Graph regularizations

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	Graph regularizations for Geometric Manifold Embeddings
Deliverable	D4.3
Document type	Report
Dissemination level	Public
Contractual date of delivery	M 30
Project reference number	213250
Status & version	Definitive version
Work package, deliverable responsible	WP 4, UNIYORK
Contributing Partners	DELFT, UNIVE
Author(s)	Richard C. Wilson, Edwin R. Hancock, Wan-Jui Lee, Robert P. W Duin, Andrea Torsello
Additional contributor(s)	-

SIMBAD Deliverable 4.3: Graph regularizations for Geometric Manifold Embeddings

Richard C. Wilson[1]
Edwin R. Hancock[1]
Wan-Jui Lee[2]
Robert P. W Duin[2]
Andrea Torsello[3]
[1]Dept. of Computer Science,University of York, UK
[2]Faculty of Electrical Engineering, Mathematics and Computer Sciences
Delft University of Technology, The Netherlands
[3]Dipartimento di Scienze Ambientali, Informatica e Statistica
Università Ca Foscari Venezia

September 19, 2011

1 Overview

The overall goal of Workpackage 4 is that, given some dissimilarity data describing objects of interest, we wish to developing algorithms for transforming them into instance-specific spatial representations (embeddings) that are suitable for geometric learning algorithms. The work package is divided into the following tasks:

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

Within WP4.1 we are interested in the problem of embedding dissimilarities onto manifolds which are not flat (i.e. that exhibit non-Euclidean behaviour). Our basic representation for the dissimilarities is a graph; the vertices represent objects and the edges encode the given dissimilarities between objects. The problem is then one of graph embedding. The embedding problem is ill-posed because the data does not fully constrain the shape of the manifold, and it is likely noise is present in the data. Regularization of embeddings is therefore a very important issue.

In this report (Deliverable 4.3) we describe our progress in the area of graph regularization for manifold embeddings. This encompasses both methods to constrain the overall form of the manifold and to regularize local measurements on the manifold such as curvature. Our main achievements are as follows:

- A set of tools to probe the cause of non-Euclidean behaviour in dissimilarity data.
- Methods to regularize quantities measured on the graph using diffusion processes.

- A method of embedding the graphs onto the surface of constant curvature manifolds in the presence of noise.
- A algorithm to evolve non-Euclidean dissimilarities towards a Euclidean manifold.
- Methods to measure and control the complexity of graph structures, and some information-theoretic kernels derived from these complexities.

These achievements have led to a number of publications, listed in the references, and we have included key works in the appendix. In the following sections we give an overview of the work in these areas. More detailed description are available in the associated papers.

2 Causes of non-Euclidean Behaviour in data

Non-Euclidean behaviour in distances may have a number of different causes, some of which are important to the structure of the data, and some which are confounding factors. Some of the main causes are:

1. **Extended objects:** Data items may not represent point-like objects, rather they may cover an extended region of some putative embedding space. If this is the case then, for example, it is possible for two objects to touch a third (i.e. have zero distance to the third object) while having a non-zero difference between each other. In this case the triangle inequality is violated and the data is certainly non Euclidean-embeddable or even metric.
2. **Non-flat manifolds** The data may exist on a manifold which is curved. Curved manifolds are not isomorphic to Euclidean space and so cannot be embedded into such a space in a distortion-free way. A well-known example of this is furnished by points on the surface of the earth, where distances are measured across the surface and distortion free maps are not possible. Such data is metric.
3. **Noisy distances** As with all measurement processes, we must account for the possibility that distances between data items are subject to errors. This means that even if our objects reside in Euclidean space, the measured distances will not be embeddable without error.

It is important to be able to distinguish between these different cases before determining the best way to approach the data. There are some simple tests available to distinguish some classes (see Workpackage 3). For example, any asymmetry or triangle violations in the data indicate that we have non-metric data.

Given a matrix $\mathbf{D}_s$ of squared distances, we can compute the centred similarity matrix:

$$\mathbf{S} = -\frac{1}{2}(\mathbf{I} - \frac{1}{n}\mathbf{J})\mathbf{D}_s(\mathbf{I} - \frac{1}{n}\mathbf{J}) \tag{1}$$

In Euclidean space, this procedure gives exactly the kernel matrix for the points. From this, we can compute the negative eigenfraction (NEF):

$$\text{NEF} = \frac{\sum_{\lambda_i < 0} |\lambda_i|}{\sum_i |\lambda_i|} \tag{2}$$

where $\lambda_i \, \forall i$ are the eigenvalues of $\mathbf{S}$. If NEF=0, then the data is Euclidean, otherwise the data may still be metric but is not Euclidean.

The presence of noise in the data makes it more problematic to use these simple tests. For example, in a noisy environment, even Euclidean data will give an non-zero NEF because of the errors introduced by noise. Triangle violations may also occur. Furthermore, extended objects do not necessarily always cause triangle violations themselves. Therefore, there is a need for further analysis of the detection of non-Euclidean behaviour.

Under this workpackage, we have investigated a number of measures aimed at separating these different causes of non-Euclidean behaviour. Details of the approach are given in [15]. We considered three measures to categorise the nature of the deviations from Euclidean behaviour. The first is produced by offsetting the distances to restore metricity[9]. By adding a constant offset to all distances, we can remove any triangle violations in the data. The smallest constant necessary to do this is a measure of the deviation from metricity. Secondly, we analyzed the shape of the negative eigenvalues produced by the similarity matrix $\mathbf{S}$. While the overall mass of eigenvalues can indicate the degree of departure from Euclidean behaviour, the actual shape of the ordered negative eigenvalues can reveal more detailed information. We noted that the eigenvalues follow an exponential trend, and fitting the exponential $y = ae^{bx}$ gives two additional parameters a, b. By analyzing these three parameters, we were able to separate cases generated by noise, extended objects and curved manifolds (specifically the sphere).

In summary, real data may be affected by both noise and some other intrinsic cause of non-Euclidean behaviour. For the remainder of this report, we will focus on data which lies on a non-Euclidean manifold, as most data we have seen is metric or nearly metric. We can represent such data as a graph, with vertices representing the objects and edges which have a length corresponding to the measured distances. Our problem is then one of *graph embedding*, i.e. finding an embedding of the graph onto a Riemannian manifold which respects the edge distances as much as possible. Since there are an infinite number of possible manifolds which could fit the data and noise in the distances, we have a classic overfitting problem and need to seek suitable regularizers for the problem.

3 Constant curvature embedding

The simplest (non-Euclidean) regularizer is to constrain the embeddings to a limited class of manifolds, namely those with constant curvature everywhere. In this workpackage, we investigated the use of both constant curvature manifolds (elliptic and hyperbolic space) to embed distance graphs[12]. The technical details are given in Appendix B.

The radius of curvature is determined directly from the distance matrix $\mathbf{D}$ via the spherical inner-product matrix $\mathbf{Z} = r^2 \cos \mathbf{D}/r$. The optimal radius can be found by minimising the magnitude of the smallest eigenvalue of $\mathbf{Z}$. We can then embed the points onto the elliptic or hyperbolic manifold.

Due to noise, distortions and the varying nature of data, a general set of dissimilarities will not lie exactly on the manifold. In the work described in Appendix B we demonstrate that when the curvature is low, the natural correction (discarding negative eigenvalues of the inner-product matrix) is a good approximation. For highly-curved dataset, this correction is not appropriate and we provide an alternative projection method which involves finding a new optimal set of eigenvalues for the inner-product matrix.

4 Smoothing processes on graphs

4.1 Curvature regularization

An alternative approach to constant-curvature embedding is to allow the curvature to vary by attaching a variable curvature to each edge in the graph. This approach is detailed in later sections of [13]. The procedure is as follows: First we embed the points into Euclidean space. This gives a Euclidean distance between two points which ignores the curvature of the manifold. Assuming a constant curvature on each edge, we can compute the curvature from the actual (geodesic) distance and the Euclidean distance:

$$d_E = \frac{2}{\sqrt{K}} \sin\left(\frac{\sqrt{K}}{2} d_G\right)$$

This equation can be simply solved for the curvature K using a Newton iteration.

However, due to the noisy nature of the distances, the individual curvature estimates are not reliable. We regularize these curvatures by using a smoothing process on the graph. The smoothing utilizes the heat kernel on the graph. The heat kernel is a fundamental solution of the heat equation on the graph and is given by

$$H(t) = \exp(-Lt)$$

where L is the Laplacian of the graph. This provides a smoothing kernel on the vertices of the graph (larger t corresponds to more smoothing).

Since we define curvatures on the edges of the graph, we cannot directly apply the vertex-based regularizer to the problem. We first construct the dual graph from the original distance graph. In the dual graph, each original edge becomes a vertex, and they are connected if the corresponding edges share a vertex in the original graph. Regularization then proceeds on the dual graph.

4.2 Quaternion Regularization

In [11] we address the problem of restoring a global geometric embedding given a set of pairwise constraints in the form of measurements of the relative pose of each node. Due to the noise in the measurement process there is no geometric embedding satisfying the constraint, but we seek a minimal tension solution through a diffusion process over the graph of geometric constraints defined over the manifold of rigid transforms. However, there is no single metric on such manifold linked with the problem at hand, since the tension depends on the (yet unknown) position of points in space. We solve this by proving that, while the tension depends on the points, the minimizer does not. The result is a graph smoothing algorithm which recovers registration by diffusion of a dual quaternion representation.

4.3 Edge-based regularization

In some ways the heat-kernel based regularization described in the previous section has weaknesses. We already noted in the previous section that the diffusion is described on the vertices only, and we must transfer the problem to the dual graph to smooth edge-based curvatures. More fundamentally, this simple diffusion process does not model diffusion well because the diffusion between vertices is instantaneous - there is no speed of propagation.

In our most recent work, we have tried to address these issues using the framework of Friedman[4]. He defines a graph Laplacian Δ which exists on both the vertices and edges. In this model, each edge has a finite length and signals propagate at a finite speed. From this, we can define a heat equation

$$\frac{df}{dt} = \Delta f$$

where f is a function defined on both the vertices and edges. The fundamental solution is then another type of heat kernel which is defined everywhere on the graph and can directly regularize over edges. For example, the edge-based heat-kernel for a graph with no boundary is given by

$$\begin{aligned} K(t, v, x_e, u, x_f) &= \sum_{\omega \in \Omega_A} e^{-\omega^2 t} C(e) \cos[B(e) + \omega x_e] C(f) \cos[B(f) + \omega x_f] \\ &\quad + \sum_{n=0}^{\infty} e^{-\pi^2 n^2 t} C_n \cos[\pi n x_e] \cos[\pi n x_f] \end{aligned}$$

Here v is a vertex and x_e is the position along an edge e which leaves the vertex v. ω^2 is an eigenvalue of the edge-based Laplacian. The heat kernel is therefore defined at a time t and between two edge positions (v, x_e) and (u, x_f). Hence this kernel can be used to directly smooth the curvature values on edges.

5 Ricci Flow Embedding

Our next piece of work involves using a Ricci-flow based smoothing process to regularize dissimilarities. This work is described in [2] The goal is to transform the data into Euclidean space. In order to prevent excessive distortion of the data, we have developed a method which gradually reduces the curvature of the manifold using Ricci flow. The Ricci flow[2] evolves a manifold so that the rate of change of the metric tensor is controlled by the Ricci curvature. Essentially, this is an analogue of a diffusion process for a manifold. The geometric evolution equation is:

$$\frac{dg_{ij}}{dt} = -2R_{ij}$$

where g_{ij} is the metric tensor of the manifold and R_{ij} is the Ricci curvature. The evolution of the metric tensor according to this differential equation results in a flattening of the manifold over time.

By assuming a locally constant curvature on the edge, the evolution of the manifold can be solved to find

$$K(t) = \frac{K_0}{1 \pm 2K_0 t}$$

Naturally, there is an interaction between the local curvatures on the edges, and so we have adopted an iterative update procedure, where each curvature is evolved for a short time and then the distances are recomputed.

The second important ingredient of this method is the coupling between curvature and distance on the manifold. Under the constant local curvature assumption, the curvature, geodesic and Euclidean distances are connected via

$$d_E = \frac{1}{\sqrt{K}} \sin \frac{\sqrt{K} d_G}{2}$$

In order to find the Euclidean distance we need to embed the points into a Euclidean space. We explored two methods to achieve this; the kernel embedding[8] and ISOMAP[10]. Each embedding has different properties. The kernel embedding discards negative eigenvalue components of the similarity matrix before embedding the kernel matrix. As a result the Euclidean distances are always shorter than the original geodesic distances, and therefore this method gives positive curvature. ISOMAP tries to reproduce the local geodesic distances in the embedding and can produce positive or negative curvatures.

The curvatures produced by either method are subject to considerable noise, and must be smoothed before use in the Ricci flow algorithm. The method described in a previous section may be used to regularize the values across the graph. We have also investigated another approach[14] based on a stronger constraint. We model the manifold as a set of local hyperspheres centred at each object. The curvature is therefore constant over a small neighbourhood of the graph, rather than single edges. We use the constant-curvature embedding to find the position of the points and the curvature. It is this curvature which is then evolved according to the Ricci flow. Distances are preserved relative to the central object by using the tangent-plane embedding of the hypersphere.

In summary, the Ricci flow embedding method proceeds as follows; firstly a Euclidean embedding is produced from the geodesic distances. Then the curvatures are computed from the distances and the embedding. The curvatures evolve according to the Ricci flow dynamics. Finally, new geodesic distances are computed from the curvature, and the process repeats. This process gradually flattens the manifold on which the data lies.

6 Measuring and controlling graph complexity

We have explored how information theoretic complexity measures can be used to a) control the complexity of generative models learned from sets of example graphs, and b) construct information theoretic graph kernels. The complexity measures studied are based on entropy estimates for the graphs. Here we have explored two alternatives. These are the thermodynamic depth complexity (Appendix E), and the Von-Neumann entropy [7, 5]. The thermodynamic depth complexity, associates with heat flow on a graph the concept of an entropy history, and this together with the notion of phase-change can be used to assign a complexity measure to a graph. The second approach has been to develop a simplified computation of the Von Neumann entropy from quantum systems. This is the Shannon entropy associated with the normalised Laplacian spectrum. By making a quadratic approximation to the Shannon entropy, we have shown how to approximate the Von Neumann entropy using simple node degree statistics.

In the first instance, we have used these entropies to regulate the model-order of generative models fitted to sets of examples graphs using a description length information coding criterion. Here the goodness of fit of the model is traded against the complexity of the fitted model [6]. Secondly, we used the entropies to construct information theoretic graph-kernels. The thermodynamic depth complexity has been used for the problem of matching embedded graphs by alignment [3]. The approximate von Neumann entropy has been used to construct Shannon-Jensen kernels for pairs of graphs [1].

References

[1] Lu Bai and E. R. Hancock. Graph clustering using the Jensen-Shannon kernel. In *CAIP*, 2010.

[2] B. Chow and F. Luo. Regularising the ricci flow embedding. *J. Differential Geom.*, 63(1):97–129, 2003.

[3] Francisco Escolano, Edwin R. Hancock, and Miguel Angel Lozano. Graph matching through entropic manifold alignment. In *CVPR*, pages 2417–2424, 2011.

[4] Joel Friedman and Jean-Pierre Tillich. Calculus on graphs. *CoRR*, arXiv:cs/0408028v1, 2004.

[5] Lin Han, Edwin R. Hancock, and Richard C. Wilson. Entropy versus heterogeneity for graphs. In *GbRPR*, pages 32–41, 2011.

[6] Lin Han, Edwin R. Hancock, and Richard C. Wilson. An information theoretic approach to learning generative graph prototypes. In *SIMBAD*, 2011.

[7] Lin Han, Edwin R. Hancock, and Richard C. Wilson. Learning generative graph prototypes using simplified von neumann entropy. In *GbRPR*, pages 42–51, 2011.

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

[9] 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.

[10] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[11] A. Torsello, E. Rodolà, and A. Albarelli. Multiview registration via graph diffusion of dual quaternions. In *IEEE International Conference on Computer Vision and Pattern Recognigion (CVPR2011).* IEEE Computer Society, 2011.

[12] Richard C. Wilson, Edwin R. Hancock, Elzbieta Pekalska, and Robert P. W. Duin. Spherical embeddings for non-euclidean dissimilarities. In *CVPR*, pages 1903–1910, 2010.

[13] Weiping Xu, Edwin R. Hancock, and Richard C. Wilson. Rectifying non-euclidean similarity data using ricci flow embedding. In *ICPR*, pages 3324–3327, 2010.

[14] Weiping Xu, Edwin R. Hancock, and Richard C. Wilson. Rectifying non-euclidean similarity data through tangent space reprojection. In *IbPRIA*, pages 379–386, 2011.

[15] Weiping Xu, Richard C. Wilson, and Edwin R. Hancock. Determining the cause of negative dissimilarity eigenvalues. In *CAIP (1)*, pages 589–597, 2011.

A Causes of Non-Euclidean Behaviour

Determining the Cause of Negative Dissimilarity Eigenvalues

Weiping Xu, Richard C. Wilson, and Edwin R. Hancock

Dept. of Computer Science, University of York, UK
{*elizaxu, wilson, erh*}*@cs.york.ac.uk*

Abstract. Pairwise dissimilarity representations are frequently used as an alternative to feature vectors in pattern recognition. One of the problems encountered in the analysis of such data, is that the dissimilarities are rarely Euclidean, and are sometimes non-metric too. As a result the objects associated with the dissimilarities can not be embedded into a Euclidean space without distortion. One way of gauging the extent of this problem is to compute the total mass associated with the negative eigenvalues of the dissimilarity matrix. However,this test does not reveal the origins of non-Euclidean or non-metric artefacts in the data. The aim in this paper is to provide simple empirical tests that can be used to determine the origins of the negative dissimilarity eigenvalues. We consider three sources of the negative dissimilarity eigenvalues, namely a) that the data resides on a manifold (here for simplicity we consider a sphere), b) that the objects may be extended and c) that there is Gaussian error. We develop three measures based on the non-metricity and the negative spectrum to characterize the possible causes of non-Euclidean data. We then experimentally test our measures on various real-world dissimilarity datasets.

Keywords: non-Euclidean pairwise data; metric; embedding

1 Introduction

Pairwise dissimilarity representations offer a powerful alternative to vectorial or feature-based characterisations of objects. Specifically, they provide a natural way of capturing the relationships between objects that are not characterised by ordinal measurements or feature vectors [6]. One way to translate such data into a vector representation is to represent the similarity data using a kernel matrix, and to embed the data into a vector space using kernel principal components analysis. In this way a vector representation is obtained by projecting the dissimilarity data into a vector space of fixed dimension.

However, one of the problems with dissimilarity representations and their embeddings is that the distance measures can not be used to construct a Euclidean vector space if the underlying Gram matrix contains negative eigenvalues. If this is the case, then the data can not be embedded into a real-valued Euclidean space, and must instead be embedded into a complex valued or Krein space [5].

In order to analyse non-Euclidean dissimilarity data using traditional geometric machine learning or pattern recognition techniques, we must first attempt to rectify the data so as to minimize the non-Euclidean artifacts. Examples of translating similarities into vector representation include using only the positive definite subspace of the distances, adding a constant amount to the off diagonal elements, i.e. the constant shift embedding [4], or manifold embedding (e.g. the spherical embedding in [1]).

Each of these approaches is based on assumptions concerning the sources of the negative eigenvalues. The positive definite subspace embedding assumes that metric violations are an artifact of noise and that the distances in the negative sub-space do not carry any significant discriminative information. The manifold embedding assumes that the Euclidean violations are geodesic and that the data resides on a manifold. Recent studies [2, 4] have showed that the negative eigenspace can contain valuable information. Moreover, Euclidean correction can lead to poor classification performance. Thus, before using any of the above approaches to attempt to rectify non-Euclidean data, it is advisable to analyze the underlying causes.

We model the distribution of non-Euclidean pairwise data in the following three situations: a) that the objects reside on the surface of a sphere (a simple manifold) and that the pairwise similarities are geodesic distances across the manifold, b) a non-metric dataset based on the distances between the surfaces of randomly positioned balls having different radii (Delft's balls data) and c) a noisy dataset with the Gaussian noise added to the distance between points in Euclidean space. By observing the spectrum of negative eigenvalues of the resulting Gram matrices and the additive constant required to render it metric, we identify three measures that can be used to characterise the above sources of negative eigenvalues. A variety of dissimilarity datasets are tested on the measures. Our analysis provides insight into the non-Euclidean behaviour of dissimilarity datasets and can be used to select appropriate embedding methods suited to the non-Euclidean data in hand.

Another secondary contribution of the paper is to develop a measure that assesses the contribution of each object to the mass of negative eigenvalues that provides further insight into the cause of non-Euclidean behavior. In this paper, we test a finer measure that assesses the contribution of each object to the mass of negative eigenvalues. In this way it is possible to determine whether the non-Euclidean artifacts are attributable to the dissimilarities associated with a few outlying objects or are uniformly distributed throughout the dataset.

2 Characterising the causes of non-Euclidean data

In this paper we are concerned with the sources of non-Euclidean data. Our overall aim is to identify the causes of a given set of non-Euclidean dissimilarity data so as to find out suitable correction methods to make them more Euclidean.

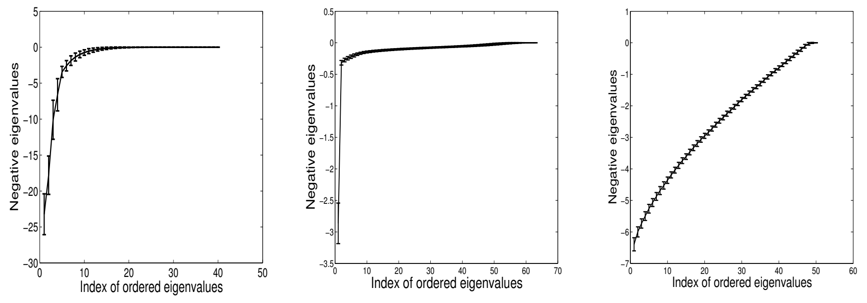
2.1 The causes of non-Euclidean data

We begin by identifying three reasons for non-Euclidean behaviour [2].

Manifold If the data points reside on a curved manifold, then the distances between them are intrinsically non-Euclidean (but still metric). This is one possible source of non-Euclidean distances. Here we model such data as points on the surface of a sphere, a simple surface where distances are easy to compute. It is simple to simulate patches with various degrees of curvature that depart from Euclidean behavior by changing the curvature of the patch. The dissimilarity measurements on the sphere are metric but non-Euclidean.

Extended objects If objects are not point-like but rather are extended in space, then the distances between them are measured between the closest points on their surface. As a result the distances will be non-Euclidean and possibly non-metric. Delft's balls data [2] is a typical example. Randomly positioned balls are generated with varying radius. The pairwise dissimilarities are the surface distances between the balls. As a result only the pairwise distances between balls with zero radius are Euclidean. It is also simple to modify the degree of non-Euclidean behaviour by adjusting the radii of the balls.

Gaussian noise The final source is Gaussian noise added to the original Euclidean dissimilarities. This will generate data that is both non-Euclidean and non-metric.



(a) on the sphere (b) balls data (c) Gaussian noise

Fig. 1: The negative eigenvalues of the resulting Gram matrix of 100 points on the sphere, from extended objects and Gaussian noise as a function to the index of ordered negative eigenvalues.

2.2 Negative spectrum

We study with the three simple modes of the occurrence of non-Euclidean pairwise data. The Gram matrix of non-Euclidean dissimilarity data is indefinite, i.e. it has negative eigenvalues. One way to gauge the degree to which a pairwise distance matrix exhibits non-Euclidean artefacts is to analyse the properties of its centralised Gram matrix. For an $N \times N$ symmetric pairwise dissimilarity matrix D with the pairwise distance as elements, the centralized Gram matrix $G = -\frac{1}{2}JD^2J$,where $J = I - \frac{1}{N}11^T$ is the centering matrix and 1 is the all-ones vector of length N. The degree to which the distance matrix departs from being Euclidean can be measured by using the relative mass of negative eigenvalues or "negative eigenfraction " $F_{eigS} = \sum_{\lambda_i<0} |\lambda_i| / \sum_{i=1}^{N} |\lambda_i|$ [3]. This measure is zero when the distances are Euclidean and increases as the distance becomes increasingly non-Euclidean.

We commence by examining the negative spectrum of the Gram matrix under the three models. Figure 1 shows the non-Euclidean dissimilarities from the sphere and balls data-sets have spectrum which contain a strong negative component, with a concentration towards the low end of the spectrum. The non-Euclidean dissimilarities from Gaussian noise have a more slowly decreasing negative spectrum. Each of the negative spectrum appear to follow an exponential decay. Thus the slope and the intercept from an exponential fit should be able to discriminate at least the Guassian noise model from the remaining two models. An exponential curve of the form $y = ae^{bx}$ is fitted to the data, with b the slope and a the intercept. These two parameters are used as measures to characterise the negative spectrum.

2.3 Non-metricity

A distance measure is considered to be non-metric if it is either non-symmetric, negative or violates the triangle inequality. A dissimilarity matrix rarely satisfies the triangle inequality, but is usually positive [4]. Thus the violation of the triangle inequality is considered when measuring non-metricity. A constant $C = \max_{i,j,k} |d_{ij} + d_{ik} - d_{jk}|$ is computed and added to the off-diagonal elements of the dissimilarity matrix so as to increase the amount of data that satisfies triangle equality [3]. If C is zero, the pairwise dissimilarity is considered to be metric. Moreover, the dissimilarity values over the sphere are metric. Thus a, b and C can be used as three measures to identify the three modeled sources of non-Euclidean behavior.

2.4 Object's contribution to the non-Euclidean behaviour of dissimilarities

If the non-Euclidean artefacts are created solely by the set of distances to a few outlying objects which are incorrectly placed, then it is possible to restore the data to a Euclidean state by editing these objects from the dataset. Based on this idea the notion of measuring the contribution of each object to the negative eigenfraction of a dissimilarity matrix is introduced.That is, the fraction given by the ratio of the sum of the negative distances originating from an individual object to each of the remaining objects, divided by the total.

The points can be embedded in Krein space as follows $Y = \sqrt{\Lambda}\Phi^T$ where Λ is the diagonal matrix with the ordered eigenvalues of centralised Gram matrix as elements and Φ is the eigenvector matrix with the ordered eigenvectors as columns. When the centered Gram matrix has negative eigenvalues then those dimensions of the embedding associated with negative eigenvalues are represented by imaginary numbers, and those associated with positive eigenvalues by real numbers. In other words, the data are embedded into a pseudo Euclidean or Krein space [5]. Under the embedding,the coordinate vector of point j is $y_j = (\sqrt{\lambda_1}\Phi_{1j}, ..., \sqrt{\lambda_i}\Phi_{ij}, \sqrt{\lambda_N}\Phi_{Nj})^T$. The contribution to the squared distance between two points k and e is

$$d_{ke}^2 = \sum_i (y_k(i) - y_e(i))^2 = \sum_i \lambda_i(\phi_{ik} - \phi_{ie})^2$$

The sum of negative squared distances and the sum of positive squared distances from point k to all the remaining points are:

$$d_{k-}^2 = \sum_{\lambda_i<0} \lambda_i \sum_{e\neq k} (\phi_{ik} - \phi_{ie})^2, \quad d_{k+}^2 = \sum_{\lambda_i>0} \lambda_i \sum_{e\neq k} (\phi_{ik} - \phi_{ie})^2$$

Thus the fraction of negative squared distances from point k is

$$f_{pneig} = \frac{|d_{k-}^2|}{|d_{k-}^2| + |d_{k+}^2|}$$

This measure is zero for all objects (or points) when the distances are Euclidean and non-zero for outlier objects. Thus the measure can be useful to identify whether the non-Euclidean is caused by the second sources.

3 Experiments

To model distances sampled from a manifold, we commence with 100 points uniformly distributed on the surface of a 3D sphere with unit radius. The spherical coordinates of an object are $x = (r \sin\theta \cos\phi, r \sin\theta \sin\phi, r \cos\theta)^T$ where r is the radius of the sphere, θ is the elevation angle($[0, \pi]$) and $\phi([0, 2\pi])$ is azimuth angle. The pairwise geodesic distances are computed as the lengths of great circle arcs between pairs of objects. We can use the tangent space projection and increase the radius or change the range of the elevation angle to control the extent to which the patches deviate from a Euclidean surface, i.e. the degree of non-Euclideanness in the dissimilarity matrix. In total 100 initial configurations of points are used.

To model the extended objects, we pick 100 randomly positioned points in a 7D hypercube with length 100, and we take each point as the center of a ball with radius $r(r \geq 0)$. The balls do not overlap. The pairwise distance is the Euclidean distances between the centers of two balls minus the radii of the two balls. We regard the balls with radius greater than 0 as non-Euclidean balls. We vary the fraction of non-Euclidean balls, and take the fraction to be 0.1, 0,3, 0.5, 0.7 or 0.9 in our experiments. The radii of the non-Euclidean balls are 2, 3 or 4. We also generate 100 balls with uniformly distributed radii ranging from 0 to 4.

To model the Gaussian noise, we commence with 100 randomly positioned points in a 3D Euclidean space and calculate the Euclidean dissimilarity matrix.

Then we add Gaussian noise with zero mean and various values of standard deviation to the off-diagonal elements of the dissimilarity matrix to generate a non-Euclidean dissimilarity matrix. The value of the standard deviation of the Gaussian noise is 0.1, 0.3, 0.5, 0.7 and 0.9.

To ensure the results are comparable over the dissimilarity data in various ranges and scales, all of the dissimilarity metrics are scaled such that the average dissimilarity is unity. We calculate the negative eigenvalues of each dissimilarity matrix and fit the average negative spectrum by an exponential curve to obtain the slope b, the intercept a and the average metric constant C. The whole process is repeated for a sample sizes of 500 and 1000 points.

Figure 2 shows the slope b as a function of the metric constant value C from the non-Euclidean dissimilarities on the sphere, the "balls" data and Gaussian noise. As the negative spectrum of the Gram matrix from the Euclidean points with Gaussian noise appears to be in a flat and linear in shape, so the value of slope b is very small with a value around -0.04. For the dissimilarities from the extended objects, the negative spectrum has a very sharp decreasing negative tail (just few significant negative eigenvalue), so the value for the slope b has a larger magnitude. Comparing the points on sphere and the ball data, there are several negative eigenvalues in the tail and the decrease is less sharp. This may explain why the slope of the non-Euclidean dissimilarities on the sphere is intermediate between that of the Gaussian noise and the non-Euclidean balls data. Another interesting finding is that the number of objects is not correlated with the slope, especially for points on the sphere and Gaussian noise. In terms of the parameters the three sources of negative eigenvalues are well separated from each other.

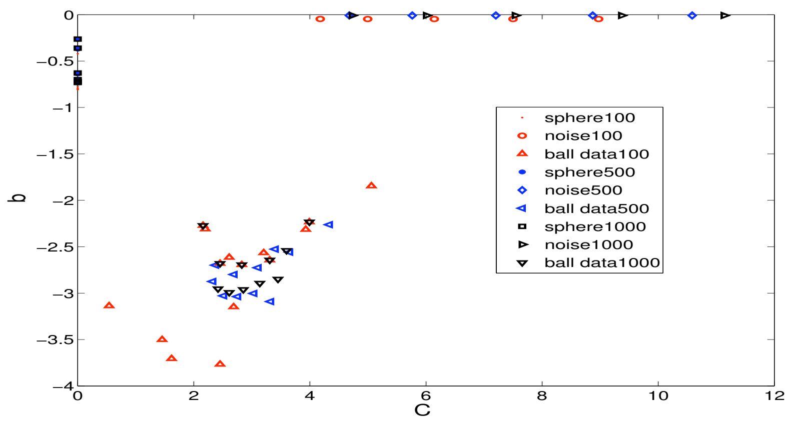
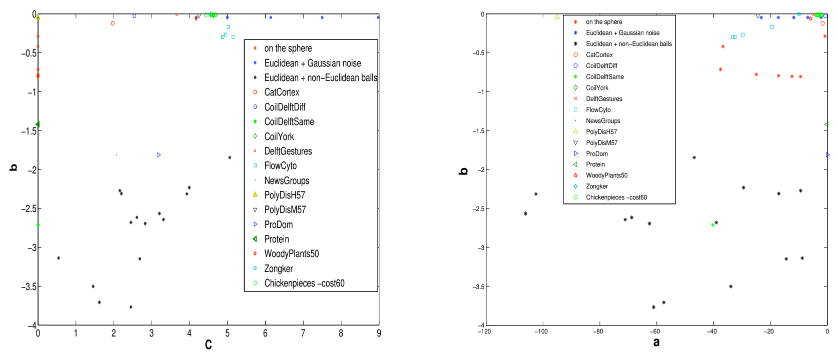


Fig. 2: The artificial non-Euclidean dissimilarity data caused by the manifold the data resides on, the extended objects and the Gaussian noise

We therefore use the above models to analyze a set of public domain dissimilarity data provided by the EU SIMBAD project consortium [2]. The Catcortex dataset contains dissimilarities based on the connection strengths between 65 cortical areas of the cat brain from four regions. CoilDelftDiff, CoilDelftSame and CoilYork are three dissimilariy datasets extracted from feature points detected in the COIL image database computed using different graph edit distances. FlowCyto contains four histogram dissimilarities for samples of breast cancer tissue. Newsgroups contains dissimilarities for messages in four classes of newsgroups. PolyDisH57 and PolyDisM57 are the dissimilarites of randomly generated polygons based on the standard and the modified Hausdorff distance. Protein contains the dissimilarities of protein sequences based on an evolutionary measure of distance. Woodyplants50 contains the shape dissimilarities between leaves of woody plants. Zongker contains the dissimilarities between handwritten digits based on deformable templates. Chickenpieces-cost60 contains 7 dissimilarity matrices from a weighed edit distance.



(a) (b)

Fig. 3: (a)The slope b as a function of the metric constant C; (b)The slope b as a function of the intercept a.

The left and right plots in Figure 3 respectively show the slope b and the intercept a as a function of the metric constant value C, for the artificial samples of 100 objects. The plots indicate that the non-Euclidean behaviour of Defilt-Gestures, PolyDisM57, Woodyplants50, Zongker, Chicken pieces, Catcortex and FlowCyto are likely to arise from Gaussian noise. On the other hand, the non-Euclidean behaviour of the Newgroups, ProDom and DelftSame datasets is likely to arise the non-Euclidean distances of a few outlying objects. We are unsure about the origin of the negative eigenvalues for the Protein and PolyDisH57 datasets. For PolyDisH57 the cause may be a combination of data residing on a manifold and the Gaussian noise. For the Protein dataset it may be a combination of data on the manifold and extended objects.

We plot the individual contribution to the negative eigenmass for the Protein dataset in Figure 4. This shows that the negative eigenvalues are caused by the non-Euclidean distances of just a few objects. The protein data is almost Euclidean with a very small negative eigenfraction value of 0.001. We have explored the effect of applying a leave one out nearest neighbor classifier to the dataset. When we edit out the effect of the outlier objects distances by adding a constant to the squared distances to the remaining objects, we obtain only a slightly smaller error rate of 0.47% compared to 1.9% for the original distances.

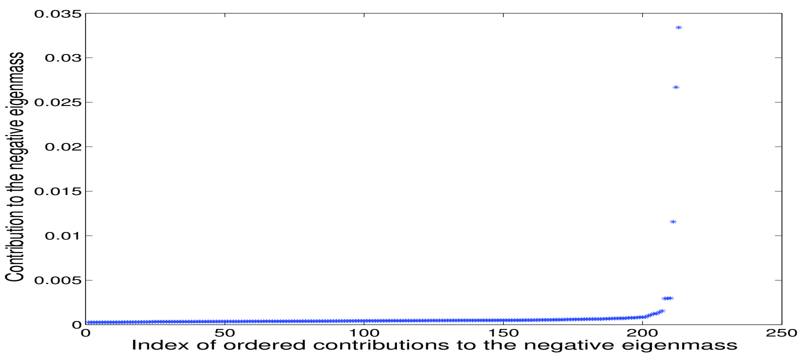


Fig. 4: Sorted each object's contribution to the negative eigenvalues at Protein

4 Conclusion

This paper discusses three possible sources of non-Euclidean behavior in dissimilarity data. We present three measures for analysing and determining the causes of negative eigenvalues in a non-Euclidean dissimilarity matrix. The three measures are based on distribution of the negative eigenvalues, and allow us to determine if the case is a) that data resides on a manifold, b)that the objects may be extended and c) that there is Gaussian noise.

Acknowledgments. Edwin R. Hancock is supported by the EU FET project SIMBAD and by a Royal Society Wolfson Research Merit Award.

References

1. Richard Wilson and Edwin Hancock :Spherical embedding and classification, SSPR, 2010
2. Duin, Robert P. W. and Pkekalska, Elżbieta: Non-Euclidean dissimilarities causes and informativeness, SSPR, pp.324 – 333 2010
3. Pekalska, Elżbieta and Harol, Artsiom and Duin, Robert and Spillmann, Barbara and Bunke, Horst: Non-Euclidean or Non-metric Measures Can Be Informative. SSPR, pp.871–880 2006
4. Julian Lauba, Volker Rothb, Joachim M. Buhmannb and Klaus-Robert Mllera:On the information and representation of non-Euclidean pairwise data, Pattern Recognition, pp. 1815-1826 2006
5. Goldfarb L.: A new approach to pattern recognition, Progress in pattern recognition, pp.241–402 (1985)
6. Sanfeliu, Alberto and Fu, King-Sun:A Distance measure between attributed relational graphs for pattern recognition,IEEE transactions on systems, man, and cybernetics, pp.353–362, 1983

B Spherical embedding

Spherical and Hyperbolic Embeddings of Data

Richard C. Wilson
Edwin R. Hancock
Department of Computer Science, University of York
UK

Elżbieta Pękalska
School of Computer Science, University of Manchester, United Kingdom

Robert P. W. Duin
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, Netherlands

August 26, 2011

Abstract

Many computer vision and pattern recognition problems may be posed by defining a way of measuring **dissimilarities** between patterns. For many types of data, these dissimilarities are not Euclidean, and may not be metric. In this paper, we provide a means of embedding such data. We aim to embed the data on a hypersphere whose radius of curvature is determined by the dissimilarity data. The hypersphere can be either of positive curvature (elliptic) or of negative curvature (hyperbolic). We give an efficient method for solving the elliptic and hyperbolic embedding problems on symmetric dissimilarity data. This method gives the radius of curvature and a method for approximating the objects as points on a hyperspherical manifold. We then develop a optimisation-based procedure for embedding objects on hyperspherical manifolds from a given set of dissimilarities. We use the Lie group

We acknowledge financial support from the FET programme within the EU FP7, under the SIMBAD project (contract 213250). Edwin Hancock was also supported by a Royal Society Wolfson Research Merit Award.

representation of the hypersphere and its associated Lie algebra to define the exponential map between the manifold and its local tangent space. We can then solve the optimisation problem locally in Euclidean space. This process is efficient enough to allow us to embed datasets of several thousand objects. We apply our method to a variety of data including shape-similarities, graph-similarity and gesture-similarity data. In each case the embedding maintains the local structure of the data while placing the points in a metric space.

1 Introduction

Many computer vision and pattern recognition problems may be posed by defining a way of measuring *dissimilarities* between patterns[7]. In other words, there are no features or vectors associated with the objects to hand, but instead there is a set of dissimilarities between the objects. Some examples include shape-similarities, gesture interpretation and graph comparison, but there are many more. There are two challenges to the analysis of such data. First, the objects can not clustered or classified using standard pattern recognition techniques, since they are not characterised by pattern-vectors. Instead, pairwise rather than central clustering techniques must be used. Alternatively, the objects can be embedded into a vector-space using techniques such as multidimensional scaling[2] or IsoMap[18]. Once embedded in such a space then the objects can be characterised by their embedding co-ordinate vectors, and analysed in a conventional manner.

Most embedding methods produce an embedding that is Euclidean. However, dissimilarity data cannot always be embedded exactly into a Euclidean space. This is the case when the similarity matrix (the equivalent of a kernel matrix) contains negative eigenvalues, and where the embedding (which depends on the square-root of the eigenvalue matrix) is non-real. Examples of such dissimilarity data occur in a number of data sources furnished by applications in computer vision. For instance, shape-similarity measures and graph-similarity measures are rarely Euclidean. Previous work[14] has shown that there is potentially useful information in the non-Euclidean part of the dissimilarities. Such data can be embedded in a pseudo-Euclidean space, i.e. one where certain dimensions are characterised by negative eigenvalues and the squared-distance between objects has positive and negative components which sum together together to give the total distance. A pseudo-Euclidean space is however non-metric which makes it difficult to correctly compute geometric properties. Another alternative, which we explore in this paper, is to embed the data on a Riemannian manifold, which is metric but non-Euclidean.

A third alternative, which we explore here, is to use a non-Euclidean, but metric, embedding space. A Riemannian manifold is curved, and the geodesic distances are metric. However they can also be indefinite and so can represent indefinite dissimilarities in a natural way. In this paper, we explore the embedding of objects onto the hypersphere with its associated spherical geometry. We aim to embed the available data on a hypersphere whose radius of curvature is determined by the dissimilarity data. The hypersphere can be either of positive curvature (i.e. an elliptic surface) or of negative curvature (i.e. a hyperbolic surface). We show how to approximate a distribution of dissimilarity data by a suitable hypersphere. Our analysis commences by defining the embedding in terms of a co-ordinate matrix that minimises the Frobenius norm with a similarity matrix. We show how the curvature of the embedding hypersphere is related to the eigenvalues of this matrix. In the case of an elliptic embedding, the radius of curvature is given by an optimisation problem on the smallest eigenvalues of a similarity matrix, while in the case of a hyperbolic embedding it is dependent on the second-smallest eigenvalue. Under the embedding, the geodesic distances between points are metric but non-Euclidean. Once embedded, we can characterise the objects using a revised dissimilarity matrix based on geodesic distances on the hypersphere. We apply our method to a variety of data including shape-similarities, graph comparison and gesture interpretation data. In each case the embedding maintains the local structure of the data while placing the points in a metric space.

2 Related Work

Multidimensional scaling (MDS) has its roots in Pyschometrics and has a long history. Initially, the goal was to analyze perceptual similarities in order to visualize the results of pyschological experiments. In classical MDS, the embedding space is generally Euclidean and an embedding space is sought into two or three dimensional for visualization purposes. However, it was soon realized that some types of data do not seem to lie naturally on a Euclidean manifold; the perceptual similarities of color and musical notes are good examples. Rather, these seem to lie on curves or circles in the embedding space.

Motivated by this observation, there are a number of works which look at the problem of embedding dissimilarities onto other manifolds, most typically circles or spheres. For example, Hubert et al have investigated the use of unidimensional embeddings on circles[8]. In particular, the problem of mapping distances onto the sphere S^2 has received particular attention since it has a number of applications such as the embedding of points on the surface of the Earth, and texture mapping spheroid objects[6]. Cox and Cox[3] were one of the first to look in detail at the problem of spherical embedding. They employ the Krustral stress[9] of the point configuration and use spherical-polar coordinates for the points. The stress can then be optimized with respect to the inclination and azimuth angles of the points. Similarly, Elad et al[6] use a stress measure which is then optimized with respect to the spherical polar coordinates of the points. These methods are effective and specifically designed for the two-dimensional sphere S^2. However, they do not easily extend to spheres of higher dimension.

These methods all follow a pattern which is typically of approaches to non-Euclidean MDS. The key idea is to define a measure of the quality of the embedding, called the stress, and then optimize the position of the points to minimize the stress. This is a very general approach which can be used to embed into all kinds of manifold. The optimization is an important step; here the SMACOF algorithm has proved very popular[4, 5].

The possibility of embedding onto higher dimensional spheres (elliptic space) has been explored by Lindman and Caelli in the context of interpreting psychological data[11]. As with other methods, their method involves optimizing a stress which is an extension of the original MDS method of Torgerson[19]. Interestingly, Lindman and Caelli note that the mathematics of hyperbolic space is very similar to that of elliptic space, and propose a method for embedding into hyperbolic space as well. This suggests that hyperbolic space may also be a viable alternative for representing dissimilarity-based data, although problems may arise from the different topologies - for example elliptic space is closed, whereas hyperbolic space is not. Hyperbolic embeddings have also been explored by Shavitt and Tankel, who have used the hyperbolic embedding as a model of internet connectivity[17]. In other work, Robles-Kelly and Hancock[15] show how to preprocess the available dissimilarity data so that it conforms either to elliptic or hyperbolic geometry. In practice the former corresponds to a scaling of the distance using a sine function, and the latter scaling the data using a hyperbolic sine function.

In this paper, we propose a number of novel extensions to address the problem of embedding into elliptic and hyperbolic spaces. Firstly, we show how to find and appropriate radius of curvature for the manifold directly from the data. We then develop a method of embedding into these spaces which, in contrast to other approaches, is not based on optimization. Finally, we develop an optimization scheme to refine the results which is specifically tailored to the problem of constant-curvature embeddings and easily extends to any number of dimensions in elliptic or hyperbolic space.

3 Indefinite spaces

We begin with the assumption that we have measured a set of dissimilarities between all pairs of patterns in our problem. This is denoted by the matrix $\mathbf{D}$, where D_{ij} is dissimilarity between i and j. We can define an equivalent set of similarities by using the matrix of squared dissimilarities $\mathbf{D}'$, where $D'_{ij} = D^2_{ij}$. This is achieved by identifying the similarities as $-\frac{1}{2}\mathbf{D}'$ and centering the resulting matrix:

$$\mathbf{S} = -\frac{1}{2}(\mathbf{I} - \frac{1}{n}\mathbf{J})\mathbf{D}'(\mathbf{I} - \frac{1}{n}\mathbf{J}) \tag{1}$$

Here $\mathbf{J}$ is the matrix of all-ones, and n is the number of objects. In Euclidean space, this procedure gives exactly the inner-product or kernel matrix for the points.

If $\mathbf{S}$ is positive semi-definite, then the original dissimilarities are Euclidean and we can use the kernel embedding to find positions $\mathbf{x}_i$ for the points in Euclidean space; $\mathbf{X} = \mathbf{U}_S\mathbf{\Lambda}_S^{\frac{1}{2}}$ where $\mathbf{U}_S$ and $\mathbf{\Lambda}_S$ are the eigenvector and eigenvalue matrices of $\mathbf{S}$, respectively. The position-vector $\mathbf{x}_i$ of the i^{th} point corresponds to the i^{th} row of $\mathbf{X}$. In this case, the relationship between the squared distance and the kernel is

$$D'_{ij} = S_{ii} + S_{jj} - 2S_{ij} \tag{2}$$

If $\mathbf{S}$ is indefinite, which is often the case, then the objects cannot exist in Euclidean space with the given dissimilarities, and $\mathbf{S}$ is not a kernel. This does not necessarily mean the the dissimilarities are non-metric; metricity is a separate issue. One measure of the deviation from definiteness which has proved useful is the negative eigenfraction (NEF) which measures the fractional weight of eigenvalues which are negative:

$$\text{NEF} = \frac{\sum_{\lambda_i<0}|\lambda_i|}{\sum_i|\lambda_i|} \tag{3}$$

If NEF=0, then the data is Euclidean. We can measure the *non-metricity* of the data by counting the number of violations of metric properties. It is very rare to have an initial distance measure which gives negative distance, so we will assume than the dissimilarities are all positive. The two measures of interest are then the fraction of triples which violate the triangle inequality (TV) and the degree of asymmetry of the dissimilarities. The methods applied in this paper assume symmetry - some of the data we have studied shows mild asymmetry which is corrected before processing. We give figures in the experimental section for triangle violations.

One way to treat such non-Euclidean data is to correct it before embedding. An example of this is to disregard the negative eigenvalues present in $\mathbf{S}$. We then obtain

$$\mathbf{S}_+ = \mathbf{U}_S \mathbf{\Lambda}_+ \mathbf{U}_S^T \tag{4}$$

Now $\mathbf{S}_+$ is a kernel matrix and we can find the embedding in the standard way. We refer to this as the *kernel embedding* of $\mathbf{S}$ to highlight its derivation from the kernel matrix, but essentially this is identical to classical multidimensional scaling.

Another alternative is to embed the non-Euclidean dissimilarity data can be embedded in a non-Riemannian pseudo-Euclidean space[13, 7]. This space uses the non-Euclidean inner product

$$< \mathbf{x}, \mathbf{y} >= \mathbf{x}^T \mathbf{M} \mathbf{y},\ \mathbf{M} = \begin{pmatrix} \mathbf{I}_p & 0 \\ 0 & -\mathbf{I}_q \end{pmatrix} \tag{5}$$

The values of -1 correspond to the 'negative' part of the space. The space has a signature (p, q) with p positive dimensions and q negative dimensions. 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 similarity as

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

where the negative part of the space corresponds to the negative eigenvalues of $\mathbf{S}$, and the pseudo-Euclidean embedding is

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

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 squared-distances to each other. Locality is not preserved in such a space, and geometric constructions such as lines are difficult to define in a consistent way. The space is more general than needed to represent the given dissimilarities (as it allows negative squared-distances) and the projection of new objects is ill defined. In order to overcome these problems, we would like to embed the points in a space with a metric distance measure which produces indefinite similarity matrices; this means that the space must be curved.

4 Riemannian Manifolds

In this paper, we use Riemannian manifolds to embed a set of objects. On the manifold, distances are measured by geodesics (the shortest curve between points), so we will employ manifolds where the geodesics are easy to compute. The manifold must also be curved in order to produce an indefinite similarity matrix. Two prime candidates for the embedding are the *elliptic* manifold and the *hyperbolic* manifold.

A manifold embedding is important because it allows the use of geometric and statistical tools. On a Riemannian manifold, distances are defined between any pair of points in the manifold in a consistent way (not just between the sample data-points). We can define geodesics as the equivalent of straight lines, and find the distance between points and lines. We can also compute statistics such as the mean in a consistent way. This means that all the standard classifiers can be applied, at least in theory, to the data; the exact formulation will however be different.

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

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

The metric tensor must be positive definite, and any metric tensor defines a particular Riemannian space. However, this definition is not always the most convenient - alternatively we can define a manifold as a subspace embedded in a larger 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 a 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. Two examples are furnished by the elliptic and hyperbolic manifolds.

5 Geometry of Constant-curvature manifolds

5.1 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:

$$\mathbf{x} = (r \sin u \sin v, r \cos u \sin v, r \cos v)^T \tag{10}$$

This embedding implies a particular metric tensor:

$$\begin{aligned} ds^2 &= dx^2 + dy^2 + dz^2 \\ &= r^2 \sin^2 v du^2 + r^2 dv^2 \end{aligned} \tag{11}$$

The embedding of an $(n-1)$-dimensional sphere in n-dimensional space is a straightforward 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 is curved and 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

$$d_{ij} = r \cos^{-1}(\langle \mathbf{x}_i, \mathbf{x}_j \rangle / r^2) \tag{15}$$

The elliptic space is metric but clearly not Euclidean. It is therefore a good candidate for representing points which produce indefinite kernels. The first question we wish to answer is, to what extent do the points in a curved space produce indefinite similarities? To answer this question, we have constructed the similarity matrices of points in these spaces. The points are generated via a parameterisation (Eqn 10) and drawing points from the parameters via a normal distribution. The indefinite nature of the similarity can be characterised by the negative eigenfraction (NEF) (Eqn. 3) Figure 1 shows the NEF for points on a unit hypersphere with varying standard deviation. The curved manifold produces significant negative eigenfraction, up to 24%.

5.2 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* and has a parameterisation and metric tensor given by

$$\begin{aligned} \mathbf{x} &= (r \sin u \sinh v, r \cos u \sinh v, r \cosh v)^T \\ ds^2 &= dx^2 + dy^2 - dz^2 \\ &= r^2 \sinh^2 v du^2 + r^2 dv \end{aligned} \tag{17}$$

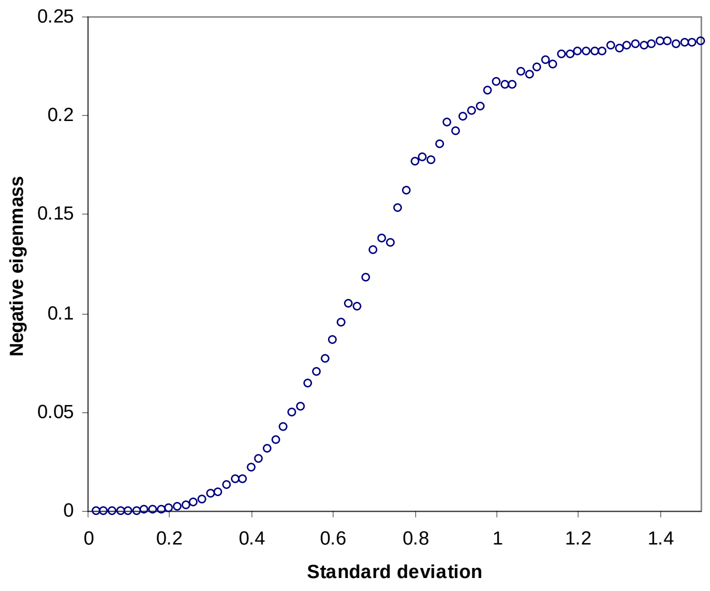


Figure 1: Negative eigenfraction of points on an elliptic surface

The metric tensor is positive definite, and so the surface is Riemannian and distances measured on the surface are metric, even though the embedding space is non-Riemannian.

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.

$$\sum_i x_i^2 - z^2 = -r^2 \tag{18}$$

It can be shown that $ds^2 > 0$ and so the hyperbolic surface is Riemannian. If there is more than one negative dimension, the surface is no longer Riemannian as it is possible to obtain $ds^2 < 0$. The hyperbolic space is therefore restricted to any number of positive dimensions but just 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

$$\begin{aligned} \langle \mathbf{x}_i, \mathbf{x}_j \rangle &= \sum_k x_{ik}x_{jk} - z_i z_j & (19) \\ &= -|\mathbf{x}_i||\mathbf{x}_j| \cosh \theta_{ij} & (20) \end{aligned}$$

which in turn defines the notion of hyperbolic angle. From this angle, the distance between two points in the space is $d_{ij} = r\theta_{ij}$. 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 $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = -r^2 \cosh \theta_{ij}$ we can also write

$$d_{ij} = r \cosh^{-1}(- \langle \mathbf{x}_i, \mathbf{x}_j \rangle / r^2) \tag{21}$$

6 Embedding

6.1 Embedding in Elliptic space

Given a distance matrix 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 $n-1$-dimensional 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

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle = r^2 \cos\theta_{ij} = r^2 \cos(d_{ij}/r) \tag{22}$$

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

$$\mathbf{X}\mathbf{X}^T = \mathbf{Z} \tag{23}$$

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[16]. 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_1. By minimising the magnitude of this eigenvalue, we can find the optimal radius.

$$r^* = \arg\min_r |\lambda_1 [\mathbf{Z}(r)]| \tag{24}$$

In practice we locate the optimal radius via search. 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:

$$\mathbf{Z}(r^*) = \mathbf{U}_Z \mathbf{\Lambda}_Z \mathbf{U}_Z^T \tag{25}$$

$$\mathbf{X} = \mathbf{U}_Z \mathbf{\Lambda}_Z^{\frac{1}{2}} \tag{26}$$

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_1 will be less than zero, and there will be residual negative eigenvalues. The embedding then is onto a 'hypersphere' of radius r, but embedded in a pseudo-Euclidean space. In order to obtain points on the hypersphere, we must correct the recovered points. The traditional method in kernel embedding is to discard the negative eigenvalues; here that will not suffice as this will change the length of the vectors and constraint 12 will be violated. In the next section we present a solution to this problem.

6.2 Approximation of the points

For a general set of dissimilarities, the points do not lie on a hypersphere, and need correction to lie on the manifold. The normal correction for a kernel embedding is to drop the negative eigenvalues and corresponding dimensions. We show that this process is justified for the spherical embedding in the next section, for a large radius of curvature. For more curved spaces, we propose a different approximation.

6.2.1 Limits of large radius

When the radius of curvature is large, clearly the space is nearly flat, and we might hope to recover the standard kernel embedding of the data. In fact we can write

$$\mathbf{Z} = r^2 \cos(\mathbf{D}/r) \simeq r^2\mathbf{J} - \frac{1}{2}\mathbf{D}' \ \ (r >> 1) \tag{27}$$

The squared distance matrix $\mathbf{D}'$ is related to the kernel matrix by

$$\mathbf{D}' = 2\mathbf{K}_l - 2\mathbf{K} \tag{28}$$

where $K_{lij} = (K_{ii} + K_{jj})/2$ is constructed from the diagonal elements of the kernel (Eqn 2), giving

$$\mathbf{Z} \simeq r^2\mathbf{J} + \mathbf{K} - \mathbf{K}_l \tag{29}$$

Since $\mathbf{K}$ is small compared to $r^2\mathbf{J}$ we can use degenerate eigenperturbation theory to show that $\mathbf{Z}$ and $\mathbf{K}$ have the same eigenvectors and eigenvalues. The exception is the leading eigenvalue of $\mathbf{Z}$, which is $\lambda_0 = nr^2 - \mathrm{Tr}(\mathbf{K})$, but zero for $\mathbf{K}$. As a result, we recover the kernel embedding for large r. This motivates us to use standard approach of neglecting any negative eigenvalues for embeddings with large radius.

6.2.2 Small radius approximation

We pose the problem as follows: The task is to find a point-position matrix $\mathbf{X}$ on the elliptic manifold which minimises the Frobenius distance to the Euclidean-equivalent matrix $\mathbf{Z}$. Given the manifold radius r, determined by the method in the previous section, we begin with the normalised matrix $\hat{\mathbf{Z}} = \mathbf{Z}/r^2$. The problem is then

$$\begin{aligned} \min_{\mathbf{X}} \quad & |\mathbf{X}\mathbf{X}^T - \hat{\mathbf{Z}}| \\ \mathbf{x}_i^T\mathbf{x}_i \ = \ & 1 \end{aligned} \tag{30}$$

This can be simplified by observing in the usual way that the Frobenius norm is invariant under an orthogonal similarity transform, so given $\hat{\mathbf{Z}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, we apply $\mathbf{U}$ as an orthogonal similarity transform to get

$$\min_{\mathbf{X}} |\mathbf{U}^T\mathbf{X}\mathbf{X}^T\mathbf{U} - \mathbf{\Lambda}| \tag{31}$$

which has a solution $\mathbf{X} = \mathbf{U}\mathbf{B}$ where $\mathbf{B}$ is some diagonal matrix, giving

$$\min_{\mathbf{B}} |\mathbf{B}^2 - \mathbf{\Lambda}| \tag{32}$$

Of course, $\mathbf{B}^2 = \mathbf{\Lambda}$ is an exact solution if all the eigenvalues are non-negative, and this is the case if the points lie precisely on a hypersphere. In the general case, there will be negative eigenvalues and we must find a minimum of the constrained optimisation problem. In the constrained setting, we are no longer guaranteed that $\mathbf{B}$ should be a diagonal matrix. Nevertheless, we make the simplifying approximation that we can find a diagonal matrix which is close to the optimal solution. This will be true if the points lie close to a hypersphere.

Let $\mathbf{b}$ be the vector of squared diagonal elements of $\mathbf{B}$, i.e. $b_i = B_{ii}^2$, $\boldsymbol{\lambda}$ be the vector of eigenvalues and U_s be the matrix of squared elements of $\mathbf{U}$, $U_{sij} = U_{ij}^2$. Then we can write the problem as

$$\begin{aligned} \min_{\mathbf{b}} \quad & (\mathbf{b} - \boldsymbol{\lambda})^T(\mathbf{b} - \boldsymbol{\lambda}) \\ b_i \quad &> \quad 0 \\ \mathbf{U}_s\mathbf{b} \quad &= \quad \mathbf{1} \end{aligned} \tag{33}$$

While this is a quadratic problem, and can be solved by quadratic programming, the solution actual has a simple form which can be found by noting that the matrix $\mathbf{U}_s$ should have rank $n-1$ and hence one singular value equal to zero. First we make the following observations: The vector of eigenvalues $\boldsymbol{\lambda}$ is an absolute minimiser of this problem, i.e. $\mathbf{b} = \boldsymbol{\lambda}$ minimises the Frobenius norm and satisfies constraint 2, but not constraint 1. Secondly, $\mathbf{b} = \mathbf{1}$ satisfies both constraints since $\sum_i U_{ij}^2 = 1$ (as $\mathbf{U}$ is orthogonal). These observations, and the fact that $\mathbf{U}_s$ is rank $n-1$, means that the general solution to the second constraint is

$$\mathbf{b} = \mathbf{1} + \alpha(\boldsymbol{\lambda} - \mathbf{1}) \tag{34}$$

It only remains then to find the value of α which satisfies the first constraint and minimises the criterion. Since the criterion is quadratic, the solution is simply given by the largest value of α for which the first constraint is satisfied. Given the optimal value α^* we can find $\mathbf{b}^*$ and

$$\mathbf{X}^* = \mathbf{U}\mathbf{B}^* \tag{35}$$

6.3 Embedding in Hyperbolic space

In hyperbolic space, we have

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle = -r^2 \cosh \theta_{ij} = -r^2 \cosh(d_{ij}/r) \tag{36}$$

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

$$\mathbf{X}\mathbf{M}\mathbf{X}^T = \mathbf{Z} \tag{37}$$

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:

$$r^* = \arg\min_r |\lambda_2 [\mathbf{Z}(r)]| \tag{38}$$

The embedded positions become

$$\mathbf{X} = \mathbf{U}_Z |\mathbf{\Lambda}_Z|^{\frac{1}{2}} \tag{39}$$

As with the elliptic embedding, in general the points do not lie on the embedding space and there will be residual negative eigenvalues.

6.4 Approximation of the points

A similar procedure may applied for hyperbolic space as for the elliptic space, but the optimisation problem is modified by the indefinite inner product. As with the elliptic embedding, we may drop residual negative eigenvalues for large r. For small radius, the equivalent analysis is as follows. Here $\mathbf{M}$ is defined by Eqn. 5, with $q = 1$.

$$\begin{aligned} \min_{\mathbf{X}} & \quad |\mathbf{X}\mathbf{M}\mathbf{X}^T - \hat{\mathbf{Z}}| \\ \mathbf{x}_i^T \mathbf{M} \mathbf{x}_i & = \; -1 \end{aligned} \tag{40}$$

As before, we apply the orthogonal similarity transform given by $\mathbf{U}$, where $\hat{\mathbf{Z}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$:

$$\min_{\mathbf{X}} |\mathbf{U}^T \mathbf{X}\mathbf{M}\mathbf{X}^T \mathbf{U} - \mathbf{\Lambda}| \tag{41}$$

which has a solution $\mathbf{X} = \mathbf{UB}$ where $\mathbf{D}$ is some diagonal matrix, giving

$$\min_{\mathbf{B}} |\mathbf{BMB} - \mathbf{\Lambda}| \tag{42}$$

Now we have a vector of diagonal elements given by $b_i = (\mathbf{BMB})_{ii}$. Exactly one of the b_i's must be negative (the one corresponding to the most negative element of $\boldsymbol{\lambda}$). Let b_n be the component of $\mathbf{b}$ corresponding to the negative dimension. We then have, as before

$$\begin{aligned} \min_{\mathbf{b}} & \quad (\mathbf{b} - \boldsymbol{\lambda})^T(\mathbf{b} - \boldsymbol{\lambda}) \\ b_n & < 0 \\ b_i & > 0, i \neq n \\ \mathbf{U}_s\mathbf{b} & = -\mathbf{1} \end{aligned} \tag{43}$$

Now we have a global minimiser of $\mathbf{b} = \boldsymbol{\lambda}$ which satisfies the final constraint and a second solution of the constraint is given by $\mathbf{b} = -\mathbf{1}$. We must therefore find the optimal value for α in

$$\mathbf{b} = -\mathbf{1} + \alpha(\boldsymbol{\lambda} + \mathbf{1}) \tag{44}$$

The solution is more complicated than in the elliptical case, due to the constraint $d_n < 0$. This means that it is possible that there is no solution. If a solution exists, the optimal point will lie on one of the two boundaries of the feasible region. Given the optimal solution of α^*, we get $\mathbf{b}^*$ and $\mathbf{X}^* = \mathbf{UB}^*$.

If there is not a solution, essentially this means that we cannot find a set of eigenvalues for the inner-product matrix $\mathbf{Z}^*$ which both satisfy both the conditions that only one is negative and that have unit length. We must abandon one of these properties - in this case we return to our standard procedure of neglecting negative eigenvalues.

7 Optimisation

The methods above provide the correct embeddings when the points lie exactly on the surface of a constant-curvature manifold, and a good approximation for points nearly on the manifold. Although the embeddings become unsatisfactory for larger approximations, they still provide a good initialisation for optimisation-based approaches. In this section, we develop an optimisation based on the properties of the manifold.

7.1 The Exponential Map

Non-Euclidean geometry can involve detailed calculations, and many problems are intractable on general Riemannian manifolds. However, by chosing a simple non-Euclidean manifold such as the hypersphere, we can hope to solve some problems such as embedding and classification. To do so, we require one important tool of Riemannian geometry, which is the exponential map.

The exponential map is a map from points on the manifold to points on a tangent space of the manifold. The map has an origin, which defines the point at which we construct the tangent space of the manifold. The formal definition of the Exponential map is the map which connects the Lie algebra on the tangent space to the Lie group which defines the manifold. We will not concern ourselves with the technical details here, but the map has an important property which simplifies geometric computations; the geodesic distance between the origin of the map and a point on the manifold is the same as the Euclidean distance between the images of the two points on the tangent space. As the tangent space is a Euclidean space, we can compute various geometric and statistical quantities in the tangent space in the standard way. Formally, the definition of these properties as follows: Let T_M be the tangent space at some point M on the manifold, P be a point on the manifold and X a point on the tangent space. We have

$$X = \mathrm{Log}_M P \tag{45}$$

$$P = \mathrm{Exp}_M X \tag{46}$$

$$d_g(P, M) = d_e(X, M) \tag{47}$$

The Log and Exp notation defines a log-map from the manifold to the tangent space and an exp-map from the tangent space to the manifold. This is a formal notation and does not imply the normal $\log$ and $\exp$ functions - although they do co-incide for some types of data, they are not the same for the spherical space. M is the origin of the map and is mapped onto the origin of the tangent space. The distance $d_g(.,.)$ is the geodesic distance on the manifold and $d_e(.,.)$ the Euclidean distance on the tangent space.

For the elliptic manifold, the exponential map is as follows. We define a point P on our manifold as a position vector $\mathbf{p}$ with fixed length $|\mathbf{p}| = r$ (the origin is at the centre of the hypersphere). Similarly, the point M is represented by the vector

$\mathbf{m}$, and M is the origin of the map. The maps are then

$$\begin{aligned}
\mathbf{x} &= \frac{\theta}{\sin\theta}(\mathbf{p} - \mathbf{m}\cos\theta) && (48)\\
\mathbf{p} &= \mathbf{m}\cos\theta + \frac{\sin\theta}{\theta}\mathbf{x} && (49)\\
d_g(P, M) &= r\theta && (50)\\
d_e(X, M) &= |\mathbf{x}| && (51)
\end{aligned}$$

where $\theta = \cos^{-1}\frac{<\mathbf{p},\mathbf{m}>}{r^2}$. The vector $\mathbf{x}$ is the image of P in the tangent space, and the image of M is at the origin of the tangent space.

For the hyperbolic manifold, the exponential map simply becomes

$$\begin{aligned}
\mathbf{x} &= \frac{\theta}{\sinh\theta}(\mathbf{p} - \mathbf{m}\cosh\theta) && (52)\\
\mathbf{p} &= \mathbf{m}\cosh\theta + \frac{\sinh\theta}{\theta}\mathbf{x} && (53)\\
& && (54)
\end{aligned}$$

where $\theta = \cosh^{-1}\frac{<\mathbf{p},\mathbf{m}>}{r^2}$. Lengths and inner-products are calculated in the pseudo-Euclidean space (section 5.2).

7.2 Elliptic Optimisation

Given a dissimilarity matrix $\mathbf{D}$, we want to find the embedding of a set of points on the surface of a hypersphere of radius r, such that the geodesic distances are as similar as possible to $\mathbf{D}$. Unfortunately, this is a hard problem and requires an approximate optimisation-based approach. We simplify the problem by considering just the distances to a single point at a time. Let the point of interest be $\mathbf{p}_i$; we then want to find a new position for this point on the hypersphere such that the geodesic distance to point j is d^*_{ij} where $*$ denotes that this is the target distance. We formulate the estimation of position as a least-squares problem which minimises

$$E = \sum_{j \neq i}(d_{ij}^2 - d_{ij}^{*2})^2 \qquad (55)$$

where d_{ij} is the actual distance between the points. Direct optimisation on the sphere is complicated by the need to restrict points to the manifold. However, as we are considering a single point at a time, we can construct a linear embedding using the log-map and optimise in the Euclidean space. If the current point-positions on the hypersphere are $\mathbf{p}_j, \forall j$, we can use the log-map to obtain point-positions $\mathbf{x}_j$ for each object j in the tangent space as follows:

$$\mathbf{x}_j = \frac{\theta_{ij}}{\sin\theta_{ij}}(\mathbf{p}_j - \mathbf{p}_i \cos\theta_{ij}) \tag{56}$$

with $\mathbf{x}_i = 0$.

We have found standard optimisation schemes to be infeasible on larger datasets, so here we propose a gradient descent scheme with optimal step-size determined by line-search. In this iterative scheme, we update the position of the point $\mathbf{x}_i$ in the tangent space to obtain a better fit to the given distances. At iteration k, the point is at position $\mathbf{x}_i^{(k)}$. Initially, the point is at the origin, so $\mathbf{x}_i^{(0)} = 0$. Since the points lie in tangent space, which is Euclidean, we then have

$$d_{ij}^2 = (\mathbf{x}_j - \mathbf{x}_i)^T(\mathbf{x}_j - \mathbf{x}_i) \tag{57}$$

The gradient of the error is

$$\nabla E = 4\sum_{j\neq i}(d_{ij}^2 - d_{ij}^{*2})(\mathbf{x}_i - \mathbf{x}_j) \tag{58}$$

and our iterative update procedure is

$$\mathbf{x}_i^{(k+1)} = \mathbf{x}_i^{(k)} + \eta\nabla E \tag{59}$$

Finally, we can determine the optimal step size as follows: let $\Delta_j = d_{ij}^2 - d_{ij}^{*2}$ and $\alpha_j = \nabla E^T(\mathbf{x}_i - \mathbf{x}_j)$, then the optimal step size is the smallest root of the cubic

$$n|\nabla E|^4\eta^3 + 3|\nabla E|^2(\sum_j \alpha_j)\eta^2 + (2\sum_j \alpha_j^2 + |\nabla E|^2\sum_j \Delta_j)\eta + \sum_j \alpha_j\Delta_j \tag{60}$$

After finding a new point position $\mathbf{x}_i$, we apply the exp-map to locate the new point position on the spherical manifold

$$\mathbf{p}_i' = \mathbf{p}_i\cos\theta + \frac{\sin\theta}{\theta}\mathbf{x}_i \tag{61}$$

7.3 Hyperbolic Optimisation

Optimisation on the hyperbolic manifold proceeds in a very similar way. However we need to use the hyperbolic exponential map

$$\mathbf{x}_j = \frac{\theta_{ij}}{\sinh \theta_{ij}}(\mathbf{p}_j - \mathbf{p}_i \cosh \theta_{ij}) \tag{62}$$

with $\mathbf{x}_i = 0.$ and bear in mind the inner product is modified by the pseudo-Euclidean embedding space. As a result, the squared distance is

$$d_{ij}^2 = (\mathbf{x}_j - \mathbf{x}_i)^T \mathbf{M}(\mathbf{x}_j - \mathbf{x}_i) \tag{63}$$

The gradient of the error is therefore

$$\nabla E = 4 \sum_{j \neq i} (d_{ij}^2 - d_{ij}^{*2}) \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j) \tag{64}$$

which gives $\alpha_j = \nabla E^T \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)$. Additionally, the squared length of the gradient is $|\nabla E|^2 = \nabla E^T \mathbf{M} \nabla E$. With these expressions, Equation 60 can be used without change to determine the optimal step size.

8 Experiments

We investigate the efficacy of constant curvature embeddings using a variety of datasets including synthetic and real distance. Since scaling distances by a constant factor does not alter the geometry of the points, we first rescale the distance matrix so that the mean distance between points is 1. By doing this, we ensure that radii and distance errors are directly comparable between datasets.

Our baseline comparison is with the kernel embedding (or classical MDS), which is exact for Euclidean distances. For non-Euclidean distances, this is given by kernalising the similarity matrix by eliminating negative eigenvalues:

$$\begin{aligned} \mathbf{S} &= \mathbf{U}_S \mathbf{\Lambda} \mathbf{U}_S^T \\ \mathbf{X}_K &= \mathbf{U}_S \mathbf{\Lambda}_+^{1/2} \end{aligned} \tag{65}$$

We use two measures of distortion of the embedded points. The first is the RMS difference between the distance matrix and the distances between the embedded points (in the embedding space). This measures the overall distortion introduced by the embedding. However, it is possible that there could be a local distortion which alters the local position of points close to each other, but which is small when measured over the whole space. Since local configuration is important to applications such as clustering and classification, we also measure the change in neighbourhood order. This is achieved by ordering the points by distance away from a central point and measuring the distortion in the ranked list using Spearman's rank correlation coefficient ρ. The structural error is $1 - \bar{\rho}$.

8.1 Texture mapping

As an initial evaluation and comparison to the literature, we begin with a set of texture mapping experiments, similar to those conducted by Elad et al[6]. We begin with a triangulated mesh describing the 3D surface of an object. We then compute the geodesic distance across the mesh[12]. These distances between the vertices form the starting point for our algorithm. We then 'unwrap' these geodesic distances onto a two dimensional surface; the sphere S^2 for the spherical embeddings and the plane R^2 for the kernel embedding. This embedding is used to map a texture back onto the surface of the object. The texture is defined on the surface of a sphere for the spherical embeddings and on the plane for kernel embedding. For visualization purposes only, we subdivide the mesh after embedding to enable us to view the texture in high resolution. Any distortions in the embedding are revealed as distortions in the surface texture map.

The first model is a simple test case of the sphere. Figure 2 shows the results of texture-mapping the surface with a triangular texture. These results are summarized in Table 1. Both spherical embedding methods produce near-perfect embeddings of the surface. Since our method is not based on optimization, it is considerably quicker that the method of Elad et al. However, these times are only indicative as neither algorithm was optimized for speed. While the kernel method is much quicker, there is naturally considerable distortion in mapping the sphere to a plane.

The second model is the Stanford bunny. This model is subsampled to 1016 vertices and 2000 faces. The embedding results are shown in Figure 3 and Table 2. Again, the spherical embedding methods produce very similar results. However, our method finds a radius of curvature of 0.78 which is considerably different from 1. Note the distortion of the texture around the ears for the kernel methods, which is not present in the other methods.

The methods were implemented in Matlab on a Intel Core2 Duo 3GHz machine

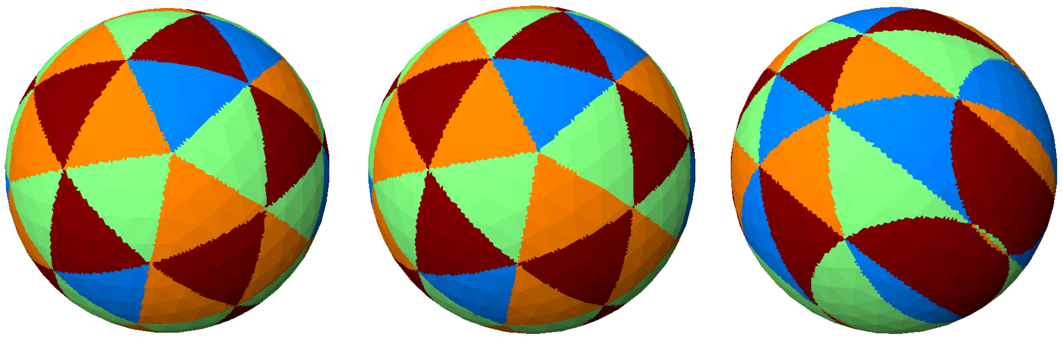


Figure 2: Texture mapping of the sphere. Left: using our method, middle: the method of Elad et al, and right: kernel embedding. Both spherical methods produce virtually perfect embeddings, whereas the expected distortion is evident in the kernel embedding.

	Points	Radius	RMS err	Struct. err	Time
Our method	642	1.00	0.0030	9×10^{-6}	7.2s
Elad et al		-	0.0029	3×10^{-5}	26.9s
Kernel		-	0.39	1×10^{-5}	0.8s

Table 1: The performance of embedding methods on the sphere texturing problem.

	Points	Radius	RMS err	Struct. err	Time
Our method	1016	0.78	0.21	0.11	31s
Elad et al		-	0.21	0.11	206s
Kernel		-	0.24	0.14	4.3s

Table 2: The performance of embedding methods on the Stanford bunny texturing problem.

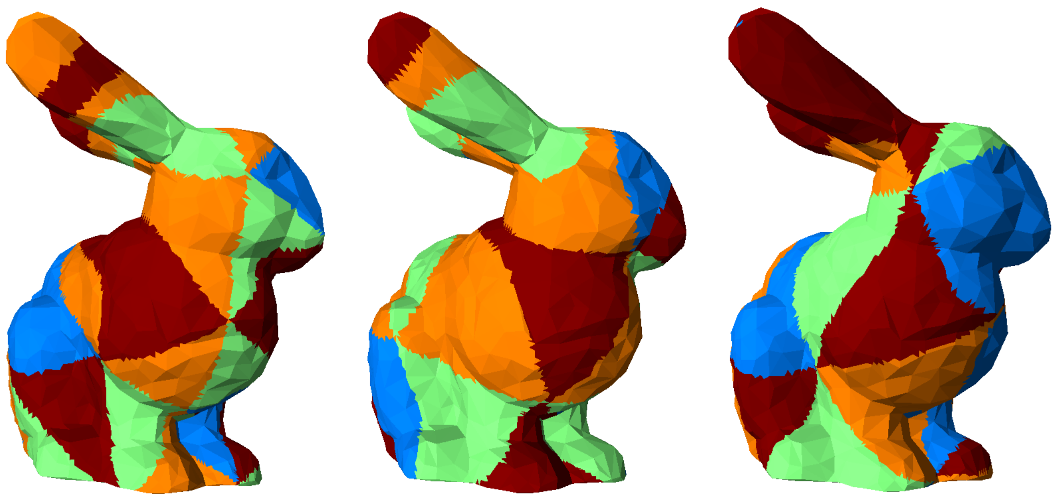


Figure 3: Texture mapping of the Stanford bunny. Left: using our method, middle: the method of Elad et al, and right: kernel embedding. The spherical embeddings produce similar results. Note the distortion around the ears in the kernel embedding.

8.2 Robustness to Noise

We now turn our attention to more challenging problems, where the embedding manifold is more than two-dimensional and noise and distortions are present. The methods based on spherical-polar coordinates are not applicable to this situation[6, 3] Although our method finds the embedding exactly when the points lie on the hypothesised surface, in realistic situations there is invariably noise in the measurements. In order to assess the performance of our methods on noisy data, we generate synthetic data with controlled levels of noise as follows. We begin by generating points on the surface of a sphere (or hyperboloid); in this experiment the sphere is embedded in 50-dimensional space and 50 points are generated. We then construct the distance matrix for the points using the geodesic distance on the sphere. These distances are then perturbed by Gaussian noise of varying amounts; we then process the resulting distance matrix to remove any negative distances and maintain symmetry of the distances. We finally apply our embedding methods to the noisy distances. The results are shown in Figures 8.2 for the elliptic space and 8.2 for the hyperbolic space. The errors are computed between the noisy distance matrix and its embedding (i.e. they are the errors cause by the embedding process only). For comparison, we include the difference between the noisy distance matrix and the original(with no noise).

It is clear from Figure 8.2 that the spherical embedding is effective even in the presence of large amounts of noise. At all noise levels, the distortion of the spherical embedding is less than that caused by the kernel embedding. The spherical embedding also shows a remarkable ability to preserve the neighborhood order (structural error). This is also apparent, although to a lesser extent, in real-world datasets (section 9. Optimization of the spherical embedding produces good embedding results but increases the structural error significantly.

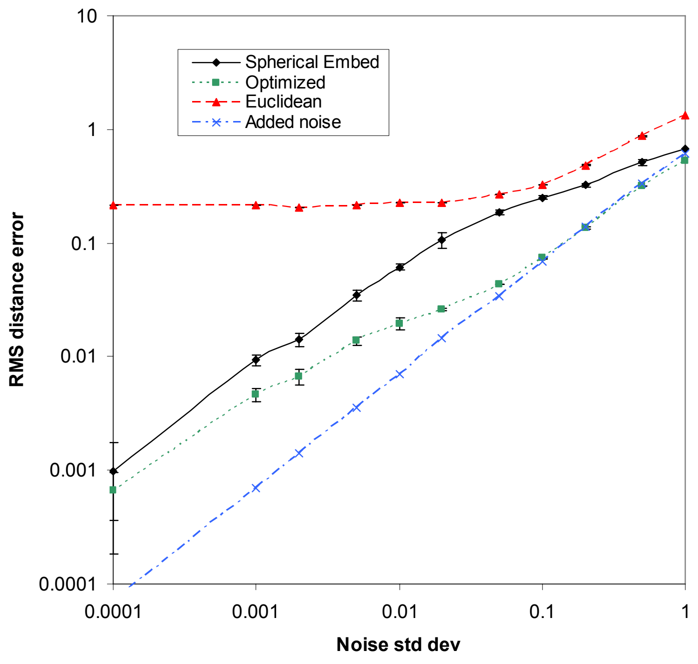
The hyperbolic embedding (Figure 8.2) is affected more significantly by noise - the performance is good for low noise levels but at moderate to high noise the errors are similar to that of the kernel embedding. Optimization does give a significant improvement in embedding accuracy. It appears to be more difficult to accurately locate the correct radius of curvature for the hyperbolic space in the presence of noise.

9 Similarity-based datasets

Finally, we use some data from a selection of similarity-based pattern recognition problems[14, 1, 20, 10]. These are a subset of the data analyzed under the SIMBAD project (simbad-fp7.eu), selected on the basis that they have small radius-of-curvature under the elliptic or hyperbolic model, and therefore are significantly non-Euclidean under those models. The data is based on classification problems, and so as well as showing measurements of distortion, we also calculate the nearest-neighbour classification error rate (using leave-one-out).

9.1 Elliptic embedding

For these non-Euclidean similarity-based datasets, the elliptic space offers an embedding which performs very well (Table 3. The optimized RMS error of the embedding is always less than the kernel embedding, and the original elliptic embedding is better in 3 of the 5 examples. The structural error is always superior for the elliptic embedding and is very low for DelftGestures, FlowCyto-1 and Chickenpieces-25-45 (as suggested by the synthetic data above). There is also an improvement in the classification accuracy, excepting the Catcortex data. In fact the Catcortex data appears different in a number of ways; it has a much higher radius of curvature and high structural error. We suspect that this dataset conforms to a different geometry than that of elliptic space.



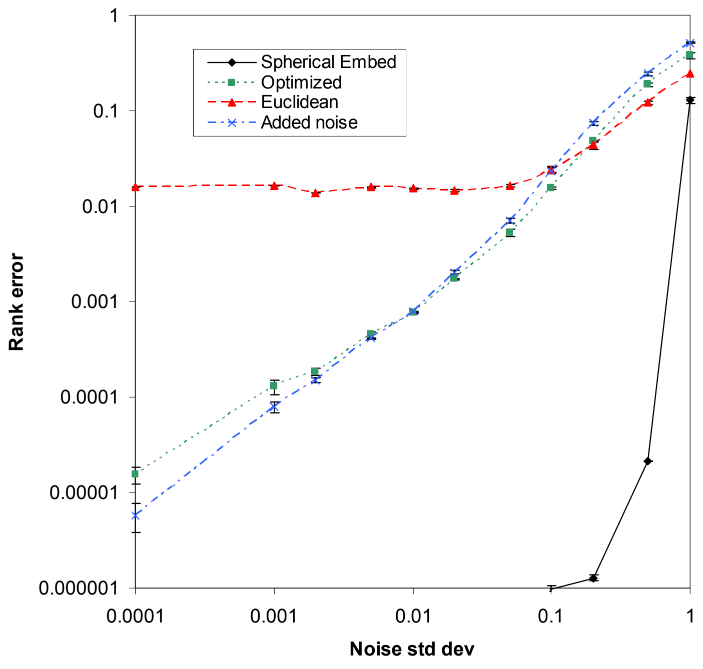
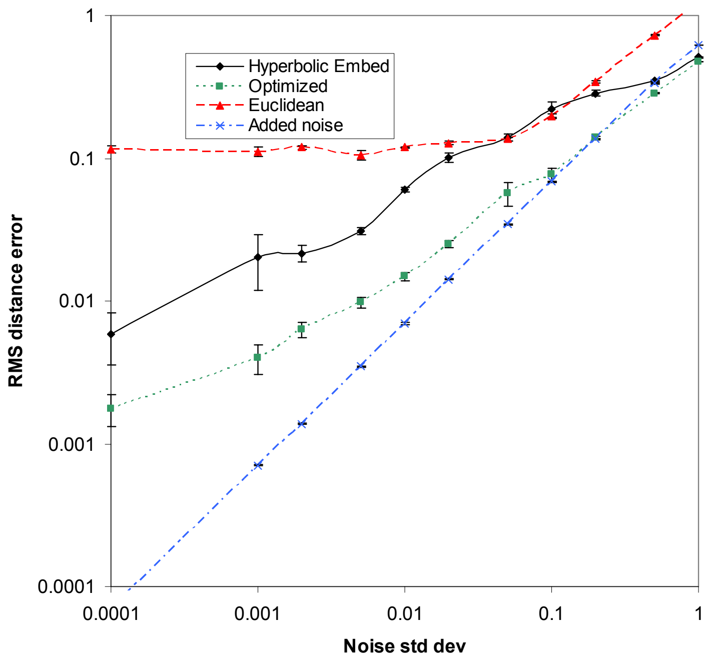


Figure 4: Reconstruction of noisy distances via a range of different embedding methods.



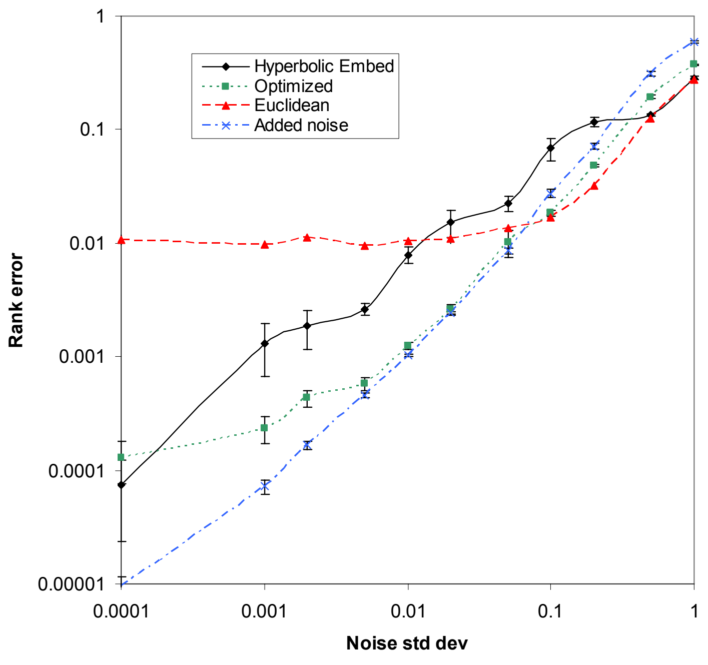


Figure 5: Reconstruction of noisy distances via a range of different embedding methods.

	Radius	RMS err	Struct. err	1NN
DelftGestures	0.908	0.508	7.29×10^{-11}	0.0407
optimized	0.719	0.0605	0.0308	0.226
kernel	–	0.410	0.0449	0.104
FlowCyto-1	0.923	0.572	3.38×10^{-9}	0.361
optimized	0.866	0.078	0.0317	0.418
kernel	–	0.329	0.0557	0.410
Chickenpieces-25-45	1.15	0.236	0.0170	0.0673
optimized	0.972	0.0545	0.030	0.0897
kernel	–	0.423	0.0375	0.141
Catcortex	0.687	0.162	0.162	0.138
optimized	0.662	0.0879	0.194	0.108
kernel	–	0.215	0.211	0.062
Zongker	1.73	0.990	0.236	0.194
optimized	1.09	0.208	0.195	0.048
kernel	–	1.81	0.403	0.589

Table 3: Elliptic embeddings of some similarity-based datasets, compared to the kernel embedding.

	Radius	RMS err	Struct. err	1NN
Newsgroups	1.85	0.0642	0.442	0.28
optimized	1.86	0.0312	0.335	0.263
kernel	–	0.170	0.418	0.273
CoilDelftSame	0.567	0.0331	0.00990	0.594
optimized	0.568	0.00649	0.00110	0.635
kernel	–	0.0199	0.00165	0.608

Table 4: Hyperbolic embeddings of two similarity-based datasets, compared to the kernel embedding.

9.2 Hyperbolic embedding

Again we analyzed our similarity-based datasets for hyperbolic-like examples by looking for those with significant curvatures under the hyperbolic model. However, such examples were rare. The two we found are detailed in Table 9.2. Here the results are not so clear-cut. The RMS errors and structural errors are improved after optimization, but the original embedding is not always better and the classification results are quite similar.

10 Conclusion

Spaces of constant-curvature offer a useful alternative for the embedding of non-Euclidean datasets. This allows intrinsically non-flat geometry between objects and may be a better description for many datasets.

In this paper we have presented efficient methods of embedding points into elliptic and hyperbolic spaces using their distance matrices. This simple method is based on the eigendecomposition of a similarity matrix, followed by a correction to project into constant-curvature space. We also developed an optimization procedure for improving the accuracy of the embedding for more difficult datasets which utilized the exponential map to transform the problem into an optimization in tangent space.

Our results on synthetic and real data show that the elliptic embedding performs well under noisy conditions and can deliver low-distortion embeddings for a wide variety of datasets. Hyperbolic-like data seems to be much less common (at least in our datasets) and is more difficult to accurately embed. Nevertheless, in low-noise cases and for some datasets, the hyperbolic space can also be used to accurately embed non-Euclidean dissimilarity data.

While accurate embedding is our goal here, it is natural to want to apply pattern recognition techniques to the embedded data. Unfortunately many methods rely, either explicitly or implicitly, on an underlying kernel space which is Euclidean. We believe that much more work needs to be done on applying such techniques in non-flat spaces.

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] M. A. A. Cox and T. F. Cox. Multidimensional scaling. In *Handbook of Data Visualization*, pages 315–347. Springer Handbooks of Computational Statistics, 2008.

[3] T. F. Cox and M. A. A. Cox. Multidimensional scaling on a sphere. *Communications in Statistics - Theory and Methods*, 20(9):2943–2953, 1991.

[4] J. De Leeuw. Applications of convex analysis to multidimensional scaling. In J. R. Barra, F. Brodeau, G. Romier, and B. van Cutsem, editors, *Recent Developments in Statistics*, pages 133–145, 1977.

[5] J. De Leeuw and W. J. Heiser. Multidimensional scaling with restrictions on the configuration. In P. R. Krishnaiah, editor, *Multivariate Analysis*, pages 501–522, 1980.

[6] A. Elad, Y. Keller, and R. Kimmel. Texture mapping via spherical multi-dimensional scaling. In *Scale-Space 2005*, volume 3459 of *LNCS*, pages 443–455, 2005.

[7] L. Goldfarb. A unified approach to pattern recognition. *Pattern Recognition*, 17:575–582, 1984.

[8] L. Hubert, P. Arabie, and J. Meulman. Linear and circular unidimensional scaling for symmetric proximity matrices. *British Journal of Mathematical & Statistical Psychology*, 50:253–284, 1997.

[9] J.B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[10] J. Lichtenauer, E. A. Hendriks, and M. J. T. Reinders. Sign language recognition by combining statistical DTW and independent classfication. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:2040–2046, 2008.

[11] H. Lindman and T. Caelli. Constant curvature Riemannian scaling. *Journal of Mathematical Psychology*, 17:89–109, 1978.

[12] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM Journal of Computing*, 16:647–668, 1987.

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

[14] 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.

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

[16] I.J. Schoenberg. On certain metric spaces arising from Euclidean spaces by a change of metric and their imbedding in hilbert space. *Annals of Mathematics*, 38:787–797, 1937.

[17] Y. Shavitt and T. Tankel. Hyperbolic embedding of internet graph for distance estimation and overlay construction. *IEEE/ACM Transactions on Networking*, 16:25–36, 2008.

[18] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[19] 1958. Torgerson, W.S. *Theory and methods of scaling.* Wiley, New York, 1958.

[20] B. Xiao and E. R. Hancock. Geometric characterisation of graphs. *ICIAP 2005*, LNCS 3617:471–478, 2005.

C Ricci Flow

Regularising the Ricci Flow Embedding

Weiping Xu, Edwin R. Hancock, and Richard C. Wilson

Dept. of Computer Science
University of York, UK
{elizaxu,erh,wilson}@cs.york.ac.uk

Abstract. This paper concerns the analysis of patterns that are specified in terms of non-Euclidean dissimilarity or proximity rather than ordinal values. In prior work we have reported a means of correcting or rectifying the similarities so that the non-Euclidean artifacts are minimized. This is achieved by representing the data using a graph, and evolving the manifold embedding of the graph using Ricci flow. Although the method provides encouraging results, it can prove to be unstable. In this paper we explore how this problem can be overcome using a graph regularisation technique. Specifically, by regularising the curvature of the manifold on which the graph is embedded, then we can improve both the stability and performance of the method. We demonstrate the utility of our method on the standard "Chicken pieces" dataset and show that we can transform the non-Euclidean distances into Euclidean space.

1 Introduction

Dissimilarity representations [1] provide a powerful and natural way of capturing the relationships between objects that are not characterised by ordinal measurements or feature vectors. The idea is to use a pairwise dissimilarity (or proximity) measure [2,3] to describe the properties of objects in terms of their attribute differences. Examples of such representations are provided by weighted proximity graphs. The advantages of such a representation are that if characterised in terms of a dissimilarity matrix, then pattern matching can be effected without the need for explicit alignment. However, the dissimilarities are quite frequently non-Euclidean and this prevents the use of many geometrically based learning techniques.

One way to overcome these problems is to represent the dissimilarity data using a weighted graph, and to embed the graph on a manifold. This produces a vectorial representation of the data by projecting dissimilarity data into a fixed-dimensional vector space. Examples of this approach include multidimensional scaling (MDS) [4], Isomap [5], locally linear embedding [6] and the Laplacian eigenmap embedding [7]. The common aim is to locate a low-dimensional representation. In order to apply non-Euclidean dissimilarity data with traditional geometric learning techniques, we must attempt to rectify the data so as to minimize the non-Euclidean artifacts. One route is to consider the positive definite subspace of the distances [8]. An alternative route adopted by Pekalska et al. [9] is to add a suitable constant to the squared off-diagonal elements of the dissimilarity matrix. It is equivalent to adding a certain constant to all eigenvalues of the related Gram matrix, and thus compensating for the effect of the negative eigenvalues, while maintaining the same eigenvector structure.

In prior work [10] we have shown how to correct the dissimilarity data, giving a set of new Euclidean distances. The method uses Ricci flow on a constant curvature Riemannian manifold to evolve the distance measures. This is effected by updating the curvatures on the edges of the graph representing the data. Unfortunately, the method can prove unstable due to local fluctuations in edge curvature. To overcome this problem, in this paper we show how to stabilise the method by regularising the curvatures of the embedded graph. To do this we use the heat kernel to smooth the curvatures on the edges. The result shows both improved numerical stability and lower classification error in the embedded space.

2 Embedding Non-Euclidean Data

In this paper we are concerned with embedded data represented in terms of pairwise dissimilarities or distances, and in particular the case where the data is non-Euclidean. Our overall aim is to rectify a given set of non-Euclidean dissimilarity data so as to make them more Euclidean. One way to gauge the degree to which a pairwise distance matrix contains non-Euclidean artefacts is to analyse the properties of its centralised Gram matrix. For an $N \times N$ symmetric pairwise dissimilarity matrix D with the pairwise distance as elements, the centered Gram matrix $G = -\frac{1}{2}JD^2J$,where D^2 is element-wise squaring of elements in D, $J = I - \frac{1}{N}11^T$ is the centering matrix and 1 is the all-ones vector of length N. The degree to which the distance matrix departs from being Euclidean can be measured by using the relative mass of negative eigenvalues or "negative eigenfraction" $F_{eigS} = \sum_{\lambda_i<0} |\lambda_i| / \sum_{i=1}^{N} |\lambda_i|$ [11]. This measure is zero for Euclidean distances and increases as the distance becomes increasingly non-Euclidean.

The kernel embedding is obtained from the centered Gram matrix using the factorisation $G = YY^T$, where Y is the $N \times N$ matrix with the embedded co-ordinates of the data as columns. To determine whether the Gram matrix is positive semi definite [11], we perform the eigendecomposition $G = \Phi\Lambda\Phi^T$ on the Gram matrix, where $\Lambda = diag(\lambda_1, ..., \lambda_N)$ is the diagonal matrix with the ordered eigenvalues as elements and $\Phi = (\phi_1|...|\phi_N)$ is the eigenvector matrix with the ordered eigenvectors ϕ_1, ..., ϕ_N as columns. In terms of the eigenvalues and eigenvectors, the matrix of embedded co-ordinates is given by $Y = \Phi\sqrt{\Lambda}$ where the eigenvalues Λ are positive. In Isomap embedding, the dimension and the number of nearest neighbors are estimated to be the optimal values by looking at the residue variances[5].

3 Ricci Flow

Our aim is to develop a method that can be used to rectify the non-Euclidean artefacts in such a dissimilarity matrix. The approach is as follows. Firstly, we consider the objects of interest to be represented by points on a manifold, and the given dissimilarities to be the geodesic distances on the manifold between these points (geodesic distances). For an arbitrary set of non-Euclidean similarities the manifold will be curved. By contrast, a Euclidean space will be flat and the geodesic and Euclidean distances will be identical. Our task is then to remove the curvature from the manifold to create a corrected set of Euclidean distances. We achieve this by evolving the manifold using Ricci flow.

The Ricci flow [12] evolves a manifold so that the rate of change of the metric tensor is controlled by the Ricci curvature. Essentially, this is an analogue of a diffusion process for a manifold. The geometric evolution equation is:

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

where g_{ij} is the metric tensor of the manifold and R_{ij} is the Ricci curvature.

We model the embedding manifold as consisting of a set of local patches with individual constant Ricci curvatures. These patches can be either elliptic (of positive sectional curvature) or hyperbolic (of negative sectional curvature). It is straightforward to re-express the Ricci flow in terms of the sectional curvature K:

$$\frac{dK}{dt} = \begin{cases} -2K^2 & \text{elliptic hypersphere,} \\ 2K^2 & \text{hyperbolic space.} \end{cases} \tag{2}$$

Under this evolution, the curvature moves towards zero for both types of patch, flattening the manifold. The solution of the differential equation is straightforward. Commencing with the initial conditions $K = K_0$ at time $t = 0$, then at time t we have

$$K = \frac{K_0}{1 \pm 2K_0 t} \tag{3}$$

with the positive sign for the elliptic space.

4 Curvature Computation

Our aim is to evolve a non-Euclidean dissimilarity measure into a Euclidean one using the Ricci flow described in the previous section. We commence by representing the dissimilarity data using a weighted graph $G = (V, E, D)$, where the node set V represents the set of objects and the edges E are weighted with the pairwise dissimilarities. We embed the graph onto a manifold so that the geodesic distance $d_g(u, v), (u, v) \in E$ between the positions of the nodes u and v is equal to the dissimilarity on the edges. Let $\boldsymbol{y}_u$ be the embedded co-ordinates of the node $u \in V$ and $Y = (\boldsymbol{y}_1 | ... | \boldsymbol{y}_{|V|})$ be the matrix with the embedded co-ordinates as columns. Under this embedding the edges acquire a curvature determined by the difference between geodesic distance (dissimilarity) $d_G(u, v)$ and Euclidean distance $d_E(u, v) = \sqrt{(\boldsymbol{y}_u - \boldsymbol{y}_v)^T (\boldsymbol{y}_u - \boldsymbol{y}_v)}$. The Ricci flow, modifies the Gaussian curvatures on the edges, so as to flatten the manifold. Adopted from [13] we use a Euclidean embedding of the points and use the difference between the geodesic distance d_G on the manifold (from the similarity or dissimilarity matrix) and the Euclidean distance in the embedded space d_E to compute the curvature. We compare experimental results for embeddings obtained with both Isomap [5] and the kernel embedding in Section 7. Lindman and Caelli [14] give the relationship between the two distances on elliptic, hyperbolic and Euclidean constant curvature manifolds as

$$d_E = \begin{cases} \frac{2}{K^{\frac{1}{2}}} \sin(\frac{K^{\frac{1}{2}}}{2} d_G) & \text{Elliptic,} \\ \frac{2}{|K|^{\frac{1}{2}}} \sinh(\frac{|K|^{\frac{1}{2}}}{2} d_G) & \text{Hyperbolic,} \\ d_G & \text{Euclidean.} \end{cases}$$

However, the adopted curvature approximations used only hold for small curvatures. In the data under study here, we find that the curvatures are too large for these approximations to hold. We therefore use it as the initialization and estimate curvature from Equation 4 using Newton's method. Taking the curvature in an elliptic space as an example, the Newton iteration is

$$K_{n+1}^{\frac{1}{2}} = K_n^{\frac{1}{2}} - \frac{K_n^{\frac{1}{2}} d_E - 2\sin\frac{K_n^{\frac{1}{2}}}{2} d_G}{d_E - d_G \cos\frac{K_n^{\frac{1}{2}}}{2} d_G} \tag{4}$$

Finally, we can compute new geodesic distances for the points based on the updated curvature. We keep the Euclidean distance between the points fixed, while updating the curvature.The updated geodesic distance under the new Gaussian curvature can be represented in terms of the old geodesic distance at the previous iteration. The update equation for the geodesic distance is

$$d_{G_{n+1}} = \begin{cases} \frac{2}{K_{n+1}^{\frac{1}{2}}} \arcsin\left(\frac{K_{n+1}^{\frac{1}{2}}}{K_n^{\frac{1}{2}}} \sin(\frac{K_n^{\frac{1}{2}}}{2} d_{G_n})\right) & \text{elliptic hypersphere} \\ \frac{2}{|K_{n+1}|^{\frac{1}{2}}} \operatorname{arcsinh}\left(\frac{|K_{n+1}|^{\frac{1}{2}}}{|K_n|^{\frac{1}{2}}} \sinh(\frac{|K_n|^{\frac{1}{2}}}{2} d_{G_n})\right) & \text{hyperbolic space} \end{cases} \tag{5}$$

This equation can be applied to each element of the dissimilarity matrix in turn.

5 The Algorithm

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_G. The following algorithm can be used to rectify the distance matrix from being non-Euclidean to Euclidean.

Begin with a pairwise distance matrix $D_G^{(0)}$,

1. Embed the objects in a Euclidean space using either Isomap or the kernel embedding. In the embedded space compute the Euclidean distances d_E.
2. From the geodesic distance d_G and Euclidean distance d_E, compute the constant curvature space with curvature K for a pair of objects using Equation 4.
3. Update the Gaussian curvature with a small time step using Equation 3.
4. Obtain the new geodesic distance $d_{G_{n+1}}$ from the previously available geodesic distance matrix together with the curvatures under a fixed Euclidean distance using Equation 5.
5. Obtain the updated distance matrix $D_G^{(1)}$ containing rectified geodesic distances between objects, and repeat from step 1 until D_G is Euclidean, that is its centered Gram matrix has no negative eigenvalues.

6 Regularizing Curvature

As posed above, the Ricci flow embedding updates the Gaussian curvature separately for each individual edge. This is because we use piecewise constant curvature manifolds for each edge. This places no constraint on the smoothness of the manifold, and this can lead to numerical instability in the embedding. Graph regularization provides a way to smooth data samples over a graph and overcome the numerical stability problems. One such regularization process is a graph diffusion. A diffusion process is analogous to the flow of heat, which flows from high to low concentrations, and over time creates a smooth distribution of heat. In a similar way, a diffusion of a function on the graph will create a smoother function. The diffusion is defined in terms of a random walk on the edges of the graph[15], and is represented by the diffusion (or heat) kernel:

$$H = \exp(-Lt) \tag{6}$$

The evolution of a function under this kernel is simply

$$f(t) = H(t)f(0) \tag{7}$$

The evolution is 'mass-preserving' in the sense that the sum of the values of the function over vertices is preserved.

We can use this process for smoothing curvatures before the application of the Ricci flow, to remove extreme values. However, our curvatures are defined pairs of objects and we therefore need to construct a graph which has vertices corresponding to object-pairs and edges describing a neighbourhood structure of these pairs. We construct this graph as follows. Firstly, we build the nearest-neighbours graph of the objects $G = \{V, E\}$. Each vertex represents an object u and an undirected edge E_{uv} exists if u is in the n nearest neighbours of v or v is in the n nearest neighbours of u. We then construct the dual of this graph $G_D = \{V_D, E_D\}$; each edge of the original graph becomes a vertex V_{uv} and an edge exist between two vertices if they share a common vertex from the original graph. In the dual graph, each vertex represents a pair of objects and the edges reflect the neighbourhood structure of the pairs. We can then define the curvature between object pairs as a function over the vertices of this graph and apply the diffusion kernel.

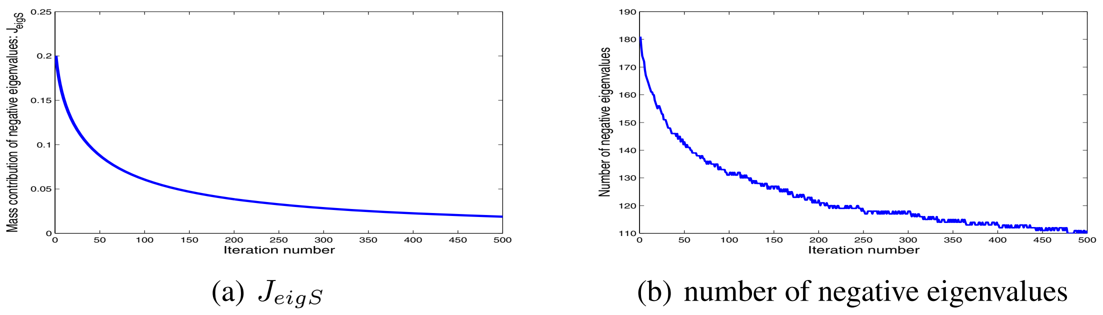
We therefore add an additional step in which we smooth the Gaussian curvatures over the dual of the nearest neighbor graph prior to performing the Ricci flow updating of the curvatures. All of the remaining steps of the algorithm remain as above. The following steps shows how to smooth Gaussian curvatures over the nearest neighbour edges.

Commence with initial Gaussian curvatures K from step 2 above,

1. Construct the n nearest neighbour graph over the available dissimilarity data. Node u and v are connected by an edge if u is among n nearest dissimilarity neighbors of v or v is among n nearest dissimilarity neighbors of u.
2. Construct the dual graph of the nearest neighbour graph. Each edge in the nearest neighbour graph is a vertex of the dual graph. If two edges in the nearest neighbour graph share a one common vertex, then the corresponding two vertices in the dual graph are connected by an edge.
3. Obtain the updated and regularised curvature K. Suppose that $\hat{L}$ is the normalized Laplacian of the dual nearest neighbour graph, then the heat-kernel of the dual graph is $\exp[-\hat{L}t]$. If V_D is the node-set of the dual graph, then we construct a

vector $\boldsymbol{K}$ of Gaussian curvatures $\boldsymbol{K} = (K_1, ..., K_{|V_D|})^T$. The vector of regularised Gaussian curvatures after heat kernel smoothing is $\boldsymbol{K}_{reg} = \exp[-\hat{L}t]\boldsymbol{K}$.

In summary, the above approach commences from a nearest neighbor graph over the dissimilarity matrix, and then constructs the dual graph where a node corresponds to an edge in the original graph. The heat kernel on the dual graph smooths the curvatures on the original nearest neighbour graph.



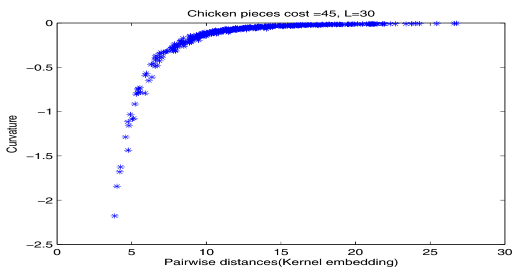
(a) J_{eigS} (b) number of negative eigenvalues

Fig. 1. (a) is the negative eigenfraction during iteration. (b) is the number of negative eigenvalues during iteration.

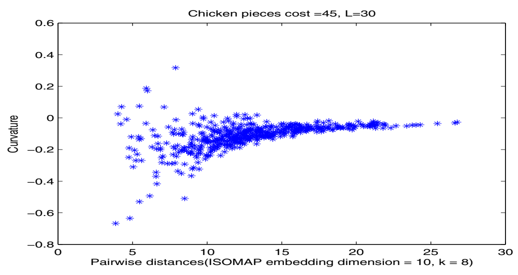
7 Experiments

We use the well known "Chicken pieces" dataset [8] for experimentation. The data-set concerns classifying binary images of a different types of chicken joint into shape-classes. It contains 446 binary images falling into five shape classes, namely a) breast (96 examples), b) back (76 examples), c) thigh and back (61 examples), d) wing (117 examples) and e) drumstick (96 examples). The data exists in the form of a set of non-Euclidean shape dissimilarity matrices, generated using different settings for the parameters in which, L is the length of straight line segments of chicken contours and C is the insertion and deletion costs for computing edit distances between boundary segments. Our experimental results are for the dissimilarity data with C = 45 and L = 5, 10, 15, 20, 25 and 30.

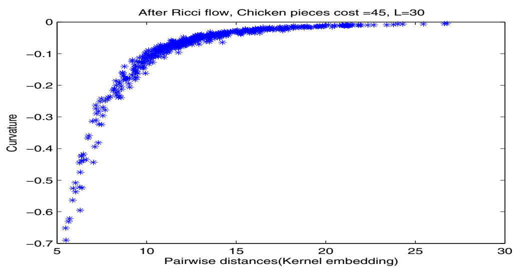
The negative eigenfraction for the Chicken Pieces data with $L = 5.0$, $C = 45$ is shown in Figure 1 as the manifold evolves with iteration number. As the curvatures are updated both the negative eigenfraction and the negative eigenvalues decrease, indicating that the dissimilarity measure becomes increasingly Euclidean. Figure 2 shows the curvatures as a function of distances obtained using the kernel embedding and Isomap embedding. It demonstrates how the Ricci flow process affects distances commencing from the two embedding methods with and without regularization. It indicates that the embedding method affects the magnitude of curvatures. The figure also shows that the Kernel embedding preserves the global distances. Here, the larger the distances, the smaller the curvatures. On the other hand, the Isomap embedding preserves some of the local distances. This maybe the due to the fact that the chicken pieces data does not reside on simple manifold such as Swiss roll. The embedding method determines



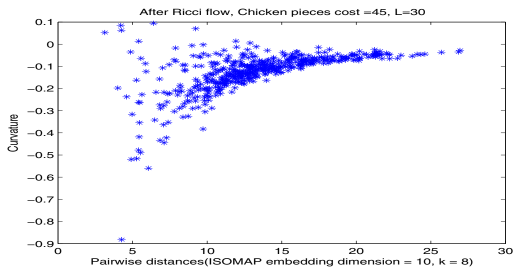
(a) Kernel embedding curvature



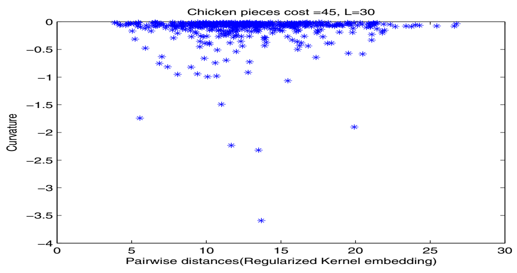
(b) Isomap embedding



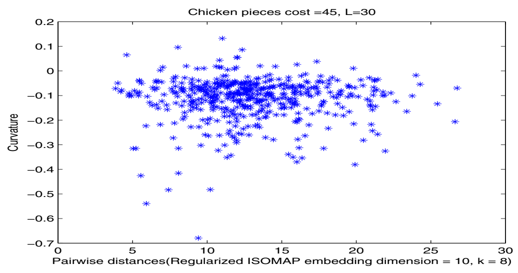
(c) Kernel embedding



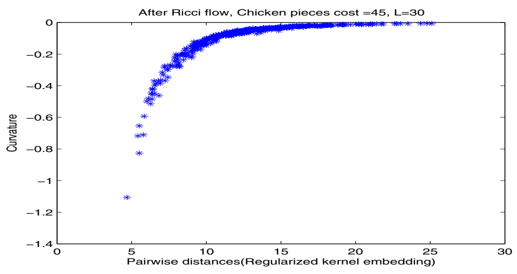
(d) Isomap embedding



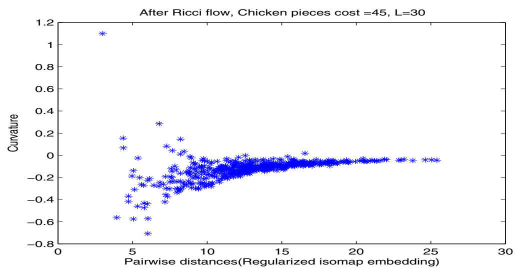
(e) Kernel embedding curvature



(f) Isomap embedding



(g) Kernel embedding



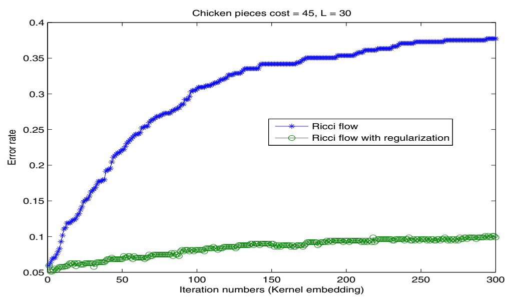
(h) Isomap embedding

Fig. 2. (a) and (b) are initial edge curvatures for the kernel and Isomap embeddings. (c) and (d) are edge curvatures after Ricci Flow for the kernel and Isomap embeddings.(e) and (f) are initial regularised edge curvatures for the kernel and Isomap embeddings. (g) and (h) are edge curvatures after Ricci Flow for the kernel and Isomap embeddings.

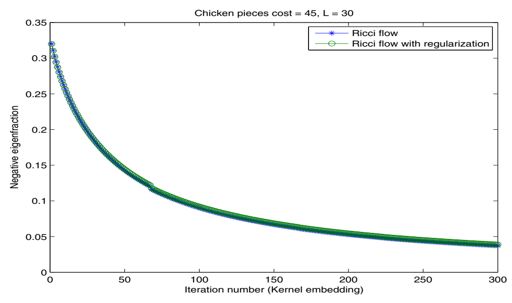
the magnitude of curvatures. From our Ricci flow curvature updating process, the larger the magnitude of the original curvatures, the larger the curvature reduction in the update process. As a result in the case of the kernel embedding, those locations associated with large curvature expand more rapidly than those associated with small curvatures. In other words, the initial smaller distances expand more rapidly than larger distances.

This effect can be observed from Figure 2(a) and Figure 2(c). As a result. it disrupts the local pattern of distances without influencing the larger ones.

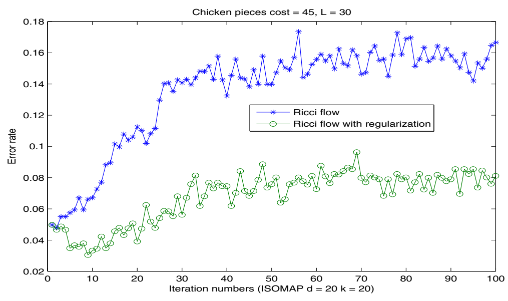
During the regularization step, the curvatures are smoothed over nearest neighbour edges. The result is to reduce local curvature fluctuations, and this may reduce some locally large curvature values. Figure 2(e) shows that when regularisation is used, the curvatures are smoothed over local distance scales compared to the initial curvatures in Figure 2(a). Hence the local distance structure is preserved under the embedding, and this is demonstrated in Figure 2(g). As a result the regularization step preserves local distances and stabilizes the local structure.



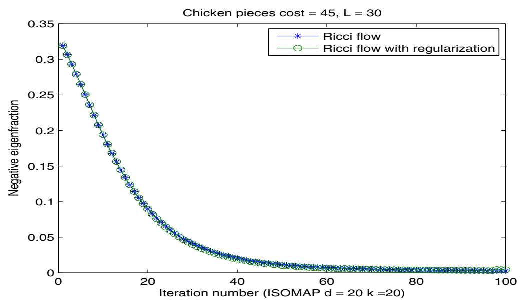
(a) 1NN error rate



(b) negative eigenfraction



(c) 1NN error rate



(d) negative eigenfraction

Fig. 3. (a) is the 1NN error rate with and without the regularization step using the kernel embedding during iteration. (b)is the negative eigenfraction with and without the regularization step using the kernel embedding during iteration. (c) is the 1NN error rate with and without the regularization step using the Isomap embedding during iteration. (d)is the negative eigenfraction with and without the regularization step using the Isomap embedding during iteration.

Next, we turn our attention to the effect of regularisation and the choice of embedding on the results of classification. The classification results were obtained with the 1-NN classifier and 10-fold cross validation. In Figure 3 we compare the 1-NN error rates and the negative eigenfaction obtained with regularised and unregularised versions of Ricci flow on the two embedding schemes. The first point to note is that for both the kernel embedding and Isomap, we obtain better classification results when heat kernel regularisation is used. However, in each case the application of the Ricci flow scheme causes the classification error to increase with iteration number. However, in the case of the regularised kernel embedding, the effect is smallest. Finally, the choice of embedding scheme strongly affects the rate of decrease of the negative eigenfraction, with

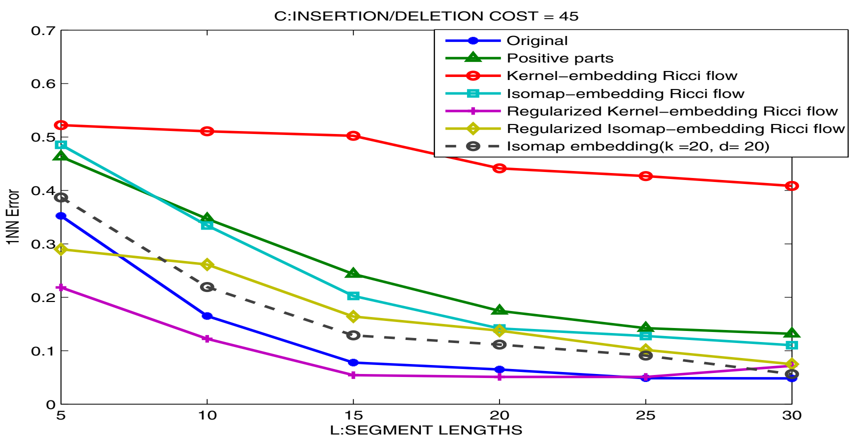


Fig. 4. Error rate from 1NN

Isomap giving a faster rate of decrease with iteration number than the kernel embedding. However, for both embedding schemes the use of regularisation has little effect on the rate of decrease.

Finally, we have compared our results with the known manifold embedding technique Isomap and those obtained using some alternative non-Euclidean distance rectification procedures. The methods explored were a) using the original distances, b) projecting onto the positive subspace and taking the distance here, unregularised Ricci flow on c) the kernel embedding and d) the Isomap embedding, regularised Ricci flow on e) the kernel embedding and f) the Isomap embedding. Figure 4 shows the 1-NN error rate as function of the shape parameter L (the segment length). The best results are obtained with Ricci flow on the regularised kernel embedding. All of the remaining methods give poorer results than applying the classifier to the original distance data.

8 Conclusion

In this paper we have explored how to evolve a non-Euclidean dissimilarity measure into a Euclidean one using Ricci flow. We commence by representing the dissimilarity data using a weighted graph, where the nodes represent objects and the edge weights dissimilarities between objects. We embed the graph onto a manifold so that the geodesic distance between nodes is equal to the dissimilarity on the edges. Under the embedding the edges acquire a curvature determined by the difference between geodesic distance (dissimilarity) and Euclidean distance. The Ricci flow, modifies the Gaussian curvatures on the edges, so as to flatten the manifold. We explore in depth the effect of stabilising this process by using heat-kernel regularisation to smooth the Gaussian curvatures prior to evolving the manifold.

We apply our method to the Chicken Pieces data. When applied without regularisation, although the distance measures can be transformed into a Euclidean space there is some loss of discriminating power and the classifier performance degrades. The loss of information is attributable to the effect of the Ricci evolution process which acts independently on each edge and ignores the local structure of the manifold. When heat kernel regularization is used the ranking of distance measures is preserved, and better performance is achieved. Although the method degraded the error obtained with a 1NN classifier, it does deliver data in a form where geometric classification methods can be applied to the data. The Ricci flow evolution minimise the curvatures, when the curvatures reach zero, then the geodesic and the Euclidean distances are equal and the negative eigenfraction is zero.

As the embedding methods affects the magnitude of curvatures a lot, one way to develop our work is to reduce the reliance on the embedding methods by using spherical embedding and tangent space projection. Another direction is to develop incremental learning, in which new points can be mapped on the manifold, as our current method is performed in a batch mode, i.e., all training points are processed simultaneously.

Acknowledgements

This work was supported by the EU FET project SIMBAD (213250). Edwin Hancock was supported by a Royal Society Wolfson Research Merit Award.

References

1. Pekalska, E., Duin, R.P.W.: Beyond traditional kernels: classification in two dissimilarity-based representation spaces. IEEE Transactions on Systems Man and Cybernetics-Part C 38(6) (November 2008)
2. Sanfeliu, A., Fu, K.S.: A distance measure between attributed relational graphs for pattern recognition. IEEE transactions on systems, man, and cybernetics 13(3), 353–362 (1983)
3. Bunke, H.: A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters 19(3-4), 255–259 (1998)
4. Borg, I., Groenen, P.: Modern multidimensional scaling: Theory and applications. Springer, Heidelberg (2005)
5. Tenenbaum, J., Silva, V., Langford, J.: A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2323 (2000)
6. Roweis, S., Saul, L.: Nonlinear dimensionality reduction by locally linear embedding (2000)
7. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. Advances in neural information processing systems 1, 585–592 (2002)
8. Duin, R.P.W., Pekalska, E., Harol, A., Lee, W.J., Bunke, H.: On euclidean corrections for non-euclidean dissimilarities. In: SSPR/SPR, pp. 551–561 (2008)
9. Pekalska, E., Duin, R., Gunter, S., Bunke, H.: On not making dissimilarities euclidean. Lecture notes in computer science, pp. 1145–1154 (2004)
10. Xu, W., Hancock, E.R.W.: Rectifying non-euclidean similarity data using ricci flow embedding. In: To appear ICPR 2010 (August 2010)
11. Pekalska, E., Harol, A., Duin, R., Spillmann, B., Bunke, H.: Non-euclidean or non-metric measures can be informative, pp. 871–880 (2006)
12. Chow, B., Luo, F.: Combinatorial Ricci flows on surfaces. J. Differential Geom. 63(1), 97–129 (2003)
13. ElGhawalby, H., Hancock, E.R.: Measuring graph similarity using spectral geometry. In: ICIAR, pp. 517–526 (2008)
14. Lindman, H., Caelli, T.: Constant curvature riemannian scaling. Journal of Mathematical Psychology 17, 89–109 (1978)
15. Kondor, R., Lafferty, J.: Diffusion kernels on graphs and other discrete structures. In: Proceedings of the ICML, pp. 315–322 (2002)

Rectifying Non-euclidean Similarity Data through Tangent Space Reprojection

Weiping Xu, Edwin R. Hancock, and Richard C. Wilson

Dept. of Computer Science,University of York, UK
{elizaxu,erh,wilson}@cs.york.ac.uk

Abstract. This paper concerns the analysis of shapes characterised in terms of dissimilarities rather than vectors of ordinal shape-attributes. Such characterisations are rarely metric, and as a result shape or pattern spaces can not be constructed via embeddings into a Euclidean space. The problem arises when the similarity matrix has negative eigenvalues. One way to characterise the departures from metricty is to use the relative mass of negative eigenvalues, or negative eigenfraction. In this paper, we commence by developing a new measure which gauges the extent to which individual data give rise to departures from metricity in a set of similarity data. This allows us to assess whether the non-Euclidean artifacts in a data-set can be attributed to individual objects or are distributed uniformly. Our second contribution is to develop a new means of rectifying non-Euclidean similarity data. To do this we represent the data using a graph on a curved manifold of constant curvature (i.e. hypersphere). Xu et. al. have shown how the rectification process can be effected by evolving the hyperspheres under the Ricci flow. However, this can have effect of violating the proximity constraints applying to the data. To overcome problem, here we show how to preserve the constraints using a tangent space representation that captures local structures. We demonstrate the utility of our method on the standard "chicken pieces" dataset.

Keywords: Dissimilarity, Embedding, Ricci flow, Spherical embedding, Tangent space.

1 Introduction

Geometric shape representation and recognition is an active area of research in computer vision and pattern recognition. Graph-based representations have found widespread use in shape analysis, for example, in the use of shock graphs to represent shape-skeletons [5]. When such a representation is adopted, measures such as graph-edit distance provide the natural way of capturing the similarity of different shapes. This provides a powerful and natural way of capturing the relationships between objects that are not characterised by ordinal measurements or feature vectors [6]. One way to construct a shape-space for such data is to represent the dissimilarity data using a weighted graph, and to embed the graph on a manifold. This produces a vectorial representation of the data by projecting dissimilarity data into a fixed-dimensional vector space. Examples of this approach include multidimensional scaling (MDS) and Isomap [10].

However, one of the problems with dissimilarity representations and their embeddings, is that the distance measures can not be used to construct a shape space if the underlying dissimilarity matrix contains negative eigenvalues. If this is the case the shapes can not be embedded into a real-valued Euclidean space [3], and must instead be embedded into a complex valued or Krein space.

In order to analyse non-Euclidean dissimilarity data using traditional geometric machine learning or pattern recognition techniques, we must first attempt to rectify the data so as to minimize the non-Euclidean artifacts. Before the analysis of such data is attempted, it is advisable to assess the degree and extent to which non-Euclidean artefacts affect the data-set. One measure that has proved useful in this respect is the negative eigenfraction [1] which is the total mass of negative eigenvalues as a fraction of the total mass of unsigned eigenvalues. However, in this this paper, we introduce a finer measure that assesses the contribution of each object to the mass of negative eigenvalues. In this way it is possible the determine whether the non-Euclidean artefacts are attributable to the outlying dissimilarities of a few objects or are uniformly distributed throughout the dataset.

However, our main contribution in the paper is to consider how to rectify the data to minimise the effects of non-Euclidean artefacts. Xu et. al. [8] have explored the idea of embedding the disimilarity data on a locally hyperspherical surface of constant curvature. They then flatten the manifold composed of local hyperspherical patches by reducing the curvatue according to a Ricci flow. As a result both local and global distances are modified by the flattening. One of the problems they have encountered in applying Ricci flow to a constant curvature Riemannian manifold to evolve the distance measures is that due to the piecewise nature of the manifold, the structure of the data is distorted. Moreover, they also encounter instabilities due to local fluctuations in edge curvature. Although this latter problem can to some extent be remedied by regularizing the Gaussian curvature [8], the problem of preserving structure persists.

To overcome this problem, we aim to reduce the reliance on the piecewise embedding and its effect on individual edges. We turn to the tangent space representation of data using the exponential and log maps [2], which provide a means of preserving the distance between the points on the manifold and the origin of the map. This allows us to to flatten the manifold while preserving the global structure of the data.

2 Characterising Non-euclidean Data

In this paper we are concerned with embedding data represented in terms of pairwise dissimilarities or distances, and in particular the case where the data is non-Euclidean. Our overall aim is to rectify a given set of non-Euclidean dissimilarity data so as to make them more Euclidean. One way to gauge the degree to which a pairwise distance matrix exhibits non-Euclidean artefacts is to analyse the properties of its centralised Gram matrix. For an $N \times N$ symmetric pairwise dissimilarity matrix D with the pairwise distance as elements, the centralized Gram matrix $G = -\frac{1}{2}JD^2J$,where $J = I - \frac{1}{N}11^T$ is the centering matrix and 1 is the all-ones vector of length N. The degree to which the distance matrix departs from being Euclidean can be measured by using the relative mass of negative eigenvalues or "negative eigenfraction " $F_{eigS} = \sum_{\lambda_i<0}|\lambda_i|/\sum_{i=1}^{N}|\lambda_i|$ [1]. This measure is zero when the distances are Euclidean and increases as the distance becomes increasingly non-Euclidean.

If the non-Euclidean artefacts are contributed solely by the set of distances to a few "outlier" objects, it is possible to restore the data to a Euclidean state by editing (i.e. removing) these objects from the dataset. Based on this idea we introduce the notion of measuring the contribution of each object to the negative eigenfraction of a dissimilarity matrix. That is, the fraction given by the sum of the negative distances originating from an individual object to all the remaining objects, divided by the total.

The matrix of kernel embedding co-ordinates is given by $Y = \sqrt{\Lambda}\Phi^T = (y_1, ..., y_N)$, where $\Lambda = diag(\lambda_1, ..., \lambda_N)$ is the diagonal matrix with the ordered eigenvalues of centered Gram matrix as elements and $\Phi = (\phi_1|...|\phi_N)$ is the eigenvector matrix with the ordered eigenvectors ϕ_1, ..., ϕ_N as columns. When the centered Gram matrix has negative eigenvalues then those dimensions of the embedding associated with negative eigenvalues are represented by imaginary numbers, and those associated with positive eigenvalues by real numbers. In other words, the data are embedded into a pseudo Euclidean or Krein space [3].

Under the embedding, the coordinate vector of point j is $y_j = (\sqrt{\lambda_1}\Phi_{1j}, ..., \sqrt{\lambda_i}\Phi_{ij}, \sqrt{\lambda_N}\Phi_{Nj})^T$. The contribution to the negative squared distance between two points k and e is $d_{ke}^2 = \sum_i (y_k(i) - y_e(i))^2 = \sum_i \lambda_i(\phi_{ik} - \phi_{ie})^2$. The sum of negative squared distances from point k to all the remaining points is $d_{k-}^2 = \sum_{\lambda_i<0} \lambda_i \sum_{e\neq k} (\phi_{ik} - \phi_{ie})^2$. On the other hand, the sum of positive distances from point k to the remaining points is $d_{k+}^2 = \sum_{\lambda_i>0} \lambda_i \sum_{e\neq k} (\phi_{ik} - \phi_{ie})^2$. Thus the fraction of negative squared distances from point k is $C_{neig} = \frac{|d_{k-}^2|}{|d_{k-}^2|+|d_{k+}^2|}$.

3 Spherical Embedding

Spherical embedding [9] provides a means by which to embed objects represented in terms of dissimilarity data onto a hypersphere. The optimal radius of the hypersphere minimises the distortion of the geodesic distances between objects. It is desirable that the degree of the nodes of the embedded graph and the ranking of distances (dissimilarities) are preserved under the embedding. Given the hypershere of optimal raidus, the embedding coordinates are obtained through the eigendecomposition of the inner product matrix[9], and for the node indexed p the vector of embedding co-ordinates is $y_p = \sqrt{\Lambda}\Phi^T$, where Λ is the diagonal matrix with the ordered eigenvalues of the inner product matrix $Z = cos(\frac{D}{r})$

as elements and Φ is the matrix with the ordered eigenvectors of Z as columns, D is the distance matrix and $r \in R^+$ is the optimal radius of the embedding hypersphere.

4 The Exponential and Log Map

The exponential map $\mathrm{Exp_p}[.]$ is a mapping from points on the manifold to points in the tangent space at a reference point on a manifold. The log map $Log_p[.]$ the inverse mapping from points in the tangent space at the reference point to points on the manifold. The exponential map [2] preserves the distance between the points on the manifold and the reference point or origin of the tangent space to the manifold. Our aim is to flatten the global manifold by gradually

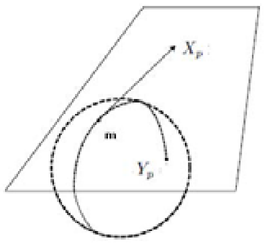


Fig. 1. The exponential map and log map

smoothing out the local patches. This is achieved by representing sub-graphs of objects on the local hyperspheres, mapping the points to the tangent space through the log-map function, reducing the curvatures (i.e.increasing the radii) of the individual hyperspherical patches, and then mapping the data back onto the inflated hyperspheres through the exponential-map function. The increase in radius of the hypersheres is determined by the Ricci flow [7] and satisfies the equation

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

where g_{ij} is the metric tensor of the manifold and R_{ij} is the Ricci curvature. We model the embedding manifold as consisting of a set of local patches with individual constant Ricci curvatures. The solution of the differential equation is straightforward. Commencing with the initial conditions curvature $K = K_0$ at time $t = 0$, then at time t we have $K_t = \frac{K_0}{1 \pm 2K_0 t}$ with the positive sign for the elliptic space (hypersphere).

On the spherical manifold, the log and exp maps give the following co-ordinate trasformations [9]:

$$x_p = \frac{\theta}{\sin\theta}(y_p - y_m \cos\theta); y_p = y_m \cos\theta + \frac{\sin\theta}{\theta} x_p. \tag{2}$$

where y_p is the coordinate vector for point p on the manifold, x_p is the coordinate vector for point p in the tangent space, m is the reference point or origin of the map with the length equals to the corresponding radius, and θ is the angle between radius vectors to the points m and p on the hypersphere. This set of transformations is illustrated in Figure 1.

Once, the inflation and reprojection onto the hypersphere are complete, we compute the new coordinate vector of a point on the inflated hypersphere based on the old coordinate vector on the original hypersphere by using Equation 2:

$$y_{p_{n+1}} = (1 + 2K_n t)^{\frac{1}{2}} (y_{m_n} \cos\theta_{n+1} + \frac{\sin\theta_{n+1}}{\sin\theta_n}(y_{p_n} - y_{m_n}\cos\theta_n)). \tag{3}$$

where the angle on the original sphere is $\theta_n = K_n acos < y_m, y_p >$.

As the geodesic distances to the origin are preserved, we can compute the angles on the inflated sphere $\theta_{n+1} = \frac{K_{n+1}^{\frac{1}{2}}}{K_n^{\frac{1}{2}}}\theta_n$, given the curvatures of the original and the inflated spheres, and updated radial angles on the original sphere.

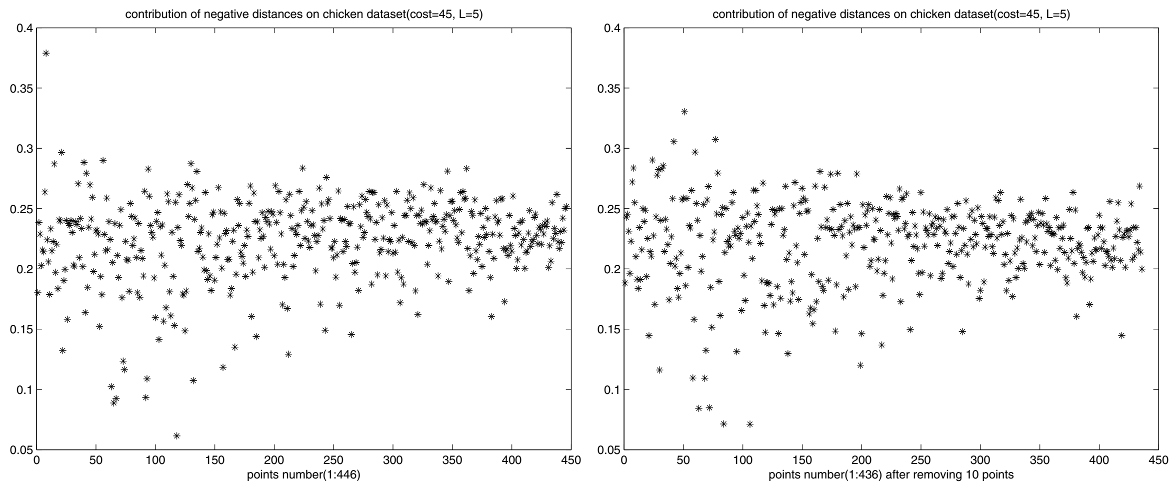
Then we compute new geodesic distances for the points on the inflated hypersphere. Reprojection under the log map preserves the geodesic distances to the origin of the tangent space. However, the geodesic distances between points are modified by the inflation and reprojection. The updated geodesic distances on the inflated hypersphere can be computed using the new co-ordinates on the inflated hypersphere. The update equation for the geodesic distance between point m and p on the inflated hypersphere is

$$d_{Gmp} = r_{n+1}\theta_{n+1} = \frac{acos(< y_{m_{n+1}}, y_{p_{n+1}} > K_{n+1})}{K_{n+1}^{\frac{1}{2}}}. \tag{4}$$

5 The Algorithm

Given a set $Y = \{y_1, \cdots, y_N\}$ of N objects and a dissimilarity measure d, a dissimilarity representation is an $N \times N$ matrix D_G with the elements $d_G(u, v)$ representing the pairwise geodesic distance between objects y_u and y_v. The following algorithmic steps can be used to perform Euclidean rectification of the distance matrix, and suppress its non-Euclidean artefacts:

1. Construct a local patch for a second order k-NN graph, consisting of the first and second neighbors of the reference object.
2. Perform hyperspherical embedding to obtain the initial curvature and the co-ordinates of the objects in the local patch.
3. Update the hyperspherical radius (i.e. curvature) with a small time step derived from Equation 1.
4. Obtain the new coordinates on the inflated hypersphere using Equation 3.
5. Obtain the new geodesic distance matrix $d_{G_{n+1}}$ for the local patch using Equation 4. The distances between pairs of objects external to the patch are approximated using the old dissimilarity matrix. The geodesic distance between pairs of objects external to the patch and objects internal to the patch are approximated by adding the geodesic distances over the of edge-connected path between the objects.



(a) The original distribution

(b) The distribution after removing 10 highly-contributed points

Fig. 2. The distribution of negative contribution of each point before and after removing 10 most highly-contributing objects

6. Obtain the updated global distance matrix $D_G^{(1)}$ containing rectified geodesic distances between objects, and repeat from step 1 until D_G stabilises, i.e. there are no further decreases in the negative eigenfraction. Ideally, the centralized Gram matrix should have no negative eigenvalues.

6 Experiments

We use the well known "Chicken pieces" shape dataset [4] for experimentation. The data-set poses the problem of classifying binary images of different types of chicken joint into shape-classes. It contains 446 binary images falling into five shape classes, namely a) breast (96 examples), b) back (76 examples), c) thigh and back (61 examples), d) wing (117 examples) and e) drumstick (96 examples). The data exists in the form of a set of non-Euclidean shape dissimilarity matrices, generated using different parameter settings. The parameters are the length of straight line segments of the chicken contours L and the insertion and deletion costs for computing edit distances between boundary segments C. Our experimental results are for the dissimilarity data with parameters $C = 45$ and $L = 5$, 10, 15, 20, 25 and 30. The originally asymmetric dissimilarities are made symmetric by averaging.

We commence by showing the distribution of the individual object contributions to the negative eigenfraction for the chickenpieces data. In Figure 2 we show the distribution for the data with L =5.0; C=45, both before and after removing the 10 most strongly ontributing points. Figure 3 shows the negative eigenfraction of the chickenpieces data with $C = 45$ and $L = 5$, 10, 15, 20, 25 and 30, again both before and after removing the 10 most strongly contributing objects. Removing the most strongly contributing objectss has little effect

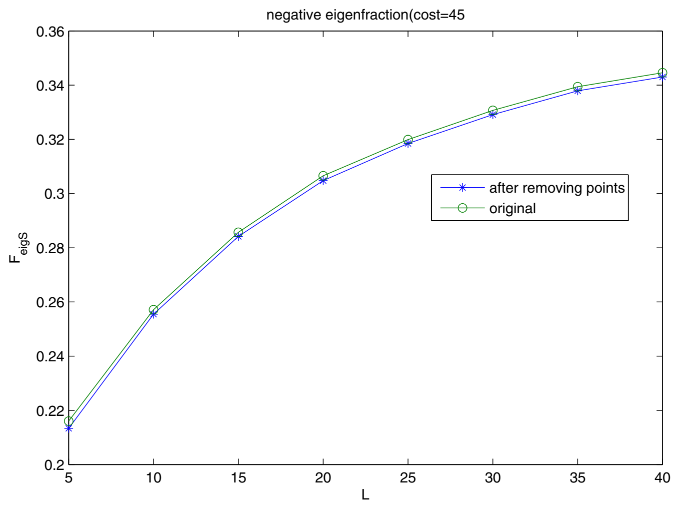
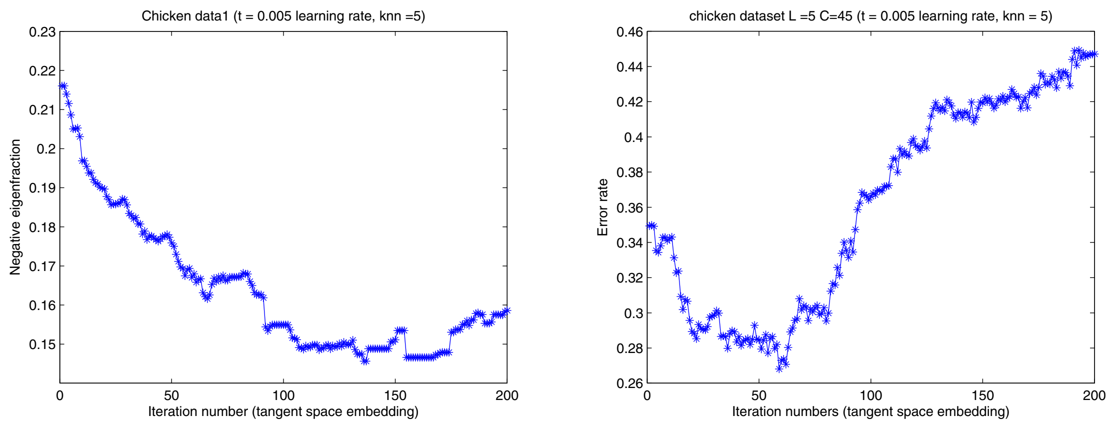


Fig. 3. The mass contribution of negative eigenvalues before and after removing the 10 most highly-contributing objects



(a) The negative eigenfraction (b) The 1NN error rate

Fig. 4. The negative eigenfraction and 1NN error rate as a function of iteration number

on he distribution, and this indicates that the non-Eucludean artefacts can not be attributed to outliers. Next, we explore the effectiveness of the rectification process. Figure 4 shows the negative eigenfraction and 1NN error rate for the shape-classes as the distance matrix is evolved. The negative eigenfraction drops from 22% to 15% and then increases again, indicating that the evolution has succeeded in flattening the manifold, but then deteriorates. This demonstrates that the distance measures can be corrected or flattened, but it is essential to have a halting criterion. The deterionation is caused by the path-based approximation of geodesic distances between internal and external objects on the local patches. These distances have been inflated more rapidly than the local distances on the hyperspherical surface. This exaggerates the overall curvature.

7 Conclusion

In this paper, we have made two contributions. First, we have presented a method to gauge the distribution of non-Euclidean artefacts in dataset Second, we have shown how to evolve a patchwise hyperspherical manifold so as to rectify such artefacts in a dataset. The method uses a tangent-space reprojection method to inflate the local hyperspherical patches, while maintaining the consistency of the pattern of geodesic distances. Applying our method to the chicken pieces shape dataset, we demonstrate that the method can be used to improve the classification of shape-data.

Acknowledgments. Work was supported by the project SIMBAD (213250). Edwin Hancock was supported by a Royal Society Wolfson Research Merit Award.

References

1. Pękalska, E., Harol, A., Duin, R., Spillmann, B., Bunke, H.: Non-euclidean or non-metric measures can be informative. In: Yeung, D.-Y., Kwok, J.T., Fred, A., Roli, F., de Ridder, D. (eds.) SSPR 2006 and SPR 2006. LNCS, vol. 4109, pp. 871–880. Springer, Heidelberg (2006)
2. Fletcher, P.T., Lu, C., Pizer, S.M., Joshi, S.: Barbara and Bunke, Horst: Principal geodesic analysis for the study of nonlinear statistics of shape. IEEE Transactions on Medical Imaging, 995–1005 (2004)
3. Goldfarb, L.: A new approach to pattern recognition. Progress in Pattern Recognition, 241–402 (1985)
4. Andreu, G., Crespo, A., Valiente, J.M.: Selecting the toroidal self-organizing feature maps (TSOFM) best organized to object recognition. In: ICNN, pp. 1341–1346 (1997)
5. Torsello, A., Hancock, E.R.: Computing approximate tree edit distance using relaxation labeling. Structural, Pattern Recognition Letters, 1089–1097 (2003)
6. Sanfeliu, A., Fu, K.-S.: A Distance measure between attributed relational graphs for pattern recognition. IEEE Transactions on Systems, Man, and Cybernetics, 353–362 (1983)
7. Chow, B., Luo, F.: Combinatorial Ricci flows on surfaces. J. Differential Geom., 97–129 (2003)
8. Xu, W., Hancock, E.R., Wilson, R.C.: Regularising the ricci flow embedding. In: Hancock, E.R., Wilson, R.C., Windeatt, T., Ulusoy, I., Escolano, F. (eds.) SSPR&SPR 2010. LNCS, vol. 6218, pp. 579–588. Springer, Heidelberg (2010)
9. Wilson, R.C., Hancock, E.R.: Spherical embedding and classification. In: Hancock, E.R., Wilson, R.C., Windeatt, T., Ulusoy, I., Escolano, F. (eds.) SSPR&SPR 2010. LNCS, vol. 6218, pp. 589–599. Springer, Heidelberg (2010)
10. Tenenbaum, J.B., Silva, V., Langford, J.C.: A global geometric framework for non-linear dimensionality reduction. Science (2000)

D Graph Diffusion of Quaternions

Multiview Registration via Graph Diffusion of Dual Quaternions

Andrea Torsello, Emanuele Rodolà, and Andrea Albarelli
torsello@dsi.unive.it rodola@dsi.unive.it albarelli@unive.it
Dipartimento di Scienze Ambientali, Informatica e Statistica - Università Ca' Foscari Venezia

Abstract

Surface registration is a fundamental step in the reconstruction of three-dimensional objects. While there are several fast and reliable methods to align two surfaces, the tools available to align multiple surfaces are relatively limited. In this paper we propose a novel multiview registration algorithm that projects several pairwise alignments onto a common reference frame. The projection is performed by representing the motions as dual quaternions, an algebraic structure that is related to the group of 3D rigid transformations, and by performing a diffusion along the graph of adjacent (i.e., pairwise alignable) views. The approach allows for a completely generic topology with which the pairwise motions are diffused. An extensive set of experiments shows that the proposed approach is both orders of magnitude faster than the state of the art, and more robust to extreme positional noise and outliers. The dramatic speedup of the approach allows it to be alternated with pairwise alignment resulting in a smoother energy profile, reducing the risk of getting stuck at local minima.

1. Introduction

Surface registration is a fundamental step in the reconstruction of three-dimensional objects. This is typically a two step process where all the views are first registered against each other, and then all the pairwise transformations are lowered to a common coordinate frame through a process commonly referred to as multiview registration.

The literature on pairwise registration is quite ample, with modifications to the original ICP proposed by Zhang [23] and Besl and McKay [4] taking the lion's share. ICP-based methods start from an initial pose estimate and iteratively refine it by minimizing a distance function measured between pairs of selected neighboring points. The variants generally differ in the strategies used to sample points from the surfaces, reject incompatible pairs, or measure error. In general, the precision and convergence speed of these techniques is highly data-dependent and very sensitive to the fine-tuning of the model parameters. Several approaches that combine these variants have been proposed in the literature in order to overcome these limitations (see [19] for a comparative review). Some recent variants avoid hard culling by assigning a probability to each candidate pair by means of evolutionary techniques [15] or Expectation Maximization [10]. ICP variants, being iterative algorithms based on local, step-by-step decisions, are very susceptible to the presence of local minima. Other fine registration methods include the well-known approach by Chen [6] and signed distance fields matching [16].

By contrast, the literature of multiview registration is more diverse. In [6] Chen and Medioni propose to iteratively merge new views into a single metaview: The registration of a new view against the metaview is obtained with a common pairwise registration technique, such as ICP, and then the points of the new registered view are merged to the metaview; the approach is iterated until all the range images are merged. This metaview approach has problems since registration errors are accumulated rather than mediated. To solve this problem Bergevin *et al.* [3] match points in every view with all the views overlapping with it, and calculate a transformation that registers the first view using all the mating points. This process is iterated to convergence, thus diffusing the errors among all views. This is implicitly a diffusion process where the random walk in the transformation space is governed by the constraints offered by nearby views in the view-graph; however, convergence toward the steady-state is extremely slow and computationally demanding. Eggert *et al.* [9] constrain the pairings so that the points of each scan map with exactly one other point and then minimize the total distance between the paired points. This speeds up convergence, but can prevent the algorithm from converging to a correct solution as the views may cluster into groups that are well registered, without improving inter-group registration. With these iterative algorithms based on global point correspondences, how and when to apply the transformation remains an open issue: For example, Bergevin *et al.* [3] calculate a transformation for each view separately and then apply them simultaneously before the next round of matchings, while Benjemaa and Schmitt [2] apply the new transformations independently as soon as they are calculated, and Eggert *et al.* [9] solve for the update by simulating a spring model. An alternative was explored in [11] where an approximate surface model is created and the view are registered against the model. The surface model is then iteratively refined using the new registrations.

In [18] Pulli takes a simplifying view that pairwise registrations are "as good as it gets" and that the role of multiview registration is only to project the transformation into a common reference frame in such a way as to limit the accumulation of registration errors. To this end, he proposes a greedy approach that tries to limit the difference between the position of point sets as positioned in two frames and transformed by the pairwise registration of the two frames. More formally, he tries to keep the distortion $\mathcal{D}(S)$ of the points from a set S within a given tolerance ϵ, where

$$\mathcal{D}(S) = \sum_{s\in S} \sum_{(i,j)\in\mathcal{V}} ||P_i(s) - T_{ij}\big(P_j(s)\big)||^2 .$$

Here P_i is the transformation that maps a point into the coordinate system of view i, T_{ij} is the transformation that maps the coordinate frame j into the coordinate frame i obtained through pairwise registration, and $\mathcal{V}$ is the set of pairs of neighboring views for which pairwise registration is performed. Pulli suggests to sample the set of fiduciary points S from the surface of the object. Interestingly, by working only on the space of transformations, this approach limits the memory requirements since it does not need to keep all the points from all the views in memory at once. Note, however, that the approach cannot guarantee that an optimal solution will be found, nor that any solution within the given tolerance will be found.

More recently, Williams and Bennamoun [22] adopted a similar view, posing the problem as the minimization of the distortion on a set of fiduciary points and computing the minimization by an iterative approach optimizing each rotation via singular value decomposition.

In this paper we propose a novel multiview registration algorithm where the poses are estimated through a diffusion process on the view-adjacency graph. The diffusion process is over dual quaternions [7], a non-commutative and non-associative algebraic structure that is related to the group $SE(3)$ of 3D rigid transformations, leading to an approach that is both orders of magnitude faster than the state of the art, and more robust to extreme positional noise and outliers.

2. Dual Quaternions and 3D Transformations

Quaternions have been a popular geometrical tool for more than 20 years as they represent 3D rotations in a way that is arguably more efficient and robust than 3×3 rotation matrices [20]. Quaternions are an algebraic extension of complex numbers with 3 imaginary bases i, j, and k, thus a quaternion is a number of the form $q = a + ix + jy + kz$. The multiplication of two quaternions is defined through the following multiplication rules for the three imaginary bases: $i^2 = j^2 = k^2 = -1$, $ij = k = -ji$, $jk = i = -kj$, $ki = j = -ik$. The *conjugate* of a quaternion $q = a + ix + jy + kz$ is the quaternion $q^* = a - ix - jy - kz$, while the *norm* of a quaternion is the quantity $||q|| = \sqrt{qq^*} = \sqrt{q^*q} = \sqrt{a^2 + x^2 + y^2 + z^2}$. Quaternions with unitary norm are called *unit quaternions*. In the following we will use the vectorial representation of quaternions: Let $\mathbf{i} = (i, j, k)$ be the row-vector of the imaginary bases, we can write the quaternion q as $a + \mathbf{iv}$, where $\mathbf{v} = (x, y, z)^T$ is a 3D vector. A right-handed 3D rotation of angle θ around the axis of unit vector $\mathbf{v}$ is in relation with the unit quaternion $q = \cos(\theta/2) + \sin(\theta/2)\mathbf{iv}$. In fact, let $\mathbf{p} = (p_x, p_y, p_z)^T$ be a 3D point and $\mathbf{p}_r$ its rotation, we have $\mathbf{ip}_r = q(\mathbf{ip})q^*$. The ring of quaternions, however, is a dual cover of the group $SO(3)$ of 3D rotations, as q and $-q$ represent the same rotation. Quaternions are particularly interesting since they allow for optimal interpolation between rotations. The famous Spherical Linear Interpolation (SLERP) algorithm [20] interpolates quaternions on the unit hypersphere and exhibits the following useful properties:

- **Shortest path:** the motion between the initial rotation R_0 and the final rotation R_1 is a rotation about a fixed axis with the smallest angle.
- **Constant speed:** the angle of the interpolated rotation varies linearly with respect to parameter t.
- **Coordinate system invariance:** the interpolation path does not change if we change the coordinate system.

On the other hand, linear interpolation followed by reprojection onto the unit hypersphere guarantees the first and third properties, but exhibits changes in speed, in particular it accelerates around the middle of the interpolation. This implies that a linear averaging of quaternions does not minimize the squared geodesic distance in the unit quaternion manifold in the same way that the mean of a set of points minimizes the squared Euclidean distances to the points. Unfortunately, SLERP does not generalize to the blending of several rotations. Buss and Fillmore [5] provide an iterative algorithm to find the proper (weighted) mean in the unit-quaternion manifold, while Kavan and Žára [14] show that the difference between spherical and linear interpolation is always less than 0.071 radians.

Dual quaternions are less known than quaternions, but their ability to efficiently represent rigid transformations has been successfully adopted in 3D animation and skinning [12], robot control and registration [1, 8], and have been used in theoretical kinematics for a long time [17]. Dual quaternions are an algebraic extension of quaternions

Table 1. Multiplicative table of dual quaternions.

	1	i	j	k	ϵ	ϵi	ϵj	ϵk
1	1	i	j	k	ϵ	ϵi	ϵj	ϵk
i	i	-1	k	$-j$	ϵi	$-\epsilon$	ϵk	$-\epsilon j$
j	j	$-k$	-1	i	ϵj	$-\epsilon k$	$-\epsilon$	ϵi
k	k	j	$-i$	-1	ϵk	ϵj	$-\epsilon i$	$-\epsilon$
ϵ	ϵ	ϵi	ϵj	ϵk	0	0	0	0
ϵi	ϵi	$-\epsilon$	ϵk	$-\epsilon j$	0	0	0	0
ϵj	ϵj	$-\epsilon k$	$-\epsilon$	ϵi	0	0	0	0
ϵk	ϵk	ϵj	$-\epsilon i$	$-\epsilon$	0	0	0	0

much like complex numbers are an extension of the reals. They are defined in terms of a dual basis ϵ that commutes with the imaginary bases; thus, a dual quaternion is a number of the form $q + \epsilon r$, where q and r are quaternions, and the product follows the multiplicative rule $\epsilon^2 = 0$, yielding

$$(q + \epsilon r)(s + \epsilon t) = qs + \epsilon(qt + rs)\,.$$

This results in the multiplicative table shown in Table 1. Dual quaternions have three different conjugates:

$$(q+\epsilon r)^* = q^*+\epsilon r^* \quad (q+\epsilon r)^\dagger = q^*-\epsilon r^* \quad (q+\epsilon r)^+ = q-\epsilon r$$

The norm of a dual quaternion dq is $||dq|| = \sqrt{dq^* dq} = \sqrt{dq\, dq^*} = ||dq^*||$ and the inverse of a dual quaternion dq is $dq^{-1} = \frac{dq^*}{||dq||^2}$. The rigid transformation obtained by a rotation defined by the unit quaternion r and then a translation by $\mathbf{t} = (t_x, t_y, t_z)^T$, is represented by the dual quaternion

$$dq = r + \tfrac{1}{2}\epsilon\mathbf{i}\mathbf{t}\, r\,.$$

In fact, if we represent a 3D point $\mathbf{p}$ as $1+\epsilon\mathbf{i}\mathbf{p}$, the following holds:

$$\begin{aligned}(r + \tfrac{1}{2}\epsilon\mathbf{i}\mathbf{t}rt)(1 + \epsilon\mathbf{i}\mathbf{p})(r + \tfrac{1}{2}\epsilon\mathbf{i}\mathbf{t}r)^\dagger &=\\ (r + \tfrac{1}{2}\epsilon\mathbf{i}\mathbf{t}r + \epsilon r\mathbf{i}\mathbf{p})(r^* - \tfrac{1}{2}\epsilon(\mathbf{i}\mathbf{t}r)^*) &=\\ (r + \tfrac{1}{2}\epsilon\mathbf{i}\mathbf{t}r + \epsilon r\mathbf{i}\mathbf{p})(r^* + \tfrac{1}{2}\epsilon r^*\mathbf{i}\mathbf{t}) = 1 + \epsilon(r\mathbf{i}\mathbf{p}r^* + \mathbf{i}\mathbf{t})\,.\end{aligned}$$

Thus, $1 + \epsilon\mathbf{i}\mathbf{p}$, i.e., the dual quaternion representation of the point $\mathbf{p}$, gets mapped into the dual quaternion $1+\epsilon(r\mathbf{i}\mathbf{p}r^* + \mathbf{i}\mathbf{t})$, which is the representation of $\mathbf{p}$ after the rotation and translation have been applied. In fact, $r\mathbf{i}\mathbf{p}r^*$ represents the rotation by r with the usual quaternion notation, while the addition of $\mathbf{i}\mathbf{t}$ takes care of the subsequent translation. Further, any dual quaternion $q + \epsilon r$ with $||q|| = 1$ and $q \cdot r = 0$ represents a rigid transformation. Here $\cdot$ represents the standard dot product in the quaternion viewed as a four-dimensional vector space over $\mathbb{R}$. However, the dual quaternion representation is not unique since, as with the normal quaternions, dq and $-dq$ represent the same transformation.

In [13, 12] the authors present ScLERP, a generalization of the SLERP interpolation algorithm for dual quaternions. Let $\alpha(t)$, $\mathbf{a}(t)$, $\delta(t)$, and $\mathbf{d}(t)$ be respectively the rotation angle and axis, and the translation magnitude and direction, then ScLERP was shown to have the following properties: a) $\mathbf{a}(t)$, and $\mathbf{d}(t)$ are constant and $\alpha(t) \in [-\pi;\pi]$ (shortest path); b) $\frac{d}{dt}\alpha(t) = 0$ and $\frac{d}{dt}\delta(t) = 0$ (constant speed). Further, it is invariant to changes in the coordinate system. In [13] was presented an iterative algorithm called Dual quaternion Iterative Blending (DIB) for averaging dual quaternions in a way that minimizes the (weighted) squared geodesic distances between the target mean and the input quaternions in the Riemannian manifold of unit dual quaternions, and it was shown that the variation between the proper geodesic average and a linear blending followed by a reprojection has an upper bound of 0.143 radians in rotation and a relative variation of 15% in translation, while in general the differences remain much smaller. In particular, if the set of dual quaternions we want to blend has small variance, the linear average and the geodesic average converge rapidly, since the difference between the geodesic and Euclidean distance is $O(\theta)$ where θ is the angle of rotation. In the following we will use the notation $\mathrm{ScAVG}(q_1, \dots, q_n)$ to refer to geodesic mean of the quaternions $(q_1, \dots, q_n)$ as obtained by applying DIB with uniform weights.

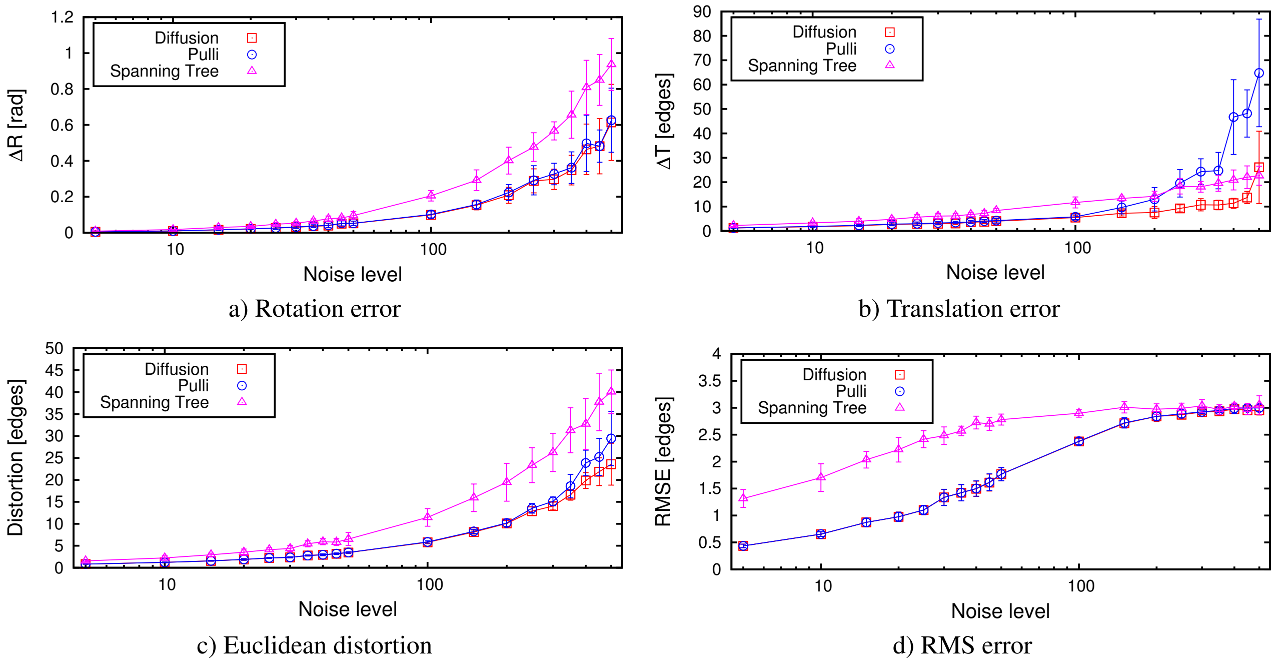
3. View-Graph Diffusion

We cast the multiview registration problem into a diffusion of rigid transformations over the view-graph, i.e., a graph in which nodes correspond to the range images and the edges reflect the adjacency relation between views, or, equivalently, the existence of an overlap between the scans. Let V_i be a transformation taking the coordinate frame of view i into a global coordinate frame, and T_{ij} the result of the pairwise registration taking the coordinate frame of view j into the frame of view i. Then, if the pairwise registration was noise-free, we would have $T_{ji} * V_i = V_j$ for all adjacent views i and j. In this setting the problem of multiview registration is that of finding a set of rigid motions from each view to a common frame of reference, say that of view 0, such that a measure of distortion between the position V_i of view i and the position $T_{ij}V_j$ obtained from the composition of position V_j and the pairwise registration T_{ij} is minimized, i.e., we seek to minimize the functional

$$D = \sum_i \sum_{j\in N(i)} d(T_{ij}V_j, V_i)$$

for an appropriate distortion function d. Here $N(i)$ is the set of neighbors of view i. This is in spirit similar to the approach taken by Pulli [18], where the distortion function is the (squared) Euclidean distance between the final position of a set of points S transformed with motions V_i and $T_{ij}V_j$:

$$D_P = \sum_i \sum_{j\in N(i)} \sum_{\mathbf{p}\in S} ||T_{ij}V_j\mathbf{p} - V_i\mathbf{p}||^2.$$



a) Rotation error b) Translation error c) Euclidean distortion d) RMS error

Figure 1. Comparison of the different methods in the synthetic experiments at various levels of noise.

We adopt a different measure of distortion that derives from the fact that any rigid transformation is in fact a screw motion, i.e., a rotation around an axis placed anywhere in the 3D space, and a translation along the direction of the axis. We define the *screw distance* $d_{SC}(q_i, q_2, \mathbf{p})$ as the length of the screw path of the point $\mathbf{p}$ along the transformation $\hat{q} = q_i^\dagger q_2$, i.e., $d_{SC}(q_i, q_2, \mathbf{p}) = \sqrt{t^2 + \alpha^2 r_{\mathbf{p}}^2}$ where t is the length of the translation along the axis of $\hat{q}$, α is the rotation angle, and $r_{\mathbf{p}}$ is the distance of $\mathbf{p}$ from the rotation axis. The *screw distortion* is then defined as

$$D_{SC} = \sum_i \sum_{j \in N(i)} \sum_{\mathbf{p} \in S} d_{SC}(T_{ij}V_j, V_i, \mathbf{p})^2 .$$

A direct consequence of the fact that ScLERP interpolation is both shortest path and constant speed is that, given a set of dual quaternions $Q = q_i, \dots, q_n$, their screw average $\hat{q} = \text{ScAVG}(q_i, \dots, q_n)$ minimizes the sum of squared screw distances $\sum_i d_{SC}(q_i, \hat{q}, \mathbf{p})$ for any point $\mathbf{p} \in \mathbb{R}^3$ [21]. That is, by measuring along the curved screw path, the transformation that minimizes the distortion does not depend on the points selected, which was arguably the most problematic aspect with the Euclidean distortion adopted by Pulli. Further, when the variation in orientation among the dual quaternions $q_i, \dots, q_n$ is very small, we have that $d_{SC}(T_{ij}V_j, V_i, \mathbf{p}) \approx ||T_{ij}V_j\mathbf{p} - V_i\mathbf{p}||$ for any point $\mathbf{p} \in \mathbb{R}^3$. In fact, we have $d_{SC}(T_{ij}V_j, V_i, \mathbf{p})^2 = {d_E^{\parallel}}^2 + \frac{\theta/2}{\sin(\theta/2)} {d_E^{\perp}}^2$ where θ is the rotation angle, and $d_E^{\parallel}$ and $d_E^{\perp}$ are respectively the components of the Euclidean distance parallel and orthogonal to the axis.

The optimal multiview alignment is thus obtained by computing the steady-state of the following process

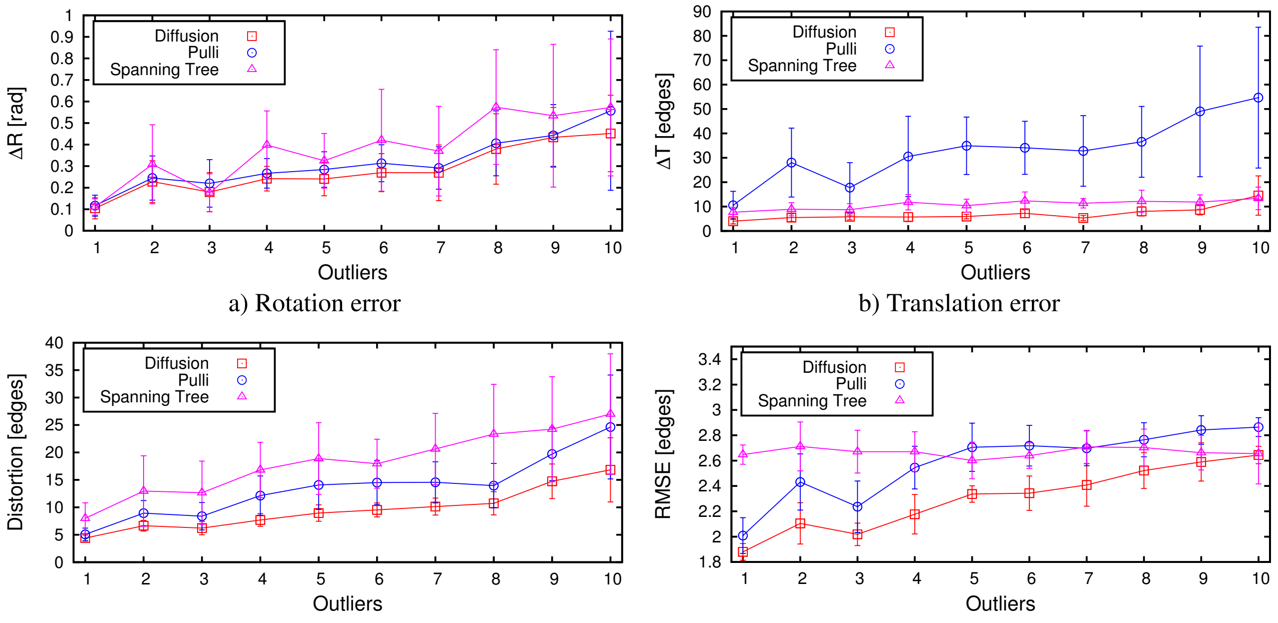
$$V_i^{t+1} = \underset{j \in N(i)}{\text{ScAVG}}(\pm T_{ij} V_j^t)$$

where the sign uncertainty is a consequence of the sign uncertainty in the dual quaternion representation, and is chosen so that $\pm T_{ij}V_j^t \cdot V_i^t > 0$. Further, since for small and moderate rotational variability in the vectors the linear average approximates well the screw average, while being much faster, in all our experiments we are using linear averages.

Finally, the proposed approach is a refinement method that requires initial motion estimates, but these can be computed by simple composition of the transformations along adjacent views with a breadth-first visit starting from the view 0. A demo application and code is available at `http://www.dsi.unive.it/~rodola/`.

4. Experimental Evaluation

Multiview registration techniques have both a sparse and diverse coverage in literature, and as such they suffer from the lack of a robust and fair methodology for performance assessment and comparison. Specifically, in real scenarios, where ground-truth data is not available, it can be very hard to evaluate and quantify the results of a global alignment and settle for a solution, without resorting to a thorough and time-consuming analysis of the registered views.



a) Rotation error

b) Translation error

c) Euclidean distortion

d) RMS error

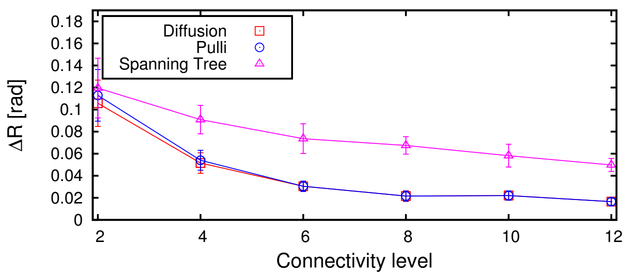
Figure 2. Comparison of the different methods in the synthetic experiments with different numbers of outliers.

To this end, we performed a wide range of experiments with both synthetic and real-world data. For each complete 3D model (from Georgia Tech's large geometric models archive[1]), a total of 36 orthographic snapshots were taken, each at a different angle of view; these, together with the ground-truth rigid motions used to produce the range images, constitute the dataset over which synthetic experiments were performed. In all the experiments we compare our method against Pulli's algorithm (as implemented by the author in the Scanalyze[2] software package), currently the method of choice in many applications. We evaluated the performance of the two algorithms, together with the initialization results obtained through a breadth-first coverage of the view graph (indicated as *Spanning Tree*), under different noise conditions and connectivity levels. For all the experiments we show the relative displacement with respect to ground-truth motion (with ΔR being the angle between the two unit quaternions, and ΔT the translation error expressed in median edge length), the RMS error among all the ranges and range 0 (point pairs were obtained through normal-shooting in both directions), and the Euclidean distortion metric adopted by Pulli (indicated as *Distortion*).

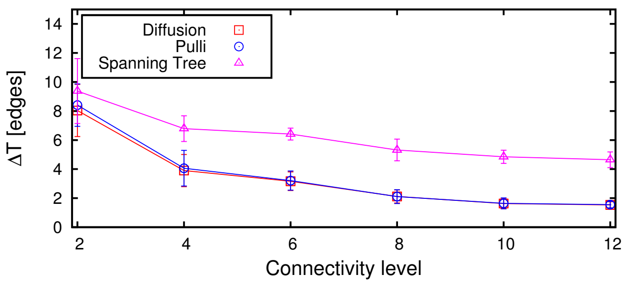
In Fig. 1 we show the results at different levels of initial displacement, where every view in the graph is connected with the next two in a ring topology. *Noise level* refers to a quantity which is proportional to the amount of Gaussian noise applied to the ground-truth pairwise motions, and ranges from a few units of edges and radians to tens of units; for each noise level, 10 independent runs of each method were performed. Both the Diffusion and Pulli methods compensate well rotational errors and are comparably good at low levels of noise, whereas the latter is outperformed when positional noise increases, both in terms of translation error and Euclidean distortion. It is worth noting that when we perturbed either the rotational or translational part of the rigid motion, keeping the other fixed, Spanning Tree and Pulli's methods performed a joint optimization modifying both components, while our method never changes the already optimal part of the motion, yielding better results at all levels of noise.

In Fig. 2 we assess the resilience of the tested methods to the presence of outliers: starting from a close-to-optimal initialization, we introduced strong pairwise misalignments so as to simulate a realistic scenario in which pairwise registrations get stuck at local minima. Connectivity is the same as in the previous experiments. In these graphs, the x-axis grows with the number of such mis-registrations; here, 20 runs were performed at each level, and for each run random pairs were picked from a uniform distribution and perturbed strongly. It can be seen that all the methods handle well rotational errors, with the Diffusion method giving particularly good performance constantly, even when the number of outlying pairs becomes large. Fig. 2b) and Fig. 2d) interestingly show the inability of Pulli's method to deliver good results in such situations: this is an inherent weakness of the method, since it acts in such a way to minimize all

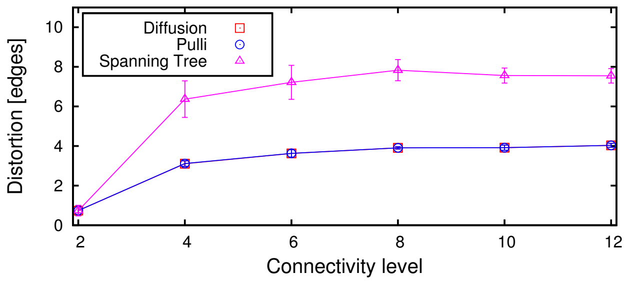
[1]http://www.cc.gatech.edu/projects/large_models
[2]http://www.graphics.stanford.edu/software/scanalyze



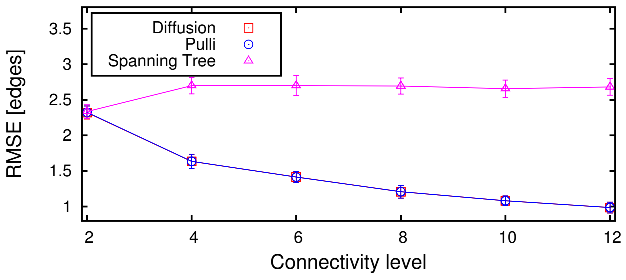
a) Rotation error



b) Translation error



c) Euclidean distortion



d) RMS error

Figure 3. Comparison of the different methods in the synthetic experiments at various connectivity levels.

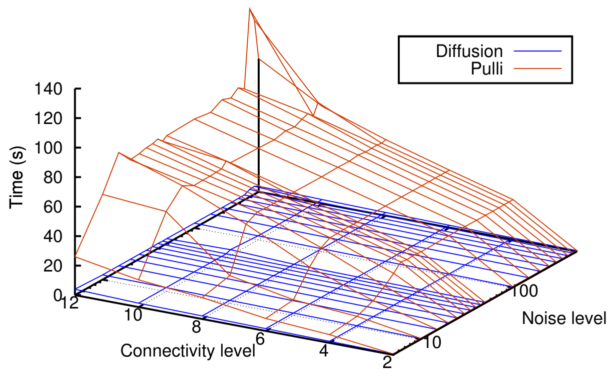


Figure 4. Computation times versus noise and connectivity levels.

motion, relying on the assumption that pairwise alignments are nearly perfect.

The next set of experiments (Fig. 3) is aimed at studying the effect of view-graph connectivity on the registration results. In these figures, *Connectivity level* refers to the number of links per view. As it can be readily seen, performance tends to increase with the number of edges in the view graph, as all three methods greatly benefit from more structure being brought in. It must be noted, however, that connectivity augmentation dramatically increases the time requirements of Pulli's method, bringing up convergence times by orders of magnitude (see Fig. 4).

Figure 4 compares the average convergence time of the proposed approach with the time required by Pulli's method. The times are shown as a function of noise level and connectivity. We can see that, with the exception of extreme values, the times required by Pulli's approach are independent from the level of noise, but grow linearly with the connectivity level and are in almost all tested situations in the order of 10 to 100 seconds. The proposed approach exhibits the same independence with respect to noise and linear growth with respect to connectivity level, but it is always around 2 to 3 orders of magnitude faster.

In order to simulate a more realistic type of noise, we also tested the following setup: we perturbed the initial pairwise motion as for the first set of experiments, and then applied ICP to refine the alignment. This setup simulates a normal registration process with increasingly bad coarse registration, with the possibility that ICP gets stuck on local minima providing more structured outliers than the previous experiments. The results of this set of experiments can be seen in Fig. 5. Note that our approach and Pulli's method yield very similar results in all the metrics except for ΔT. This can be justified by the fact that, when caught in local minima, ICP slides the surface one over the other, resulting in strong translational outliers which Pulli's method cannot deal with. The very low RMSE derives from the fact that sliding along the surface each point still finds close-by mates on the other mesh. The proposed method, on the other hand, manages to smooth all the outliers effectively, yielding low errors in all the metrics.

Figure 6 shows an example of a real set of range images acquired with a scanner and aligned using Pulli's method and the proposed approach. While at a large scale the overall alignment appears similar, by examining closeups of various sections of the glasses we see that Pulli's method pro-

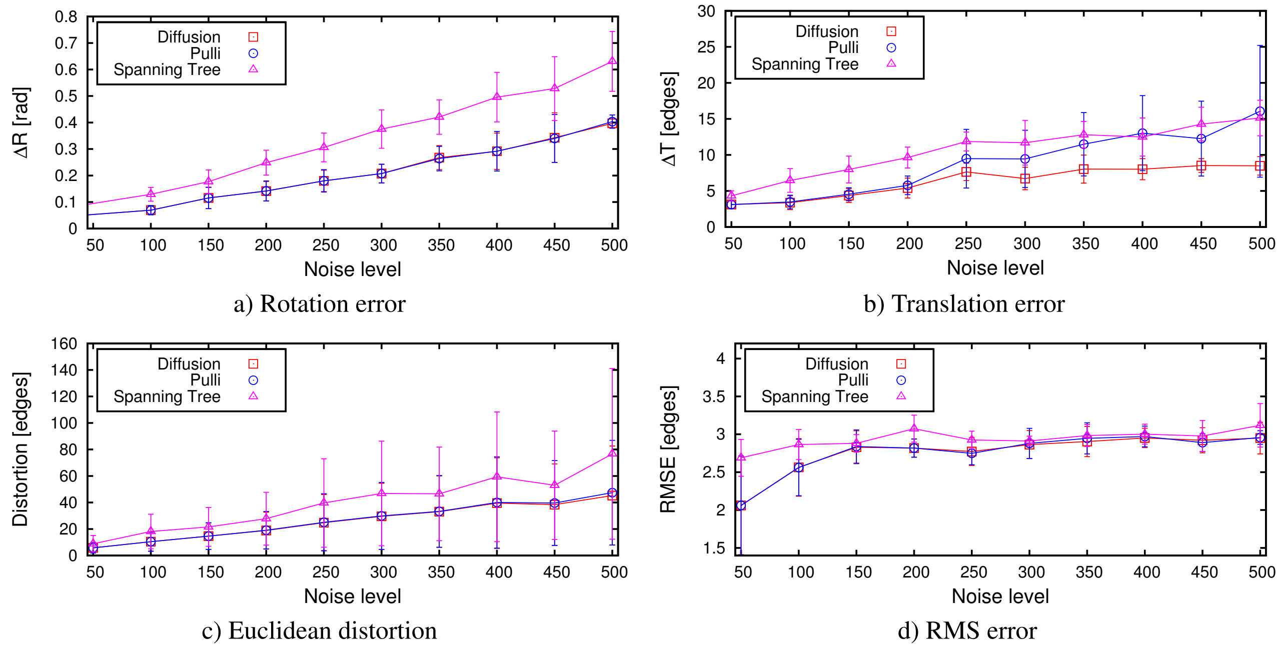


Figure 5. Comparison of the different methods in the experiments with motion refined by ICP at various levels of noise.

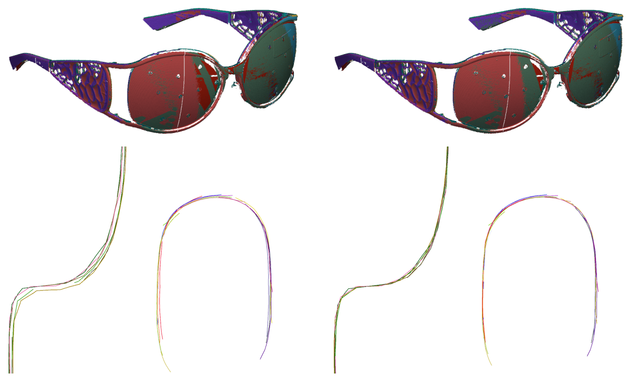


Figure 6. Global registration and closeup of slices of Pulli's method and our approach.

vides a slightly worse motion estimate, resulting in a wider stratification of the meshes.

It is worth noting that the orders of magnitude speedup provided by our approach makes it possible to run it several times combining it with pairwise registration. The idea is to alternate a few steps of ICP performed on all adjacent views with the diffusion. This way the diffusion process can be seen as a projection operator taking the incremental pairwise motion onto a set of consistent motion estimates. The advantage of this projection is that the constrained motion space smooths the energy profile of the resulting "global" ICP, reducing the risk of getting stuck in local minima. Figure 7 shows two examples of alignments obtained by performing ICP from bad initial motion estimates and then performing diffusion at the end (Trailing diffusion) and alternating between 10 steps of ICP and a diffusion process until convergence (Alternating diffusion). Clearly alternating pairwise registration allows to avoid local minima in these examples without incurring in any noticeable penalty in running times.

5. Conclusions

In this paper we proposed a novel multiview registration algorithm that projects several pairwise alignments onto a common reference frame. The projection is performed by representing the motions as dual quaternions which are then diffused along the graph of adjacent (i.e., pairwise alignable) views. The approach is general allowing for any topology of the view-adjacency graph.

An extensive set of experiments have shown that the proposed approach is both orders of magnitude faster than the state of the art, and more robust to extreme positional noise and outliers. Finally, the dramatic speedup of the approach allows it to be alternated with the pairwise alignment process resulting in a "global" ICP that exhibits a smoother energy profile, reducing the risk of getting stuck at local minima.

Acknowledgments

We acknowledge the financial support of the Future and Emerging Technology (FET) Programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open project SIMBAD grant no. 213250.

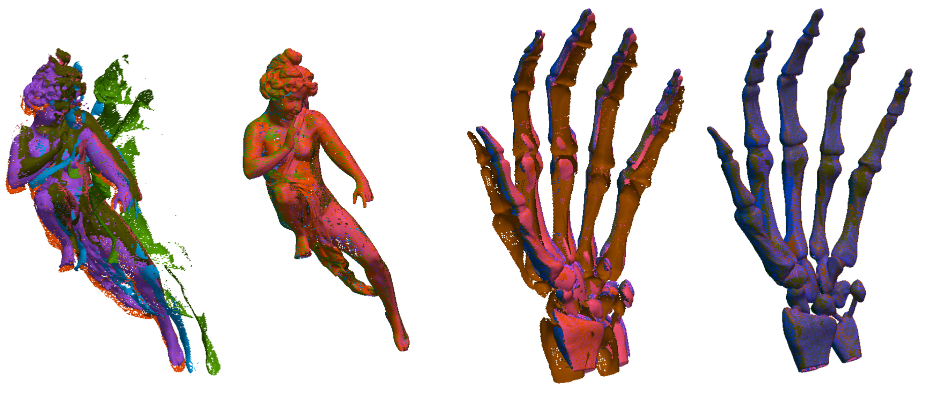


Figure 7. Alignments obtained with the trailing and alternating approaches, and respective timings.

References

[1] M. Agrawal. A lie algebraic approach for consistent pose registration for motion estimation. In *Proc. Int. Robotics Symposium*, 2006. 2442

[2] R. Benjemaa and F Schmitt. Fast global registration of 3d sampled surfaces using a multi-z-buffer technique. In *Proc. Int. Conf. on Recent Advances in 3-D Digital Imaging and Modeling*, pages 113–120, 1997. 2441

[3] R. Bergevin et al. Towards a general multi-view registration technique. *IEEE Trans. Patt. Anal. Machine Intell.*, 18(5):540–547, 1996. 2441

[4] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992. 2441

[5] S. R. Buss and J. P. Fillmore. Spherical averages and applications to spherical splines and interpolation. In *ACM Transactions on Graphics*, volume 20, pages 95–126, 2001. 2442

[6] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Image and Vision Computing*, pages 145–155, 1992. 2441

[7] W. Clifford. *Mathematical Papers*. Macmillan, London, 1882. 2442

[8] K. Daniilidis. Hand-eye calibration using dual quaternions. *Int. J. of Robotics Research*, 18:286–298, 1999. 2442

[9] D. W. Eggert, A. W. Fitzgibbon, and Fisher R. B. Simultaneous registration of multiple range views for use in reverse engineering. Technical Report 804, Dept. of Artificial Intelligence, University of Edinburgh, 1996. 2441, 2442

[10] S. Granger, X. Pennec, and A. Roche. Rigid point-surface registration using an em variant of icp for computer guided oral implantology. In *MICCAI*, pages 752–761, London, UK, 2001. Springer-Verlag. 2441

[11] H. Jin, T. Duchamp, H. Hoppe, J. A. McDonald, K. Pulli, and W. Stuetzle. Surface reconstruction from misregistered data. In *Proc. SPIE vol. 2573: Vision Geometry IV*, pages 32–328, 1995. 2442

[12] L. Kavan, S. Collins, J. J. Žára, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.*, 27(4):105, 2008. 2442, 2443

[13] L. Kavan, S. Collins, C. O'Sullivan, and J. Žára. Dual quaternions for rigid transformation blending. Technical Report TCD-CS-2006-46, Trinity College Dublin, 2006. 2443

[14] L. Kavan and J. Žára. Spherical blend skinning: a real-time deformation of articulated models. In *Proc. 2005 Symp. on Interactive 3D Graph. and Games*, pages 9–16, 2005. 2442

[15] Y. Liu. Replicator dynamics in the iterative process for accurate range image matching. *Int. J. Comput. Vision*, 83(1):30–56, 2009. 2441

[16] T. Masuda. Registration and integration of multiple range images by matching signed distance fields for object shape modeling. *Comput. Vis. Image Underst.*, 87(1-3):51–65, 2002. 2441

[17] J. M. McCarthy. *An Introduction to Theoretical Kinematics*. MIT Press, 1990. 2442

[18] K. Pulli. Multiview registration for large data sets. *Int. conf. on 3D Digital Imaging and Modeling*, pages 160–168, 1999. 2442, 2443

[19] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings of the Third Intl. Conf. on 3D Digital Imaging and Modeling*, pages 145–152, 2001. 2441

[20] K. Shoemake. Animating rotation with quaternion curves. In *Proc. of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, 1985. 2442

[21] A. Torsello. Point invariance of the screw tension minimizer. Technical Report DAIS-2011-2 `http://www.dsi.unive.it/~atorsell/DAIS-2011-2.pdf`, DAIS, Università Ca' Foscari Venezia, 2011. 2444

[22] J. Williams and M. Bennamoun. Simultaneous registration of multiple corresponding point sets. *Comput. Vis. Image Underst.*, 81(1):117–142, 2001. 2442

[23] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput. Vision*, 13(2):119–152, 1994. 2441

E Thermodynamic Depth Complexity of Networks

The Heat Diffusion - Thermodynamic Depth Complexity of Networks

Francisco Escolano* and Miguel A. Lozano†
University of Alicante, Spain

Edwin R. Hancock‡
University York, UK

(Dated: September 6, 2011)

In this paper we use the Birkhoff-von Neumann decomposition of the diffusion kernel to compute a polytopal measure of graph complexity. We decompose the diffusion kernel into a series of weighted Birkhoff combinations, and compute the entropy associated with the weighting proportions (polytopal complexity). The maximum entropy Birkhoff combination can be expressed in terms of matrix permanents. This allows us to introduce a phase-transition principle that links our definition of polytopal complexity to the heat flowing through the network at a given inverse temperature. The result is an efficiently computed complexity measure, which we refer to as flow complexity. Moreover, the flow complexity measure allows us to analyse graphs and networks in terms of Thermodynamic Depth (TD). We compare our method with three alternative methods described in the literature (Estrada's Heterogeneity Index, the Laplacian Energy and the von Neumann Entropy). Our study is based on 217 PPI networks including histidine kinases from several species of bacteria. We find a correlation between structural complexity and phylogeny (more evolved species have statistically more complex PPIs). Although our methods outperform the alternatives, we find similarities with Estrada's Heterogeneity Index in terms of network size independence and predictive power.

PACS numbers: 89.75.Fb, 89.75.Hc, 89.75.-k

I. INTRODUCTION

The quantification of the *intrinsic complexity* of undirected graphs and networks has attracted significant attention due to its fundamental practical importance, not only in network analysis [1] but also in other areas such as pattern recognition and control theory. Such a quantification not only allows the complexity of different graph or network structures to be compared, but also allows the complexity to be traded against goodness of fit to data when a structure is being learned in an information theoretic description length framework [2]. Some of the existing quantifications are easily computable, i.e., they have polynomical computational complexity, but others are not since they rely on NP-hard problems or non-computable quantities. In this paper we focus on the polynomial class of methods, although we also briefly discus the most influential methods falling into the second class. Instead, we focus in more detail on the dichotomy induced by distinguishing between *randomness complexity* and *statistical complexity.* Randomness complexity aims to quantify the degree of randomness or disorganisation of a combinatorial structure, this approach aims to characterise an observed graph structure probabilistically and compute its associated Shannon entropy. Statistical complexity, on the other hand, aims to characterise a combinatorial structure using statistical features such as node degree statistics, edge density or the Laplacian spectrum. Viewed historically, most early work in this area falls into the randomness class, while recent work is statistically based.

I.1. Related Literature

Turning our attention first to randomness complexity, one of the earliest contributions was Körner's graph entropy [3, 4]. The method poses complexity characrerisation as an optimization problem. Assuming there is a probability distribution associated with the vertices of the graph, the complexity is the minimal cross entropy between the probability distribution and the vertex packing polytope of the graph. Unfortunately the approach is not applicable to more general unweigthed graphs. Körner's approach relies on computing chromatic numbers which is in general an NP-complete problem, but is polynomial for perfect graphs. Recently, the von Neumann entropy (or quantum entropy) has been applied to the domain of graphs [5, 6] through a mapping between discrete Laplacians and quantum states [7]. If the discrete Laplacian [8] is scaled by the inverse of the volume of the graph we obtain a density matrix whose entropy can be computed using the spectrum of the discrete Laplacian. The measure distinguishes between different structures. For instance it is maximal for random graphs, minimal for complete ones and takes on intermediate values for star graphs. In addition, when there is degree heterogeneity then the Shannon (classical) and von Neumann (quantum information theoretic) entropies are correlated. Turning out attention to [9, 10], entropy is used to decompose graphs into substructures by grouping vertices are grouped based on the discrete entropy computed from local vertex density. In [11], this approach is gener-

* F. Escolano: sco@dccia.ua.es
† M.A. Lozano: malozano@dccia.ua.es
‡ E. Hancock: erh@cs.york.ac.uk

alized by using j-spheres (subgraphs around a vertex having a maximum topological distance j) to construct the required probability distribution over vertices. Moreover, this approach allows a complexity trace to be defined and this can be used to characterise phase changes in structure. In these two approaches, a complete graph must be as complex as any other graph with the same number of vertices. Off-diagonal complexity [12], which relies on taking statistics from the number of joint occurrences of pairs of nodes having a given pair of cardinalities. After normalization the method allows a discrete entropy to be computed. This entropy, takes on the value zero for both regular lattices and complete graphs, has small values for random graphs and large values for apparently complex structures.

The main drawback of randomness complexity is that it does not capture properly the correlations between vertices [13]. Statistical complexity aims to overcome this problem by measuring regularities beyond randomness, and does not necessarily grow monotonically with randomness. Both completely random systems and completely ordered ones must have a minimal statisttical complexity. Logical depth [14] (the time required by an universal Turing machine to run the minimal program that reproduces it) is the statistical counterpart of Kolmogorov-Chaitin [15, 16] randomness complexity because it is based on the notion of a process rather than a measure. In [17], graph complexity is characterised using the minimal dimension into which its corresponding intersection graph can be embedded. Jukna [18] uses similar ideas and defines the complexity of a graph to be the minimum number of union and intersection (i.e. Boolean operations) needed to obtain its complete set of edges commencing from stars. Neel's linear complexity of a graph [19] is given by the number of additions, subtractions and multiplications needed to perform matrix multiplication of the adjacency matrix with an arbitrary vector. Linear complexity is independent of the permutations of the vertices. If we reduce the original adjacency matrix by removing redundant rows (those with identical connectivity patterns), rows of zeros (isolated vertices) and rows with only a single one (vertices of unit degree), the linear complexity of the reduced matrix does not change. Relating the linear complexity of a reduced matrix to that of its transpose, we can obtain recursively the linear complexity of the original matrix. Gell-Mann's effective complexity [20, 21] (the length of a highly compressed description of its regularities) leads to quantifying a high degree of randomness in terms of high complexity, and regularity in terms of low complexity. One interesting property of effective complexity is its ability to distinguish purely random structures from regular ones. This property is shared with off-diagonal complexity. The latter approach is formally linked with the Minimum Description Length (MDL) principle, and has been used in practice for learning tree-structure [2].

Turning our attention to graph spectral methods, Song [22]) has recently explored the use of the Laplacian energy [23], i.e. the sum of absolute differences between the eigenvalues and the average vertex degree, as a complexity measure for graphs. In a regular graph, the Laplacian energy is equal to the energy of the graph (sum of absolute values of the eigenvalues of its adjacency matrix). The Laplacian energy is also low for graphs associated to polygons. An important recent addition to the graph-spectral literature is Estrada's network heterogeneity index [24]. The index gauges differences in degree for all pairs of connected vertices. The latter index can be expressed as a quadratic form of the Laplacian. It is equal to zero for both regular graphs and random ones, and is equal to one for the star graph. Finally, thermodynamic Depth [25] is another interesting measure which takes on low values for both random and ordered systems. The main problem in extending the method to graphs, is to define the required macroscopic states [26]. When dealing with graphs, thermodynamic depth (TD) aims to quantify the difficulty in contructing a given graph (the macroscopic state) from scratch (microscopic states). If there is little uncertainty about the process and all the possible causal trajectories have restricted variability, then the graph is narrow (simple). Otherwise, when historical uncertainty arises and many causal trajectories have been excluded for reaching a given structural design, then the graph is deep (complex).

I.2. Paper Overview

In this paper we link the ideas of information flow and thermodynamic depth to construct a novel characterisation of graph complexity. To commence, we establish a link between heat kernels and Birkhoff polytopes. This choice is motivated by the notion of Körner's entropy. The use of Birkhoff polytopes, on the oter hand, is morivated by the fact that the heat kernel of a graph can be decomposed into permutation matrices. We draw on the work of Birkhoff and von Neuman to characterize structural complexity in terms of the entropy of the polytopal decomposition coefficients. As the heat kernel is parameterized by time (or equivalently by inverse temperature) we obtain a complexity trace as inverse temperature evolves. In practice, we observe that these traces are endowed with a phase change. Unfortunately, the mathematical framework for polytopes makes it difficult to characterize the phase change. To overcome this problem, we show the relationship between heat kernels and the permanent of a graph. The permanent of a graph is a necessary prerequisite to computing the maximum entropy decomposition of the kernel. As the latter problem is known to be $\#P$, we derive a fast complexity measure (*flow complexity*) which has a similar qualitative behavior. With this representation to hand, we establish that the existence of a phase transition point in the flow complexity trace implies that the entropy of the corresponding Birkhoff decomposition is maximal.

As a result, we can assimilate phase transitions (maximum flow) into the maximum entropy framework. Once the flow complexity trace is defined, we proceed to specify a) the microstates, i.e. the nodes of the graphs and b) the causal histories, i.e. sequence of subgraphs rooted at each node, where each subgraph is generated from the previous one by reaching non-visited nodes through one edge). According to this picture, each node generates a history which is a path to the macroscopic state. Each history is characterized by a flow complexity trace. We quantify the historical variability with the minimal radius of the Bregman ball that encloses all traces. We also consider the projection of the flow complexity trace corresponding to the van der Waerden matrix to exploit the notion of entropy as the divergence from the uniform distribution. The proposed measure of depth combines both historical variability and divergence from the uniform. The experimental evaluation of the method focusses on the analysis of protein-protein interaction (PPI) networks. Given a key protein implied in signal transduction, our approach finds a significant correlation between *Heat Kernel - Thermodynamic Depth* complexity and the evolution of PPIs that include the protein in question in samples from different species of bacteria. We compare the predictive power of our proposed method with three alternative graph spectral complexity measures, namely a) Estrada's heteronegeity index, b) the von Neumann entropy and c) the Laplacian energy (all embedded in the framework of thermodynamic depth).

I.3. Illustration

As an example of how our approach works, in Fig. 1 we show the Heat Kernel - Thermodynamic Depth complexities for several graphs with the same number of nodes. As regularity increases complexity decreases. In our approach any complete graph has zero complexity. A complete graph is characterized by having the maximal regularity degree. In this example, it is particularly interesting to note the difference between the loop (zero complexity) and the linear graph. From a thermodynamic depth point of view, each node in the loop uses the same history to reach the macroscopic state. However, a non-zero thermodynamic depth is derived from the fact that each node has a *different view* of the process which leads to the emergence of the graph. The flow complexity traces register the amount of information flowing through the network at each particular inverse termperature. When the inverse temperature is zero, each node retains a unit of mass. As the inverse temperature increases, we eventually reach a "freezing point" where we have the same amount of heat-mass both in the nodes and in the edges. The equilibrium state of the diffusion process is always the van der Waerden matrix for any graph and there is no loss of heat-mass (conservation law). Consequently, part of the heat-mass encodes indirect or transitive relationships between the vertices of the graph, and which are not encoded by the graph itself. Different histories exist to the macroscopic state, due to high variability in the set of complexity traces emmanating from different vertices. In the case of complete graphs, the variability is zero. This provides a paradigmatic example of building a graph without imposing any constraint in the design (topology) of the microstate. Without constraints, information flows freely through the network. This situation is similar to that encountered in the case of the loop, which also has zero complexity because information flows isotropically or symmetrically. However, when we break a link of the loop we have that the rate of mass loss at the extreme vertices is smaller than that from the internal nodes. This is true for all inverse temperatures, but quantitively different for different histories, and this leads to a small but non-zero complexity.

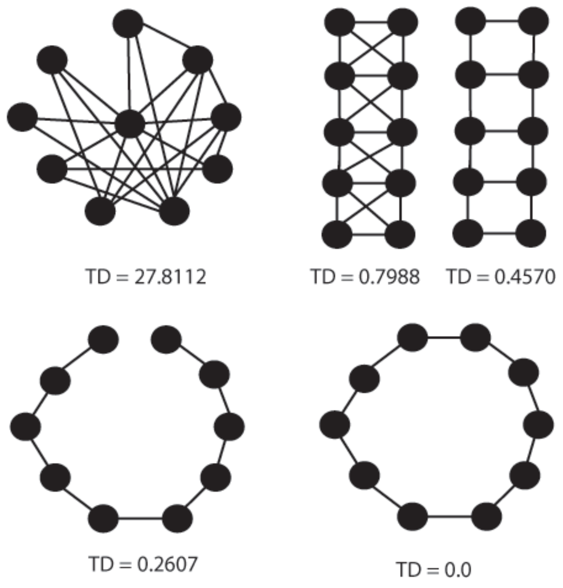


FIG. 1. Heat Kernel - Thermodynamic Depth complexities for graphs of $|V| = 10$.

II. PRELIMINARIES

In this section we review the prerequisites to the development of our complexity characterisation. We commence by defining the Birkhoff polytope, then explain links to the heat kernel and finally show how to define complexity using the Shannon entropy over the expansion co-efficients of the Birkoff-von Neumann decomposition of the heat kernel.

II.1. The Birkhoff Polytope

The $n-$th Birkhoff polytope $\mathcal{B}_n$ is the set of doubly stochastic matrices of dimension $n \times n$:

$$\mathcal{B}_n = \left\{ B = [b_{ij}]_{n\times n} \;:\; \begin{array}{c} \sum_i b_{ij} = 1,\ \forall j \\ \sum_j b_{ij} = 1,\ \forall i \\ b_{ij} \geq 0,\ \forall i \forall j \end{array} \right\} . \quad (1)$$

Such a polytope is convex, has dimension $(n-1)^2$ and its extreme points are the set of $n!$ permutation matrices $P = [p_{ij}]_{n\times n}$ (entries are 0 except for a single 1 in each row and column).

According to the **Birkhoff-von Neumann (BvN) Theorem**, Every doubly stochastic matrix (DSM) can be expressed as a convex combination of permutation matrices (PM) [27]:

$$B = \sum_\alpha p_\alpha P_\alpha,\ \forall B \in \mathcal{B}_n \text{ and } \begin{array}{c} \sum_\alpha p_\alpha = 1 \\ p_\alpha \geq 0\ \forall \alpha \end{array} . \quad (2)$$

Thus $\mathcal{B}_n$ is the convex hull of the set of the $n \times n$ permutation matrices. However, the representation of a DSM in terms of many PMs is not unique because $\mathcal{B}_n$ is not a simplex. The barycenter of $\mathcal{B}_n$ is the so-called van der Waerden constant matrix B_* with all entries equal to $1/n$.

II.2. Birkhoff von Neuman Decomposition of Heat Kernels

Consider an undirected graph $G = (V, E)$ where $|V| = n$ with $n \times n$ adjacency matrix A. The Laplacian of the graph is the $n \times n$ matrix $\mathcal{L} = D - A$ where $D = diag(d_i = \sum_{j \in V} A_{ij}, i \in V)$ is the degree matrix. The $n\times n$ *heat/diffusion kernel* $K_\beta(G)$ of the graph is the solution to the heat/diffusion equation: $\frac{\partial K_\beta}{\partial \beta} = -\mathcal{L}K_\beta$, and is given by the matrix exponentiation $K_\beta(G) = exp(-\beta\mathcal{L})$, (with $\beta \geq 0$). The matrix K_β is a doubly stochastic matrix *DSM*, where $K_{\beta_{ij}}$ encodes the probability of reaching vertex j from vertex i, and vice versa, through lazy random walks [28]. In fact the state vector of the random walk at time β is $q_\beta = K_\beta(G)q_0$, where q_0 is the intitial state vector of the walk. For a given value of β, the decomposition $K_\beta(G) = \sum_\alpha p_\alpha P_\alpha$ will not only map a graph (for a given β) into the Birkhoff polytope $\mathcal{B}_n$, but will also associate with the mapping a set of real numbers (the expansion coefficients) satisfying the axioms of probability. In other words it associates with the mapping a probability distribution over the convex combination of polytopes P_α. This defines the solution to a heat transportation problem for a given β. An interesting example is the complete graph of n vertices C_n whose diffusion kernel is the barycenter $K_{\beta>0}(C_n) = B_*$, that is, the van der Waerden matrix. In addition, the probability distribution, with exactly n coefficients, associated to the BvN decomposition is the uniform one $\mathcal{U}_n$ where $p_\alpha = 1/n$, $\forall\alpha$.

III. POLYTOPAL COMPLEXITY

Having established the link between the heat kernel and the Birkoff von Neumann decomposition, in this section we explot this link to define a complexity measure for a graph using the Shannon entropy over the expansion co-efficients.

III.1. The *BvN* Constructive Decomposition

We compute the BvN decomposition recursively. The procedure is as follows: (i) initially express the doubly stochastic matrix $K(G)$ as a convex combination of a single permutation matrix and residual doubly sotchastic matrix $K(G) \equiv K_1 = \lambda_1 P_1 + (1-\lambda_1)K_2$; and (ii) iteratively repeat this procedure using the decomposition $K_r = \lambda_r P_r + (1-\lambda_r)K_{r+1}$ until the final residual DSM is itself a permutation. Let γ be the number of permutations needed to encode K, satisfying $2 < \gamma < n^2$, and let r be the step number of the recursion process. With these ingredients, the decomposition is

$$\begin{aligned} K(G) &= \lambda_1 P_1 + (1-\lambda_1)\lambda_2 P_2 + (1-\lambda_1)(1-\lambda_2)\lambda_3 P_3 \ldots \\ &= \sum_{\alpha=1}^{\gamma} \underbrace{\{\prod_{r=1}^{\alpha-1}(1-\lambda_r)\}\lambda_\alpha}_{p_\alpha} P_\alpha = \sum_{\alpha=1}^{\gamma} p_\alpha P_\alpha\ , \end{aligned} \quad (3)$$

III.2. *BvN* Graph Complexity

Given an undirected and unweighted graph $G = (V, E)$ with diffusion kernel $K_\beta(G)$, and BvN decomposition $K_\beta(G) = \sum_{\alpha=1}^{\gamma} p_\alpha P_\alpha$, we define the *polytopal complexity of* G as the normalised entropy

$$\mathcal{BC}_\beta(G) = \frac{H(\mathcal{P})}{\log_2 n} \quad (4)$$

where $\mathcal{P} = \{p_1, \ldots, p_\gamma\}$ are is the set of Birkoff von Neumann expansion co-efficients and $H(\mathcal{P}) = \sum_{\alpha=1}^{\gamma} p_\alpha \log p_\alpha$ is the Shannon entropy. The definition depends on the value of β and also takes into account both the size of the graph and the number of components of the decomposition. For a complete graph C_n, where the set of poytope co-efficients is $\mathcal{U}_n$, our definition of graph complexity reduces to the entropy ratio $H(\mathcal{P})/H(\mathcal{U}_n)$. Moreover, when β approaches zero and infinity, we have the following limiting values

$$\lim_{\beta\to 0} \mathcal{BC}_\beta(G) = 0 \text{ and } \lim_{\beta\to\infty} \mathcal{BC}_\beta(G) = 1\ \forall G\ . \quad (5)$$

The polytopal complexity is determined by the structure of the heat-kernel and the value of β. To analyse the polytopal complexity in more detail, we perform the Taylor expansion

$$K_\beta(G) = e^{-\beta\mathcal{L}} = I_n - \beta\mathcal{L} + \frac{\beta^2}{2!}\mathcal{L}^2 - \frac{\beta^3}{3!}\mathcal{L}^3 + \dots\ , \quad (6)$$

where I_n is the $n \times n$ identity matrix, and $\lim_{\beta\to 0} K_\beta(G) \approx I_n - \beta\mathcal{L}$. That is, for low values of β the kernel is dominated by the Laplacian. Under these conditions, the role of I_n is to preserve the double stochasticity of the kernel. In terms of the diffusion process, having $K_\beta(G) = I_n$ at $\beta = 0$ means that every node still retains all its heat (the unit) and therefore the total heat in the system is n. As I_n is itself a member of the set of permuation matrices, its decomposition is trivial and so $\mathcal{BC}_{\beta=0}(G) = 0$.

To understand the behaviour of the polytopal complexity at large values of β we use the spectral decomposition of the heat kernel:

$$K_\beta(G) = e^{-\beta\lambda_1}\phi_1\phi_1^T + e^{-\beta\lambda_2}\phi_2\phi_2^T + \dots + e^{-\beta\lambda_n}\phi_n\phi_n^T\ , \quad (7)$$

where $\lambda_1 = 0 \leq \lambda_2 \leq \dots \leq \lambda_n$ are the ordered eigenvalues of $\mathcal{L}$ and $\phi_1, \phi_2, \dots, \phi_n$ are the corresponding eigenvectors. As a result for large β the spectrum of K_β is dominated by the smallest eigenvalues of $\mathcal{L}$. Since $\lambda_1 = 0$, and we are dealing with graphs having only a single connected component (i.e. the multiplicity of the smallest eigenvalue is one), we have that $\lim_{\beta\to\infty} K_\beta(G) \approx \phi_1\phi_1^T + e^{-\beta\lambda_2}\phi_2\phi_2^T$. Since $\phi_1 = \frac{1}{n}e$, where e is the all-ones vector of length n, then $\phi_1\phi_1^T = B_*$, the so-called van der Waarden matrix. As a result when β becomes large enough, say β_{max}, the diffusion process reaches the equilibrium state, that is, we have $\mathcal{BC}_{\beta_{max}}(G) \equiv K_{\beta>0}(C_n) = 1$. One consequence of this result is that a complete graph will have unit polytopal complexity.

The above analysis suggests that the graph complexity trace $\mathcal{BC}_\beta(G)$ is a signature of the interaction between the heat diffusion process and the structure/topology of the graph as β (and thus the range of interactions between vertices) changes. It can also be interpreted as a trajectory in $\mathcal{B}_n$ between the extreme point given by the identity permutation $P_I = I_n$ and the barycenter $B_* = K_{\beta>0}(C_n)$. The typical signature is heavy tailed, monotonically increasing from 0 to $\beta^+ = \arg\max\{\mathcal{BC}_\beta(G)\}$ and either monotonically decreasing or constant from β^+ to β_{max} (the smallest $\beta \in R^+$ satisfying the equilibrium condition). Thus, β^+ represents the most significant *topological phase transition* concerning the impact of the diffusion process on the topology of the input graph. In prior work [29, 30] we present experimental results that support the above hypothesis concerning the polytopal complexity trace. However, no way of characterizing this principle has been enunciated.

IV. POLYTOPAL VS HEAT FLOW COMPLEXITY

IV.1. Links with Matrix Permanents

The analysis of the behavior of the diffusion process in the interval $[0, \beta^+)$ is key to understanding the phase transition point. In such an interval, as β increases there is a decreasing number of elements of the heat kernel $K_\beta(G)$ satisfying the condition $K_{\beta_{ii}} \approx 1$, and equivalently an increasing number of elements for which $K_{\beta_{ij}} = 0,\ i \neq j$. This motivates the analysis of polytopal complexity in terms of *the rate of loss of perfect matchings over this interval.* Let A be the $n \times n$ adjacency matrix associated with a given heat kernel $K_{\beta_{ij}}$, such that $A_{ij} = 1$ if $K_{\beta_{ij}} > 0$ for a given value of β. The *permanent* of A is

$$per(A) = \sum_{\pi\in\mathcal{S}_n}\prod_{i=1}^{n} A_{i\pi(i)}\ , \quad (8)$$

where $\mathcal{S}_n$ is the set of permutations of $\{1, 2, \dots, n\}$ and $\pi(i)$ is the $i-$th element of the permutation π. As each product can only take on the value 1 when a perfect matching exists in the bipartite graph induced by A, the overall sum counts the number of perfect matchings in the bipartite graph. As $K_{\beta=0}(G) = I_n$ the minimum number of perfect matchings in the interval $[0, \beta^+]$ is 1.

For purposes of computing the polytopal complexity it is useful to note that $\forall B \in \mathcal{B}_n : 0 < per(B) \leq 1$ and $per(B) = 1$ and this implies that B is a permutation matrix. Thus, an alternative definition of graph complexity relies on $\mathcal{BC}^{'}_\beta(G) = 1 - per(K_\beta(G))$ defining the $\mathcal{BC}^{'}_\beta(G)$ profile in $[0, \beta_{max}]$. However, the latter profile is only informative in the interval $[0, \beta^+)$ and is typically flat and equal to 1 for $\beta \geq \beta^+$. In addition, the computation of the permanent is a #P problem and one of the best approximate algorithms [31] (the MCMC fully polynomial fully polynomial-time approximate scheme (FPTS)) takes on average $O(n^{23})$ only for generating a sample of the Markov chain. However, when small graphs are available we may use Ryser's exact algorithm which requires $\Theta(n2^n)$ operations.

The problem of finding $per(B)$ is closely related to that of finding the maximum entropy (and unique) *BvN* decomposition. As pointed out by Slater [32], the problem of finding such a decomposition is #P-hard, since it involves the computation of matrix permanents. Therefore, finding a unique *BvN* decomposition is intractable in practice unless the graph is small or can be simplified (for instance using the method proposed in [33]). Graph simplification is required for $n \geq 70$ even when we exploit the constructive proof of the Birkhoff-von Neumann theorem to find one of the possible decompositions. Unfortunately, since the number of iterations of the *BvN* decomposition is $O(n^2)$ and a Kuhn-Munkres algorithm $(O(n^3))$ is executed at each iteration, we have a $O(n^5)$

complexity per β value. This complexity precludes the use of the descriptor for practical analysis of complex networks. In addition the analysis of phase change is somewhat cumbersome in the polytopal framework. Thus, a new descriptor which is qualitatively similar to but more efficient than the current one is needed to provide a simpler analytical framework.

IV.2. Heat Flow Complexity

The connection between phase change at β^+ and the gain in the number of weighted perfect matchings, and consequently with the permanents of doubly stochastic matrices is an intriguing one. Although it does not yield a practical method for simplifying the computation of polytopal complexity, it has inspired the dynamic analysis of the *diffusion flow over the structure* as β changes. The spectral decomposition of the diffusion kernel is $K_\beta(G) = \exp(-\beta\mathcal{L}) \equiv \Phi\Lambda\Phi^T$, where $\Lambda = diag(e^{-\beta\lambda_1}, e^{-\beta\lambda_2}, \ldots, e^{-\beta\lambda_n})$, and $\Phi = [\phi_1, \phi_2, \ldots, \phi_n]$. Thus

$$K_{\beta_{ij}} = \sum_{k=1}^{n} \phi_k(i)\phi_k(j)e^{-\lambda_k\beta} , \qquad (9)$$

and $K_{\beta_{ij}} \in [0,1]$ is the (i,j)th entry of a doubly stochastic matrix. Double stochasticity for all β implies *heat conservation* in the system as a whole. That is, not only in the nodes and edges of the graph but also in the transitivity links eventually iteratively established between non-adjacent nodes. If i is not adjacent to j, eventually there will appear an entry $K_{\beta_{ij}} > 0$ when β is suffiently large. The overall amount of heat to be diffused is always $n = |V|$. The manner in which the available heat is diffused for $\beta \in [0, \beta_{max}]$ is highly dependent on the structure of the graph. One of the features of our approach is that for simplicity we do not incorporate *design* constraints. For instance, a linear graph of n nodes is more complex than a complete graph since the number of diffusion constraints is higher despite its high 2-regularity. A higher number of diffusion constraints implies the creation of more transitivity links as β grows. We illustrate this process in Fig. 2 where we show the heat kernel for a linear graph of $n = 100$ nodes.

The latter example suggests a simple way of defining a complexity trace which is qualitatively similar to that of the polytopal case. This new trace accounts for the total heat flowing through the graph at a given β *(instantaneous flow)* which is given by

$$F_\beta(G) = \sum_{i=1}^{n}\sum_{j\neq i}^{n} \delta_{ij}\left(\sum_{k=1}^{n} \phi_k(i)\phi_k(j)e^{-\lambda_k\beta}\right) , \qquad (10)$$

where $\delta_{ij} = 1$ if $A_{ij} = 1$ and $\delta_{ij} = 0$ otherwise, that is, if $\delta_{ij} = 1$ if $(i,j) \in E$. A more compact definition of the flow is $F_\beta(G) = A : K_\beta$, where $X : Z = \sum_{ij} X_{ij}Z_{ij} = trace(XZ^T)$ is the Frobenius inner product. Instantaneous flow accounts only for the heat flowing through the edges of the graph. Neither the heat remaining in the nodes nor the heat flow corresponding to the transitivity links are accounted for. The limiting cases are $F_0 = 0$ and $F_{\beta_{max}} = \frac{2|E|}{n}$.

Therefore the *heat flow complexity* $\mathcal{FC}_\beta(G)$ is a $\beta-$dependent function simply defined as the instantaneous flow $F_\beta(G)$. The edge-normalized version of the instantaneously flow $\mathcal{FC}_\beta(G) = n/(2|E|)F_\beta(G)$ can be used to satisfy the conditions

$$\lim_{\beta\to 0} \mathcal{FC}_\beta(G) = 0 \text{ and } \lim_{\beta\to\infty} \mathcal{FC}_\beta(G) = 1\ \forall G\ . \qquad (11)$$

Heat flow complexity takes $O(n^2)$ for each β, and is not normalized in the same way as the polytopal complexity. Nonetheless, its complexity trace is qualitatively similar to that of polytopal complexity.

IV.3. Characterization of Polytopal and Flow Complexity

By formally linking the complexity traces for both the polytopal and the heat flow complexity we are able to not only better understand the heat flow process and matrix permanents, but can also justify the practical use of a simple structural complexity measure (heat flow trace).

Given an undirected graph $G = (V, E)$ with node set V ($|V| = n$) and edge set E, the existence of a *Phase Transition Point* (PTP) at a $\beta^+ \geq 0$ can be inferred from the analysis of the difference between the sum of the off-diagonal elements of the diffusion kernel $K_\beta(G)$ and its trace (sum of diagonal elements). For $K_{\beta^+}(G)$ we have $\sum_{i=1}^{n}\sum_{j\neq i}^{n} K_{\beta^+_{ij}} < trace(K_{\beta^+})$ whereas for $\beta > \beta^+$ we always obtain $\sum_{i=1}^{n}\sum_{j\neq i}^{n} K_{\beta_{ij}} \geq trace(K_\beta)$. This means that at a PTP, the quantity $\Xi_\beta = trace(K_\beta) - \sum_{i=1}^{n}\sum_{j\neq i}^{n} K_{\beta_{ij}}$ is negative. At $K_0 = I_n$ we have $\Xi_0 = n$ and at $K_\infty = \mathcal{B}_*$ we obtain $\Xi_\infty = 1 - (n-1) = 2 - n$. In practice, the value $\Xi_\infty = 2 - n$ may be reached as soon as the kernel converges to $\mathcal{B}_*$ (i.e. it reaches the *equilibrium point*). Local maxima of Ξ_β are precluded by the monotonic nature of the diffusion process. Therefore Ξ_β is a monotonically decreasing function with a minimum at equilibrium

The existence of a unique PTP is key to relating heat flow and maximal entropy. However, it is not sufficient to enunciate a characterization principle for the change of phase, which involves a dependency between heat flow and entropy. Such a principle is the

Phase Transition Principle: *Let $\beta^+ > 0$ define a PTP. Then, the heat flow $F_{\beta^+}(G)$ is maximal among all choices of β. This implies that if $K_{\beta^+}(G) = \sum_{\alpha=1}^{\gamma} p_\alpha P_\alpha$ is the maximum entropy BvN decomposition for β^+ then its entropy, i.e. $H_{\beta^+}(\mathcal{P})$ with $\mathcal{P} = \{p_1, \ldots, p_\gamma\}$, is maximal with respect to all maximum entropy decompositions*

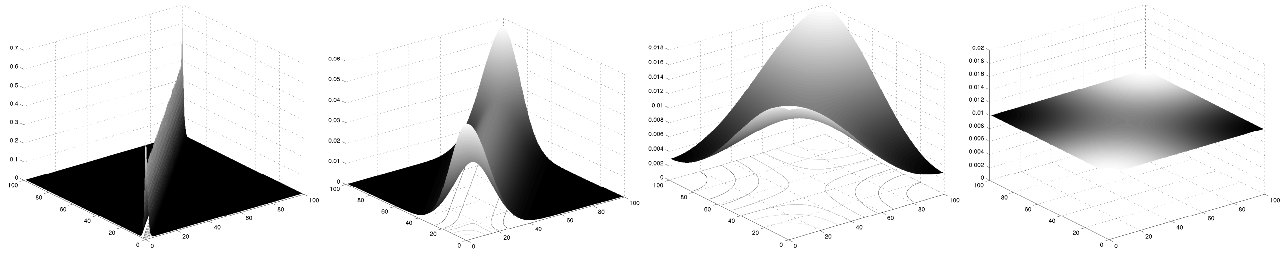


FIG. 2. Diffusion process in a linear graph with $n = 100$ nodes. From $\beta = 0.64$ to $\beta = 6,070$ (left to right). In each case lighter gray means higher $K_{\beta_{ij}}$. In all cases the kernel overlays a contour plot. Extreme nodes (1 and 100) loose less heat that the rest of nodes. As $\beta > 0.64$ increases, the closer the nodes to the extremes the less heat they loose. The true value of β_{max} is $10,717$ which gives an idea of the slow convergence of the diffusion process. In terms of flow we have: $F_{\beta=0.64} = 43.274$, $F_{\beta=100} = 6.5185$ and $F_{\beta=1,000} = 2.7483$. For $F_\beta = 1.9849$ we are close to the equilibrium flow ($\frac{2\times 99}{100} = 1.9800$)

in the range $[0, \beta_{max}]$.

In other words, flow maximality occurs at β^+ together with entropy maximality with respect to all *BvN* maximum-entropy decompositions.
For both $\beta < \beta^+$ and $\beta \geq \beta^+$ *flow maximality* relies on the elements of the kernel matrix associated with edge links (the first elements in the system receiving heat from the nodes). When $\beta < \beta^+$, elements associated with edge links must transfer heat to create transitivity links. Consequently the sum of their heat values decreases with respect to that at β^+. We can write $\sum_{ij} \delta_{ij} K_{ij} = A : K$, where A the adjacency matrix of G. It follows that $A : K_\beta < A : K_{\beta+}$, that is $F_{\beta+} > F_\beta$. When $\beta > \beta^+$ the individual values of both the diagonal and off-diagonal elements tend to $1/n$ as β increases. Elements associated with transitivity (indirect) links i.e. paths in the graph are always smaller than or equal to those associated with direct edge links. If the equilibrium is reached later than β, only the off-diagonal transitivity links increase. However, the elements associated with edge links decrease in value. This implies that $A : K_{\beta+} > A : K_{\beta+}$, that is $F_{\beta+} > F_\beta$. If equilibrium is reached at β^+, we have $F_{\beta+} = \frac{2|E|}{n}$.

Turning now to the *equivalence of flow maximality and entropy maximality* at β^+, we link the mechanisms driving a PTP with the dual formulation of finding the maximum entropy *BvN* decomposition. From [34] (Lemma 4) we have that the maximum entropy (ME) *BvN* decomposition of a *DSM* B is the solution the primal optimization problem (shown of the left) whose dual solution is shown on the right:

$$\begin{array}{llll} \min & \displaystyle\sum_{\alpha\in\mathcal{S}_n} p_\alpha(\log p_\alpha - 1) & \max & \displaystyle B : Y - \sum_\alpha e^{Y:P_\alpha} \\ s.t. & \displaystyle\sum_\alpha p_\alpha P_\alpha \leq B & s.t. & \displaystyle\sum_\alpha e^{(Y:P_\alpha)} P_\alpha \leq B \\ & p_\alpha \geq 0 & & 0 \geq Y \end{array} \tag{12}$$

where $\mathcal{S}_n$ is the set of permutations of $\{1, 2, \ldots, n\}$, and $Y \in R^{n\times n}$ is the matrix of Lagrange multipliers each corresponding to one constraint (component) in $B = \sum_\alpha p_\alpha P_\alpha$. Although the primal problem (left) is relaxed in the sense that the equality appearing in $\sum_\alpha p_\alpha P_\alpha \leq B$ is not required, the proof in Lemma 4 in [34] (see Appendix B.3) shows that it is impossible to have $\sum_\alpha p_\alpha P_\alpha < B$ for the optimal solution. Consequently, the relaxed primal problem shares its optima with the non-relaxed one. Anyway both problems are #P hard since $\sum_\alpha e^{Y:P_\alpha} = per(e^Y)$ (component-wise exponentiation). Therefore, the analysis of the second element of the phase transition principle (flow maximality-entropy maximality equivalence) relies on the analyis of how the Lagrange multipliers in Y interact with the the diffusion process encoded in $K_\beta = B$ at the optimal solutions (maximum entropy decompositions). For instance, as in the dual problem we must maximize $K_\beta : Y - \sum_\alpha e^{Y:P_\alpha}$, and we have that the optimal choice of the multipliers actually maximizes $K_\beta : Y - 1$. In addition, the relaxed formulation of the primal problem leads to a bound on the optimal multipliers in the sense that $-n\log n \leq K_\beta : Y \leq 0$ and $0 \geq Y_{ij} \geq \frac{-n\log n}{k_{min}}$ where $k_{min} = \min_{ij}\{K_{\beta_{ij}}\}$. Therefore, we reduce our analysis to ensure that $-n\log n \leq K_\beta : Y_\beta < K_{\beta+} : Y_{\beta+} \leq 0$ for $\beta \neq \beta^+$ in order to verify the flow maximality-entropy maximality equivalence. The rationale for this approach comes from analyzing the possibility of assigning close-to-zero multipliers to the maximum number of kernel elements in order to maximize the Frobenius product. The number of available close-to-zero multipliers decreases as β increases although this does not preclude us having a maximal $K_\beta : Y_\beta$ at β^+. The lower bound of $K_\beta : Y_\beta$ for a diffusion process occurs at $\beta = \beta_{max}$, and the bound is $K_{\beta_{max}} : Y_{\beta_{max}} = -\log n$.

V. HEAT FLOW - THERMODYNAMIC DEPTH COMPLEXITY

The application of thermodynamic depth (TD) to characterize network complexity demands the formal specification of the micro-states whose history leads to the macro-state (of the network). Here we define such micro-states in terms of a set of *expansion subgraphs*.

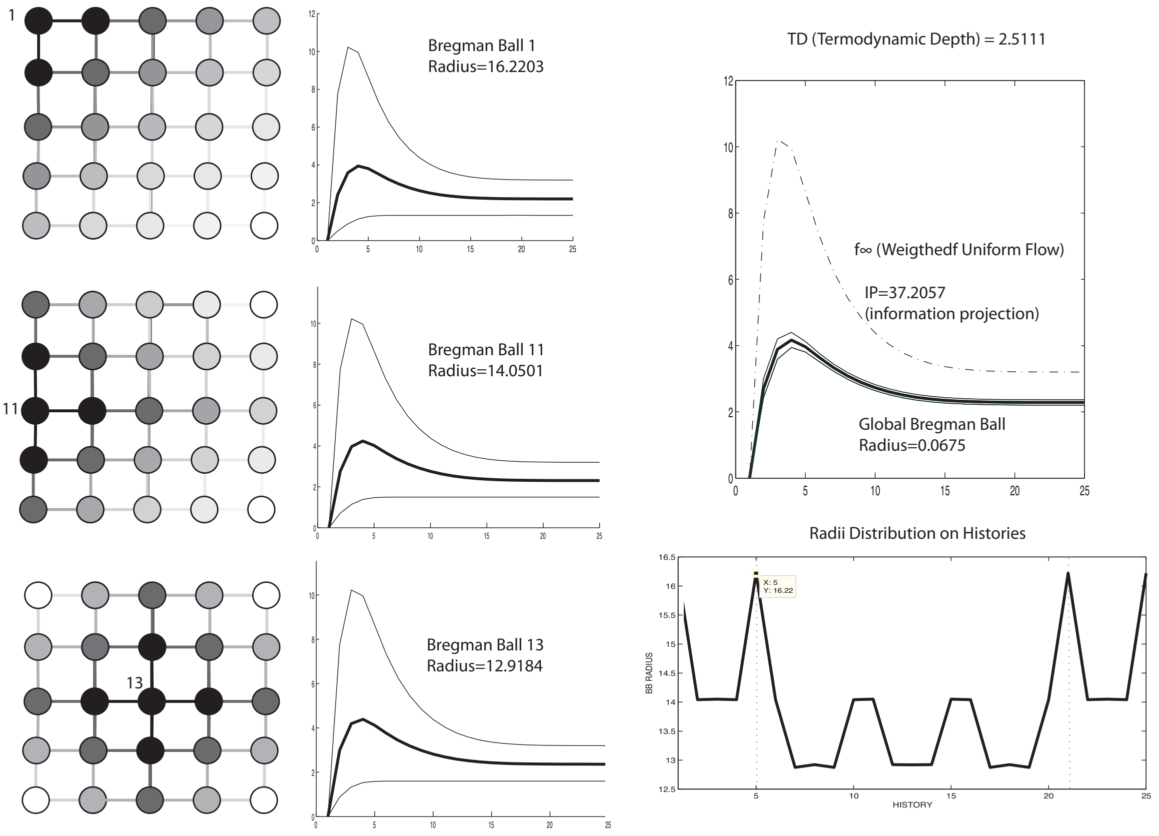


FIG. 3. Computing *Heat Flow-Thermodynamic Depth* of a 4-connected grid graph of $n = 25$ nodes. Left column: expansion subgraphs and node histories at nodes 1 (top), 11 (mid) and 13 (bottom); in each case we show the first-order expansion subgraph in black, the next one in dark gray, and so on. Center column: Minimum enclosing Bregman balls for the latter histories with their radii; in each case, the center of the ball (in bold) and the boundaries (given by the support vectors) are heat flow complexity traces inferred from the corresponding node history. Right column-top: projecting f^{∞} (van der Waerden trace) on the Bregman ball given by the centers of all the $n = 25$ balls. Right column-botton: plot of radii corresponding for all node histories; symmetry is given by the structure and the maximal radii correspond to nodes 1, 5, 21 and 25 (the four corners). In all cases I-KL divergence is used for building the Bregman balls.

V.1. Node History, Expansion Subgraphs and Causal Trajectories

Let $G = (V, E)$ with $|V| = n$. The *history of a node* $i \in V$ is $h_i(G) = \{e(i), e^2(i)), \ldots, e^p(i)\}$ where $e(i) \subseteq G$ is the *first-order expansion subgraph* given by i and all $j \sim i$, $e^2(i) = e(e(i)) \subseteq G$ is the *second-order expansion* consisting on $z \sim j : j \in V_{e(i)}, z \notin V_{e(i)}$, and so on until p cannot be increased. If G is connected $e^p(i) = G$, otherwise $e^p(i)$ is the connected component to which i belongs.

Every $h_i(G)$ defines a different causal trajectory leading to G itself, if it is connected, or to one of its connected components if it is not (see Fig. 3-left for a connected graph). In terms of thermodynamic depth (TD) the full graph G or the union of its connected components is the macro-state (macroscopic state). The *depth* of such a macro-state relies on the variability of the causal trajectories leading to it. The higher the variability, the more complex it is to explain how the macro-state is reached and the deeper is this state. In order to characterize each trajectory we combine the heat flow complexities of its expansion subgraphs by means of defining *minimal enclosing Bregman balls* (MEBB) [35]. The Bregman divergences D_F define an asymmetric family of similarity measures, each one characterized by a strictly convex generator function $F : \mathcal{X} \to R^+$, where $\mathcal{X} \subseteq R^d$ is a convex domain, and d the data dimension (in this case the number of discretized β i.e. inverse temperature). Given two patterns (discretized functions in this case) $\vec{f}$ and $\vec{g}$, $D_F(\vec{f}||\vec{g}) = F(\vec{f}) - F(\vec{g}) - (\vec{f} - \vec{f})^T \nabla F(\vec{f})$. Here, we use the I-Kullback-Leibler (I-KL) divergence $D_F(\vec{f}||\vec{g}) = \sum_{i=1}^{d} f_i \log \frac{f_i}{g_i} - \sum_{i=1}^{d} f_i + \sum_{i=1}^{d} g_i$ with $F(\vec{f}) = \sum_{i=1}^{d}(f_i \log f_i - f_i)$ (un-normalized Shannon entropy) which yields better results (more representative centroids of the heat flow complexities) than alternative divergence or distorion measures. Examples here include the Itakura-Saito divergence. When using the I-KL divergence in R^d, we have that $\nabla F(f_i) = \log f_i$ and also that $\nabla^{-1}F(f_i) = e^{f_i}$ (obviously the natural logarithm is assumed). Using these formal ingredients we define the *causal trajectory* in terms of MEBBs.

Given $h_i(G)$, the heat flow complexity $\vec{f_t} = f(e^t(i))$ for the $t-th$ expansion of i, a generator F and a Bregman divergence D_F, the *causal trajectory* leading to G (or one of its connected components) from i is characterized by the center $\vec{c_i} \in R^d$ and radius $r_i \in R$ of the MEBB $\mathcal{B}^{\vec{c}_i,r_i} = \{\vec{f_t} \in \mathcal{X} : D_F(\vec{c_i}||\vec{f_t}) \leq r_i\}$. Solving for the center and radius implies finding $\vec{c}^*$ and r^* minimizing r subject to $D_F(\vec{c_i}||\vec{f_t}) \leq r \; \forall t \in \mathcal{X}$ with $|\mathcal{X}| = T$. For the Lagrange multipliers α_t we have that $\vec{c}^* = \nabla^{-1}F(\sum_{t=1}^{T}\alpha_t\vec{f_t}\nabla F(\vec{f_t}))$. The efficient algorithm in [35] estimates both the center and the multipliers. The idea underpinning this method is closely related to Core Vector Machines [36]. It is interesting to focus on the non-zero multipliers (and their support vectors) used to compute the optimal radius. More precisely, the multipliers define a convex combination. As a result we have $\alpha_t \propto D_F(\vec{c}^*||\vec{f_t})$, and the radius is $r^* = \max_{\alpha_t>0} D_F(\vec{c}^*||\vec{f_t})$. In Fig. 3-center we show some Bregman balls corresponding to different nodes of a 4-connected grid graph.

V.2. Thermodynamic Depth of a Network

Given $G = (V, E)$, with $|V| = n$ and all the n centroid radius pairs $(\vec{c}_i, r_i)$, the *heat flow-thermodynamic depth complexity* of G is characterized by the MEBB $\mathcal{B}^{\vec{c},r} = \{\vec{c_t} \in \mathcal{X}_i : D_F(\vec{c}||\vec{c_i}) \leq r\}$ and $D_{min} = \min_{f\in\mathcal{B}^{\vec{c},r}} D_F(f^\infty||f)$, where $f^\infty = f(B_*) \in R^d$ is the van der Waerden complexity trace . As a result, the *TD depth of network* is given by $\mathcal{D}(G) = r \times D_{min}$.

The above definitions of complexity and depth are highly consistent with summarizing node histories to find a global causal trajectory which is as tightly bounded as possible. Here, r quantifies the *historical uncertainty.* The smaller r the simpler (shallower) is G. However, this is not sufficient for structures because many networks with quite different complexities may have the same value of r. To overcome this problem, we define the depth of the network in a manner to complement randomness complexity and as suggested by the thermodynamic depth approach. In our case, the projection of f^∞ onto the MEBB preserves the definition of entropy in terms of the distance to the uniform distribution (see Fig. 3-left). The combinations or hierarchies of MEBBs have proved to be more effective than ball trees for nearest-neighbor retrieval [37]. In the computation of depths, the Legendre duality (convex conjugate) is key because it establishes a one-to-one correspondence between the gradients ∇F and ∇F^{-1} due to the convexity of F. Therefore, the Bregman projection f of f^∞ onto the the border of $\mathcal{B}^{\vec{c},r}$ falls on the curve $f_\theta^{-1} = \theta\nabla F(\vec{c}) + (1-\theta)\nabla F(f^\infty)$ with $\theta \in [0, 1]$ and $f_\theta = \nabla^{-1}F(f_\theta^{-1})$. The projection f be easily found (approximately) through bisection search over θ.

Although we use the heat flow complexity trace for constructing the Bregman balls, the thermodynamic depth approach can be applied to any structural complexity measure (we will compare some of them in the experimental section). One key ingredient of our proposal, which is especially useful when the basic complexity measure (e.g. heat flow trace) is based on spectral graph theory, is the way we construct node histories. Node histories rely on extending subgraphs and then computing the basic measure. If such a measure is spectrally-based, then it can not distinguish structurally different graphs that are iso-spectral. In addition, since only the variability of the centers is considered, the thermodynamic depth approach introduces partial independence of graph size. As a result graphs with more nodes do not have necessarily have higher complexity.

VI. COMPLEXITY OF PROTEIN-PROTEIN INTERACTION NETWORKS

VI.1. Experimental Setup

We have performed experiments using protein-protein interaction networks (PPI's) extracted using STRING–8.2 [38] (`http://string82.embl.de/`). STRING–8.2 is a software tool that allows the user to select a protein and then a species (typically bacteria). In our experiments we are interested in *histidine kinase* which is a key protein in the development of signal transduction in bacteria. Signal transduction allows to adjust metabolic processes to adapt to environmental changes and, thus,it is a fundamental mechanism for their survival, even in extreme external conditions. Signal transduction implies a reversible protein phosphorylation mechanism (energy consumption) and histidine kinase is an enzime which activate another internal molecules producing enzimatic cascades inside the cell . Histidine kinase enables the transfer of growing factors through the membrane, which implies a significant role in the feeding of bacteria. Therefore, tracing histidine kinase complexity is closely related to trace bacteria survival, that is, bacterial evolution. Given the histidine kinase and the bacteria species we can select one of the proteins of the same family (query protein) and then STRING–8.2 composes a PPI network centered on the latter protein. We only consider links based on physical and functional interactions. Other settings in STRING–8.2 are a) selecting high-confidence interactions (above 0.7), b) a maximum of 50 interactions per node and c) 5 levels of interaction. These are customizable parameters in the STRING interface. Lower confidence and interactions yield simple and/or noisy networks.

Our first experimental study considers PPIs related to *histidine kinase* corresponding to 10 species belonging to 10 different phyla of bacteria. We select 3 PPIs from each species corresponding to simple, medium complexity and high-complexity structures, and compute their thermodynamic depths. In 70% of the cases, the thermodynamic depth matches intuition. When compared with Estrada's early spectral homogeneity descriptor [39], we find that the ratio between intraclass and interclass variability is slightly better (smaller) for thermodynamic depth (0.6840 vs 0.7383).

Our main investigation consists of analyzing 217 PPIs, related to histidine kinase, from 6 different groups (all the PPIs in the same group correspond to the same species) with an following evolutive order from older to more recent. The groups are a) *Aquifex* –4 PPIs from *Aquifex aeloicus*, b) *Thermotoga*–4 PPIs from *Thermotoga maritima*, c) *Gram-Positive*–52 PPIs from *Staphylococcus aureus*, d) *Cyanobacteria*–73 PPIs from *Anabaena variabilis*, and e) *Proteobacteria*–40 PPIs from *Acidovorax avenae*. There is an additional class f) (*Acidobacteria*—46 PPIs) whose bacterial evolution since discovery is more controversial [40]. There are studies which relate many Acidobacteria to different sub-phyla of *Proteobacteria* (see [41]) and also to *Actinobacteria* [42]. However, the *Candidatus Koribacter* genus of Acidobacteria is not included in the latter classifications [41]. One of them is the *Candidatus Koribacter versatilis Ellin345*, despite sharing the property of being Gram-negative with some Proteobacteria. In addition, *Ellin345* has been recently placed very early in the phylogenic tree [43]. In Table I we show the origin of all of the 217 PPIs analyzed in this experiment, including the ones related to *Ellin345*. In Fig. 4 we show one example PPI corresponding for each of the 6 species together with the complexities obtained using our method.

VI.2. Experimental results

In order to provide a fair comparison between our method (heat flow-thermodynamic depth) and previous measures of structural complexity, we have chosen three spectral methods from the literature (Estrada's network heterogeneity index [24], the Laplacian energy [23] and von Neumann entropy [5, 6]). We have also embedded these three spectral characterisations into the thermodynamic depth approach. As we have different thermodynamic depth complexities for each method, histogramming reveals a typically long tailed distributions with most of the thermodynamic depths concentrated in a narrow range. We aim to determine whether the thermodynamic depths are ordered according to the evolutive order. This question can be answered by studying the cumulative distributions instead of the histograms. Examples are shown in Fig. 5. Here we have set the abcisas bounds in order to give the best results obtainable with each method. If the cumulative distribution saturates rapidly, then this indicates a low value of thermodynamics depth, whereas slow saturation high thermodynamic depth. This observation motivates using the area under the cumulative distribution (AUC) as a charcterisation. The greater the AUC the (statistically) simpler the PPIs. We show the values of AUC in Table II, where incorrect orderings (those not consistent with evolutibve order) are shown in bold. The joint analyisis of both the cumulative distributions and their AUC values shows that Thermodynamic-depth-Heat Diffusion yields the best results. When combined with thermodynamic depth, then Estrada's Heterogeneity Index gives results that are close in quality, except that it underestimates the complexity of both the Gram-possitive and Acidobacteria species. When combined with thermodynamic depth, the Laplacian energy overestimates the complexity of Thermotoga-Aquifex and Proteobacteria, whereas it underestimates the complexity of Acidobacteria. Finally, when combined with thermodynamic depth the von Neumann Entropy overestimates the complexity of Thermotoga-Aquifex and underestimates the complexity of Acidobacteria.

The results obtained above are consistent with the degree dependency of network complexity upon network size introduced by each of the characterisations. As we show in Fig. 6, the two best methods (Heat Flow and Heterogeneity) are quite independent of size, whereas the others (Laplacian Energy and von Neumann Entropy) are highly depedent on the size of the network. As we show in Table I, most of the PPI classes are quite heterogeneous (high variability). Additionally, there are networks which are significantly smaller than the remainder, but which have greater complexity, and viceversa (see Fig. 4).

Thus, we can conclude that thermodynamic depth combined with Heat Flow is an effective and principled tool for analaying the complexity of networks.

VII. CONCLUSIONS

In this paper, we have proposed and successfully tested a novel measure of graph complexity, the so called Heat Flow-Thermodynamic Depth. We have characterized formally its link with Birkhoff polytopes and polytopal complexity. The link comes from the observation that high-entropic decompositions correspond to maximal flow, and this has been established through exploring the connections between polytopal complexity and matrix permanents, and also by analyzing the dynamics of heat diffusion in graphs. In addition, we have also proposed a theoretical framework for Thermodynamic Depth, into which any structural complexity measure can be embedded. This framework reduces the risk of iso-spectrality when measures drawn from spectral graph theory are used. We compare our method with three alternative methods reported in the literature, namely a) Estrada's Heterogeneity, b) the Laplacian Energy and c) the von Neumann Entropy. These methods are also embedded into our Thermodynamic depth framework. Our experiments are based on the analysis of 217 PPI networks corresponding to several phyla of bacteria. We find that the more evolved species have (statistically) a more complex structure. Both Heat Flow and Estrada's Heterogeneity are proved to be quite independent on network size. Given the relatively good results obtained with the Estrada's Heterogeneity, our future work will include an in depth analysis of the formal connections between the methods.

[1] E. Estrada, M. Fox, D. Higham, and G.-L. O. (Eds), *Network Science: Complexity in Nature and Technology* (Springer, London, 2010).

[2] A. Torsello and E. Hancock, IEEE TPAMI **28**, 954 (2006).

[3] J. Körner, in *Trans. of the 6th Prague Conference on Information Theory* (1973) pp. 411–425.

[4] G. Simonyi, in *Perfect Graphs* (John Wiley and Sons, 2001) pp. 293–328.

[5] F. Passerini and S. Severini, "The von Neumann entropy of networks," (2008), arXiv:cond-mat.dis-nn/0812.2597v1.

[6] K. Anand, G. Bianconi, and S. Severini, "The Shannon and the Von Neumann entropy of random networks with heterogeneous expected degree," (2010), arXiv:cond-mat.dis-nn/1011.1565v2.

[7] S. Braunstein, S. Ghosh, and S. Severini, Ann. of Combinatorics **10**, 291 (2006).

[8] N. Biggs, *Algebraic graph theory* (Cambridge University Press, Cambridge Tracts in Mathematics, No. 67., 1974).

[9] D. Bonchev, *Information Theoretic Indices for Characterization of Chemical Structures* (Research Studies Press, Chichester, 1983).

[10] D. Bonchev, *Complexity in Chemistry. Introduction and Fundamentals* (Taylor and Francis, 2003).

[11] M. Dehmer, Applied Mathematics and Computation , 82 (2008).

[12] J. Claussen, Physica A **375**, 365 (2007).

[13] D. Feldman and J. Crutchfield, Phys. Letters A , 244 (1998).

[14] C. Bennett, Found. Phys. , 585 (1986).

[15] A. Kolmogorov, Prob. Inf. Trans. **1** (1965).

[16] G. Chaitin, J. ACM. **13**, 547 (1965).

[17] Pudlák and V. Rödl, Combinatorica **12**, 221 (1992).

[18] S. Jukna, Combinatorics, Probability and Computing **15**, 855 (2006).

[19] D. Neal and M. Orrison, The Electronic Journal of Combinatorics **15**, R9 (2006).

[20] M. Gell-Mann, Complexity **1**, 16 (1995).

[21] M. Gell-Mann and S. Lloyd, in *Nonextensive Entropy-Interdisciplinary Applications* (2003) pp. 387–425.

[22] Y.-Z. Song, P. Arbelaez, P. Hall, C. Li, and A. Balikai, in *ECCV (4)* (2010) pp. 694–707.

[23] I. Gutman and B. Zhou, Linear Algebra and its Applications , 29 (2006).

[24] E. Estrada, Phys. Rev. E **82**, 066102 (2010).

[25] S. Lloyd and H. Pagels, Ann. Phys. , 186 (1988).

[26] J. Crutchfield and C. Shalizi, Physical Review E , 275 (1999).

[27] G. D. Birkhoff, Universidad Nacional de Tucuman Revista, Serie A **5**, 147 (1946).

[28] R. I. Kondor and J. Lafferty, in *ICML* (2002).

[29] F. Escolano, E. Hancock, and M. Lozano, in *ICPR* (2008) pp. 1–5.

[30] F. Escolano, E. Hancock, and M. Lozano, in *SSPR/SPR* (2008) pp. 237–246.

[31] M. Jerrum, A. Sinclair, and E. Vigoda, Journal of the ACM **51**, 671 (2004).

[32] P. B. Slater, Env. and Planning **21**, 1541 (1989).

[33] H. Qiu and E. Hancock, Pattern Recognition **40**, 2874 (2007).

[34] S. Agrawal, Z. Wang, and Y. Ye, in *WINE* (2008) pp. 126–137.

[35] R. Nock and F. Nielsen, in *ECML* (2005) pp. 649–656.

[36] I. Tsang, A. Kocsor, and J. Kwok, in *ICLM* (2007) pp. 911–918.

[37] L. Cayton, in *ICLM* (2008) pp. 112–119.

[38] L. Jensen, M. Kuhn, M. Stark, S. Chaffron, C. Creevey, J. Muller, T. Doerks, P. Mullien, A. Roth, M. Simonovic, P. Bork, and C. von Mering, Nuc. Acid Res. **37** (2009).

[39] E. Estrada, Europhys. Lett. **73**, 649 (2006).

[40] S. B. C.R. Kuske and J. Bush, Appl. Environ. Microbiol **63**, 3614 (1997).

[41] P. H. M. Sait and P. Janssen, Environ. Microbiol **4**, 654 (2002).

[42] F. Ciccarelli, F. Doerks, C. von Mering, C. Creevey, B. Snell, and P. Bork, Science **311**, 1283 (2006).

[43] M. Wu and J. Eisen, Genome Biol. **9** (2008).

TABLE I. Table with the bacteria species studied in our main experiment (first column). We include the mean and deviation of the PPI size (second column) and all the query proteins used (their respective number of nodes are inside brackets) for building PPIs with the settings described in text

Species	Size [a]	Proteins [b]
Aquifex aeolicus	204.2500 ± 2.7538	TM_0127(201) TM_0400(203) TM_0853(207) TM_1258(206)
Thermotoga maritima	206 ± 3.1623	hksP1 (210) hksP2 (207) hksP3 (203) hksP4 (204)
Staphilococcus aureus	75.6346 ± 67.1203	MW0199 (48) NWMN_1327 (200) NWMN_1741 (54) NWMN_2523 (14) SA0216 (39) SAB0162c (46) SAB1782c (56) SAB2499c (50) SACOL0202 (50) SACOL1739 (201) SACOL1906 (54) SACOL2645 (15) SAOUHSC_00185 (48) SAOUHSC_01799 (109) SAOUHSC_01981 (40) SAOUHSC_02099 (202) SAOUHSC_02955 (14) SAR0215 (43) SAR1940 (54) SAS0199 (49) SAS1770 (34) SAUSA300_0218 (45) SAUSA300_1799 (52) SAUSA300_2558 (14) SAV0224 (35) SAV1321 (204) SAV1849 (205) SAV2624 (205) SaurJH1_0214 (46) SaurJH1_1937 (42) SaurJH1_1973 (186) SaurJH9_0058 (31) SaurJH9_0208 (30) SaurJH9_0245 (16) SaurJH9_0729 (32) SaurJH9_1475 (32) SaurJH9_1749 (37) SaurJH9_1903 (17) SaurJH9_1939 (32) SaurJH9_2115 (22) SaurJH9_2647 (18) arlS (109) graS (32) hssS (109) kdpD (44) lytS (18) nreB (137) phoR (200) saeS (112) tycG (206) vraS (202) yhcS (43)
Anabena variabilis	75.4521 ± 68.4091	Ava_0024 (66) Ava_0055 (238) Ava_0062 (224) Ava_0064 (20) Ava_0066 (33) Ava_0066 (33) Ava_0355 (16) Ava_0413 (66) Ava_0505 (224) Ava_0521 (66) Ava_0612 (31) Ava_0647 (211) Ava_0792 (54) Ava_0799 (153) Ava_1003 (13) Ava_1005 (13) Ava_1149 (13) Ava_1149 (13) Ava_1168 (31) Ava_1175 (207) Ava_1191 (101) Ava_1210 (66) Ava_1486 (16) Ava_1559 (13) Ava_1628 (17) Ava_1719 (16) Ava_1954 (66) Ava_2027 (55) Ava_2149 (66) Ava_2152 (66) Ava_2176 (66) Ava_2328 (24) Ava_2420 (17) Ava_2439 (214) Ava_2466 (80) Ava_2524 (167) Ava_2562 (52) Ava_2563 (36) Ava_3003 (9) Ava_3004 (13) Ava_3207 (66) Ava_3226 (202) Ava_3368 (60) Ava_3467 (204) Ava_3526 (66) Ava_3721 (66) Ava_3779 (224) Ava_3850 (47) Ava_3852 (41) Ava_3854 (35) Ava_3865 (31) Ava_4086 (31) Ava_4135 (43) Ava_4136 (66) Ava_4267 (66) Ava_4305 (66) Ava_4325 (13) Ava_4326 (17) Ava_4343 (13) Ava_4345 (13) Ava_4401 (66) Ava_4432 (44) Ava_4433 (66) Ava_4457 (202) Ava_4696 (36) Ava_4723 (208) Ava_4723 (40) Ava_4783 (40) Ava_4885 (66) Ava_4928 (205) Ava_B0028 (37) Ava_B0190 (101) Ava_B01974 (101) Ava_B0208 (55) Ava_C0117 (32)
Acidovorax avenae	132.7556 ± 85.4209	Aave_0042 (161) Aave_0173 (126) Aave_0317 (6) Aave_0867 (202) Aave_0874 (204) Aave_0905 (205) Aave_1122 (8) Aave_1444 (208) Aave_1494 (9) Aave_1948 (211) Aave_2036 (56) Aave_2118 (201) Aave_2218 (76) Aave_2236 (231) Aave_2267 (203) Aave_2461 (207) Aave_2525 (27) Aave_2535 (25) Aave_2632 (205) Aave_2974 (232) Aave_2975 (155) Aave_2976 (155) Aave_3001 (149) Aave_3093 (230) Aave_3275 (161) Aave_3541 (203) Aave_3564 (26) Aave_3778 (3) Aave_3863 (8) Aave_3993 (204) Aave_3996 (204) Aave_4137 (202) Aave_4230 (8) Aave_4320 (21) Aave_4378 (220) Aave_4381 (8) Aave_4526 (209) Aave_4637 (161) Aave_4654 (201) Aave_4732 (206)
Acidobacterium sp. (Ellin345)	84.5435 ± 73.6875	Acid345_0310 (118) Acid345_0375 (150) Acid345_0419 (20) Acid345_0474 (136) Acid345_0507 (6) Acid345_0508 (6) Acid345_0821 (48) Acid345_0843 (20) Acid345_0938 (8) Acid345_0973 (77) Acid345_1393 (203) Acid345_1524 (215) Acid345_1590 (202) Acid345_1659 (29) Acid345_1692 (90) Acid345_1733 (24) Acid345_1771 (209) Acid345_1812 (8) Acid345_1814 (8) Acid345_2403 (30) Acid345_2411 (106) Acid345_2412 (140) Acid345_2442 (39) Acid345_2554 (25) Acid345_2603 (53) Acid345_2781 (140) Acid345_2895 (185) Acid345_2989 (210) Acid345_3062 (201) Acid345_3087 (77) Acid345_3100 (29) Acid345_3292 (20) Acid345_3488 (20) Acid345_3503 (104) Acid345_3712 (128) Acid345_3715 (20) Acid345_3749 (40) Acid345_3837 (20) Acid345_3845 (7) Acid345_4032 (210) Acid345_4055 (91) Acid345_4063 (120) Acid345_4169 (206) Acid345_4289 (77) Acid345_4461 (7) Acid345_4561 (7)

[a] $\mu \pm \sigma$

[b] with $|V|$ inside brackets

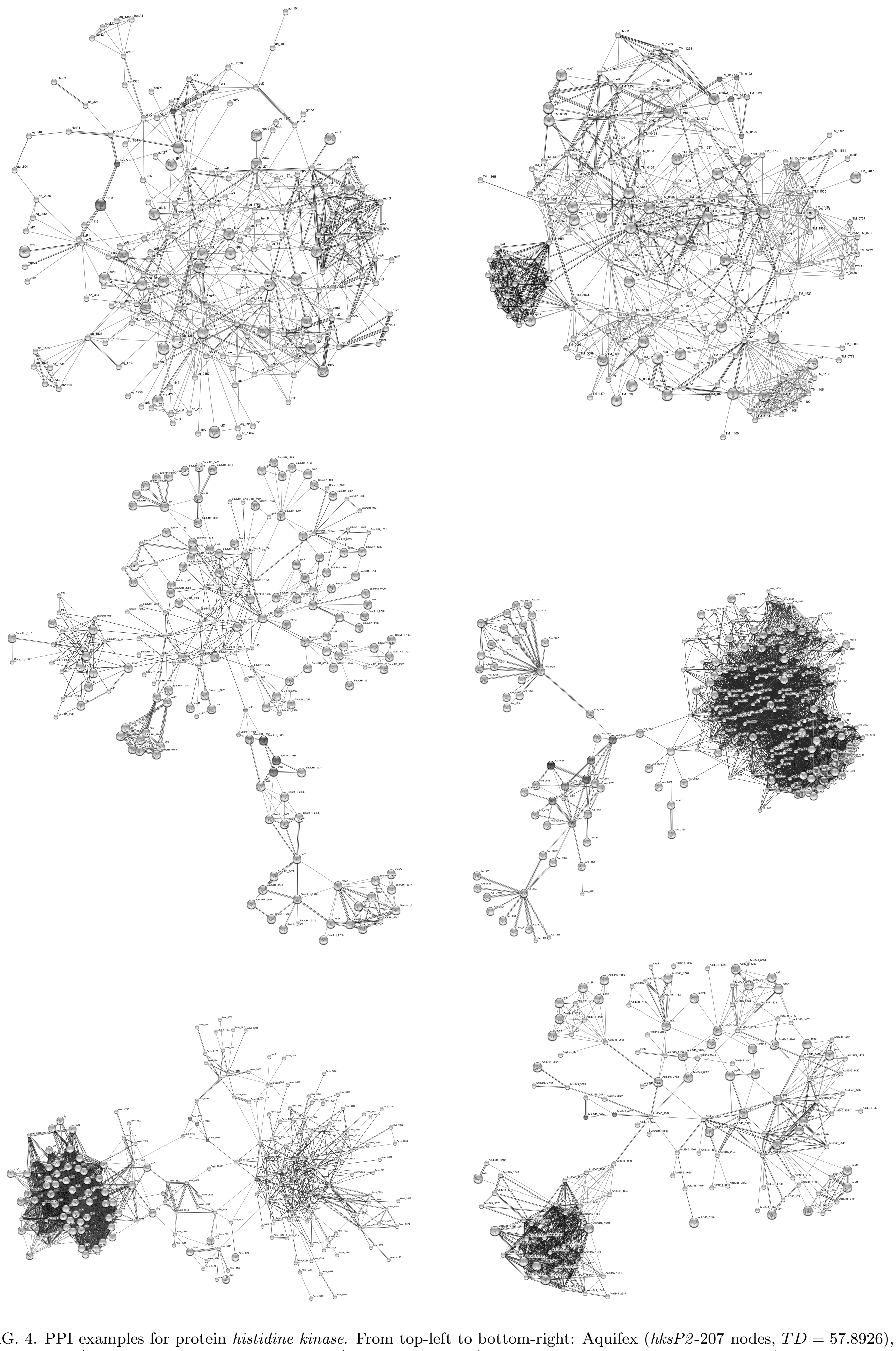


FIG. 4. PPI examples for protein *histidine kinase*. From top-left to bottom-right: Aquifex (*hksP2*-207 nodes, $TD = 57.8926$), Thermotoga (*TM_0127*-201 nodes, $TD = 69.2323$), Gram-possitive (*SaurJH1_1973*-186 nodes, $TD = 85.6689$), Cyanobacteria (*Ava_0062*-224 nodes, $TD = 4,6383$), Proteobacteria (*Aave_0867*-202 nodes,$TD = 58.3774$) and Acidobacteria (*Acid345_0474*-136 nodes, $TD = 618.1457$)

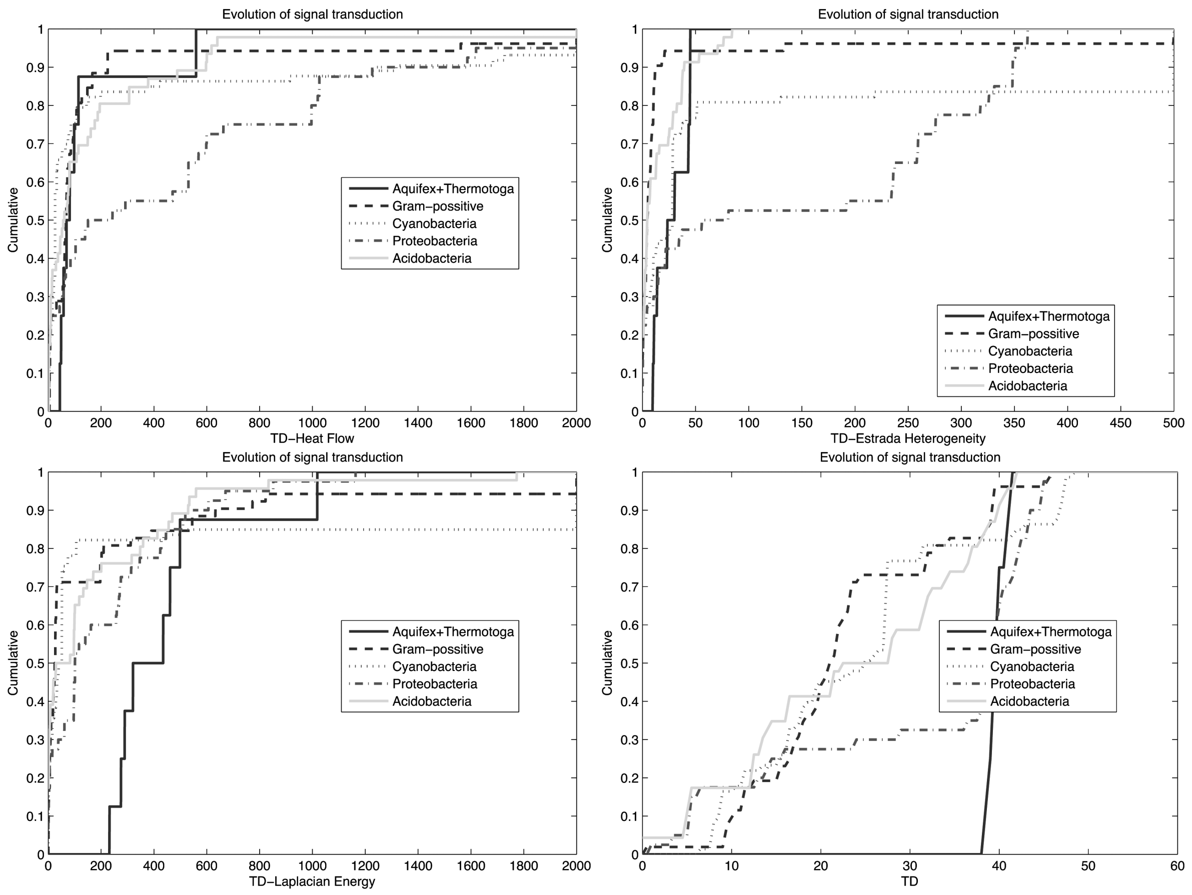


FIG. 5. Cumulatives for: Heat Flow (top-left), Estrada's Heterogeneity (top-right), Laplacian Energy (bottom-left) and von Neumann Entropy (bottom-right)

TABLE II. Table of AUCs

Bacteria	Heat[a]	EHN[b]	LEN[c]	VNE[d]
Aquifex-Thermotoga	93.29%	94.44%	**77.93%**	**34.09%**
Gram-possitive	91.33%	**94.61%**	88.85%	62.00%
Cyanobacteria	86.42%	79.94%	83.00%	59.49%
Proteobacteria	75.70%	70.62%	**88.86%**	48.20%
Acidobacteria	91.39%	**96.77%**	**91.10%**	**61.00%**

[a] Heat Flow
[b] Estrada's Heterogeneity
[c] Laplacian Energy
[d] Von Neumann Entropy

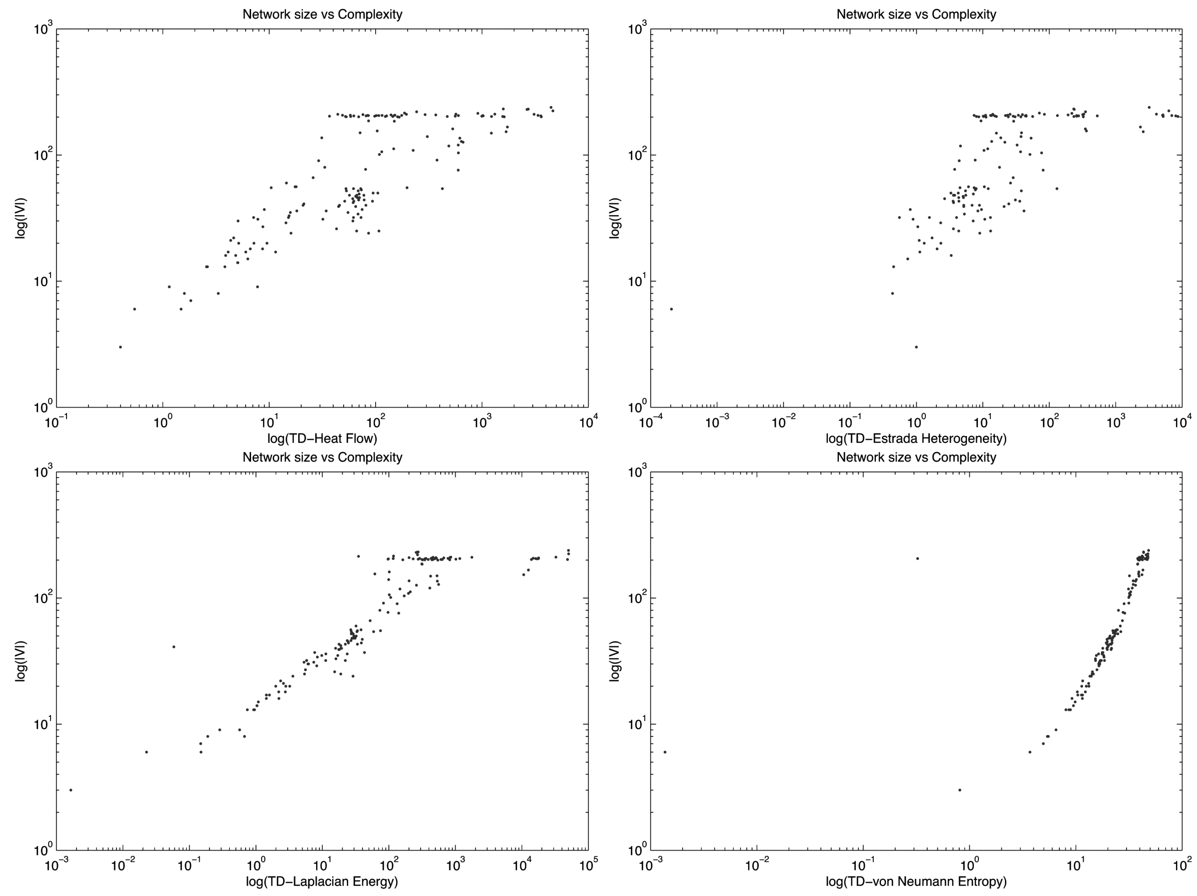


FIG. 6. Network size vs Complexity loglog plots for: Heat Flow (top-left), Estrada's Heterogeneity (top-right), Laplacian Energy (bottom-left) and von Neumann Entropy (bottom-right)

F Information-Theoretic approaches

Graph Clustering Using the Jensen-Shannon Kernel

Lu Bai and Edwin R. Hancock*

Department of Computer Science, University of York
{lu,erh}@cs.york.ac.uk

Abstract. This paper investigates whether the Jensen-Shannon divergence can be used as a means of establishing a graph kernel for graph classification. The Jensen-Shannon kernel is nonextensive information theoretic kernel which is derived from mutual information theory, and is defined on probability distributions. We use the von-Neumann entropy to calculate the elements of the Jensen-Shannon graph kernel and use the kernel matrix for graph classification. We use kernel principle components analysis (kPCA) to embed graphs into a feature space. Experimental results reveal the method gives good classification results on graphs extracted from an object recognition database.

Keywords: Jensen-Shannon kernel, Divergence, Entropy, Graph Kernel.

1 Introduction

Graph based representations have proved to be powerful tools for structural pattern analysis in computer vision. One of the problems that arises with large amounts of graph data is that of edit distance computation which itself is dependant on accurate correspondence analysis.

There have been several successful attempts to classify graph data using clustering techniques. The methods developed include a) using vectors of structural characteristics or permutation invariants [18], b) applying pairwise clustering to edit distances [17], c) embedding graph structures in dissimilarity-based feature spaces, and d) through central clustering based on class prototypes [16] [15]. An alternative to this methods is to use kernel methods formulated in the graph domain [13]. Graph kernels aim to overcome the computational bottleneck associated with graph similarity or edit distance computation which is known to be NP-complete. This is analogous to the use of kernel methods with vector data, which allow efficient algorithms to be developed that can deal with high dimensional data without the need to construct an explicit high dimensional feature space [2]. A number of graph kernels have been reported in the literature and successfully applied to real world data [4]. Most of the reported methods share

* Edwin Hancock is supported by the EU FET project SIMBAD and by a Royal Society Wolfson Research Merit Award.

the feature of exploiting topological information concerning the arrangement of nodes and edges in a graphs. There are three popular methods, namely a) diffusion kernels defined with respect to similarity [9] [10], b) random walk kernels based on counting the number of nodes with the same label in a random walk [8] [4], and c) the shortest path kernel [1].

Recently, information theory has been used to define a new family of kernels on probability distributions, and these have been applied to structured data [13] [14] [3] [6]. These so-called nonextensive information theoretic kernels are derived from the mutual information between probability distributions, and are related to the Shannon entropy. Examples include the Jensen-Shannon kernel [12].

One of the problems in constructing Jensen-Shannon kernels for graphs is that of constructing the required probability distributions or computing the entropy associated with graph structures. Both of these problems have proved elusive, and this is turn has provided an obstacle to the successful construction of information theoretic graph kernels [12]. Recently, we have shown to efficiently compute the von Neumann entropy of a graph [11]. The von Neumann entropy is the Shannon entropy associated with the Laplacian eigenvalues of a graph, and requires the computation of the graph spectrum, and this is cubic in the number of nodes. By approximating the Shannon entropy by its quadratic counterpart, we have shown how to the computation can be rendered quadratic in the number of nodes. In this paper, we explore how this simplification can be used to efficiently compute the Jensen-Shannon kernel between graphs. The resulting computations depend on the node degree distribution over the graph and can be simply computed for both the original graphs and their tensor product. Once the Jensen-Shannon kernel is to hand, we use kernel principle components analysis (kPCA) [7] to embed the graphs into a low dimensional feature space where classification is performed.

This paper is organised as follows. Section 2 briefly reviews the basic concepts of Jensen-Shannon kernel. Section 3 reviews how we construct a node degree probability distribution over both graphs and product graphs. This distribution is used to approximate the von-Neumann entropy, and we show how to calculate Jensen-Shannon graph kernel. Section 4 provides our experimental evaluation. Finally, Section 5 provides conclusions and directions for future work.

2 Jensen-Shannon Kernel

In this section we review the basic theory of the Jensen-Shannon Kernel used in our work. The Jensen-Shannon Kernel is a nonextensive mutual information kernel which is defined using the extensive and nonextensive entropy. It is defined on probability distributions over structured data. Assume $M_+^1(\chi)$ is a set of probability distributions where χ is a set provided with some $\sigma - algebra$ of measurable subsets, the kernel $k_{JS} : M_+^1(\chi) \times M_+^1(\chi) \to R$, is positive definite (pd) with the following kernel function [12]:

$$k_{JS}(P_1, P_2) = \ln 2 - JS(P_1, P_2) \tag{1}$$

where $JS(P_1, P_2)$ is the Jensen-Shannon divergence $k_{JS} : M^1_+(\chi) \times M^1_+(\chi) \rightarrow [0, \infty)$defined as

$$JSD(P_1, P_2) = H(\frac{P_1 + P_2}{2}) - \frac{1}{2}(H(P_1) + H(P_2)) \tag{2}$$

and for a mixture of n probability distribution $P_1, ...P_n$ with mixing proportions $\pi_1, ..., \pi_n$, the divergence is given by:

$$JSD(P_1, P_2, ..., P_n) = H(\Sigma^n_{i=1}\pi_i P_i) - \Sigma^n_{i=1}\pi_i H(P_i) \tag{3}$$

where $H(P_i)$ is an Shannon entropy for distribution P_i.

3 Jensen-Shannon Graph Kernel

In this section we explore how to compute the Jensen-Shannon kernel for pairs of graphs. We commence by defining a probability distribution over the node degree distribution and then show how this can be used to compute the Shannon entropy appearing in the definition of the kernel.

3.1 Node Degree Distribution

We use the node degree distribution to calculate the Jensen-Shannon graph kernel. To commence, we denote the graph as $G = (V, E)$ where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. The adjacency matrix A of graph G has elements

$$A(u, v) = \begin{cases} 1 & if(u, v) \in E, \\ 0 & otherwise. \end{cases} \tag{4}$$

The degree matrix of graph G is a diagonal matrix D with the nodes degrees as diagonal elements $D(u, u) = d_u = \sum_{u,v \in V} A(u, v)$. From the adjacency matrix and the degree matrix we compute the Laplacian matrix $L \equiv D - A$, which has elements

$$L(u, v) = \begin{cases} d_v & if u = v, \\ -1 & if(u, v) \in E, \\ 0, & otherwise. \end{cases} \tag{5}$$

The spectral decomposition of the Laplacian matrix is $L = \sum_{u=1}^{|V|} \lambda_i \phi_i \phi_i^T$ where λ_i are the eignevalues and ϕ_i are the eigenvectros of L.

We can define the node degree distribution as the node degree divided by the volume of the graph, and for node $u \in V$ the probability is

$$P_G(u) = d_u / \sum_{v \in V} d_v \tag{6}$$

3.2 Graph Product

Before we introduce the Jensen-Shannon graph kernel, we first introduce the graph product concept. For the graphs $G(V, E)$ and $G'(V', E')$ the product graph $G\times = (V\times, E\times)$, has node and edge sets

$$V_\times = \{(v_i, v_i^{'}) : v_i \in V \wedge v_i^{'} \in V\} \tag{7}$$

$$E_\times = \{((v_i, v_i^{'}), (v_i, v_i^{'})) : (v_i, v_i^{'}) \in E \wedge (v_i, v_i^{'}) \in E^{'}\} \tag{8}$$

If A and A' are the adjacency matrices of graphs G and G' respectively $A\times = A \prod A'$ is the adjacency matrix of the product graph $G\times$. The most common graph products are formed by taking the Cartesian product, tensor product or the union. For reasons of efficiency here we take the union graph. We compute the difference in entropy between the two graphs and their union. To construct the union graph, we perform pairwise correspondence matching. Details of the construction are outside the scope of this paper. Our approach follows that of Lin, Wilson and Hancock [5], and the adjacency matrix of the union is denoted by A_U.

3.3 Jensen-Shannon Graph Kernel Graph Kernel

Consider a a graph set $\{G_1, ..., G_i, ..., G_j, ..., G_n\}$. A graph kernel can be defined using a similarity measure to compute the $n \times n$ positive matrix. Associated with the degree distribution P_i and P_j of graphs G_i and G_j, the Jensen-Shannon kernel is defined as

$$k_{JS}(P_i, P_j) = \ln 2 - H(\frac{P_i + P_j}{2}) + \frac{1}{2}(H(P_i) + H(P_j)) \tag{9}$$

We suppose $\frac{P_i+P_j}{2}$ represents the degree distribution of the product graph $G_\times$ of $G_i(V_i, E_i)$ and $G_j(V_j, E_j)$. As a result (9) can be written as

$$k_{JS}(P_i, P_j) = \ln 2 - H(P_\times) + \frac{1}{2}(H(P_i) + H(P_j)) \tag{10}$$

Here we use the von Neumann entropy to compute the Jensen-Shannon kernel. The von Neumann entropy for graph G is $H_{VN} = \sum_{i=1}^{|V|} \frac{\lambda_i}{2} ln \frac{\lambda_i}{2}$. By approximating the non-Neumann entrpy by its quadratic counterpart, Han and Hancock [11] have shown that the approximate von-Neumann entropy is given by

$$H_{VN} = \frac{|V|}{4} - \sum_{(u,v)\in E} \frac{1}{4d_u d_v} \tag{11}$$

As the node degree distribution $P_i(u)$ and $P_j(v)$ can be written as $P_i(u) = d_u/\Sigma_u^V d_u$ and $P_j(v) = d_v/\Sigma_v^V d_v$, so associated with function (10) and (11), the Jensen-Shannon graph kernel can be approximated as

$$k_{JS}(P_i, P_j) = \ln 2 - (\frac{|V_\times|}{4} - \sum_{v_\times \neq u_\times} \frac{1}{P_\times(u_\times)P_\times(v_\times)}) + \frac{|V_i| + |V_j|}{2}$$
$$- \frac{1}{2}(\sum_{u_1 \neq v_1, v_1(i) \neq u_1(j)}^{V_1} \frac{1}{P_i(u)P_i(v)} + \sum_{u_2 \neq v_2, v_2(i) \neq u_2(j)}^{V_2} \frac{1}{P_j(u)P_j(v)}) \quad (12)$$

where $P_\times(i)$ represents the degree distribution of $p_\times$.

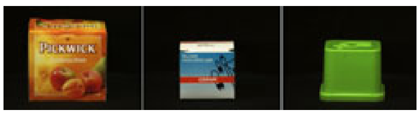
4 Experiments

In this section, we will explore whether the Jensen-Shannon kernel can be used for object recognition, and evaluate its stability. we commence by classifying synthetic graph abstracted from real-world image, and then calculate relationship between the kernel value and number of edit operation.

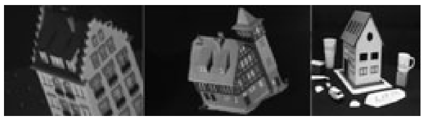
4.1 Graph Characterization

In the first experiment, we use three different graph datasets to illustrate the classification performance of the Jensen-Shannon graph kernel. The first dataset consists of graphs are extracted from digital images of three similar boxes in the ALOI database Fig.1(a), the second of graphs extracted from images of three toy houses in the MOVI and CMU databases, and the third of graphs extracted from images of cups from the COIL database Fig.1(3). For each object there are 18 images captured from different viewpoints. The graphs are the Delaunay triangulations of feature points extracted from the different images.

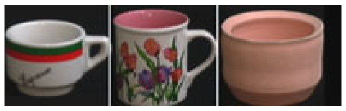
For the three different datasets we compute the Jensen-Shannon kernel matrices. We perform kernel Principal Components Analysis (kPCA) on the kernel matrices to embed the graphs into a 3-dimensional feature space. Fig.2(1),



(a) Similar Boxes from ALOI Dataset

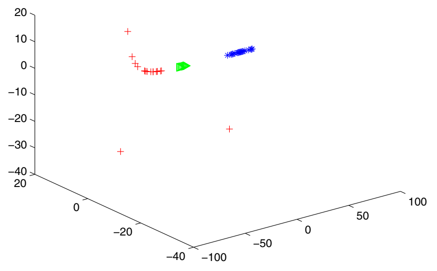


(b) Similar Houses from CMU and MOVI Datasets

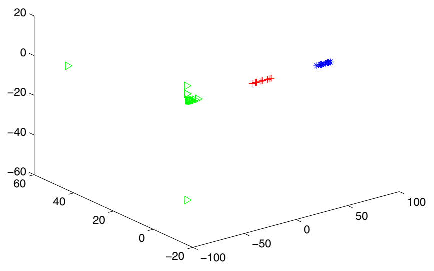


(c) Similar Cups from COIL Datasets

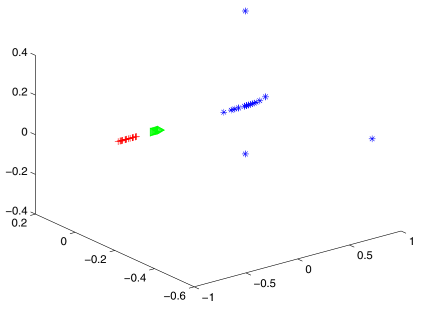
Fig. 1. Datastes for Experiments



(a) Experiments Performance of ALOI



(b) Experiments Performance of CMU and MOVI



(c) Experiments Performance of COIL

Fig. 2. Experiment Performance of Jensen-Shannon Kernel

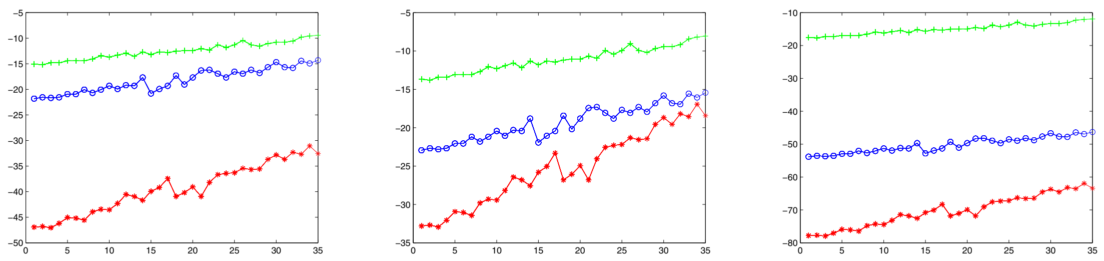
Table 1. Accurancy of Classification with Jensen-Shannon Kernel

Datasets	Object1	Object2	Object3
Boxes (ALOI)	100%	100%	100%
Houses (CMU and MOVI)	100%	100%	100%
Cups (COIL)	100%	100%	100%

Fig.2(b) and Fig.2(c) show the resulting embeddings. In each case the different objects are well separated in the embedding space. To place our analysis on a more quantitative footing, we apply K-means clustering method to the embedded graphs, and compute the classification accuracy for the three datasets. Table.1 summaries the results, and indicate that an accuracy of 100% is achievable.

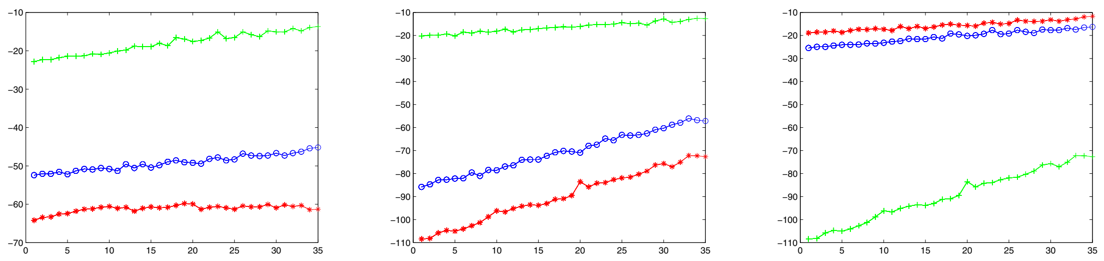
4.2 Stability Evaluation

Next we evaluate the stability of the Jensen-shannon kernel. We select nine seed graphs from the three groups of real-world images shown in Fig.1. We then apply random edit operations to the nine seed graphs to simulate the effects of noise. The edit operations are node deletion and edge deletion. For each seed graph, we randomly delete a predetermined fractions of the nodes or edges to obtain noise corrupted variants. Fig.3. and Fig.4. show the effects of node deletion and



(a) Evaluation of ALOI (node) (b) Evaluation of CMU and MOVI (node) (c) Evaluation of COIL (node)

Fig. 3. Jensen-Shannon Kernel Evaluation with Node Deletion Edit Operation



(a) Evaluation of ALOI (edge) (b) Evaluation of CMU and MOVI (edge) (c) Evaluation of COIL (edge)

Fig. 4. Jensen-Shannon Kernel Evaluation with Edge Deletion Edit Operation

edge deletion for each group of graphs respectively. The x-axis shows the fraction of nodes or edges deleted, and the y-axis shows value of the kernel $K(G_o, G_n)$ between the original graph G_o and its noise corrupted counterpart G_n. The plots show that there is an approximately liner relationship between the Jensen-Shannon kernel and the number of deleted nodes or edges, i.e. the graph edit distance.

5 Conclusions

In this paper, we have shown how to construct Jensen-Shannon kernels for graph data-sets using the von-Neumann entropy. The method is based on a probability distribution over the node degree in a graph, and uses the von Neumann entropy to measure the mutual information between pairs of graphs. By applying kernel PCA to the Jensen-Shannon kernel matrix, we embed sets of graphs into a low dimensional space. Here we use K-means clustering to assign the graphs to classes. Experimental results reveal that the method gives good results for graph datasets extracted from image data.

References

1. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs (2005)
2. Bunke, H., Riesen, K.: Graph classification based on dissimilarity space embedding. In: Structural, Syntactic, and Statistical Pattern Recognition, pp. 996–1007 (2010)
3. Desobry, F., Davy, M., Fitzgerald, W.J.: Density kernels on unordered sets for kernel-based signal processing. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2007, vol. 2, p. II–417. IEEE, Los Alamitos (2007)
4. Gartner, T.: A survey of kernels for structured data. ACM SIGKDD Explorations Newsletter 5(1), 49–58 (2003)
5. Han, L., Hancock, E., Wilson, R.: Learning generative graph prototypes using simplified von neumann entropy. In: Graph-Based Representations in Pattern Recognition, pp. 42–51 (2011)
6. Jebara, T., Kondor, R., Howard, A.: Probability product kernels. The Journal of Machine Learning Research 5, 819–844 (2004)
7. Jolliffe, I.: Principal component analysis (2002)
8. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernels between labeled graphs. In: International Workshop then Conference on Machine Learning, vol. 20, p. 321 (2003)
9. Kondor, R.I., Lafferty, J.: Diffusion kernels on graphs and other discrete input spaces. In: International Workshop then Conference on Machine Learning, pp. 315–322. Citeseer (2002)
10. Lafferty, J., Lebanon, G.: Diffusion kernels on statistical manifolds. The Journal of Machine Learning Research 6, 129–163 (2005)
11. Lin, H., Hancock, E.R.: Characterizing Graphs Using Approximate von-Neumann Entropy (2011)
12. Martins, A.F.T., Smith, N.A., Xing, E.P., Aguiar, P.M.Q., Figueiredo, M.A.T.: Nonextensive information theoretic kernels on measures. The Journal of Machine Learning Research 10, 935–975 (2009)
13. Scholkopf, B., Smola, A.J.: Learning with kernels. Citeseer (2002)
14. Shawe-Taylor, J., Cristianini, N.: Kernel methods for pattern analysis. Cambridge Univ Pr., Cambridge (2004)
15. Suau, P., Escolano, F.: Bayesian optimization of the scale saliency filter. Image and Vision Computing 26(9), 1207–1218 (2008)
16. Torsello, A., Hancock, E.R.: Graph embedding using tree edit-union. Pattern recognition 40(5), 1393–1405 (2007)
17. Torsello, A., Robles-Kelly, A., Hancock, E.R.: Discovering shape classes using tree edit-distance and pairwise clustering. International Journal of Computer Vision 72(3), 259–285 (2007)
18. Xiao, B., Hancock, E.R., Wilson, R.C.: Graph characteristics from the heat kernel trace. Pattern Recognition 42(11), 2589–2606 (2009)

Learning Generative Graph Prototypes Using Simplified von Neumann Entropy

Lin Han, Edwin R. Hancock, and Richard C. Wilson

Department of Computer Science, University of York

Abstract. We present a method for constructing a generative model for sets of graphs by adopting a minimum description length approach. The method is posed in terms of learning a generative supergraph model from which the new samples can be obtained by an appropriate sampling mechanism. We commence by constructing a probability distribution for the occurrence of nodes and edges over the supergraph. We encode the complexity of the supergraph using the von-Neumann entropy. A variant of EM algorithm is developed to minimize the description length criterion in which the node correspondences between the sample graphs and the supergraph are treated as missing data.The maximization step involves updating both the node correspondence information and the structure of supergraph using graduated assignment. Empirical evaluations on real data reveal the practical utility of our proposed algorithm and show that our generative model gives good graph classification results.

1 Introduction

The main obstacle to learning in the graph domain is solving the problem of how to capture variability in graph or tree structure. The main reason for the lack of progress in this area is the difficulty in developing representations that can capture variations in graph-structure. This variability can be attributed to a) variations in either node or edge attributes, b) variations in node or edge composition and c) variations in edge-connectivity. This trichotomy provides a natural framework for analyzing the state-of-the-art in the literature. Most of the work on Bayes nets in the graphical models literature can be viewed as modeling variations in node or edge attributes [1]. Examples also include the work of Christmas et al.[2] and Bagdanov et al. [3] who both use Gaussian models to capture variations in edge attributes. The problems of modeling variations in node and edge composition are more challenging since they focus on modeling the structure of the graph rather than its attributes.

The problem of learning edge structure is probably the most challenging of those listed above. Broadly speaking there are two approaches to characterizing variations in edge structure for graphs. The first of these is graph spectral, while the second is probabilistic. In the case of graph spectra, many of the ideas developed in the generative modeling of shape using principal components analysis can be translated relatively directly to graphs using simple vectorization procedures based on the correspondences conveyed by the ordering of Laplacian eigenvectors [4]. Although these methods are simple and effective, they are limited by the stability of the Laplacian spectrum under perturbations in graph-structure. The probabilistic approach is potentially more robust, but requires accurate correspondence information to be inferred from the available graph structure. If this is to hand, then a representation of edge structure can be learned.

In this paper, we focus on the third problem and aim to learn a generative model that can be used to describe the distribution of structural variations present in a set of sample graphs, and in particular to characterize the variations of the edge structure present in the set. We follow Torsello and Hancock [5] and pose the problem as that of learning a generative supergraph representation from which we can sample. However, their work is based on trees, and since the trees are rooted the learning process can be effected by performing tree merging operations in polynomial time. This greedy strategy does not translate tractably to graphs where the complexity becomes exponential, and we require different strategies for learning and sampling. Torsello and Hancock realize both of these using edit operations, here on the other hand we use a soft-assign method for optimization.

In prior work Han, Wilson and Hancock propose a method of learning a supergraph model in [13] which overlooks the complexity of the supergraph model. Here, we take an information theoretic approach to estimating the supergraph structure by using a minimum description length criterion. By taking into account the overall code-length in the model, MDL allows us to select a supergraph representation that trades-off goodness-of-fit with the observed sample graphs against the complexity of the model. We show how to gauge the complexity of the supergraph using von-Neumann entropy[8] (i.e. the entropy associated with the Normalized Laplacian eigenvalues), and how to efficiently approximate this entropy without the need to compute the Laplacian spectrum. We use a variant of EM algorithm to minimize the total code-length criterion, in which the correspondences between the nodes of the sample graphs and those of the supergraph are treated as missing data. In the maximization step, we update both the node correspondence information and the structure of supergraph using graduated assignment. This novel technique is applied to a large database of object views, and used to learn class prototypes that can be used for the purposes of object recognition.

2 Probabilistic Framework

We are concerned with learning a structural model represented in terms of a so-called supergraph that can capture the variations present in a sample of graphs. In Torsello and Hancock's work [5] this structure is found by merging the set of sample trees, and so each sample tree can be obtained from it by edit operations. Here, on the other hand, we aim to estimate an adjacency matrix that captures the frequently occurring edges in the training set. To commence our development we require the *a posteriori* probabilities of the sample graphs given the structure of the supergraph and the node correspondences between each sample graph and the supergraph. To compute these probabilities we use the method outlined in [6].

Let the set of sample of graphs be $\mathcal{G} = \{G_1, ...G_i, ...G_N\}$, where the graph indexed i is $G_i = (V_i, E_i)$ with V_i the node-set and E_i the edge-set. Similarly, the supergraph which we aim to learn from this data is denoted by $\varGamma = (V_\varGamma, E_\varGamma)$, with node-set $V_\varGamma$ and edge-set $E_\varGamma$. Further, we represent the structure of the two graphs using a $|V_i| \times |V_i|$ adjacency matrix D^i for the sample graph G_i and a $|V_\varGamma| \times |V_\varGamma|$ adjacency matrix M for the supergraph model $\varGamma$. The elements of the adjacency matrix for the sample graph and those for the supergraph are respectively defined to be

$$D^i_{ab} = \begin{cases} 1 & \text{if } (a,b) \in E_i \\ 0 & \text{otherwise} \end{cases} , \quad M_{\alpha\beta} = \begin{cases} 1 & \text{if } (\alpha,\beta) \in E_\varGamma \\ 0 & \text{otherwise} \end{cases} . \tag{1}$$

We represent the correspondence matches between the nodes of the sample graph and the nodes of the supergraph using a $|V_i| \times |V_\varGamma|$ assignment matrix S^i which has elements

$$s^i_{a\alpha} = \begin{cases} 1 \text{ if } a \rightarrow \alpha \\ 0 \text{ otherwise} \end{cases} . \tag{2}$$

where $a \rightarrow \alpha$ implies that node $a \in V_i$ is matched to node $\alpha \in V_\varGamma$.

With these ingredients, according to Luo and Hancock [6] the *a posteriori* probability of the graphs G_i given the supergraph $\varGamma$ and the correspondence indicators is

$$P(G_i|\varGamma, S^i) = \prod_{a \in V_i} \sum_{\alpha \in V_\varGamma} K^i_a \exp[\mu \sum_{b \in V_i} \sum_{\beta \in V_\varGamma} D^i_{ab} M_{\alpha\beta} s^i_{b\beta}] . \tag{3}$$

where $\mu = \ln \frac{1-P_e}{P_e}$ and $K^i_a = P_e^{|V_i| \times |V_\varGamma|} B^i_a$. In the above, P_e is the error rate for node correspondence and B^i_a is the probability of observing node a in graph G_i , the value of which depends only on the identity of the node a . $|V_i|$ and $|V_\varGamma|$ are the number of the nodes in graph G_i and supergraph $\varGamma$.

3 Model Coding Using MDL

Underpinning minimum description length is the principle that learning, or finding a hypothesis that explains some observed data and makes predictions about data yet unseen, can be viewed as finding a shorter code for the observed data [9,7]. To formalize this idea, we encode and transmit the observed data and the hypothesis, which in our case are respectively the sample graphs $\mathcal{G}$ and the supergraph structure $\varGamma$. This leads to a two-part message whose total length is given by $\mathcal{L}(\mathcal{G}, \varGamma) = LL(\mathcal{G}|\varGamma) + LL(\varGamma)$.

Encoding sample graphs: We first compute the code-length of the graph data. For the sample graph-set $\mathcal{G} = \{ G_1, ...G_i, ...G_N \}$ and the supergraph $\varGamma$, the set of assignment matrices is $\mathcal{S} = \{S^1, ...S^i, ...S^N\}$ and these represent the correspondences between the nodes of the sample graphs and those of the supergraph. Under the assumption that the graphs in $\mathcal{G}$ are independent samples from the distribution, using the *a posteriori* probabilities from Section 2 the likelihood of the set of sample graphs is

$$P(\mathcal{G}|\Gamma,\mathcal{S}) = \prod_{G_i\in\mathcal{G}} P(G_i|\Gamma,S^i) = \prod_{G_i\in\mathcal{G}} \prod_{a\in V_i} \sum_{\alpha\in V_\Gamma} K_a^i \exp[\mu \sum_{b\in V_i} \sum_{\beta\in V_\Gamma} D_{ab}^i M_{\alpha\beta} s_{b\beta}^i] \ . \tag{4}$$

Instead of using the *Shannon-Fano code*, which is equivalent to the negative logarithm of the above likelihood function, we measure the code-length of the graph data using its average. Our reason is that if we adopt the former measure, then there is a bias to learning a complete supergraph that is fully connected. The reason will become clear later-on when we outline the maximization algorithm in Section 4, and we defer our justification until later. Thus, the graph code-length is $LL(\mathcal{G}|\Gamma) = -\frac{1}{|\mathcal{G}|}\sum_{G_i\in\mathcal{G}} \log P(G_i|\Gamma,S^i)$ which is the average over the set of sample graphs $\mathcal{G}$.

Encoding the supergraph model: Next, we must compute a code-length to measure the complexity of the supergraph. For two-part codes the MDL principle does not give any guideline as to how to encode the hypotheses. Hence every code for encoding the supergraph structure is allowed, so long as it does not change with the sample size N. Here the code-length for describing supergraph complexity is chosen to be measured using the von-Neumann entropy [8] $H = -\sum_k \frac{\lambda_k}{2}\ln\frac{\lambda_k}{2}$ where λ_k are the eigenvalues of the normalized Laplacian matrix of the supergraph $\hat{L}$ whose elements are

$$\hat{L}_{\alpha\beta} = \begin{cases} 1 & \text{if } \alpha=\beta \\ -\frac{1}{\sqrt{T_\alpha T_\beta}} & \text{if } (\alpha,\beta)\in E_\Gamma \\ 0 & \text{otherwise} \end{cases} \ . \tag{5}$$

where $T_\alpha = \sum_{\xi\in V_\Gamma} M_{\alpha\xi}$ and $T_\beta = \sum_{\xi\in V_\Gamma} M_{\beta\xi}$. The normalized Laplacian matrix is commonly used as a graph representation and graph cuts, and its eigenvalues are in the range $0 \le \lambda_k \le 2$ [11]. Divided by 2, the value of $\frac{\lambda_k}{2}$ is constrained between 0 and 1, and the von-Neumann entropy derived thereby is an intrinsic property of graphs that reflects the complexity of their structures better than other measures. We approximate the entropy $-\frac{\lambda_k}{2}\ln\frac{\lambda_k}{2}$ by the quadratic entropy $\frac{\lambda_k}{2}(1-\frac{\lambda_k}{2})$, to obtain

$$H = -\sum_k \frac{\lambda_k}{2}\ln\frac{\lambda_k}{2} \simeq \sum_k \frac{\lambda_k}{2}(1-\frac{\lambda_k}{2}) = \frac{\sum_k \lambda_k}{2} - \frac{\sum_k \lambda_k^2}{4} \ . \tag{6}$$

Using the fact that $Tr[\hat{L}^n] = \sum_k \lambda_k^n$, the quadratic entropy can be rewritten as $H = \frac{Tr[\hat{L}]}{2} - \frac{Tr[\hat{L}^2]}{4}$. Since the normalized Laplacian matrix $\hat{L}$ is symmetric and it has unit diagonal elements, then according to equation(5) for the trace of the normalized Laplacian matrix we have $Tr[\hat{L}] = |V_\Gamma|$. Similarly, for the trace of the square of the normalized Laplacian, we have

$$Tr[\hat{L}^2] = \sum_{\alpha\in V_\Gamma}\sum_{\beta\in V_\Gamma} \hat{L}_{\alpha\beta}\hat{L}_{\beta\alpha} = |V_\Gamma| + \sum_{(\alpha,\beta)\in E_\Gamma} \frac{1}{T_\alpha T_\beta} \ . \tag{7}$$

Then the simplified entropy becomes

$$H = \frac{|V_\Gamma|}{2} - \frac{|V_\Gamma|}{4} - \sum_{(\alpha,\beta)\in E_\Gamma} \frac{1}{4\, T_\alpha T_\beta} = \frac{|V_\Gamma|}{4} - \sum_{(\alpha,\beta)\in E_\Gamma} \frac{1}{4\, T_\alpha T_\beta} \quad . \tag{8}$$

As a result, the approximated complexity of the supergraph depends on two factors. The first is the order of supergraph, i.e. the number of nodes of the supergraph. The second is the degree of the nodes of the supergraph.

Finally, by adding together the two contributions to the code-length, the overall code-length is

$$\mathcal{L}(\mathcal{G},\Gamma) = LL(\mathcal{G}|\Gamma) + LL(\Gamma) = \tag{9}$$
$$-\frac{1}{|\mathcal{G}|}\sum_{G_i\in\mathcal{G}}\sum_{a\in V_i} \log\{\sum_{\alpha\in V_\Gamma} K_a^i \exp[\mu \sum_{b\in V_i}\sum_{\beta\in V_\Gamma} D_{ab}^i M_{ab} s_{b\beta}^i]\,\} + \frac{|V_\Gamma|}{4} - \sum_{(\alpha,\beta)\in E_\Gamma} \frac{1}{4\, T_\alpha T_\beta} \;.$$

Unfortunately, due to the mixture structure, the direct estimation of the supergraph structure M from the above code-length criterion is not tractable in closed-form. For this reason, we resort to using the expectation maximization algorithm.

4 Expectation-Maximization

Having developed our computational model which poses the problem of learning the supergraph as that of minimizing the code-length, in this section, we provide a concrete algorithm to locate the supergraph structure using our code-length criterion using expectation-maximisation. With the above likelihood function and the code-length developed in the previous section, Figueiredo and Jain's formulation of EM[14] involves maximizing

$$\Lambda^{(n+1)}(\mathcal{G}|\Gamma,\mathcal{S}^{(n+1)}) = \frac{1}{|\mathcal{G}|}\sum_{G_i\in\mathcal{G}}\sum_{a\in V_i}\sum_{\alpha\in V_\Gamma} Q_{a\alpha}^{i,(n)}\{\ln K_a^i + \mu \sum_{b\in V_i}\sum_{\beta\in V_\Gamma} D_{ab}^i M_{\alpha\beta}^{(n)} s_{b\beta}^{i,(n+1)}\}$$
$$-\frac{|V_\Gamma|}{4} + \sum_{(\alpha,\beta)\in E_\Gamma} \frac{1}{4\, T_\alpha^{(n)} T_\beta^{(n)}} \quad . \tag{10}$$

The expression above can be simplified since the first term under the curly braces contributes a constant amount. Based on this observation, the critical quantity in determining the update direction is

$$\hat{\Lambda}^{(n+1)} = \tag{11}$$
$$\frac{1}{|\mathcal{G}|}\sum_{G_i\in\mathcal{G}}\sum_{a\in V_i}\sum_{\alpha\in V_\Gamma}\sum_{b\in V_i}\sum_{\beta\in V_\Gamma} Q_{a\alpha}^{i,(n)} D_{ab}^i M_{\alpha\beta}^{(n)} s_{b\beta}^{i,(n+1)} - \frac{|V_\Gamma|}{4} + \sum_{(\alpha,\beta)\in E_\Gamma} \frac{1}{4\, T_\alpha^{(n)} T_\beta^{(n)}} \;.$$

In order to optimize our weighted code-length criterion, we use graduated assignment [10] to update both the assignment matrices $\mathcal{S}$ and the structure of the supergraph, i.e. the supergraph adjacency matrix M. The updating process is realized by computing the derivatives of $\hat{\Lambda}^{(n+1)}$, and re-formulating the underlying discrete assignment problem as a continuous one using softmax.

In the **maximization step**, we have two parallel iterative update equations. The first update mode involves softening the assignment variables, while the second aims to modify the edge structure in the supergraph. Supergraph edges that are unmatchable disappear by virtue of having weak connection weights and cease to play any significant role in the update process. Experiments show that the algorithm appears to be numerically stable and appears to converge uniformly.

To update the assignment matrices, we commence by computing the partial derivative of the weighted code-length function in Equation (11) with respect to the elements of the assignment matrices, which gives

$$\frac{\partial \hat{\Lambda}^{(n+1)}}{\partial s_{b\beta}^{i,(n+1)}} = \frac{1}{|\mathcal{G}|} \sum_{a \in V_i} \sum_{\alpha \in V_\Gamma} Q_{a\alpha}^{i,(n)} D_{ab}^{i} M_{\alpha\beta}^{(n)} \quad . \tag{12}$$

To ensure that the assignment variables remain constrained to lie within the rage [0,1], we adopt the soft-max update rule

$$s_{a\alpha}^{i,(n+1)} \leftarrow \exp[\frac{1}{\tau} \frac{\partial \hat{\Lambda}^{(n+1)}}{\partial s_{a\alpha}^{i,(n+1)}}] / \sum_{\alpha' \in V_\Gamma} \exp[\frac{1}{\tau} \frac{\partial \hat{\Lambda}^{(n+1)}}{\partial s_{a\alpha'}^{i,(n+1)}}] \quad . \tag{13}$$

The value of the temperature τ in the update process has been controlled using a slow exponential annealing schedule of the form suggested by Gold and Rangarajan[10]. Initializing τ^{-1} with a small positive value and allowing it to gradually increase, the assignment variable $s_{a\alpha}^{i,(n+1)}$ corresponding to the maximum $\frac{\partial \hat{\Lambda}^{(n+1)}}{\partial s_{a\alpha}^{i,(n+1)}}$ approaches 1 while the remainder approach 0.

The partial derivative of the weighted code-length function in Equation (11) with respect to the elements of the supergraph adjacency matrix is equal to

$$\frac{\partial \hat{\Lambda}^{(n+1)}}{\partial M_{\alpha\beta}^{(n)}} = \frac{1}{|\mathcal{G}|} \sum_{G_i \in \mathcal{G}} \sum_{a \in V_i} \sum_{b \in V_i} Q_{a\alpha}^{i,(n)} D_{ab}^{i} s_{b\beta}^{i,(n+1)} - \frac{1}{(T_\alpha^{(n)})^2} \sum_{(\alpha,\beta') \in E_\Gamma} \frac{1}{4\, T_{\beta'}^{(n)}} \quad . \tag{14}$$

The soft-assign update equation for the elements of the supergraph adjacency matrix is

$$M_{\alpha\beta}^{(n+1)} \leftarrow \exp[\frac{1}{\tau} \frac{\partial \hat{\Lambda}^{(n+1)}}{\partial M_{\alpha\beta}^{(n)}}] \; / \sum_{(\alpha',\beta') \in E_\Gamma} \exp[\frac{1}{\tau} \frac{\partial \hat{\Lambda}^{(n+1)}}{\partial M_{\alpha'\beta'}^{(n)}}] \quad . \tag{15}$$

Recall that in Section 3 we discussed the encoding of the sample graphs, and chose to use the average of *Shannon-Fano code*. We can now elucidate that the reason for this choice is that as the number of the sample graphs increases, for instance in the limit as the size of the graph sample-set $\mathcal{G}$ increases, i.e. $N \to \infty$, the sum of permuted adjacency matrices of the sample graphs might dominate

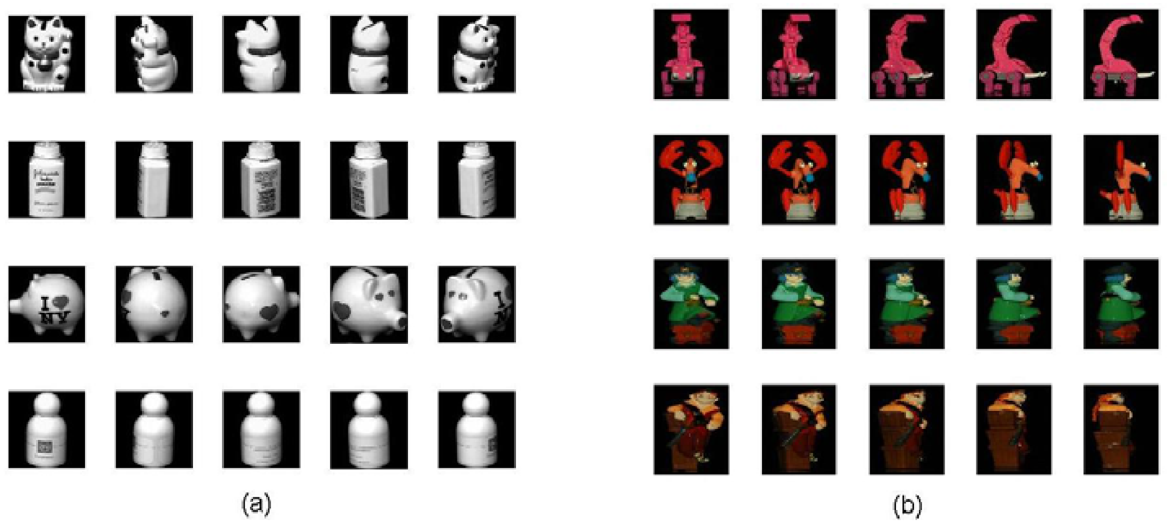


Fig. 1. (a)Example images in the COIL dataset. (b)Example images in the toys dataset.

the magnitude of the second term in Equation (14). Thus the update algorithm might induce a complete supergraph that is fully connected. Hence, we choose to use its average rather than its sum.

In the **expectation step** of the EM algorithm, we compute the *a posteriori* correspondence probabilities for the nodes of the sample graphs to the nodes of the supergraph. Applying Bayes rule, the *a posteriori* correspondence probability for the nodes of the sample graph G_i at iteration $n+1$ are given by

$$Q^{i,(n+1)}_{a\alpha} = \frac{\exp[\sum_{b\in V_i}\sum_{\beta\in V_\Gamma} D^i_{ab} M^{(n)}_{\alpha\beta} s^{i,(n)}_{b\beta}]\pi^{i,(n)}_{\alpha}}{\sum_{\alpha'\in V_\Gamma}\exp[\sum_{b\in V_i}\sum_{\beta\in V_\Gamma} D^i_{ab} M^{(n)}_{\alpha'\beta} s^{i,(n)}_{b\beta}]\pi^{i,(n)}_{\alpha'}} \ . \tag{16}$$

In the above equation, $\pi^{i,(n)}_{\alpha'} = \langle Q^{i,(n)}_{a\alpha'}\rangle_a$, where $\langle\ \rangle_a$ means average over a.

5 Experiments

In this section, we report experimental results aimed at demonstrating the utility of our proposed generative model on real-world data. We use images from two datasets for experiments. The first dataset is the COIL which consists of images of 4 objects, with 72 views of each object from equally spaced directions over 360°. We extract corner features from each image and use the detected feature points as nodes to construct sample graphs by Delaunay triangulation. The second dataset is a dataset consisting of views of toys, and contains images of 4 objects with 20 different views of each object. For this second dataset, the feature keypoints used to construct Delaunay graphs are extracted using the SIFT detector. Some example images of the objects from these two datasets are given in Figure 1.

The first part of our experimental investigation aims to validate our supergraph learning method. We test our proposed algorithm on both of the two datasets and in order to better analyze our method, we initialize the supergraph in our EM algorithm with different structures. For the COIL dataset, we initialize the supergraph structure with the median graph, i.e. the sample graph with the largest *a posteriori* probability from the supergraph. On the other hand, to initialize the structure of the supergraph in the toys dataset, we match pairs of graphs from a same object using the discrete relaxation algorithm [12]. Then we concatenate(merge) the common structures over for the sample graphs from a same object to form an initial supergraph. The initial supergraph constructed in this way preserves more of the structural variations present in the set of sample graphs. The median graph, on the other hand, captures more of the common salient information. We match the sample graphs from the two datasets against their supergraphs both using graduated assignment[10] and initialize the assignment matrices in our algorithm with the resulting assignment matrices. Using these settings, we iterate the two steps of the EM algorithm 30 times, and observe how the complexity of the supergraph, the average log-likelihood of the sample graphs and the overall code-length vary with iteration number. Figures 2 and 3 respectively shows the results for the COIL and toys datasets illustrated in Figure 1.

From Figure 2(a) it is clear that the von-Neumann entropy of the supergraph increases as the iteration number increases. This indicates that the supergraph structure becomes more complex with an increasing number of iterations. Figure 2(b) shows that the average of the log-likelihood of the sample graphs increases during the iterations. Figure 2(c) shows that the overall-code length decreases and gradually converges as the number of iterations increases. For the toys dataset, the von-Neumann entropy in Figure 3(a) shows an opposite trend and decreases as the number of iterations increases. The reason for this is that the initial supergraph we used for this dataset, i.e. the concatenated supergraph, accommodates too much structural variation from the sample graphs. The reduction of the von-Neumann entropy implies some trivial edges are eliminated or relocated. As a result the supergraph structure both condenses and simplifies with increasing iteration number. Although the complexity of the graphs behaves differently, the average of the likelihood of the graphs in Figure 3(b) and the overall-code length in Figure 3(c) exhibit a similar behaviour to those for the COIL dataset. In other words, our algorithm behaves in a stable manner both increasing the likelihood of sample graphs and decreasing the overall code-length on both datasets.

Our second experimental goal is to evaluate the effectiveness of our learned generative model for classifying out-of-sample graphs. From the COIL dataset, we aim 1) to distinguish images of cats from pigs on the basis of their graph representations and 2) distinguish between images of different types of bottle. For the toys dataset, on the other hand, we aim to distinguish between images of the four objects. To perform these classification tasks, we learn a supergraph for each object class from a set of samples and use Equation (3) to compute the *a posteriori* probabilities for each graph from a separate (out-of-sample) test-set. The class-label of the test graph is determined by the class of the supergraph

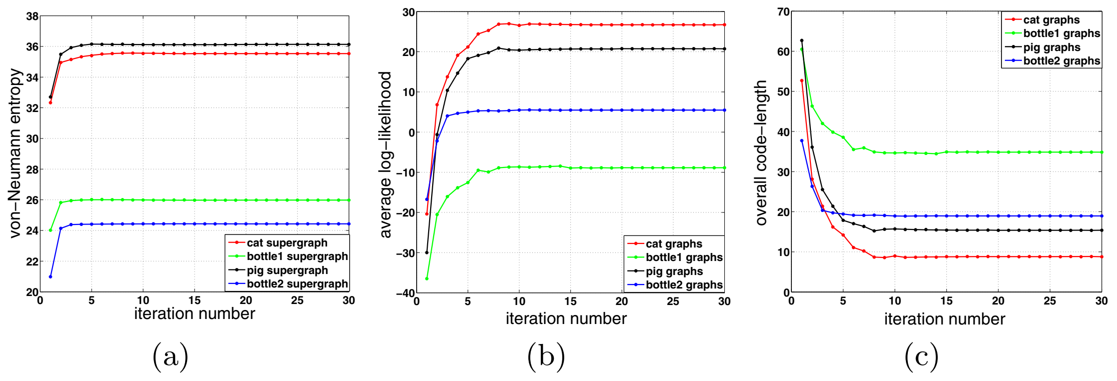


Fig. 2. COIL dataset: (a)variation of the complexity of the supergraph, encoded as von-Neumann entropy, during iterations, (b) variation of average log-likelihood of the sample graphs during iterations and (c) variation of the overall code-length during iterations.

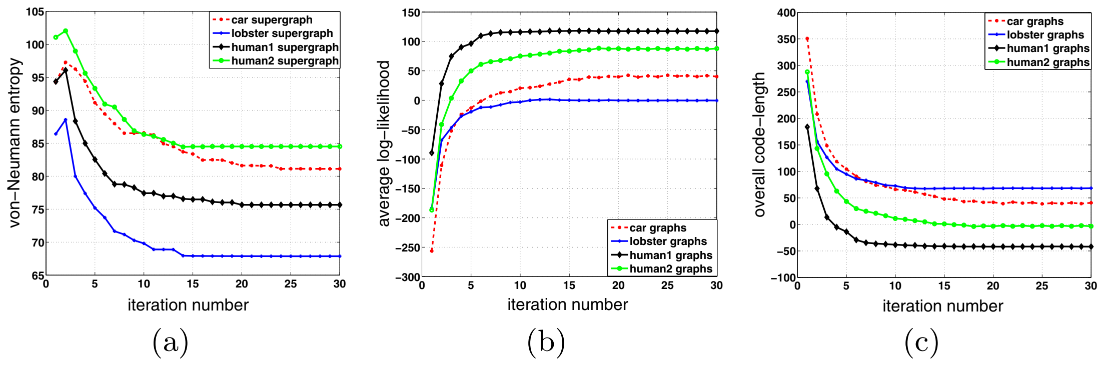


Fig. 3. Toy dataset: (a)variation of the complexity of the supergraph, encoded as von-Neumann entropy, during iterations, (b) variation of the average log-likelihood of the sample graphs during iterations and (c) variation of the overall code-length during iterations.

which gives the maximum *a posteriori* probability. The classification rate is the fraction of correctly identified objects computed using 10-fold cross validation. To perform the 10-fold cross validation for the COIL dataset, we index the 72 graphs from a same object according to their image view direction from $0°$ to $360°$, and in each fold we select 7 or 8 graphs that are equally spaced over the angular interval as test-set, and the remainder are used as as sample-set for training. The similar applies for the toys dataset. For comparison, we have also investigated the results obtained using two alternative constructions of the supergraph. The first of these is the median graph or concatenated graph used to initialize our algorithm. The second is the supergraph learned without taking its complexity into account, which means, this supergraph is learned by maximizing the likelihood function of the sample graphs given in equation (4). Table 1 shows the classification results obtained with the three different supergraph constructions. From the three constructions, it is the supergraphs learned using the MDL principle that achieve the highest classification rates on all the three classification tasks.

Table 1. Comparison of the classification results. The bold values are the average classification rates from 10-fold cross validation, followed by their standard error.

Classification Rate	cat & pig	bottle1 & bottle2	four objects (Toys)
learned supergraph(by MDL)	**0.824** ± 0.033	**0.780** ± 0.023	**0.763** ± 0.026
median graph/concatenated graph	**0.669** ± 0.052	**0.651** ± 0.023	**0.575**± 0.020
learned supergraph	**0.807** ± 0.056	**0.699** ± 0.029	**0.725** ± 0.022

6 Conclusion

In this paper, we have presented an information theoretic framework for learning a generative model of the variations in sets of graphs. The problem is posed as that of learning a supergraph. We provide a variant of EM algorithm to demonstrate how the node correspondence recover and supergraph structure estimation can be couched in terms of minimizing a description length criterion. Empirical results on real-world dataset support our proposed method by a) validating our learning algorithm and b) showing that our learned supergraph outperforms two alternative supergraph constructions. Our future work will aim to fit a mixture of supergraph to data sampled from multiple classes to perform graph clustering.

References

1. Friedman, N., Koller, D.: Being Bayesian about network structure.A Bayesian approach to structure discovery in Bayesian networks. Machine Learning, 95–125 (2003)
2. Christmas, W.J., Kittler, J., Petrou, M.: Modeling compatibility coefficient distribution. Image and Vsion Computing 14, 617–625 (1996)
3. Bagdanov, A.D., Worring, M.: First order Gaussian graphs for efficient structure classification. Pattern Recognition 36, 1311–1324 (2003)
4. Luo, B., Hancock, E.R.: A spectral approach to learning structural variations in graphs. Pattern Recognition 39, 1188–1198 (2006)
5. Torsello, A., Hancock, E.R.: Learning shape-classes using a mixture of tree-unions. IEEE PAMI 28, 954–967 (2006)
6. Luo, B., Hancock, E.R.: Structural graph matching using the EM alogrithm and singular value decomposition. IEEE PAMI 23, 1120–1136 (2001)
7. Rissanen, J.: Modelling by Shortest Data Description. Automatica, 465–471 (1978)
8. Passerini, F., Severini, S.: The von-neumann entropy of networks. arXiv:0812.2597 (2008)
9. Grunwald, P.: Minimum Description Length Tutorial. Advances in Minimum Description Length: Theory and Applications (2005)
10. Gold, S., Rangarajan, A.: A Graduated Assignment Algorithm for Graph Matching. IEEE PAMI 18, 377–388 (1996)
11. Wilson, R.C., Zhu, P.: A study of graph spectra for comparing graphs and trees. Pattern Recognition 41, 2833–2841 (2008)
12. Wilson, R.C., Hancock, E.R.: Structural matching by discrete relaxation. IEEE PAMI 19, 634–648 (1997)
13. Han, L., Wilson, R.C., Hancock, E.H.: A Supergraph-based Generative Model. In: ICPR, pp. 1566–1569 (2010)
14. Figueiredo, M.A.T., Jain, A.K.: Unsupervised learning of finite mixture models. IEEE PAMI 24, 381–396 (2002)

An Information Theoretic Approach to Learning Generative Graph Prototypes

Lin Han, Edwin R. Hancock, and Richard C. Wilson

Department of Computer Science, University of York

Abstract. We present a method for constructing a generative model for sets of graphs by adopting a minimum description length approach. The method is posed in terms of learning a generative supergraph model from which the new samples can be obtained by an appropriate sampling mechanism. We commence by constructing a probability distribution for the occurrence of nodes and edges over the supergraph. We encode the complexity of the supergraph using the von-Neumann entropy. A variant of EM algorithm is developed to minimize the description length criterion in which the node correspondences between the sample graphs and the supergraph are treated as missing data.The maximization step involves updating both the node correspondence information and the structure of supergraph using graduated assignment. In the experimental part, we demonstrate the practical utility of our proposed algorithm and show that our generative model gives good graph classification results. Besides, we show how to perform graph clustering with Jensen-Shannon kernel and generate new sample graphs.

1 Introduction

Relational graphs provide a convenient means of representing structural patterns. Examples include the arrangement of shape primitives or feature points in images, molecules and social networks. Whereas most of traditional pattern recognition and machine learning is concerned with pattern vectors, the issue of how to capture variability in graph, tree or string representations has received relatively little attention in the literature. The main reason for the lack of progress is the difficulty in developing representations that can capture variations in graph-structure. This variability can be attributed to a) variations in either node or edge attributes, b) variations in node or edge composition and c) variations in edge-connectivity.

This trichotomy provides a natural framework for analyzing the state-of-the-art in the literature. Most of the work on Bayes nets in the graphical models literature can be viewed as modeling variations in node or edge attributes [1]. Examples also include the work of Christmas et al.[2] and Bagdanov et al. [3] who both use Gaussian models to capture variations in edge attributes. The problems of modeling variations in node and edge composition are more challenging since they focus on modeling the structure of the graph rather than its attributes.

The problem of learning edge structure is probably the most challenging of those listed above. Broadly speaking there are two approaches to characterizing variations in edge structure for graphs. The first of these is graph spectral, while the second is probabilistic. In the case of graph spectra, many of the ideas developed in the generative modeling of shape using principal components analysis can be translated relatively directly to graphs using simple vectorization procedures based on the correspondences conveyed by the ordering of Laplacian eigenvectors [5, 4]. Although these methods are simple and effective, they are limited by the stability of the Laplacian spectrum under perturbations in graph-structure. The probabilistic approach is potentially more robust, but requires accurate correspondence information to be inferred from the available graph structure. If this is to hand, then a representation of edge structure can be learned. To date the most effective algorithm falling into this category exploits a part-based representation [8].

In this paper, we focus on the third problem and aim to learn a generative model that can be used to describe the distribution of structural variations present in a set of sample graphs, and in particular to characterize the variations of the edge structure present in the set. We follow Torsello and Hancock [6] and pose the problem as that of learning a generative supergraph representation from which we can sample. However, their work is based on trees, and since the trees are rooted the learning process can be effected by performing tree merging operations in polynomial time. This greedy strategy does not translate tractably to graphs where the complexity becomes exponential, and we require different strategies for learning and sampling. Torsello and Hancock realize both using edit operations, here on the other hand we use a soft-assign method for optimization and then generate new instances by Gibbs sampling.

Han, Wilson and Hancock propose a method of learning a supergraph model in [23] where they don't take into account the complexity of the supergraph model. Here, we take an information theoretic approach to estimating the supergraph structure by using a minimum description length criterion. By taking into account the overall code-length in the model, MDL allows us to select a supergraph representation that trades-off goodness-of-fit with the observed sample graphs against the complexity of the model. We adopt the probabilistic model in [7] to furnish the required learning framework and encode the complexity of the supergraph using its von-Neumann entropy[11] (i.e. the entropy of its Normalized Laplacian eigenvalues). Finally, a variant of EM algorithm is developed to minimize the total code-length criterion, in which the correspondences between the nodes of the sample graphs and those of the supergraph are treated as missing data. In the maximization step, we update both the node correspondence information and the structure of supergraph using graduated assignment. This novel technique is applied to a large database of object views, and used to learn class prototypes that can be used for the purposes of object recognition.

The remainder of this paper is organized as follows. Section 2 outlines the probabilistic framework which describes the distribution of the graph data. Section 3 explains how we encode our model so as to formulate the problem in hand in a minimum description length setting. In Section 4, we present the EM algorithm for minimizing the code-length. Section 5 provides experimental results that support our approach. Finally, section 6 offers some conclusions.

2 Probabilistic Framework

We are concerned with learning a structural model represented in terms of a so-called supergraph that can capture the variations present in a sample of graphs. In Torsello and Hancock's work [6] this structure is found by merging the set of sample trees, and so each sample tree can be obtained from it by edit operations. Here, on the other hand, we aim to estimate an adjacency matrix that captures the frequently occurring edges in the training set. To commence our development we require the *a posteriori* probabilities of the sample graphs given the structure of the supergraph and the node correspondences between each sample graph and the supergraph. To compute these probabilities we use the method outlined in [7].

Let the set of sample of graphs be $\mathcal{G} = \{G_1, ...G_i, ...G_N\}$, where the graph indexed i is $G_i = (V_i, E_i)$ with V_i the node-set and E_i the edge-set. Similarly, the supergraph which we aim to learn from this data is denoted by $\Gamma = (V_\Gamma, E_\Gamma)$, with node-set V_Γ and edge-set E_Γ. Further, we represent the structure of the two graphs using a $|V_i| \times |V_i|$ adjacency matrix D^i for the sample graph G_i and a $|V_\Gamma| \times |V_\Gamma|$ adjacency matrix M for the supergraph model Γ. The elements of the adjacency matrix for the sample graph and those for the supergraph are respectively defined to be

$$D_{ab} = \begin{cases} 1 & \text{if } (a,b) \in E_D \\ 0 & \text{otherwise} \end{cases} \quad , \quad M_{\alpha\beta} = \begin{cases} 1 & \text{if } (\alpha,\beta) \in E_\Gamma \\ 0 & \text{otherwise} \end{cases} . \tag{1}$$

We represent the correspondence matches between the nodes of the sample graph and the nodes of the supergraph using a $|V_i| \times |V_\Gamma|$ assignment matrix S^i which has elements

$$s^i_{a\alpha} = \begin{cases} 1 \text{ if } a \rightarrow \alpha \\ 0 \text{ otherwise} \end{cases} . \tag{2}$$

where $a \rightarrow \alpha$ implies that node $a \in V_i$ is matched to node $\alpha \in V_\Gamma$.

With these ingredients, according to Luo and Hancock [7] the *a posteriori* probability of the graphs G_i given the supergraph Γ and the correspondence indicators is

$$P(G_i|\Gamma, S^i) = \prod_{a \in V_i} \sum_{\alpha \in V_\Gamma} K^i_a \exp[\mu \sum_{b \in V_i} \sum_{\beta \in V_\Gamma} D^i_{ab} M_{\alpha\beta} s^i_{b\beta}] . \tag{3}$$

where

$$\mu = \ln \tfrac{1-P_e}{P_e} \quad , \quad K^i_a = P_e^{|V_i| \times |V_\Gamma|} B^i_a . \tag{4}$$

In the above, P_e is the error rate for node correspondence and B^i_a is the probability of observing node a in graph G_i , the value of which depends only on the identity of the node a . $|V_i|$ and $|V_\Gamma|$ are the number of the nodes in graph G_i and supergraph Γ.

3 Model Coding using MDL

Underpinning minimum description length is the principle that learning, or finding a hypothesis that explains some observed data and makes predictions about data yet unseen, can be viewed as finding a shorter code for the observed data [10, 13, 9]. To formalize this idea, we encode and transmit the observed data and the hypothesis, which in our case are respectively the sample graphs $\mathcal{G}$ and the supergraph structure Γ. This leads to a two-part message whose total length is given by

$$\mathcal{L}(\mathcal{G},\Gamma) = LL(\mathcal{G}|\Gamma) + LL(\Gamma) \ . \tag{5}$$

3.1 Encoding sample graphs

We first compute the code-length of the graph data. For the sample graph-set $\mathcal{G} = \{\ G_1, ...G_i, ...G_N\ \}$ and the supergraph Γ , the set of assignment matrices is $\mathcal{S} = \{S^1, ...S^i, ...S^N\}$ and these represent the correspondences between the nodes of the sample graphs and those of the supergraph. Under the assumption that the graphs in $\mathcal{G}$ are independent samples from the distribution, using the *a posteriori* probabilities from Section 2 the likelihood of the set of sample graphs is

$$P(\mathcal{G}|\Gamma,\mathcal{S}) = \prod_{G_i\in\mathcal{G}}\prod_{a\in V_i}\sum_{\alpha\in V_\Gamma} K_a^i \exp[\mu \sum_{b\in V_i}\sum_{\beta\in V_\Gamma} D^i_{ab} M_{\alpha\beta} s^i_{b\beta}] \ . \tag{6}$$

Instead of using the *Shannon-Fano code* [12], which is equivalent to the negative logarithm of the above likelihood function, we measure the code-length of the graph data using its average. Our reason is that if we adopt the former measure, then there is a bias to learning a complete supergraph that is fully connected. The reason will become clear later-on when we outline the maximization algorithm in Section 4, and we defer our justification until later. Thus, the graph code-length is $LL(\mathcal{G}|\Gamma) = -\frac{1}{|\mathcal{G}|}\sum_{G_i\in\mathcal{G}} \log P(G_i|\Gamma, S^i)$ which is the average over the set of sample graphs $\mathcal{G}$.

3.2 Encoding the supergraph model

Next, we require to compute a code-length to measure the complexity of the supergraph. For two-part codes the MDL principle does not give any guideline as to how to encode the hypotheses. Hence every code for encoding the supergraph structure is allowed, so long as it does not change with the sample size N. Here the code-length for describing supergraph complexity is chosen to be measured using the von-Neumann entropy [11]

$$H = \frac{-\sum_k \frac{\lambda_k}{2}\ln\frac{\lambda_k}{2}}{|V_\Gamma|} \ . \tag{7}$$

where $|V_\Gamma|$ is the number of nodes in the supergraph and λ_k are the eigenvalues of the normalized Laplacian matrix of the supergraph $\hat{L}$ whose elements are

$$\hat{L}_{\alpha\beta} = \begin{cases} 1 & \text{if } \alpha = \beta \\ -\frac{1}{\sqrt{T_\alpha T_\beta}} & \text{if } (\alpha,\beta) \in E_\Gamma \\ 0 & \text{otherwise} \end{cases} . \tag{8}$$

where $T_\alpha = \sum_{\xi \in V_\Gamma} M_{\alpha\xi}$ and $T_\beta = \sum_{\xi \in V_\Gamma} M_{\beta\xi}$. The normalized Laplacian matrix is commonly used as a graph representation and graph cuts [18, 19] and its eigenvalues are in the range $0 \le \lambda_k \le 2$ [17]. Divided by 2, the value of $\frac{\lambda_k}{2}$ is constrained between 0 and 1, and the von-Neumann entropy derived thereby is an intrinsic property of graphs that reflects the complexity of their structures better than other measures. We approximate the entropy $-\frac{\lambda_k}{2} \ln \frac{\lambda_k}{2}$ by the quadratic entropy $\frac{\lambda_k}{2}(1 - \frac{\lambda_k}{2})$, to obtain

$$H = \frac{-\sum_k \frac{\lambda_k}{2} \ln \frac{\lambda_k}{2}}{|V_\Gamma|} \simeq \frac{\sum_k \frac{\lambda_k}{2}(1 - \frac{\lambda_k}{2})}{|V_\Gamma|} = \frac{\sum_k \lambda_k}{2|V_\Gamma|} - \frac{\sum_k \lambda_k^2}{4|V_\Gamma|} . \tag{9}$$

Using the fact that $Tr[\hat{L}^n] = \sum_k \lambda_k^n$, the quadratic entropy can be rewritten as

$$H = \frac{Tr[\hat{L}]}{2|V_\Gamma|} - \frac{Tr[\hat{L}^2]}{4|V_\Gamma|} . \tag{10}$$

Since the normalized Laplacian matrix $\hat{L}$ is symmetric and it has unit diagonal elements, then according to equation(8) for the trace of the normalized Laplacian matrix we have

$$Tr[\hat{L}] = |V_\Gamma| . \tag{11}$$

Similarly, for the trace of the square of the normalized Laplacian, we have

$$\begin{aligned} Tr[\hat{L}^2] &= \sum_{\alpha \in V_\Gamma} \sum_{\beta \in V_\Gamma} \hat{L}_{\alpha\beta} \hat{L}_{\beta\alpha} = \sum_{\alpha \in V_\Gamma} \sum_{\beta \in V_\Gamma} (\hat{L}_{\alpha\beta})^2 \\ &= \sum_{\substack{\alpha,\beta \in V_\Gamma \\ \alpha = \beta}} (\hat{L}_{\alpha\beta})^2 + \sum_{\substack{\alpha,\beta \in V_\Gamma \\ \alpha \neq \beta}} (\hat{L}_{\alpha\beta})^2 \\ &= |V_\Gamma| + \sum_{(\alpha,\beta) \in E_\Gamma} \frac{1}{T_\alpha T_\beta} . \end{aligned} \tag{12}$$

Substituting Equation(11) and (12) into Equation (10), the entropy becomes

$$H = \frac{|V_\Gamma|}{2|V_\Gamma|} - \frac{|V_\Gamma|}{4|V_\Gamma|} - \sum_{(\alpha,\beta) \in E_\Gamma} \frac{1}{4|V_\Gamma| \; T_\alpha T_\beta} = \frac{1}{4} - \sum_{(\alpha,\beta) \in E_\Gamma} \frac{1}{4|V_\Gamma| \; T_\alpha T_\beta} . \tag{13}$$

As a result, the approximated complexity of the supergraph depends on two factors. The first is the order of supergraph, i.e. the number of nodes of the supergraph. The second is the degree of the nodes of the supergraph.

Finally, by adding together the two contributions to the code-length, the overall code-length is

$$\mathcal{L}(\mathcal{G},\mathit{\Gamma}) = LL(\mathcal{G}|\mathit{\Gamma}) + LL(\mathit{\Gamma}) = \tag{14}$$
$$-\frac{1}{|\mathcal{G}|}\sum_{G_i\in\mathcal{G}}\sum_{a\in V_i}\log\{\sum_{\alpha\in V_{\mathit{\Gamma}}} K_a^i \exp[\mu\sum_{b\in V_i}\sum_{\beta\in V_{\mathit{\Gamma}}} D_{ab}^i M_{ab} s_{b\beta}^i]\ \} + \frac{1}{4} - \sum_{(\alpha,\beta)\in E_{\mathit{\Gamma}}}\frac{1}{4|V_{\mathit{\Gamma}}|\ T_\alpha T_\beta}\ .$$

Unfortunately, due to the mixture structure, the direct estimation of the supergraph structure M from the above code-length criterion is not tractable in closed-form. For this reason, we resort to using the expectation maximization algorithm.

4 Expectation-Maximization

Having developed our computational model which poses the problem of learning the supergraph as that of minimizing the code-length, in this section, we provide a concrete algorithm to locate the supergraph structure using our code-length criterion. The minimization of the code-length is equivalent to the maximization of its negative, and we develop an EM algorithm to realize the maximization. We view the node correspondence information between the sample graphs and supergraph as missing data, and regard the structure of the supergraph as the set of parameters to be estimated. In the two interleaved steps of the EM algorithm, the expectation step involves recomputing the *a posteriori* probability of node correspondence while the maximization step involves updating both the structure of the supergraph and the node correspondence information.

4.1 Weighted code-length function

We follow Figueiredo and Jain's MDL setting of the EM algorithm [16] and make use of Luo and Hancock's log-likelihood function for correspondence matching. According to Luo and Hancock [7], treating the assignment matrix as missing data, the weighted log-likelihood function for observing a sample graph G_i, i.e. for it to have been generated by the supergraph $\mathit{\Gamma}$ is

$$\bar{\mathit{\Lambda}}^{(n+1)}(G_i|\mathit{\Gamma}, S^{i,(n+1)}) = \sum_{a\in V_i}\sum_{\alpha\in V_{\mathit{\Gamma}}} Q_{a\alpha}^{i,(n)}\{\ln\ K_a^i + \mu\sum_{b\in V_i}\sum_{\beta\in V_{\mathit{\Gamma}}} D_{ab}^i M_{\alpha\beta}^{(n)} s_{b\beta}^{i,(n+1)}\}\ . \tag{15}$$

where the superscript n indicates that quantity is taken at iteration n of the EM algorithm and $Q^{i,(n)}$ is a matrix with elements $Q_{a\alpha}^{i,(n)}$ that are set equal to the *a posteriori* probability of node a in G_i being matched to node α in $\mathit{\Gamma}$ at iteration n of the EM algorithm.

With the above likelihood function and the code-length developed in the previous section, Figueiredo and Jain's formulation of EM involves maximizing

$$\Lambda^{(n+1)}(\mathcal{G}|\Gamma,\mathcal{S}^{(n+1)}) = \frac{1}{|\mathcal{G}|}\sum_{G_i\in\mathcal{G}}\sum_{a\in V_i}\sum_{\alpha\in V_\Gamma} Q_{a\alpha}^{i,(n)}\{\ln K_a^i + \mu\sum_{b\in V_i}\sum_{\beta\in V_\Gamma} D_{ab}^i M_{\alpha\beta}^{(n)} s_{b\beta}^{i,(n+1)}\}$$
$$-\frac{1}{4}+\sum_{(\alpha,\beta)\in E_\Gamma}\frac{1}{4|V_\Gamma|\ T_\alpha^{(n)}T_\beta^{(n)}} \ . \tag{16}$$

The expression above can be simplified since the first term under the curly braces contributes a constant amount

$$\sum_{G_i\in\mathcal{G}}\sum_{a\in V_i}\sum_{\alpha\in V_\Gamma} Q_{a\alpha}^{i,(n)}\ln K_a^i = \sum_{G_i\in\mathcal{G}}\sum_{a\in V_i}\ln K_a^i \ . \tag{17}$$

Based on this observation, the critical quantity in determining the update direction is

$$\hat{\Lambda}^{(n+1)} = \tag{18}$$
$$\frac{1}{|\mathcal{G}|}\sum_{G_i\in\mathcal{G}}\sum_{a\in V_i}\sum_{\alpha\in V_\Gamma}\sum_{b\in V_i}\sum_{\beta\in V_\Gamma} Q_{a\alpha}^{i,(n)} D_{ab}^i M_{\alpha\beta}^{(n)} s_{b\beta}^{i,(n+1)} - \frac{1}{4}+\sum_{(\alpha,\beta)\in E_\Gamma}\frac{1}{4|V_\Gamma|T_\alpha^{(n)}T_\beta^{(n)}} \ .$$

4.2 Maximization

In order to optimize our weighted code-length criterion, we use graduated assignment [15] to update both the assignment matrices $\mathcal{S}$ and the structure of the supergraph, i.e. the supergraph adjacency matrix M. The updating process is realized by computing the derivatives of $\hat{\Lambda}^{(n+1)}$, and re-formulating the underlying discrete assignment problem as a continuous one using softmax[14].

In the maximization step, we have two parallel iterative update equations. The first update mode involves softening the assignment variables, while the second aims to modify the edge structure in the supergraph. Supergraph edges that are unmatchable become disjoint by virtue of having weak connection weights and cease to play any significant role in the update process. Experiments show that the algorithm appears to be numerically stable and appears to converge uniformly.

Updating Assignment Matrices: To update the assignment matrices, we commence by computing the partial derivative of the weighted code-length function in Equation (18) with respect to the elements of the assignment matrices, which gives

$$\frac{\partial\hat{\Lambda}^{(n+1)}}{\partial s_{b\beta}^{i,(n+1)}} = \frac{1}{|\mathcal{G}|}\sum_{a\in V_i}\sum_{\alpha\in V_\Gamma} Q_{a\alpha}^{i,(n)} D_{ab}^i M_{\alpha\beta}^{(n)} \ . \tag{19}$$

To ensure that the assignment variables remain constrained to lie within the rage [0,1], we adopt the soft-max update rule

$$s_{a\alpha}^{i,(n+1)} \longleftarrow \frac{\exp[\frac{1}{T}\frac{\partial \hat{\Lambda}^{(n+1)}}{\partial s_{a\alpha}^{i,(n+1)}}]}{\sum_{\alpha' \in V_\Gamma} \exp[\frac{1}{T}\frac{\partial \hat{\Lambda}^{(n+1)}}{\partial s_{a\alpha'}^{i,(n+1)}}]} \quad . \tag{20}$$

The value of the temperature T in the update process has been controlled using a slow exponential annealing schedule of the form suggested by Gold and Rangarajan[15]. Initializing T^{-1} with a small positive value and allowing it to gradually increase, the assignment variable $s_{a\alpha}^{i,(n+1)}$ corresponding to the maximum $\frac{\partial \hat{\Lambda}^{(n+1)}}{\partial s_{a\alpha}^{i,(n+1)}}$ approaches 1 while the remainder approach 0.

Updating Supergraph Structure: The partial derivative of the weighted code-length function in Equation (18) with respect to the elements of the supergraph adjacency matrix is equal to

$$\frac{\partial \hat{\Lambda}^{(n+1)}}{\partial M_{\alpha\beta}^{(n)}} = \frac{1}{|\mathcal{G}|} \sum_{G_i \in \mathcal{G}} \sum_{a \in V_i} \sum_{b \in V_i} Q_{a\alpha}^{i,(n)} D_{ab}^{i} s_{b\beta}^{i,(n+1)} - \frac{1}{4|V_\Gamma|(T_\alpha^{(n)})^2} \sum_{(\alpha,\beta') \in E_\Gamma} \frac{1}{T_{\beta'}^{(n)}} \quad . \tag{21}$$

The soft-assign update equation for the elements of the supergraph adjacency matrix is

$$M_{\alpha\beta}^{(n+1)} \longleftarrow \frac{\exp[\frac{1}{T}\frac{\partial \hat{\Lambda}^{(n+1)}}{\partial M_{\alpha\beta}^{(n)}}]}{\sum_{(\alpha',\beta') \in E_\Gamma} \exp[\frac{1}{T}\frac{\partial \hat{\Lambda}^{(n+1)}}{\partial M_{\alpha'\beta'}^{(n)}}]} \quad . \tag{22}$$

In the case of the updating of the assignment matrix elements, in each row and each column of the recovered assignment matrix no more than one element can take on unit value. By contrast, in the case of the recovered supergraph adjacency matrix there may exist multiple elements in each row or column with a unit value. To deal with this problem, in practice we set a threshold, and then recover the adjacency matrix by setting all elements larger than the threshold to unity and set the remaining elements to zero. This is repeated each time we decrease the temperature T in the annealing schedule.

From Equation (21), it is interesting to note that the derivatives of $\hat{\Lambda}^{(n+1)}$ with respect to the elements of supergraph adjacency matrix are dependent on the frequency of sample-set edges that are in correspondence with the same supergraph edge. To illustrate this point, if we approximate the matrix Q using S, then the first term in Equation (21) becomes the expectation value of the permutated adjacency matrices for the sample graphs. As a result, the elements of the supergraph adjacency matrix reflect the frequency of corresponding edges in the sample-set. The thresholding process selects frequent edges and removes unfrequent ones.

Recall that in Section 3 we discussed the encoding of the sample graphs, and chose to use the average of *Shannon-Fano code*. We can now elucidate that the reason for this choice is that as the number of the sample graphs increases, for instance in the limit as the size of the graph sample-set $\mathcal{G}$ increases, i.e. $N \to \infty$, the sum of permuted adjacency matrices of the sample graphs might dominate the magnitude of the second term in Equation (21). Thus the update algorithm might induce a complete supergraph that is fully connected. Hence, we choose to use its average rather than its sum.

4.3 Expectation

In the expectation step of the EM algorithm, we compute the *a posteriori* correspondence probabilities for the nodes of the sample graphs to the nodes of the supergraph. Applying Bayes rule, the *a posteriori* correspondence probability for the nodes of the sample graph G_i at iteration $n+1$ are given by

$$Q^{i,(n+1)}_{a\alpha} = \frac{\exp[\sum_{b\in V_i}\sum_{\beta\in V_\Gamma} D^i_{ab} M^{(n)}_{\alpha\beta} s^{i,(n)}_{b\beta}]\pi^{i,(n)}_{\alpha}}{\sum_{\alpha'\in V_\Gamma}\exp[\sum_{b\in V_i}\sum_{\beta\in V_\Gamma} D^i_{ab} M^{(n)}_{\alpha'\beta} s^{i,(n)}_{b\beta}]\pi^{i,(n)}_{\alpha'}} . \tag{23}$$

In the above equation, $\pi^{i,(n)}_{\alpha'} = \langle Q^{i,(n)}_{a\alpha'}\rangle_a$, where $\langle\ \rangle_a$ means average over a.

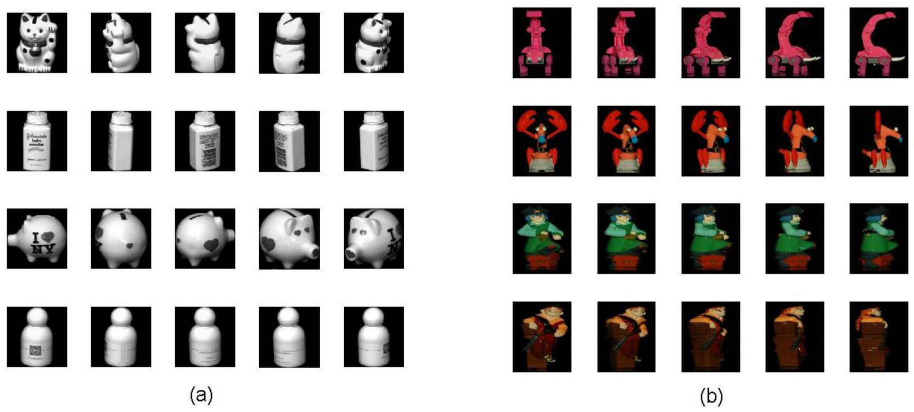


Fig. 1. (a)Example images in the COIL dataset. (b)Example images in the toys dataset.

5 Experiments

In this section, we report experimental results aimed at demonstrating the utility of our proposed generative model on real-world data. We use images from two datasets for experiments. The first dataset is the COIL [20] which consists of images of 4 objects, with 72 views of each object from equally spaced directions over 360°. We extract corner features from each image and use the detected feature points as nodes to construct sample graphs by Delaunay triangulation. The second dataset is a dataset consisting of views of toys, and contains images of 4 objects with 20 different views of each object. For this second dataset, the feature keypoints used to construct Delaunay graphs are extracted using the SIFT [21] detector. Some example images of the objects from these two datasets are given in Figure 1.

The first part of our experimental investigation aims to validate our supergraph learning method. We test our proposed algorithm on both of the two datasets and in order to better analyze our method, we initialize the supergraph in our EM algorithm with different structures. For the COIL dataset, we initialize the supergraph structure with the median graph, i.e. the sample graph with the largest *a posteriori* probability from the supergraph. On the other hand, to initialize the structure of the supergraph in the toys dataset, we match pairs of graphs from a same object using the discrete relaxation algorithm [22]. Then we concatenate(merge) the common structures over for the sample graphs from a same object to form an initial supergraph. The initial supergraph constructed in this way preserves more of the structural variations present in the set of sample graphs. The median graph, on the other hand, captures more of the common salient information. We match the sample graphs from the two datasets against their supergraphs both using graduated assignment[15] and initialize the assignment matrices in our algorithm with the resulting assignment matrices. Using these settings, we iterate the two steps of the EM algorithm 30 times, and observe how the complexity of the supergraph, the average log-likelihood of the sample graphs and the overall code-length vary with iteration number. Figures 2 and 3 respectively shows the results for the COIL and toys datasets illustrated in Figure 1.

From Figure 2(a) it is clear that the von-Neumann entropy of the supergraph increases as the iteration number increases. This indicates that the supergraph structure becomes more complex with an increasing number of iterations. Figure 2(b) shows that the average of the log-likelihood of the sample graphs increases during the iterations. Figure 2(c) shows that the overall-code length decreases and gradually converges as the number of iterations increases. For the toys dataset, the von-Neumann entropy in Figure 3(a) shows an opposite trend and decreases as the number of iterations increases. The reason for this is that the initial supergraph we used for this dataset, i.e. the concatenated supergraph, accommodates too much structural variation from the sample graphs. The reduction of the von-Neumann entropy implies some trivial edges are eliminated or relocated. As a result the supergraph structure both condenses and simplifies with increasing iteration number. Although the complexity of the graphs behaves differently, the average of the likelihood of the graphs in Figure 3(b) and the overall-code length in Figure 3(c) exhibit a similar behaviour to those for the COIL dataset. In other words, our algorithm behaves in a stable manner both increasing the likelihood of sample graphs and decreasing the overall code-length on both datasets.

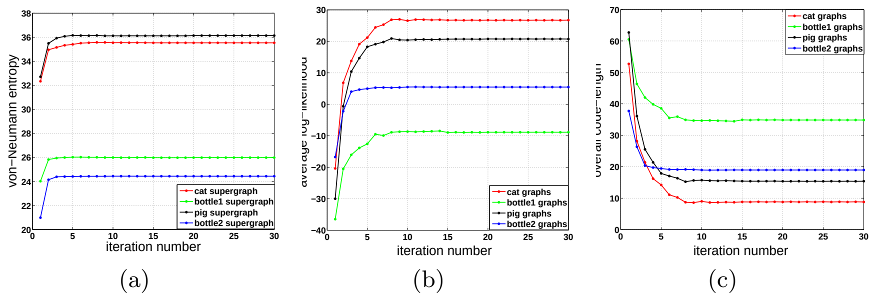


Fig. 2. COIL dataset: (a)variation of the complexity of the supergraph, encoded as von-Neumann entropy, during iterations, (b) variation of average log-likelihood of the sample graphs during iterations and (c) variation of the overall code-length during iterations.

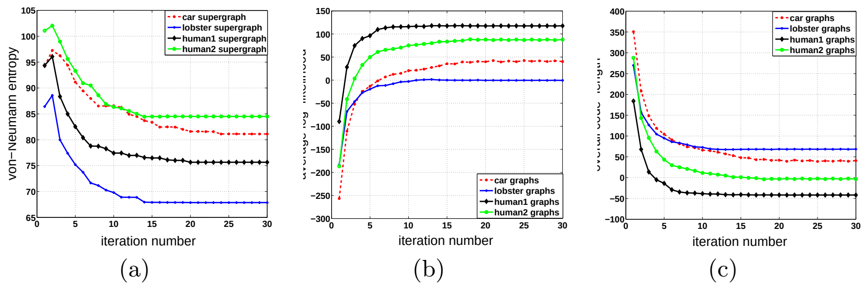


Fig. 3. Toy dataset: (a)variation of the complexity of the supergraph, encoded as von-Neumann entropy, during iterations, (b) variation of the average log-likelihood of the sample graphs during iterations and (c) variation of the overall code-length during iterations.

Our second experimental goal is to evaluate the effectiveness of our learned generative model for classifying out-of-sample graphs. From the COIL dataset, we aim 1) to distinguish images of cats from pigs on the basis of their graph representations and 2) distinguish between images of different types of bottle. For the toys dataset, on the other hand, we aim to distinguish between images of the four objects. To perform these classification tasks, we learn a supergraph for each object class from a set of samples and use Equation (3) to compute the *a posteriori* probabilities for each graph from a separate (out-of-sample) test-set. The class-label of the test graph is determined by the class of the supergraph which gives the maximum *a posteriori* probability. The classification rate is the fraction of correctly identified objects computed using 10-fold cross validation.

To perform the 10-fold cross validation for the COIL dataset, we index the 72 graphs from a same object according to their image view direction from $0°$ to $360°$, and in each fold we select 7 or 8 graphs that are equally spaced over the angular interval as test-set, and the remainder are used as as sample-set for training. The similar applies for the toys dataset. For comparison, we have also investigated the results obtained using two alternative constructions of the supergraph. The first of these is the median graph or concatenated graph used to initialize our algorithm. The second is the supergraph learned without taking its complexity into account, which means, this supergraph is learned by maximizing the likelihood function of the sample graphs given in equation (6). Table 1 shows the classification results obtained with the three different supergraph constructions. From the three constructions, it is the supergraphs learned using the MDL principle that achieve the highest classification rates on all the three classification tasks.

Classification Rate	cat & pig	bottle1 & bottle2	four objects (Toys)
learned supergraph(by MDL)	**0.824** ± 0.033	**0.780** ± 0.023	**0.763** ± 0.026
median graph/concatenated graph	**0.669** ± 0.052	**0.651** ± 0.023	**0.575**± 0.020
learned supergraph	**0.807** ± 0.056	**0.699** ± 0.029	**0.725** ± 0.022

Table 1. Comparison of the classification results. The bold values are the average classification rates from 10-fold cross validation, followed by their standard error.

We have also compared our method with a feature-based classifier. Here we apply a K-nearest neighbor classifier to the Laplacian spectrum of the graph. We perform experiments that are reported on the classification task from the COIL dataset involving images of the cat and pig. To do this, we compute the eigenvalues of the Laplacian matrix of each sample graph, and encode the spectrum as a set of eigenvalues of decreasing magnitude. Using these Laplacian spectra, we find that 10-fold cross-validation with a 3NN classifier gives an average correct classification rate of 0.625. To investigate how our learned supergraph improves the classification result. We visualize the classifications results delivered by the two methods in Figure 4. The bottom shows the classification result obtained using our generative model. Here the test images are arranged into series according to their a posterior classification probabilities. The vertical line is the Bayes decision boundary between the two objects (cat to the left and pig to the right). Each images is labeled with its index and actual identity. In the top row we show the images that are classified in error using the 3-NN classifier. Object images 56cat, 26cat, 66cat, 36cat and 26pig that are misclassified using the 3NN, are correctly classified using our learned supergraph.

Next, we investigate how to embed graphs from different objects into pattern space so as to cluster the graphs according to object identity. Here we combine the Jensen-Shannon divergence with the von-Neumann entropy to measure the pairwise dissimilarity between graphs. We then apply kernel PCA to the Jensen-

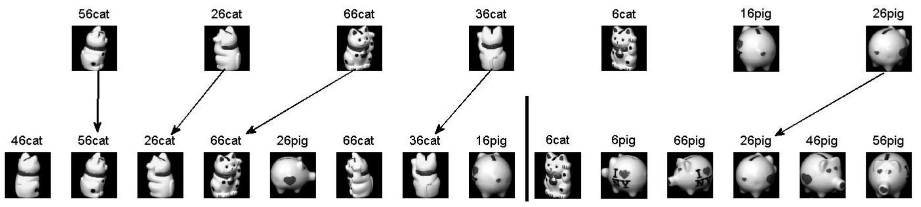


Fig. 4. Improvement of our classification result.

Shannon kernel to effect the embedding. We construct a supergraph for each pair of graphs and measure their dissimilarity using the Jensen-Shannon divergence computed from the von-Neumann entropy,

$$JSD(G_i, G_j) = H(G_i \otimes G_j) - \frac{H(G_i) + H(G_j)}{2} \quad . \tag{24}$$

In the above equation, $G_i \otimes G_j$ represents the supergraph for graphs G_i, G_j, and $H(\cdot)$ denotes the von-Neumann entropy of the corresponding graph. From the Jensen-Shannon divergence we construct a kernel $K(G_i, G_j) = JSD(G_i, G_j)$ and with the kernel matrix we embed the graphs into pattern space by kernel PCA. In order to assess the quality of the method, we compare our embedding result with that obtained by using edit distance to measure graph dissimilarity. In Figure 5, we illustrate the Jensen-Shannon embedding onto a 2D space for two object clustering tasks. The first row shows the embeddings of graphs from images of cat (red) and pig (blue). The second row shows the embedding of the graphs from two types of bottle images (bottle1 as black scatter points and bottle2 as green scatter points). The left column displays the clustering results by edit distance and the right column gives the result by Jensen-Shannon divergence. It is clear from Figure 5 that the Jensen-Shannon kernel embedding gives better clustering results than the edit distance embedding. This is especially the case for the cat and pig objects, where the cat graphs and pig graphs are heavily overlapped in the edit-distance embedding.

Finally, we explore whether our generative model can be used to generate new sample graphs. Given a supergraph structure, we use Gibbs sampling to generate some new samples. From the newly generated graphs, we select a graph that has high generating likelihood together with a graph that has low likelihood, and compare their structure with that of the median graph and the supergraph. We use black and white squares to indicate zero and unit entries respectively to represent the elements of the adjacency matrices. The adjacency matrices for the four graphs are shown in Figure 6. The example supergraph here is learned using Delaunay graphs from the 72 pig images. From Figure 6, it is clear that the supergraph, median graph and high likelihood sample graph have very similar structure. On the other hand, the low likelihood sample graph shows a very different structure. It is also important to note that the structure of the supergraph is more complex than that of the median graph, which supports

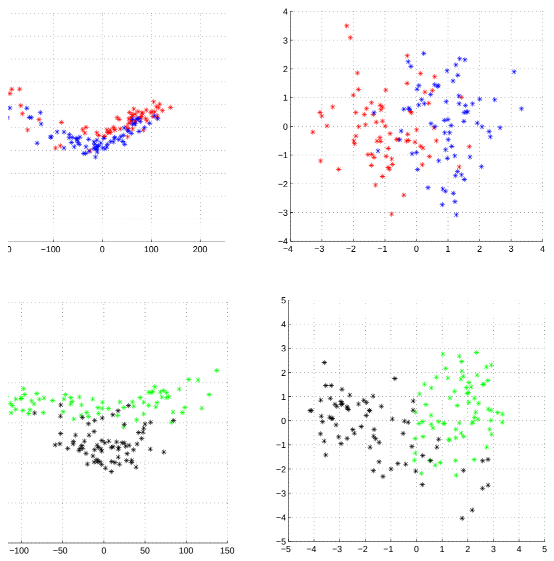


Fig. 5. Comparison of graph clusterings obtained from Jensen-Shannon kernel and edit distance. Row 1: cat (red) and pig (blue). Row 2: bottle1(black) and bottle2 (green). Column 1: edit distance and Column 2: Jensen-Shannon kernel.

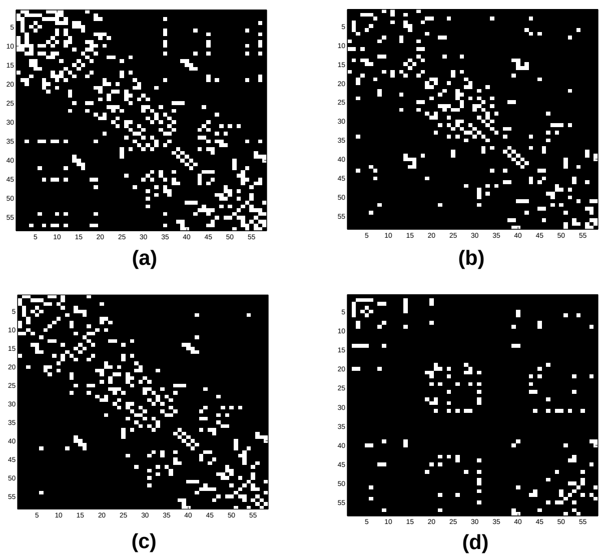


Fig. 6. The adjacency matrices of four graphs. (a) the learned supergraph, (b) a generated sample graph that has high likelihood, (c) the median graph, (d) a generated sample graph with low likelihood.

our observation that the von-Neumann entropy in Figure 2(a) increases with iteration number.

6 Conclusion

In this paper, we have presented an information theoretic framework for learning a generative model of the variations in sets of graphs. The problem is posed as that of learning a supergraph. We provide a variant of EM algorithm to demonstrate how the node correspondence recover and supergraph structure estimation can be couched in terms of minimizing a description length criterion. Empirical results on real-world dataset support our proposed method by a) validating our learning algorithm and b) showing that our learned supergraph outperforms two alternative supergraph constructions. We also have illustrated how to embed graphs using supergraphs with Jensen-Shannon divergence and investigated the performance of our generative model on generating new sample graphs. Our future work will aim to fit a mixture of supergraph to data sampled from multiple classes to perform graph clustering.

References

1. Friedman, N., Koller, D.: Being Bayesian about network structure.A Bayesian approach to structure discovery in Bayesian networks. Machine Learning, 95–125 (2003)
2. Christmas, W.J., Kittler, J., Petrou, M.: Probabilistic feature labeling schemes: modeling compatibility coefficient distribution. Image and Vision Computing 14, 617–625(1996)
3. Bagdanov, A.D., Worring, M.: First order Gaussian graphs for efficient structure classification. Pattern Recognition 36, 1311–1324 (2003)
4. Cootes, T.F., Edwards, G.J., Taylor, C.J.: Active Appearance Models. IEEE PAMI 23, 681–685 (2001)
5. Luo, B., Hancock, E.R.: A spectral approach to learning structural variations in graphs. Pattern Recognition 39, 1188–1198 (2006)
6. Torsello, A., Hancock, E.R.: Learning shape-classes using a mixture of tree-unions. IEEE PAMI 28, 954–967 (2006)
7. Luo, B., Hancock, E.R.: Structural graph matching using the EM alogrithm and singular value decomposition. IEEE PAMI 23, 1120–1136 (2001)
8. White, D., Wilson., R.C.: Parts based generative models for graphs. ICPR, pp. 1–4 (2008)
9. Rissanen., J.: Modelling by Shortest Data Description. Automatica, pp. 465–471 (1978)
10. Rissanen., J.: Stochastic complexity in statistical inquiry. World Scientific (1989)
11. Passerini, F., Severini, S.: The von neumann entropy of networks. arXiv:0812.2597 (2008)
12. Cover, T., Thomas, J.: Elements of Information Theory. New York:John Wiley&Sons (1991)
13. Grunwald, P.: Minimum Description Length Tutorial. Advances in Minimum Description Length: Theory and Applications (2005)

14. Bridle, J.S.: Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. Advances in neural information processing systems 2, pp. 211–217 (1990)
15. Gold, S., Rangarajan, A.: A Graduated Assignment Algorithm for Graph Matching. IEEE PAMI 18, 377–388 (1996)
16. Figueiredo, M.A.T., Jain, A.K.: Unsupervised learning of finite mixture models. IEEE PAMI 24, 381–396 (2002)
17. Wilson, R.C., Zhu, P.: A study of graph spectra for comparing graphs and trees. Pattern Recognition 41, 2833–2841 (2008)
18. Shi, J., Malik, J.: Normalized cuts and image segmentation. CVPR, pp. 731-737 (1997)
19. Robles-Kelly, A., Hancock, E.R.: A riemannian approach to graph embedding. Pattern Recognition 40, 1042-1056 (2007)
20. Nene, S.A., Nayar, S.K., Murase, H.:Columbiaobjectimagelibrary(COIL100). Technical Report CUCS-006-96. Department of Computer Science, Columbia University (1996)
21. Lowe, D.G.: Distinctive image features from scale invariant keypoints. IJCV 99, 91–110 (2004)
22. Wilson, R.C., Hancock, E.R.: Structural matching by discrete relaxation. IEEE PAMI 19, 634-648 (1997)
23. Han, L., Wilson, R.C., Hancock, E.R.: A Supergraph-based Generative Model. ICPR, pp. 1566-1569 (2010)

Entropy versus Heterogeneity for Graphs

Lin Han, Edwin R. Hancock, and Richard C. Wilson

Department of Computer Science, University of York

Abstract. In this paper we explore and compare two contrasting graph characterizations. The first of these is Estrada's heterogeneity index, which measures the heterogeneity of the node degree across a graph. Our second measure is the the von Neumann entropy associated with the Laplacian eigenspectrum of graphs. Here we show how to approximate the von Neumann entropy by replacing the Shannon entropy by its quadratic counterpart. This quadratic entropy can be expressed in terms of a series of permutation invariant traces, which can be computed from the node degrees in quadratic time. We compare experimentally the effectiveness of the approximate expression for the entropy with the heterogeneity index.

1 Introduction

One of the key problems that arises in the analysis on non-vectorial pattern data such as strings, trees and graphs is how to succinctly characterize such data for the purposes of clustering and classification. Unlike pattern vectors, when the analysis of tree or graph data is attempted then there is frequently no labelling or ordering of the nodes of the structure to hand.

Broadly speaking, there are three ways by which to overcome this problem. The first is to extract characteristics from the graph or tree data to-hand, and then to cluster graphs on the basis of vectors of structural characteristics [12]. The second method is to use a measure of pairwise distance between structures and resort to pairwise clustering methods [16]. The third method involves constructing a class prototype through the union or intersection of different structures[25] [19][20]. These latter two methods can prove very time consuming and even fragile since they require reliable node correspondences to hand [23][24], and this invariably requires inexact graph matching over the dataset to hand.

It is for this reason that the use of graph characteristics has proved to be an attractive one. Although there are a number of simple alternatives that can be used, such as node or edge frequency, edge density, diameter and perimeter, these have proved to be ineffective as a means of characterizing variations in intrinsic structure. Instead, it has proved necessary to resort to more complex representations. One of the most successful of these has been to use graph-spectral methods [21][22]. Here the distribution of the eigenvalues and eigenvectors can be used to construct permutation invariants that do not require node correspondences. Examples here include Laplacian spectra and characteristic polynomials. This study has recently been taken one step further by Xiao, Wilson and Hancock

[12] who have performed an analysis of the heat kernel for graphs, and have shown that the Riemann zeta function can be used to generate a number of powerful invariants from the normalized Laplacian spectrum. Another route to a unary characterization of graph structure is to define measures of intrinsic complexity. The characterization of graph complexity is a long standing problem, but recently measures based on the heat kernel have proved effective, and these include the use of Birkoff polytopes [18] and heat-flow complexity [17].

Unfortunately, both graph-spectral and heat flow complexity methods can prove computationally burdensome. The reason for this is that the computation of the graph-spectrum is cubic in the number of nodes. A much simpler alternative is the heterogeneity index recently developed by Estrada[1], who defines the heterogeneity based on simple statistics for the distribution of node degree over all pairs of linked nodes. This heterogeneity index can be expressed as a quadratic form of the Laplacian matrix of the graphs, which allows a spectral representation of graph heterogeneity.

Our aim in this paper is to explore whether more efficient complexity characterizations similar in spirit to the heterogeneity plot can be used to characterize differences in graph structure. We commence from the von Neumann entropy of a graph. This is simply the Shannon entropy associated with the spectrum of the normalized Laplacian matrix. We explore how to simplify and approximate the calculation of von Neumann entropy. Our first step is to replace the Shannon entropy by its quadratic counterpart. An analysis of the quadratic entropy reveals that it can be computed from a number of permutation invariant matrix trace expressions. This leads to a simple expression for the approximate entropy in terms of the node-degree. The expression is quadratic in the number of nodes in a graph. We compare our approximate entropy measure with Estrada's heterogeneity index. In the experiment part, we investigate whether the proposed entropy expression and the heterogeneity index are effective on graph clustering and classification tasks. We also compare how the heterogeneity H plots characterize three different kinds of graphs.

2 Graph Representation and the von Neumann Entropy

To commence, we denote the graph under study by $G = (V, E)$ where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. Further, we represent the structure of the graph using a $|V| \times |V|$ adjacency matrix whose elements are

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

The degree matrix of graph G is a diagonal matrix D whose elements are given by $D(u,u) = d_u = \sum_{v \in V} A(u,v)$. From the degree matrix and the adjacency matrix we can construct the Laplacian matrix $L = D - A$, i.e. the degree matrix minus the adjacency matrix. The elements of the Laplacian matrix are

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

The normalized Laplacian matrix is given by $\hat{L} = D^{-1/2}LD^{-1/2}$ and has elements

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

The spectral decomposition of the normalized Laplacian matrix is $\hat{L} = \Phi \Lambda \Phi^T$ where $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_{|V|})$ is a diagonal matrix with the ordered eigenvalues as elements $(0 = \lambda_1 < \lambda_2 < ... < \lambda_{|V|})$ and $\Phi = (\phi_1 | \phi_2 | ... | \phi_{|V|})$ is a matrix with the corresponding ordered orthonormal eigenvectors as columns. The normalized Laplacian matrix is positive semi-definite and so has all eigenvalues non-negative. The number of zero eigenvalues is the number of connected components in the graph. For a connected graph, there is only one eigenvalue which is equal to zero. The normalization factor means that the largest eigenvalue is less than or equal to 2, with equality only when G is bipartite. Hence all the eigenvalues of the normalize Laplacian matrix are in the range $0 \leq \lambda \leq 2$. The normalized Laplacian matrix is commonly used as a graph representation and the eigenvector ϕ_2 associated with the smallest non-zero eigenvalues λ_2 referred to as the Fiedler-vector [9] is often used in graph cuts [10][11].

The von Neumann entropy of the graph associated with the Laplacian eigenspectrum is defined as [14]

$$S = -\sum_{v=1}^{|V|} \frac{\lambda_v}{2} \ln \frac{\lambda_v}{2} \ . \tag{4}$$

We approximate the entropy $-\frac{\lambda_v}{2} \ln \frac{\lambda_v}{2}$ by the quadratic entropy $\frac{\lambda_v}{2}(1 - \frac{\lambda_v}{2})$, to obtain

$$S = -\sum_v \frac{\lambda_v}{2} \ln \frac{\lambda_v}{2} \simeq \sum_v \frac{\lambda_v}{2}(1 - \frac{\lambda_v}{2}) = \frac{\sum_v \lambda_v}{2} - \frac{\sum_v \lambda_v^2}{4} \ . \tag{5}$$

Using the fact that $Tr[\hat{L}^n] = \sum_v \lambda_v^n$, the quadratic entropy can be rewritten as

$$S = \frac{Tr[\hat{L}]}{2} - \frac{Tr[\hat{L}^2]}{4} \ . \tag{6}$$

Since the normalized Laplacian matrix $\hat{L}$ is symmetric and it has unit diagonal elements, then according to equation (3) for the trace of the normalized Laplacian matrix, we have

$$Tr[\hat{L}] = |V| \ . \tag{7}$$

Similarly, for the trace of the square of the normalized Laplacian, we have

$$\begin{aligned} Tr[\hat{L}^2] &= \sum_{u\in V}\sum_{v\in V}\hat{L}_{uv}\hat{L}_{vu} = \sum_{u\in V}\sum_{v\in V}(\hat{L}_{uv})^2 \\ &= \sum_{\substack{u,v\in V\\ u=v}}(\hat{L}_{uv})^2 + \sum_{\substack{u,v\in V\\ u\neq v}}(\hat{L}_{uv})^2 \\ &= |V| + \sum_{(u,v)\in E}\frac{1}{d_u d_v} \ . \end{aligned} \tag{8}$$

Substituting Equation (7) and (8) into Equation (6), the entropy becomes

$$S = \frac{|V|}{2} - \frac{|V|}{4} - \sum_{(u,v)\in E}\frac{1}{4\ d_u d_v} = \frac{|V|}{4} - \sum_{(u,v)\in E}\frac{1}{4\ d_u d_v} \ . \tag{9}$$

As a result, we can approximate the von Neumann entropy using two measures of graph structure. The first is the number of nodes of the graph, while the second is the degree of the nodes of the graph. The approximation bypasses calculating the Laplacian eigenvalues of a graph to estimate its von Neumann entropy.

3 Graph Heterogeneity Index and H Plot

We now turn our attention to network heterogeneity index recently developed by Estrada[1]. To develop the heterogeneity index, Estrada commences by defining a local index which measures the irregularity of an edge in the graph $(u,v)\in E$ as

$$I_{uv} = [f(d_u) - f(d_v)]^2 \ \ , \tag{10}$$

where $f(d_u)$ is a function of the node degree. Selecting $f(d_u) = d_u^{-1/2}$, the heterogeneity index proposed is defined to be the sum of the irregularity of all edges in the graph,

$$\rho'(G) = \sum_{(u,v)\in E}(d_u^{-1/2} - d_v^{-1/2})^2 \ \ . \tag{11}$$

The main advantage of defining the index as the sum of square differences of a function of node degree is that the index can be expressed in terms of a quadratic form of the Laplacian matrix of the graph. That is, let $|\mathbf{d}^{-1/2}\rangle = (d_1^{-1/2}, d_2^{-1/2}, ..., d_{|V|}^{-1/2})$ represent a column vector where d_u is the degree of the node u, the index can be written as

$$\rho'(G) = \sum_{(u,v)\in E}(d_u^{-1/2} - d_v^{-1/2})^2 = \frac{1}{2}\langle\mathbf{d}^{-1/2}|L|\mathbf{d}^{-1/2}\rangle \ \ . \tag{12}$$

The index above can also be stated in terms of the *Randić index* ${}^1R_{-1/2}$[5] of the graph,

$$\rho'(G) = \sum_{(u,v)\in E} (d_u^{-1/2} - d_v^{-1/2})^2 = |V| - 2 \sum_{(u,v)\in E} (d_u d_v)^{-1/2} = |V| - 2\ {}^1R_{-1/2} \ . \tag{13}$$

Li and Shi [2] show that for connected graphs the *Randić index* is bounded as follows

$$\sqrt{|V|-1} \le {}^1R_{-1/2} \le \frac{|V|}{2} \ , \tag{14}$$

where the lower bound is attained for star graphs and the upper bound is attained for regular graphs with $|V|$ nodes. Then the normalized heterogeneity index is defined as

$$\rho(G) = \frac{|V| - 2\ {}^1R_{-1/2}}{|V| - 2\sqrt{|V|-1}} = \frac{\sum_{(u,v)\in E} (d_u^{-1/2} - d_v^{-1/2})^2}{|V| - 2\sqrt{|V|-1}}$$

$$= \frac{1}{|V| - 2\sqrt{|V|-1}} \sum_{(u,v)\in E} \left(\frac{1}{d_u} + \frac{1}{d_v} - \frac{2}{\sqrt{d_u d_v}}\right) \ . \tag{15}$$

This is zero for regular graphs and one for star graphs, i.e., $0 \le \rho(G) \le 1$. Then heterogeneous starlike graphs are expected to have values of $\rho(G)$ close to one. On the other hand, more regular graphs are expected to have values close to zero.

Finally it is interesting to note that Von Luxburg [8] has shown that $1/d_u + 1/d_v$ is proportional to the commute time (or resistance distance) between nodes for graphs of large degree. Recall that commute time is the average of the outward hitting time and return hitting time, over all paths connecting a pair of nodes. It hence provides a non-local index of connectivity between pairs of nodes, which is non-zero even if there is no connecting edge. Apart from commute time term and constants related to the size of the graph, whereas the entropy depends on $-\sum_{(u,v)\in E} 1/d_u d_v = -\sum_{(u,v)\in E} \hat{L}^2_{u,v}$, the heterogeneity index depends on $-\sum_{(u,v)\in E} 2/\sqrt{d_u d_v} = 2\sum_{(u,v)\in E} \hat{L}_{u,v}$. Hence, the heterogeneity contains measures of both global path length distribution via commute time, and local edge structure via the elements of the normalised Laplacian. The entropy on the other hand is based only on the latter.

Using the Euler theorem [3] the *Randić index* can be expressed as follows

$${}^1R_{-1/2} = \frac{1}{2}\left[\ |V| - \frac{1}{{}^0R_{-1}} \sum_{v=2}^{|V|} \lambda_v \cos^2\theta_v\right] \ , \tag{16}$$

we can also establish a link between the normalized heterogeneity index and the spectral representation of graphs

$$\rho(G) = \frac{{}^0R_{-1}}{|V| - 2\sqrt{|V|-1}} \sum_{v=1}^{|V|} x_v^2 \ , \tag{17}$$

where $x_v = \sqrt{\lambda_v} \cos \theta_v$. λ_v is the vth eigenvalue of the Laplacian matrix of the graph and θ_v is the angle between the orthonormal eigenvector ϕ_v associated with the eigenvalue λ_v and the vector $\mathbf{d}^{-1/2}$ previously defined. Then the $\rho(G)$ can be interpreted as the sum of the squares of the projection of $\sqrt{\lambda_v}\phi_v$ onto the vector $\mathbf{d}^{-1/2}$. Define $y_v = \sqrt{\lambda_v} \sin \theta_v$, we can represent a graph by plotting x_v vs y_v for all values of v, where the heterogeneity is given by the sum of the squares of the projections of all these points on the abscissa. All the projections on y axis are positive but those on x axis can have positive and negative values. These plots are referred to as heterogeneity plots or H plots.

4 Experiments

In this section, we provide some comparative experimental evaluation of the approximate von-Neumann entropy and the heterogeneity index on both real-word dataset and synthetic dataset. The real-world dataset used is the COIL dataset[15] which consists of images of different views of several objects, with 72 views of each object from equally spaced directions over 360°. We extract corner features from each image and use the detected feature points as nodes to construct sample graphs by Delaunay triangulation. The synthetic dataset consists of Erdös-Rényi random graphs[6] generated by connecting pairs of nodes in a graph with an equal probability p ($0 \leq p \leq 1$) and scale-free graphs whose degree distribution follows the power-law distribution, the scale-free graphs here are generated with the preferential attachment algorithm of Barabási and Albert[7].

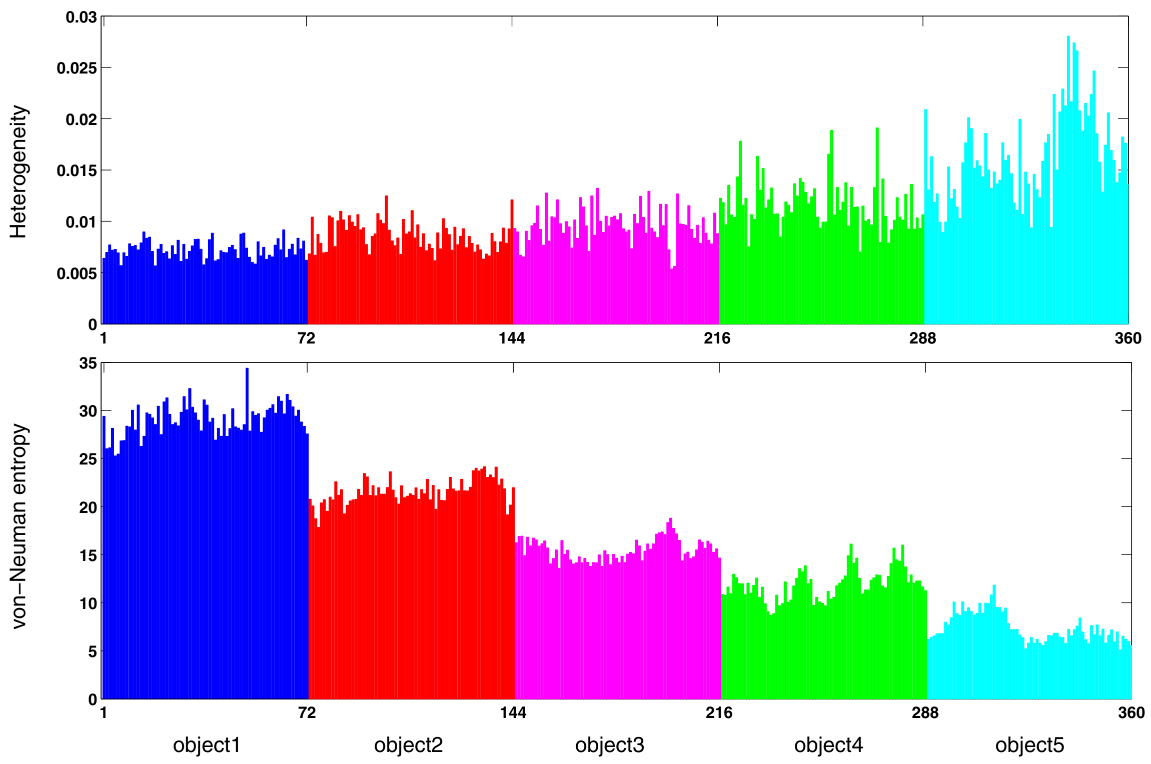


Fig. 1. (top row) The values of the heterogeneity of the Delaunay graphs. (bottom row) The values of the approximate von-Neumann entropy of the Delaunay graphs.

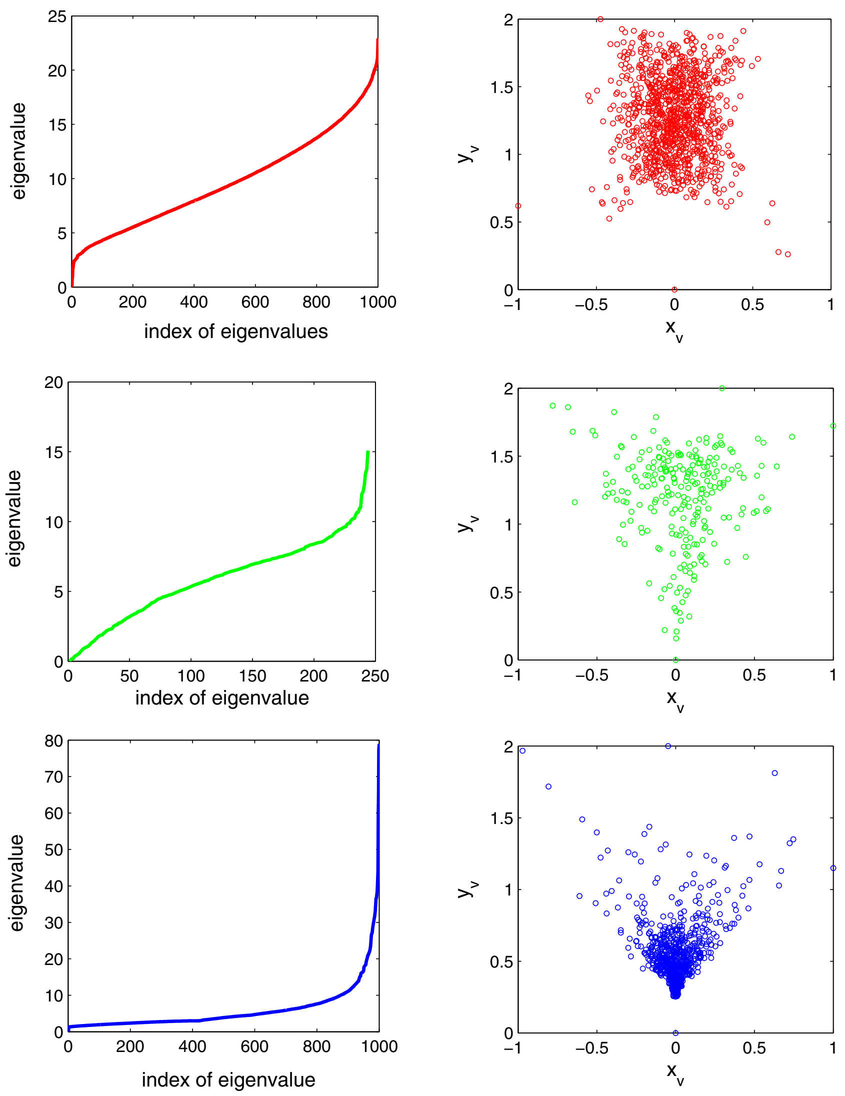


Fig. 2. Laplacian eigenvalue distributions(left column) and H plots(right column) of the ER graph (top row), the Delaunay graph(middle row) and the BA scale-free graph (bottom row)

We commerce our study by comparing the performance of the heterogeneity index and the approximate von-Neumann entropy on characterizing Delaunay graphs from the COIL dataset. To do this, we select 5 objects from the COIL dataset and plot the values of heterogeneity and the approximate von Neumann entropy for all Delaunay graphs from the 5 objects in Figure 1. Figure 1(top row) shows the values of the heterogeneity of the graphs where we use different colors to represents different objects. The values of the heterogeneity of the graphs are very small, indicating the structure of Delaunay graphs are close in structure to that of a regular graph. This can be explained by the fact that the node degree of a Delaunay graph has an average value of 5.5 and its variation is no more than 3. However, the values of the heterogeneity heavily overlap between graphs from different objects. Thus the heterogeneity index can not be used to distinguish Delaunay graphs from different objects. On the other hand, in Figure 1(bottom row), the values of the von-Neumann entropy exhibit good

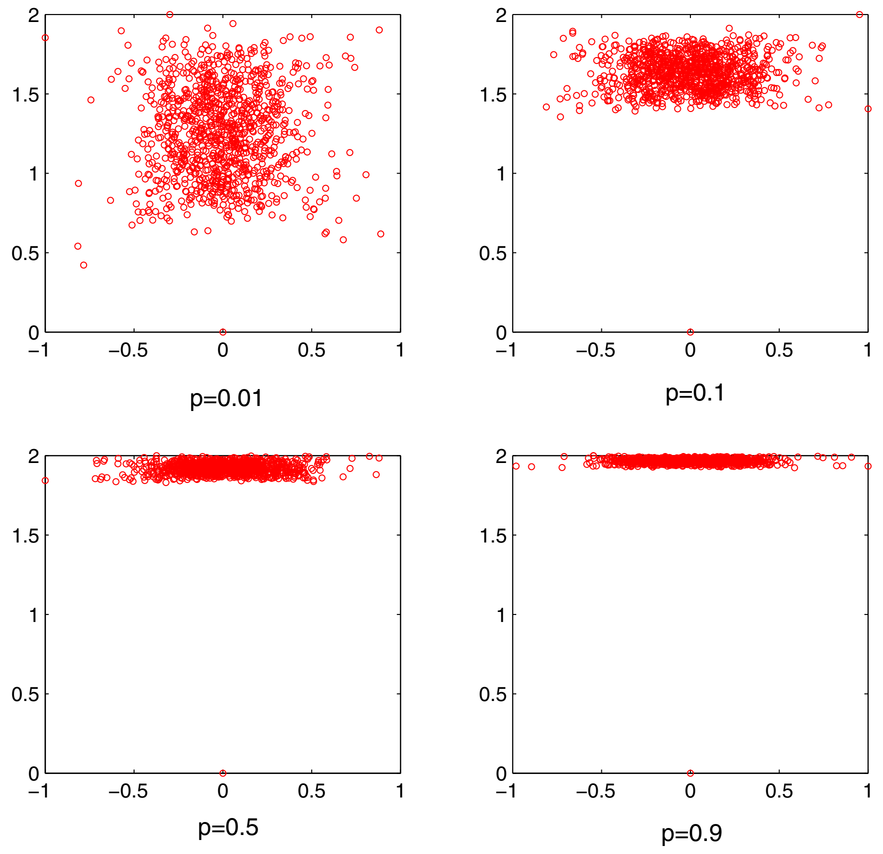


Fig. 3. H plot changes as the p value increases

separation for different objects. When we apply a 3-nearest neighbor classifier to the heterogeneity of the Delaunay graphs, its average classification rate computed using 10-fold cross-validation is 58%, much lower than the classification rate of the approximate von-Neumann entropy which is 92%.

We now turn our attention to comparing the spectral heterogeneity H plot for three kinds of graphs, namely, ER graphs, Delaunay graphs and scale-free graphs. In Figure 2 we illustrate the spectral H plots for an ER graph, a scale-free graph and a Delaunay graph as well as their Laplacian eigenvalue distributions. The ER graph here has a node-pair connecting probability $p = 0.01$, the node number of the ER graph and the scale-free graph is 1000 and that of the Delaunay graph is 250.

The left column in Figure 2 shows the Laplacian eigenvalue distributions of the three graphs. Zhang et al.[4] have observed that for the BA scale-free networks and ER random-graph networks, the Laplacian eigenvalue curves are similar to their node-degree curves. Our result is consistent with their observation by showing that the eigenvalue of the ER graph (top row) has a Poisson-like distribution, while that of the BA scale-free graph (bottom row) has a power-law distribution. Besides, we observe that eigenvalue of the Delaunay graph exhibits a similar distribution to that of the ER graph. The right column in Figure 2 shows the H plots x_v vs y_v for the three graphs where we normalize the values on the x axis between -1 and 1, and those of the y axis between 0 and 2 to have similar length in both scales. We observe that the H plot for the ER graph with $p = 0.01$ has a regular distribution of the points with an almost squared shape. In the case of the Delaunay graph, the H plot is characterized by an inverted triangle shape. The H plot for the BA scale-free graph exhibits a similar shape to that of the Delaunay graph, whereas most of the points in the plot are distributed around (0,0.2).

From the Figure 2 it is clear that the H plot for ER graph with p=0.01 has a square shape, we now investigate whether it holds for all ER graphs. To this end, we increase the value of node-pair connecting probability p from 0.01 to 1 and generate ER graphs with different p values. Figure 3 shows the H plot for four ER graphs whose p values are respectively 0.01, 0.1, 0.5 and 0.9. It is clear from Figure 3 that the shape of the H plot for ER graphs has a square shape when the value of p is very small. As p increases, the distribution of the value of y_v becomes condensed and the shape of the H plot becomes rectangle and finally closes to a line.

5 Conclusion

In this paper we show how to use the von Neumann entropy computed from the Laplacian eigenspectrum to characterize graphs. We approximate the Shannon term in the definition of the von Neumann entropy in a quadratic manner. This approximation leads to an expression for the von Neumann entropy in terms of the number of nodes and node degrees. In our experiments, we compare the von Neumann entropy measure with the heterogeneity index on an object classification task and show its effectiveness. Experimental results also reveal that the spectral heterogeneity H plot for the Delaunay graphs exhibits an inverted triangle shape and that of the ER graph tends to a rectangle shape as we increase the node-pair connecting probability p.

References

1. Estrada, E.: Quantifying network heterogeneity. E. Estrada. Quantifying network heterogeneity. Physical Review E 82, 66102 (2010)
2. Li, X., Shi, Y.: A Survey on the Randic Index. MATCH: Communication in Mathematical and in Computer Chemistry 59, 127–156 (2008)
3. Hohn, F.E.: Elementary matrix algebra. Dover, New York (1973)
4. Zhan, C., Chen, G., Yeung, L.F.: On the distributions of Laplacian eigenvalues versus node degrees in complex networks. Physica A 389, 1779–1788 (2010)
5. Randić, M.: Characterization of molecular branching. Journal of the American Chemical Society 97, 6609–6615 (1975)
6. Erdös, P., Rényi, A.: On Random Graphs. Publicationes Mathematicae 6, 290–297 (1959)
7. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science 286, 509–512 (1999)
8. Maier, M., von Luxburg, U., Hein, M.: Influence of Graph Construction on Graph-Based Clustering Measures. In: NIPS, pp. 1–9 (2010)

9. Chung, F.R.K.: Spectral Graph Theory. American Mathematical Society, Providence (1997)
10. Robles-Kelly, A., Hancock, E.R.: A Riemannian Approach to Graph Embedding. Pattern Recognition, 1042–1056 (2007)
11. Shi, J., Malik, J.: Normalized Cuts and Image Segmentation. In: CVPR, pp. 731–737 (1997)
12. Xiao, B., Hancock, E.R., Wilson, R.C.: Graph Characteristic from the Heat Kernel Trace. Pattern Recognition 42, 2589–2606 (2009)
13. McKay, B.D.: Spanning Trees in regular Graphs. European Journal of Combinatorics 4, 149–160 (1983)
14. Passerini, F., Severini, S.: The von Neumann entropy of networks. arXiv:0812.2597 (2008)
15. Nene, S.A., Nayar, S.K., Murase,H.: Columbiaobjectimagelibrary(coil100). Technical Report,Department of Computer Science, Columbia University (1996)
16. Torsello, A., Robles-Kelly, A., Hancock, E.R.: Discovering Shape Classes using Tree Edit-Distance and Pairwise Clustering. IJCV 72(3), 259–285 (2007)
17. Escolano, F., Lozano, M.A., Hancock, E.R., Giorgi, D.: What is the complexity of a network? The heat flow-thermodynamic depth approach. In: Hancock, E.R., Wilson, R.C., Windeatt, T., Ulusoy, I., Escolano, F. (eds.) SSPR&SPR 2010. LNCS, vol. 6218, pp. 286–295. Springer, Heidelberg (2010)
18. Escolano, F., Hancock, E.R., Lozano, M.A.: Polytopal graph complexity, matrix permanents, and embedding. 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. 237–246. Springer, Heidelberg (2008)
19. Torsello, A., Hancock, E.R.: Graph Embedding Using Tree Edit-union. Pattern Recognition 40(5), 1393–1405 (2007)
20. Suau, P., Escolano, F.: Bayesian Optimization of the Scale Saliency Filter. Image and Vision Computing 26(9), 1207–1218 (2008)
21. Luo, B., Wilson, R.C., Hancock, E.R.: A Spectral Approach to Learning Structural Variations in Graphs. Pattern Recognition 39(6), 1188–1198 (2006)
22. Wilson, R.C., Hancock, E.R., Luo, B.: Pattern Vectors from Algebraic Graph Theory. IEEE PAMI 27(7), 1112–1124 (2005)
23. Ferrer, M., Valveny, E., Serratosa, F., Bunke, H.: Exact median graph computation via graph embedding. 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. 15–24. Springer, Heidelberg (2008)
24. Ferrer, M., Serratosa, F., Valveny, E.: On the relation between the median and the maximum common subgraph of a set of graphs. In: Escolano, F., Vento, M. (eds.) GbRPR. LNCS, vol. 4538, pp. 351–360. Springer, Heidelberg (2007)
25. Riesen, K., Neuhaus, M., Bunke, H.: Graph embedding in vector spaces by means of prototype selection. In: Escolano, F., Vento, M. (eds.) GbRPR. LNCS, vol. 4538, pp. 383–393. Springer, Heidelberg (2007)