the smaller graph with the JoEig approach. Moreover, all the results in the following are the average over 50 repetitions of experiments resulting in a very small standard deviation.

4.1 Experiment 1: Mutagenicity Dataset

Mutagenicity is one of the numerous adverse properties of a compound that hampers its potential to become a marketable drug. The molecules are converted into graphs in a straightforward manner by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled with the corresponding chemical symbol, and there are 10 different symbols in total. The average number of nodes of a graph is 30.3177 ± 20.1201, and the average number of edges is 30.7694 ± 16.8220. The Mutagenicity dataset is divided into two classes, i.e., mutagen and nonmutagen. There are in total 4,337 elements (2,401 mutagen elements and 1,936 nonmutagen elements). In the experiments, 40% of objects are randomly selected as the training dataset, 30% are taken as the validation set and the other 30% are used as the testing dataset.

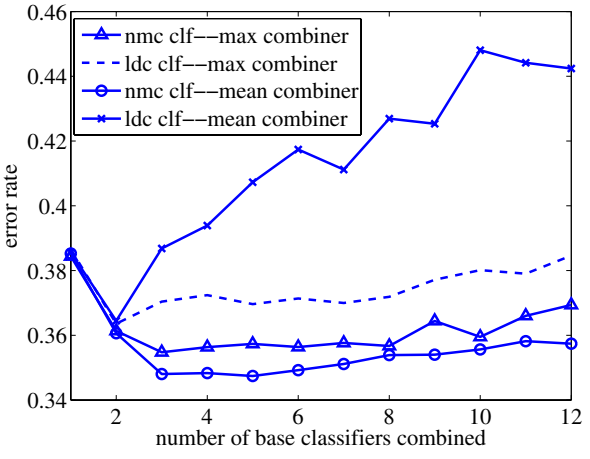


Fig. 3. Combination results of different number of base classifiers for Mutagenicity dataset

In Figure 3, we add the base classifiers (10 label base classifiers and 2 global structure classifiers) one by one with the sequential forward feature selection technique. The base classifier contributes most (with respect to the validation dataset) to the combination results of the current chosen base classifiers will be selected as the next base classifier to be combined. From Figure 3, we can see that nmc base classifiers give better combination results than ldc base classifiers with both max and mean combining rules. Combining ldc base classifiers with the mean combining rule performs much worse than the other combinations, especially when more and more base classifiers are combined. This is because some dissimilarity matrices in the label base classifier are highly correlated as some labels are absent in most graphs. As a result, most elements in the dissimilarity matrix are zero. Therefore, these base classifiers become very noisy to ldc with the mean combining rule. Also, for nmc base classifiers, the error rates of the combination results first decrease when more classifiers are included in the combination, and then remain stable, but increase again in the end when too many worse base classifiers are included in the combination. A very interesting phenomenon is that all the combiners reaching the lowest error rate have at least one of the global structure base classifiers as the base classifiers. So it is clear that the global structures can improve the classification performance.

Now the question is, would the label base classifiers also improve the performance of the combiner or is it sufficient to only combine the global structure base classifiers? In Figure 4(a) and Figure 4(b), we present the learning curve of nmc base classifiers and their max and mean combiners, respectively. From both figures, we observe that the results of combining only two global structural classifiers are much worse than combining the best 4 base classifiers. Therefore, label base classifiers also contribute significantly to the combiner.

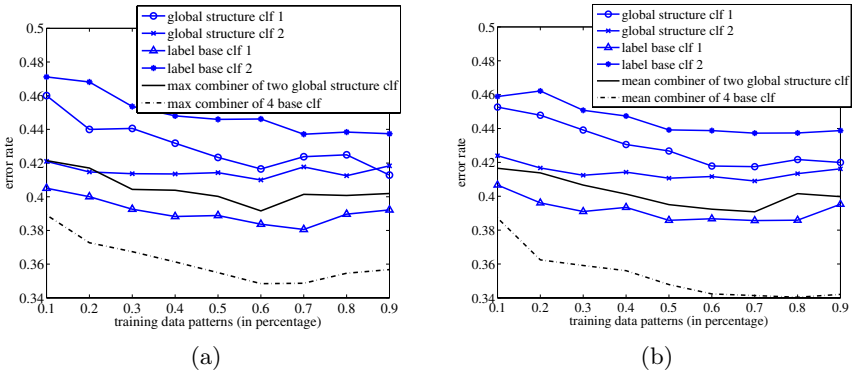


Fig. 4. Learning curves of nmc base classifiers and their (a) max and (b) mean combiners

4.2 Experiment 2: AIDS Dataset

The AIDS dataset consists of graphs representing molecular compounds. The graphs are constructed from the AIDS Antiviral Screen Database of Active Compounds (molecules). This dataset consists of two classes, active and inactive, to indicate molecules with activity against HIV or not. The molecules are converted into graphs in a straightforward manner by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled with the corresponding chemical symbol, and there are 26 labels in total. The average number of nodes of a graph is 15.6953 ± 13.1918, and the average number of edges is 16.1986 ± 15.0123. There are 2,000 elements in total (1,600 inactive elements and 400 active elements). In

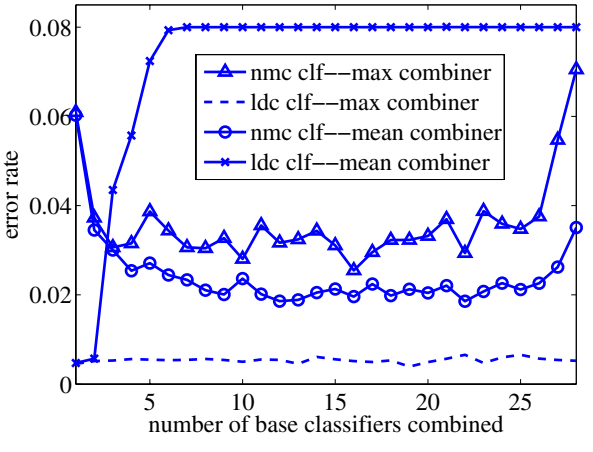


Fig. 5. Combination results of different number of base classifiers for AIDS dataset

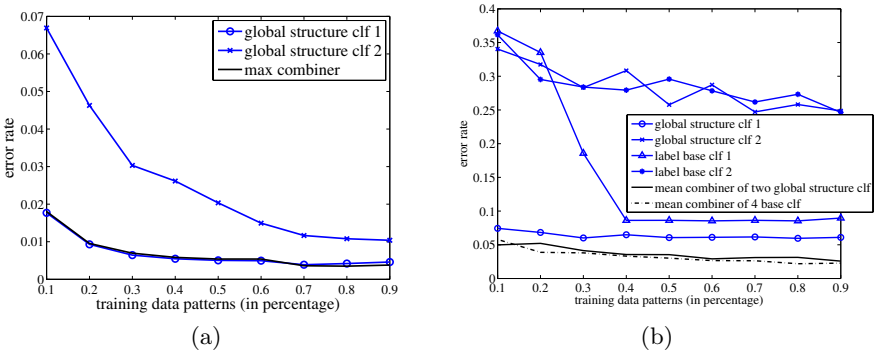


Fig. 6. Learning curves of (a) ldc base classifiers and their max combiner and (b) nmc base classifiers and their mean combiner

the experiments, 40% of objects are randomly selected as the training dataset, 30% are taken as the validation set and the other 30% are used as the testing dataset.

In Figure 5, the base classifiers (26 label base classifiers and 2 global structure classifiers) are added one by one using the same technique described above. The AIDS dataset is a much easier dataset to classify compared to the Mutagenicity dataset, and some base classifiers already reach very small error rates which makes it for the combiner difficult to improve the individual performance. From Figure 5, we can still observe that the ldc base classifiers with the mean combining rules are heavily disturbed by the correlated dissimilarity matrices and perform much worse than the other combinations. On the other hand, ldc base classifiers with the max combining perform much better than the others.

In Figure 6(a), the combiner is only slightly better than individual ldc classifiers with large amount of training data because one of the individual classifier has almost zero error rate and that leaves no much room for the combiner to improve. On the other hand, if we use weak base classifiers as in Figure 6(b), the combiners can have more significant improvements.

5 Discussions and Conclusions

We solve the labelled graph classification problem with the multiple classifier system by decomposing labelled graphs into unlabelled subgraphs with their labels and building label base classifiers from these subgraphs. Two global structural base classifiers are also considered to increase the diversity of the multiple classifier system. The subgraphs are represented by the travelling matrices instead of the adjacency matrices. The travelling matrix records the node to node travelling information. By comparing graphs/subgraphs pairwise with the JoEig approach, we can derive the dissimilarity matrix. With the dissimilarity matrix, we can construct the base classifier in the dissimilarity space.

Even though we only consider nodes with single labels and edges with no attributes in the experiments, our approach also applies to nodes with multiple labels and edges with attributes. For multiple labels, we can decompose graphs into subgraphs that might have common nodes. For attributed edges, we can simply use the weighted adjacency matrix.

References

1. Bunke, H., Irniger, C., Neuhaus, M.: Graph Matching - Challenges and Potential Solutions. In: Roli, F., Vitulano, S. (eds.) ICIAP 2005. LNCS, vol. 3617, pp. 1–10. Springer, Heidelberg (2005)
2. Bunke, H., Riesen, K.: Graph Classification Based on Dissimilarity Space Embedding. In: da Vitoria, L., et al. (eds.) Proc. SSSPR 2008, Structural, Syntacic, and Statistical Pattern Recognition, Florida, USA. LNCS, vol. 5342, pp. 996–1007. Springer, Heidelberg (2008)
3. Dijkstra, E.W.: A Note on Two Problems in Connexion with Graphs. Numerische Mathematik 1, 269–271 (1959)
4. Duin, R.P.W., Juszczak, P., Paclik, P., Pękalska, E., de Ridder, D., Tax, D.M.J.: PRTOOLS4, A Matlab Toolbox for Pattern Recognition, The Netherlands, Delft University of Technology. ICT Group (2004), `http://www.prtools.org`
5. Kuncheva, L.I.: Combining Pattern Classifiers. Methods and Algorithms. Wiley, Chichester (2004)
6. Lee, W.J., Duin, R.P.W.: An Inexact Graph Comparison Approach in Joint Eigenspace. In: Proc. SSSPR 2008, Structural, Syntacic, and Statistical Pattern Recognition, Florida, USA. LNCS, vol. 5342, pp. 35–44. Springer, Heidelberg (2008)
7. Neuhaus, M., Bunke, H.: Edit Distance-Based Kernel Functions for Structural Pattern Classification. Pattern Recognition 39, 1852–1863 (2006)
8. Pękalska, E., Duin, R.P.W.: The Dissimilarity Representation for Pattern Recognition. In: Fundations and Applications. World Scientific, Singapore (2005)
9. Qiu, H.J., Hancock, E.R.: Spectral Simplication of Graphs. In: Proceedings of the 8th European Conference on Computer Vision, Czech Republic, pp. 114–126 (2004)
10. Riesen, K., Bunke, H.: Classifier Ensembles for Vector Space Embedding of Graphs. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 220–230. Springer, Heidelberg (2007)
11. Riesen, K., Bunke, H.: IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning. In: da Vitoria, L., et al. (eds.) Proc. SSSPR 2008, Structural, Syntacic, and Statistical Pattern Recognition, Florida, USA. LNCS, vol. 5342, pp. 287–297. Springer, Heidelberg (2008)
12. Schenker, A., Bunke, H., Last, M., Kandel, A.: Building Graph-Based Classifier Ensembles by Random Node Selection. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 214–222. Springer, Heidelberg (2004)