An Introduction to Diffusion Maps

J. de la Porte†*, B. M. Herbst*†*, W. Hereman*⋆*, S. J. van der Walt*†

† Applied Mathematics Division, Department of Mathematical Sciences,
University of Stellenbosch, South Africa
⋆ Colorado School of Mines, United States of America
jolanidlp@googlemail.com, herbst@sun.ac.za,
hereman@mines.edu, stefan@sun.ac.za

Abstract

This paper describes a mathematical technique [1] for dealing with dimensionality reduction. Given data in a high-dimensional space, we show how to find parameters that describe the lower-dimensional structures of which it is comprised. Unlike other popular methods such as Principle Component Analysis and Multi-dimensional Scaling, diffusion maps are non-linear and focus on discovering the underlying manifold (lower-dimensional constrained "surface" upon which the data is embedded). By integrating local similarities at different scales, a global description of the data-set is obtained. In comparisons, it is shown that the technique is robust to noise perturbation and is computationally inexpensive. Illustrative examples and an open implementation are given.

1. Introduction: Dimensionality Reduction

The *curse of dimensionality*, a term which vividly reminds us of the problems associated with the processing of high-dimensional data, is ever-present in today's information-driven society. The dimensionality of a data-set, or the number of variables measured per sample, can easily amount to thousands. Think, for example, of a 100 by 100 pixel image, where each pixel can be seen to represent a variable, leading to a dimensionality of $10,000$. In such a high-dimensional feature space, data points typically occur sparsely, causes numerous problems: some algorithms slow down or fail entirely, function and density estimation become expensive or inaccurate, and global similarity measures break down [4].

The breakdown of common similarity measures hampers the efficient organisation of data, which, in turn, has serious implications in the field of pattern recognition. For example, consider a collection of $n \times m$ images, each encoding a digit between 0 and 9. Furthermore, the images differ in their orientation, as shown in Fig.1. A ***human***, faced with the task of organising such images, would likely first notice the different digits, and thereafter that they are oriented. The observer intuitively attaches greater value to parameters that encode larger variances in the observations, and therefore clusters the data in 10 groups, one for each digit. Inside each of the 10 groups, digits are furthermore arranged according to the angle of rotation. This organisation leads to a simple two-dimensional parametrisation, which significantly reduces the dimensionality of the data-set, whilst preserving all important attributes.

Figure 1: Two images of the same digit at different rotation angles.

On the other hand, a ***computer*** sees each image as a data point in $\mathbb{R}^{nm}$, an nm-dimensional coordinate space. The data points are, by nature, organised according to their position in the coordinate space, where the most common similarity measure is the Euclidean distance.

A small Euclidean distance between vectors almost certainly indicate that they are highly similar. A large distance, on the other hand, provides very little information on the nature of the discrepancy. This Euclidean distance therefore provides a good measure of *local similarity* only. In higher dimensions, distances are often large, given the sparsely populated feature space.

Key to non-linear dimensionality reduction is the realisation that data is often embedded in (lies on) a lower-dimensional structure or manifold, as shown in Fig. 2. It would therefore be possible to characterise the data and the relationship between individual points using fewer dimensions, if we were able to measure distances on the manifold itself instead of in Euclidean space. For example, taking into account its global structure, we could represent the data in our digits data-set using only two variables: digit and rotation.

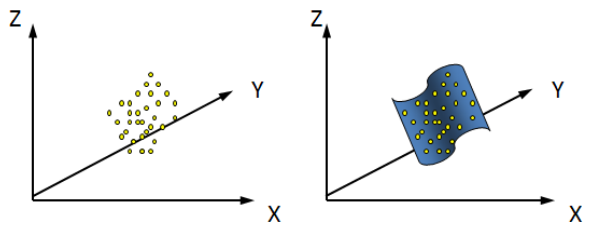

Figure 2: Low dimensional data measured in a high-dimensional space.

The challenge, then, is to determine the lower-dimensional data structure that encapsulates the data, leading to a meaningful parametrisation. Such a representation achieves dimensionality reduction, whilst preserving the important relationships between data points. One realisation of the solution is diffusion maps.

In Section 2, we give an overview of three other well known techniques for dimensionality reduction. Section 3 introduces diffusion maps and explains their functioning. In Section 4, we apply the knowledge gained in a real world scenario. Section 5 compares the performance of diffusion maps against the other techniques discussed in Section 2. Finally, Section 6 demonstrates the organisational ability of diffusion maps in an image processing example.

2. Other Techniques for Dimensionality Reduction

There exist a number of dimensionality reduction techniques. These can be broadly categorised into those that are able to detect non-linear structures, and those that are not. Each method aims to preserve some specific property of interest in the mapping. We focus on three well known techniques: Principle Component Analysis (PCA), multi-dimensional scaling (MDS) and isometric feature map (isomap).

2.1. Principal Component Analysis (PCA)

PCA [3] is a linear dimensionality reduction technique. It aims to find a linear mapping between a high dimensional space (n dimensional) and a subspace (d dimensional with $d < n$) that captures most of the variability in the data. The subspace is specified by d orthogonal vectors: the principal components. The PCA mapping is a projection into that space.

The principal components are the dominant eigenvectors (i.e., the eigenvectors corresponding to the largest eigenvalues) of the covariance matrix of the data.

Principal component analysis is simple to implement, but many real-world data-sets have non-linear characteristics which a PCA mapping fails to encapsulate.

2.2. Multidimensional Scaling (MDS)

MDS [6] aims to embed data in a lower dimensional space in such a way that pair-wise distances between data points, $X_{1..N}$, are preserved. First, a distance matrix D_X is created. Its elements contain the distances between points in the feature space, i.e. $D_X[i,j] = d(x_i, x_j)$. For simplicity sake, we consider only Euclidean distances here.

The goal is to find a lower-dimensional set of feature vectors, $Y_{1..N}$, for which the distance matrix, $D_Y[i,j] = d(y_i, y_j)$, minimises a cost function, $\rho(D_X, D_Y)$. Of the different cost functions available, *strain* is the most popular (MDS using *strain* is called "classical MDS"):

$$\rho_{\text{strain}}(D_X, D_Y) = ||J^T(D_X^2 - D_Y^2)J)||_F^2.$$

Here, J is the centering matrix, so that $J^T X J$ subtracts the vector mean from each component in X. The Frobenius matrix norm, $||X||_F$, is defined as $\sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}|x_{ij}|^2}$.

The intuition behind this cost function is that it preserves variation in distances, so that scaling by a constant factor has no influence [2]. Minimising the strain has a convenient solution, given by the dominant eigenvectors of the matrix $-\frac{1}{2}J^T D_X^2 J$.

MDS, when using Euclidean distances, is criticised for weighing large and small distances equally. We mentioned earlier that large Euclidean distances provide little information on the global structure of a data-set, and that only local similarity can be accurately inferred. For this reason, MDS cannot capture non-linear, lower-dimensional structures according to their true parameters of change.

2.3. Isometric Feature Map (Isomap)

Isomap [5] is a non-linear dimensionality reduction technique that builds on MDS. Unlike MDS, it preserves geodesic distance, and not Euclidean distance, between data points. The geodesic represents a straight line in curved space or, in this application, the shortest curve along the geometric structure defined by our data points [2]. Isomap seeks a mapping such that the geodesic distance between data points match the corresponding Euclidean distance in the transformed space. This preserves the true geometric structure of the data.

How do we approximate the geodesic distance between points without knowing the geometric structure of our data? We assume that, in a small neighbourhood (determined by K-nearest neighbours or points within a specified radius), the Euclidean distance is a good approximation for the geodesic distance. For points further apart, the geodesic distance is approximated as the sum of Euclidean distances along the shortest connecting path. There exist a number of graph-based algorithms for calculating this approximation.

Once the geodesic distance has been obtained, MDS is performed as explained above.

A weakness of the isomap algorithm is that the approximation of the geodesic distance is not robust to noise perturbation.

3. Diffusion maps

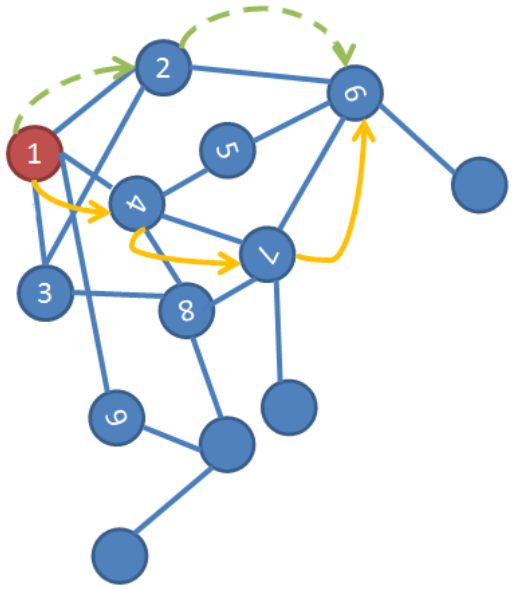

Figure 3: A random walk on a data set. Each "jump" has a probability associated with it. The dashed path between nodes 1 and 6 requires two jumps (i.e., two time units) with the probability along the path being $p(\text{node } 1,\ \text{node } 2)\, p(\text{node } 2,\ \text{node } 6)$.

Diffusion maps are a non-linear technique. It achieves dimensionality reduction by re-organising data according to parameters of its underlying geometry.

The connectivity of the data set, measured using a local similarity measure, is used to create a time-dependent diffusion process. As the diffusion progresses, it integrates local geometry to reveal geometric structures of the data-set at different scales. Defining a time-dependent diffusion metric, we can then measure the similarity between two points at a specific scale (or time), based on the revealed geometry.

A diffusion map embeds data in (transforms data to) a lower-dimensional space, such that the Euclidean distance between points approximates the diffusion distance in the original feature space. The dimension of the diffusion space is determined by the geometric structure underlying the data, and the accuracy by which the diffusion distance is approximated. The rest of this section discusses different aspects of the algorithm in more detail.

3.1. Connectivity

Suppose we take a random walk on our data, jumping between data points in feature space (see Fig. 3). Jumping to a nearby data-point is more likely than jumping to another that is far away. This observation provides a relation between distance in the feature space and probability.

The connectivity between two data points, x and y, is defined as the probability of jumping from x to y in one step of the random walk, and is

$$\text{connectivity}(x, y) = p(x, y). \tag{1}$$

It is useful to express this connectivity in terms of a non-normalised likelihood function, k, known as the diffusion kernel:

$$\text{connectivity}(x, y) \propto k(x, y). \tag{2}$$

The kernel defines a local measure of similarity within a certain neighbourhood. Outside the neighbourhood, the function quickly goes to zero. For example, consider the popular Gaussian kernel,

$$k(x, y) = \exp\left(-\frac{|x - y|^2}{\alpha}\right). \tag{3}$$

The neighbourhood of x can be defined as all those elements y for which $k(x, y) \geq \varepsilon$ with $0 < \varepsilon \ll 1$. This defines the area within which we trust our local similarity measure (e.g. Euclidean distance) to be accurate. By tweaking the kernel scale (α, in this case) we choose the size of the neighbourhood, based on prior knowledge of the structure and density of the data. For intricate, non-linear, lower-dimensional structures, a small neighbourhood is chosen. For sparse data, a larger neighbourhood is more appropriate.

The diffusion kernel satisfies the following properties:

1. k is symmetric: $k(x, y) = k(y, x)$
2. k is positivity preserving: $k(x, y) \geq 0$

We shall see in Section 8 that the first property is required to perform spectral analysis of a distance matrix, $K_{ij} = k(x_i, x_j)$. The latter property is specific to the diffusion kernel and allows it to be interpreted as a scaled probability (which must always be positive), so that

$$\frac{1}{d_X} \sum_{y \epsilon X} k(x, y) = 1. \tag{4}$$

The relation between the kernel function and the connectivity is then

$$\text{connectivity}(x, y) = p(x, y) = \frac{1}{d_X} k(x, y) \tag{5}$$

with $\frac{1}{d_X}$ the normalisation constant.

Define a row-normalised diffusion matrix, P, with entries $P_{ij} = p(X_i, X_j)$. Each entry provides the connectivity between two data points, X_i and X_j, and encapsulates what is known locally. By analogy of a random walk, this matrix provides the probabilities for a single step taken from i to j. By taking powers of the diffusion matrix[1], we can increase the number of steps taken. For example, take a 2×2 diffusion matrix,

$$P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}.$$

Each element, $P_{i,j}$, is the probability of jumping between data points i and j. When P is squared, it becomes

$$P^2 = \begin{bmatrix} p_{11}p_{11} + p_{12}p_{21} & p_{12}p_{22} + p_{11}p_{12} \\ p_{21}p_{12} + p_{22}p_{21} & p_{22}p_{22} + p_{21}p_{12} \end{bmatrix}.$$

Note that $P_{11} = p_{11}p_{11} + p_{12}p_{21}$, which sums two probabilities: that of staying at point 1, and of moving from point 1 to point 2 and back. When making two jumps, these are all the paths from point i to point j. Similarly, P^t_{ij} sum all paths of length t from point i to point j.

3.2. Diffusion Process

As we calculate the probabilities P^t for increasing values of t, we observe the data-set at different scales. This is the diffusion process[2], where we see the local connectivity integrated to provide the global connectivity of a data-set.

With increased values of t (i.e. as the diffusion process "runs forward"), the probability of following a path along the underlying geometric structure of the data set increases. This happens because, along the geometric structure, points are dense and therefore highly connected (the connectivity is a function of the Euclidean distance between two points, as discussed in Section 2). Pathways form along short, high probability jumps. On the other hand, paths that do *not* follow this structure include one or more long, low probability jumps, which lowers the path's overall probability.

In Fig. 4, the red path becomes a viable alternative to the green path as the number of steps increases. Since it consists of short jumps, it has a high probability. The green path keeps the same, low probability, regardless of the value of t .

[1]The diffusion matrix can be interpreted as the transition matrix of an ergodic Markov chain defined on the data [1]

[2]In theory, a random walk is a discrete-time stochastic process, while a diffusion process considered to be a continuous-time stochastic process. However, here we study discrete processes only, and consider random walks and diffusion processes equivalent.

Figure 4: Paths along the true geometric structure of the data set have high probability.

3.3. Diffusion Distance

The previous section showed how a diffusion process reveals the global geometric structure of a data set. Next, we define a diffusion metric based on this structure. The metric measures the similarity of two points in the observation space as the connectivity (probability of "jumping") between them. It is related to the diffusion matrix P, and is given by

$$D_t(X_i, X_j)^2 = \sum_{u \epsilon X} |p_t(X_i, u) - p_t(X_j, u)|^2 \quad (6)$$

$$= \sum_k \left| P^t_{ik} - P^t_{kj} \right|^2. \quad (7)$$

The diffusion distance is small if there are many high probability paths of length t between two points. Unlike isomap's approximation of the geodesic distance, the diffusion metric is robust to noise perturbation, as it sums over all possible paths of length t between points.

As the diffusion process runs forward, revealing the geometric structure of the data, the main contributors to the diffusion distance are paths *along* that structure.

Consider the term $p_t(x, u)$ in the diffusion distance. This is the probability of jumping from x to u (for any u in the data set) in t time units, and sums the probabilities of all possible paths of length t between x and u. As explained in the previous section, this term has large values for paths along the underlying geometric structure of the data. In order for the diffusion distance to remain small, the path probabilities between x, u and u, y must be roughly equal. This happens when x and y are both well connected via u.

The diffusion metric manages to capture the similarity of two points in terms of the true parameters of change in the underlying geometric structure of the specific data set.

3.4. Diffusion Map

Low-dimensional data is often embedded in higher dimensional spaces. The data lies on some geometric structure or manifold, which may be non-linear (see Fig. 2). In the previous section, we found a metric, the diffusion distance, that is capable of approximating distances *along* this structure. Calculating diffusion distances is computationally expensive. It is therefore convenient to map data

points into a Euclidean space according to the diffusion metric. The diffusion distance in data space simply becomes the Euclidean distance in this new *diffusion space*.

A diffusion map, which maps coordinates between data and diffusion space, aims to re-organise data according to the diffusion metric. We exploit it for reducing dimensionality.

The diffusion map preserves a data set's intrinsic geometry, and since the mapping measures distances on a lower-dimensional structure, we expect to find that fewer coordinates are needed to represent data points in the new space. The question becomes which dimensions to neglect, in order to preserve diffusion distances (and therefore geometry) optimally.

With this in mind, we examine the mapping

$$Y_i := \begin{bmatrix} p_t(X_i, X_1) \\ p_t(X_i, X_2) \\ \vdots \\ p_t(X_i, X_N) \end{bmatrix} = P^T_{i*}. \tag{8}$$

For this map, the Euclidean distance between two mapped points, Y_i and Y_j, is

$$\begin{aligned} \|Y_i - Y_j\|^2_E &= \sum_{u \epsilon X} |p_t(X_i, u) - p_t(X_j, u)|^2 \\ &= \sum_k \left|P^t_{ik} - P^t_{kj}\right|^2 = D_t(X_i, Y_j),^2 \end{aligned}$$

which is the diffusion distance between data points X_i and X_j. This provides the re-organisation we sought according to diffusion distance. Note that no dimensionality reduction has been achieved yet, and the dimension of the mapped data is still the sample size, N.

Dimensionality reduction is done by neglecting certain dimensions in the diffusion space. Which dimensions are of less importance? The proof in Section 8 provides the key. Take the normalised diffusion matrix,

$$P = D^{-1}K,$$

where D is the diagonal matrix consisting of the row-sums of K. The diffusion distances in (8) can be expressed in terms of the eigenvectors and -values of P as

$$Y'_i = \begin{bmatrix} \lambda^t_1 \psi_1(i) \\ \lambda^t_2 \psi_2(i) \\ \vdots \\ \lambda^t_n \psi_n(i) \end{bmatrix}, \tag{9}$$

where $\psi_1(i)$ indicates the i-th element of the first eigenvector of P. Again, the Euclidean distance between mapped points $Y'_i$and Y'_j is the diffusion distance. The set of orthogonal left eigenvectors of P form a basis for the diffusion space, and the associated eigenvalues λ_l indicate the importance of each dimension. Dimensionality reduction is achieved by retaining the m dimensions associated with the dominant eigenvectors, which

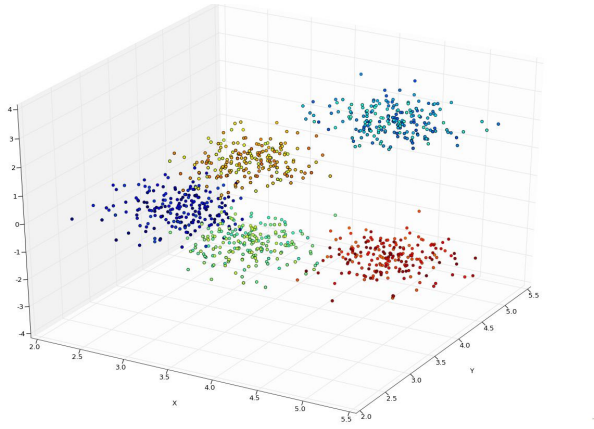

Figure 5: Original Data

ensures that $\left\|Y'_i - Y'_j\right\|$ approximates the diffusion distance, $D_t(X_i, X_j)$, best. Therefore, the diffusion map that optimally preserves the intrinsic geometry of the data is (9).

4. Diffusion Process Experiment

We implemented a diffusion map algorithm in the Python programming language. The code was adapted for the machine learning framework Elefant (Efficient Learning, Large Scale Inference and Optimisation Toolkit), and will be included as part of the next release. The basic algorithm is outlined in Algorithm 1.

Algorithm 1 Basic Diffusion Mapping Algorithm

INPUT: High dimensional data set X_i, $i = 0 \ldots N-1$.

1. Define a kernel, $k(x, y)$ and create a kernel matrix, K, such that $K_{i,j} = k(X_i, X_j)$.
2. Create the diffusion matrix by normalising the rows of the kernel matrix.
3. Calculate the eigenvectors of the diffusion matrix.
4. Map to the d-dimensional diffusion space at time t, using the d dominant eigenvectors and -values as shown in (9).

OUTPUT: Lower dimensional data set Y_i, $i = 0..N-1$.

The experiment shows how the algorithm integrates local information through a time-dependent diffusion to reveal structures at different time scales. The chosen data-set exhibits different structures on each scale. The data consists of 5 clusters which, on an intermediate scale, has a noisy C-shape with a single parameter: the position along the C-shape. This parameter is encoded in the colours of the clusters. On a global scale the structure is one super-cluster.

As the diffusion time increases, we expect different scales to be revealed. A one dimensional curve characterised by a single parameter should appear. This is the position along the C-shape, the direction of maximum global variation in the data. The ordering of colours along the C-shape should be preserved.

4.1. Discussion

In these results, three different geometric structures are revealed, as shown in figures (5) and (6):

At $t = 1$, the local geometry is visible as five clusters. The diffusion process is still restricted individual clusters, as the probability of jumping to another in one time-step is small. Therefore, even with reduced dimensions, we can clearly distinguish the five clusters in the diffusion space.

At $t = 3$, clusters connect to form a single structure in diffusion space. Paths of length three bring them together. Being better connected, the diffusion distance between clusters decreases.

At this time scale, the one-dimensional parameter of change, the position along the C-shape, is recovered: in the diffusion space the order of colours are preserved along a straight line.

At $t = 10$, a third geometric structure is seen. The five clusters have merged to form a single super-cluster. At this time scale, all points in the observation space are equally well connected. The diffusion distances between points are very small. In the lower dimensional diffusion space it is approximated as zero, which projects the super-cluster to a single point.

This experiment shows how the algorithm uses the connectivity of the data to reveal geometric structures at different scales.

5. Comparison With Other Methods

We investigate the performance of those dimensionality reduction algorithms discussed in Section 2, and compare them to diffusion maps. We use a similar data set as in the previous experiment, the only difference being an increased variance inside clusters. Which of the algorithms detect the position along the C-shape as a parameter of change, i.e. which of the methods best preserve the ordering of the clusters in a one-dimensional feature space? Being linear techniques, we expect PCA and MDS to fail. In theory, isomap should detect the C-shape, although it is known that it struggles with noisy data.

5.1. Discussion

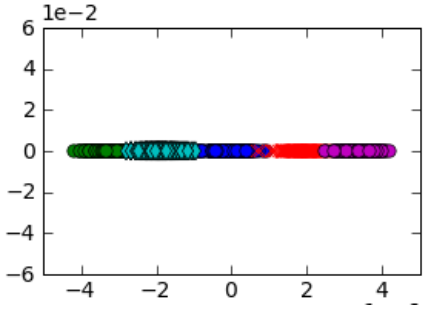

Figure 10: The one-dimensional diffusion space.

5.1.1. PCA

In observation space (Fig. (5)), most variation occurs along the z-axis. The PCA mapping in Fig. 7 preserves this axis as the first principal component. The second principal component runs parallel to $x = y$, and is orthogonal to the z-axis. The variation in the third dimension is orthogonal to these principal components, and preserves the C-shape. Once data is projected onto the two primary axes of variation, the ordering along the non-linear C-shape is lost. As expected, PCA fails to detect a single parameter of change.

5.1.2. MDS

Similar to PCA, MDS preserves the two axes along which the Euclidean distance varies most. Its one-dimensional projection fails to preserve the cluster order, as shown in Fig. 8. Suppose we had fixed the red data points in a one dimensional feature space (see Fig. 11), and wanted to plot the other data points such that Euclidean distances are preserved (this is the premise of MDS). Data points lying on the same radius would then be plotted on top of one another. Due to the non-linear structure of the data, MDS cannot preserve the underlying clusters.

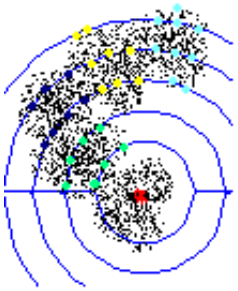

Figure 11: Failure of MDS for non-linear structures

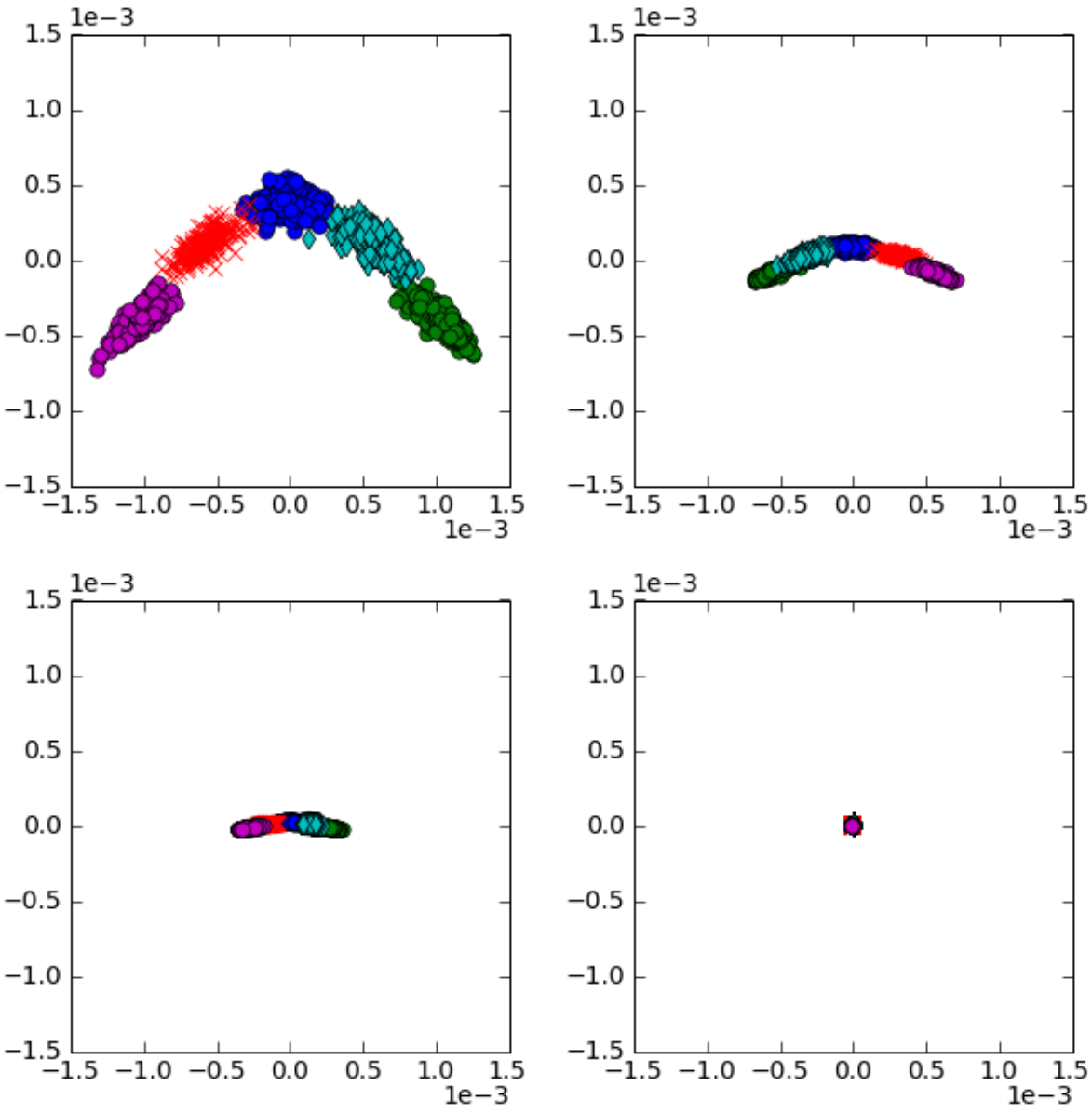

Figure 6: Projection onto diffusion space at times $t = 1$, $t = 2$, $t = 3$ and $t = 10$ (in clock-wise order, starting from the top-left).

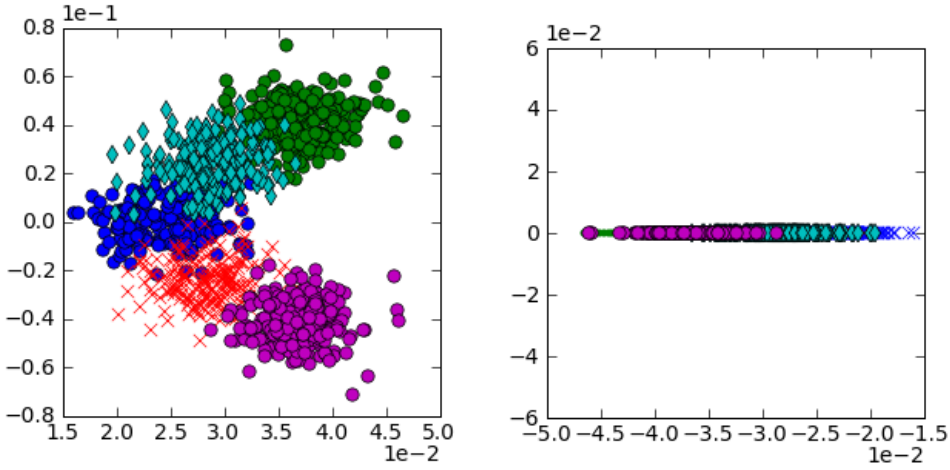

Figure 7: The two- and one-dimensional feature spaces of PCA.

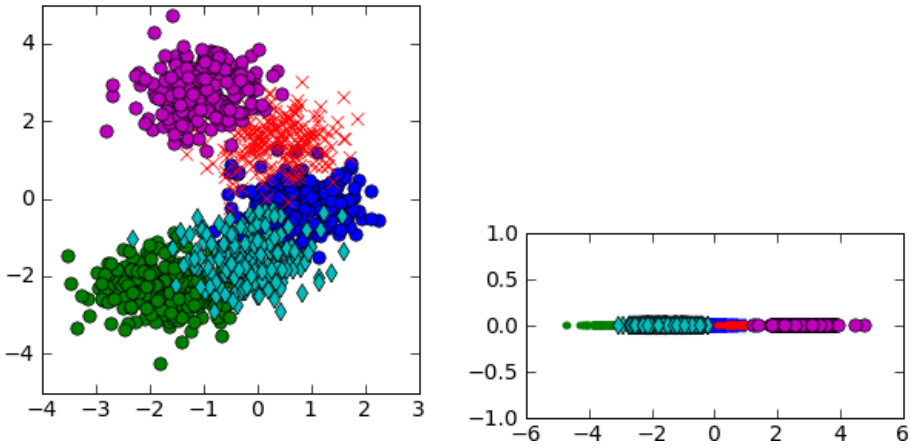

Figure 8: The two- and one-dimensional feature spaces of MDS.

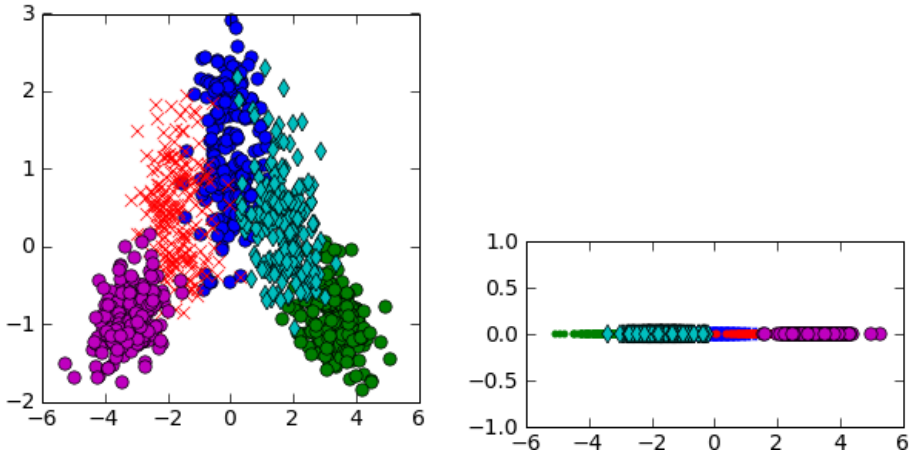

Figure 9: The two- and one-dimensional feature spaces of isomap.

5.1.3. Isomap

When determining geodesic distances, isomap searches a certain neighbourhood. The choice of this neighbourhood is critical: if too large, it allows for "short circuits" whereas, if it is too small, connectivity becomes very sparse. These "short circuits" can skew the geometry of the outcome entirely. For diffusion maps, a similar challenge is faced when choosing kernel parameters.

Isomap is able to recover all the clusters in this experiment, given a very small neighbourhood. It even preserve the clusters in the correct order. Two of the clusters overlap a great deal, due to the algorithm's sensitivity to noise.

5.1.4. Comparison with Diffusion Maps

Fig. 10 shows that a diffusion map is able to preserve the order of clusters in one dimension. As mentioned before, choosing parameter(s) for the diffusion kernel remains difficult. Unlike isomap, however, the result is based on a summation over all data, which lessens sensitivity towards kernel parameters. Diffusion maps are the only one of these techniques that allows geometric analysis at different scales.

6. Demonstration: Organisational Ability

A diffusion map organises data according to its underlying parameters of change. In this experiment, we visualise those parameters. Our data-set consists of randomly rotated versions of a 255×255 image template (see Fig. 12). Each rotated image represents a single, 65025-dimensional data-point. The data is mapped to the diffusion space, and the dimensionality reduced to two. At each 2-dimensional coordinate, the original image is displayed.

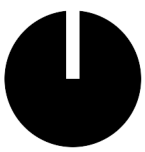

Figure 12: Template: 255 x 255 pixels.

6.1. Discussion

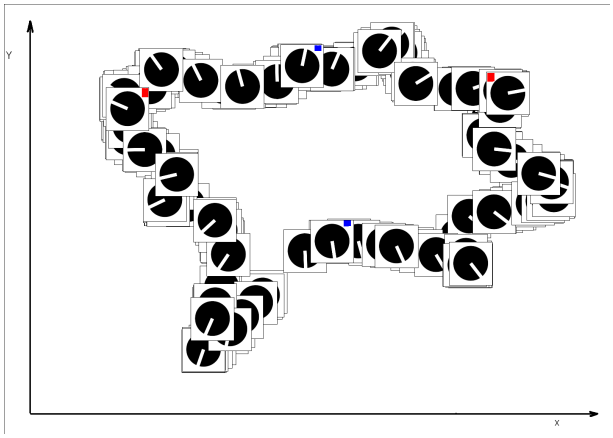

Figure 13: Organisation of images in diffusion space.

In the diffusion space, the images are organised according to their angle of rotation (see Fig. 13). Images with similar rotations lie close to one another.

The dimensionality of the data-set has been reduced from 65025 to only two. As an example, imaging running a K-means clustering algorithm in diffusion space. This will not only be much faster than in data space, but would likely achieve better results, not having to deal with a massive, sparse, high-dimensional data-set.

7. Conclusion

We investigated diffusion maps, a technique for non-linear dimensionality reduction. We showed how it integrates local connectivity to recover parameters of change at different time scales. We compared it to three other techniques, and found that the diffusion mapping is more robust to noise perturbation, and is the only technique that allows geometric analysis at differing scales. We furthermore demonstrated the power of the algorithm by organising images. Future work will revolve around applications in clustering, noise-reduction and feature extraction.

8. Proof

This section discusses the mathematical foundation of diffusion maps. The diffusion distance, given in (7), is very expensive to calculate but, as shown here, it can be written in terms of the eigenvectors and values of the diffusion matrix. These values can be calculated efficiently.

We set up a new *diffusion space*, where the coordinates are scaled components of the eigenvectors of the diffusion matrix. In the diffusion space, Euclidean distances are equivalent to diffusion distances in data space.

Lemma 1: *Suppose K is a symmetric, $n \times n$ kernel matrix such that $K[i,j] = k(i,j)$. A diagonal matrix, D,*

normalises the rows of K to produce a diffusion matrix

$$P = D^{-1}K. \tag{10}$$

Then, the matrix P', defined as

$$P' = D^{1/2}PD^{-1/2}, \tag{11}$$

1. *is symmetric*
2. *has the same eigenvalues as P*
3. *the eigenvectors, x'_k of P' are multiplied by $D^{-1/2}$ and $D^{\frac{1}{2}}$ to get the left ($e^TP = \lambda e^T$) and right eigenvectors ($Pv = \lambda v$) of P respectively.*

Proof: Substitute (10) into (11) to obtain

$$P' = D^{-1/2}KD^{-1/2}. \tag{12}$$

As K is symmetric, P' will also be symmetric.

We can make P the subject of the equation in (11),

$$P = D^{-1/2}P'D^{1/2}. \tag{13}$$

As P' is symmetric, there exists an orthonormal set of eigenvectors of P' such that

$$P' = S\Lambda S^T, \tag{14}$$

where Λ is a diagonal matrix containing the eigenvalues of P' and S is a matrix with the orthonormal eigenvectors of P' as columns.

Substituting (14) into (13),

$$P = D^{-1/2}S\Lambda S^TD^{1/2}.$$

Since S is an orthogonal matrix

$$\begin{aligned} P &= D^{-1/2}S\Lambda S^{-1}D^{1/2} \\ &= (D^{-1/2}S)\Lambda(D^{-1/2}S)^{-1} \qquad (15) \\ &= Q\Lambda Q^{-1}. \qquad (16) \end{aligned}$$

Therefore, the eigenvalues of P' and P are the same. Furthermore, the right eigenvectors of P are the columns of

$$Q = D^{-1/2}S, \tag{17}$$

while the left eigenvectors are the rows of

$$Q^{-1} = S^TD^{\frac{1}{2}}. \tag{18}$$

From (17) we see that the equation for the eigenvectors of P can be given in terms of the eigenvectors x'_k of P'. The right eigenvectors of P are

$$\mathrm{v}_k = D^{-\frac{1}{2}}\mathrm{x}'_k \tag{19}$$

and the left eigenvectors are

$$\mathrm{e}_k = D^{\frac{1}{2}}\mathrm{x}'_k. \tag{20}$$

From (15) we then obtain the eigen decomposition,

$$P = \sum_k \lambda_k \mathrm{v}_k \mathrm{e}_k^T. \tag{21}$$

When we examine this eigen decomposition further, we see something interesting. Eq. 21 expresses each row of the diffusion matrix in terms of a new basis: e_k, the left eigenvectors of the diffusion matrix.

In this new coordinate system in $\mathbb{R}^n$, a row i of P is represented by the point

$$M_i = \begin{bmatrix} \lambda_1 \mathrm{v}_1[i] \\ \lambda_2 \mathrm{v}_2[i] \\ \vdots \\ \lambda_n \mathrm{v}_n[i] \end{bmatrix},$$

where $\mathrm{v}_n[i]$ is the i-th component of the n-th right eigenvector. However, P is not symmetric, and so the coordinate system will not be orthonormal, i.e.

$$\mathrm{e}_k^T \mathrm{e}_k \neq 1$$

or, equivalently,

$$\mathrm{e}_k^T I\ \mathrm{e}_k \neq 1$$

and

$$\mathrm{e}_l^T I\ \mathrm{e}_k \neq 0 \text{ for } l \neq k.$$

This is a result of the scaling applied to the orthogonal eigenvectors of P' in (19) and (20). This scaling can be counter-acted by using a different metric Q, such that

$$\mathrm{e}_k^T Q \mathrm{e}_k = 1$$

and

$$\mathrm{e}_l^T Q\ \mathrm{e}_k = 0 \text{ for } l \neq k$$

where Q must be a positive definite, symmetric matrix. We choose

$$Q = D^{-1},$$

where D is the diagonal normalisation matrix. It satisfies the two requirements for a metric and leads to

$$\begin{aligned} \mathrm{e}_k^T Q\ \mathrm{e}_k &= \mathrm{e}_k^T (D^{-\frac{1}{2}})^T (D^{-\frac{1}{2}})\ \mathrm{e}_k \\ &= \mathrm{x}_k'^T \mathrm{x}'_k \\ &= 1, \end{aligned}$$

using (20). In the same way we can show that

$$\mathrm{e}_l^T D^{-1} \mathrm{e}_k = 0 \text{ for } l \neq k.$$

Therefore the left eigenvectors of the diffusion matrix form an orthonormal coordinate system of $\mathbb{R}^n$, given that the metric is D^{-1}. We define $\mathbb{R}^n$ with metric D^{-1} as the

diffusion space, denoted by $l_2(\mathbb{R}^n, D^{-1})$. For example, the Euclidean distance between two vectors, a and a', is normally

$$d(a,a')_{l_2} = d(a,a')_{l_2(\mathbb{R}^n,I)} = (a-a')^T(a-a')$$

in l_2. In $l_2(\mathbb{R}^n, D^{-1})$, it becomes

$$d(a,a')_{l_2(\mathbb{R}^n,D^{-1})} = (a-a')^T D^{-1}(a-a').$$

Lemma 2: *If we choose our diffusion coordinates as in (9), then the diffusion distance between points in data space (measured using the D^{-1} metric) is equal to the Euclidean distance in the diffusion space.*

***Proof**:* We are required to prove that

$$\begin{aligned} D_t(x_i,x_j)^2 &= \|p_t(x_i,\cdot) - p_t(x_j,\cdot)\|^2_{l^2(\mathbb{R}^n,D^{-1})} && (22)\\ &= \|M_i - M_j\|^2_{l^2(\mathbb{R}^n,I)} && (23)\\ &= \sum_k \lambda_k^{2t}(\mathrm{v}_k[i] - \mathrm{v}_k[j])^2. && (24)\end{aligned}$$

Here, $p_t(x_i,x_j) = P_{ij}$ are the probabilities which form the components of the diffusion matrix. For simplicity we assume t = 1. Then

$$\begin{aligned} D(x_i,x_j)^2 &= \|p(x_i,\cdot) - p(x_j,\cdot)\|^2_{l^2(\mathbb{R}^n,D^{-1})}\\ &= \|P[i,\cdot] - P[j,\cdot]\|^2_{l^2(\mathbb{R}^n,D^{-1})}.\end{aligned}$$

According to the eigen decomposition in (21), this equals

$$\begin{aligned} & \left|\sum_k \lambda_k \mathrm{v}_k[i]\mathrm{e}_k^T - \sum_{k\geq 0}\lambda_k \mathrm{v}_k[j]\mathrm{e}_k^T\right|^2\\ = & \left|\sum_k \lambda_k \mathrm{e}_k^T\left(\mathrm{v}_k[i] - \mathrm{v}_k[j]\right)\right|^2\\ = & \left|\sum_k \lambda_k \mathrm{x}_k'^T D^{\frac{1}{2}}\left(\mathrm{v}_k[i] - \mathrm{v}_k[j]\right)\right|^2\\ = & \left|\sum_k \lambda_k \mathrm{x}_k'^T\left(\mathrm{v}_k[i] - \mathrm{v}_k[j]\right) D^{\frac{1}{2}}\right|^2\end{aligned}$$

In $l_2(\mathbb{R}^n, D^{-1})$, this distance is

$$\begin{aligned} & \left(\sum_k \lambda_k \mathrm{x}_k'^T\left(\mathrm{v}_k[i] - \mathrm{v}_k[j]\right) D^{\frac{1}{2}}\right) D^{-1}\left(\sum_m \lambda_m \mathrm{x}_m'^T\left(\mathrm{v}_m[i] - \mathrm{v}_m[j]\right) D^{\frac{1}{2}}\right)^T\\ = & \left(\sum_k \lambda_k \mathrm{x}_k'^T\left(\mathrm{v}_k[i] - \mathrm{v}_k[j]\right) D^{\frac{1}{2}}\right) D^{-1}\left(D^{\frac{1}{2}}\sum_m \lambda_m \mathrm{x}_m'\left(\mathrm{v}_m[i] - \mathrm{v}_m[j]\right)\right)\\ = & \sum_k \lambda_k \mathrm{x}_k'^T\left(\mathrm{v}_k[i] - \mathrm{v}_k[j]\right)\sum_m \lambda_m \mathrm{x}_m'\left(\mathrm{v}_m[i] - \mathrm{v}_m[j]\right)\end{aligned}$$

Since $\{x_k'\}$ is an orthonormal set,

$$\mathrm{x}_m'^T\mathrm{x}_k' = 0 \qquad \text{for} \quad m \neq k.$$

Therefore

$$D(x_i,x_j)^2 = \sum_k \lambda_k^2\left(\mathrm{v}_k[i] - \mathrm{v}_k[j]\right)^2. \qquad (25)$$

We've therefore shown that the diffusion distance, $D_t(x_i,x_j)^2$, is simply the Euclidean distance between mapped points, M_i and M_j, in diffusion space.

9. References

[1] R.R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[2] A. Ihler. Nonlinear Manifold Learning (MIT 6.454 Summary), 2003.

[3] I.T. Jolliffe. *Principal component analysis*. Springer-Verlag New York, 1986.

[4] S.S. Lafon. *Diffusion Maps and Geometric Harmonics*. PhD thesis, Yale University, 2004.

[5] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.

[6] W.S. Torgerson. Multidimensional scaling: I. Theory and method. *Psychometrika*, 17(4):401–419, 1952.