

# Seminar Report

## Dimensionality Reduction and its Application to Semi-supervised Learning

Nandan Marathe  
nandan@it.iitb.ac.in  
Roll no: 05329R07

Guide: Prof. Sunita Sarawagi  
KReSIT, IIT Bombay

March 27, 2006

### Abstract

The problem of finding a low dimensional structure amongst inputs that have been sampled from high dimensional manifold is known as dimensionality reduction. When viewed from a machine learning perspective, it can be directly compared with feature selection. Another way of looking at dimensionality reduction is as a preprocessing mechanism wherein, the data of high dimensionality can be processed, so as to improve the results that can be attained after applying the standard classification techniques.

Even though classical methods exist for dimensionality reduction, they fail when presented with some nonlinear data. Powerful geometric methods have evolved recently to handle such non-linear input samples. The report provides an overview of the various methods for dimensionality reduction and looks at the advantages and disadvantages of using them. A salient feature of all these methods is that they are based on tractable, polynomial-time optimisations such as shortest path problems, least square fits, semidefinite programming, and matrix diagonalisation.

## 1 Introduction

In statistics, dimensionality reduction is mapping a multidimensional space into a space of fewer dimensions. It finds its application in varied fields like image compression, gene expression microarrays financial time series, etc. It can also be viewed as a platform for feature selection [2] (dimensionality reduction can be viewed as a step in feature selection). The area where feature extraction ends and classification begins is not very clear; ideal feature extractor would simply map the data to its class labels, thus making the task of classification a trivial one. In this report, we deal with various methods dealing with dimensionality reduction. In the end we see how they can be used for the purposes of semi-supervised learning.

We formulate the dimensionality reduction problem as follows: Given a high dimensional data set  $X = (x_1, \dots, x_n)$  of input patterns where  $x_i \in \mathbb{R}^d$ , how can we compute  $n$  corresponding output patterns  $\psi_i \in \mathbb{R}^m$  that provide a faithful low dimensional representation of the original data set with  $m \ll d$ ? The dimensionality reduction mechanism is also expected to find new dimensionality  $m$ , such that the data generated is faithfully represented in  $m$  dimensions. The principal goal in the whole process is to approximate the distances between pairs of input patterns well in lower dimensions.

Dimensionality reduction without loss of information is possible if the data in question fall exactly on a smooth, locally flat subspace; then the reduced dimensions are just coordinates in the subspace. More commonly, data are noisy and there does not exist an exact mapping, so there must be some loss of information. Dimensionality reduction is effective if the loss of information due to mapping to a lower-dimensional space is less than the gain due to simplifying the problem. Dimensionality reduction is not a very new technique. There are various methods of dimensionality reduction. They are classified depending on the way in which the reduction is done. Classical, or linear methods work well in case, the input patterns are confined to a low dimensional subspace. In other cases, the non-linear methods are applied.

The report is organised as follows. In section 2, we review the classical methods of dimensionality reduction viz. principal component analysis (PCA) and metric multidimensional scaling (MDS). These linear methods, though generally work well, fail in case the structure of the data is non-linear. We then describe in section 3, prominent graph-based nonlinear methods that can be used to analyse high dimensional data. All these algorithms, share a similar pattern, constructing a weighted graph, deriving a matrix from this graph and producing an embedding from the top or bottom eigenvectors of this matrix. In section 4 we interpret the non-linear algorithms for dimensionality reduction as kernel methods. We also see an interpretation of Kernel PCA as a form of metric multidimensional scaling. Finally, in section 5 we discuss the similarities and differences of different techniques described in the report.

## 2 Linear Methods

They are the classical methods for dimensionality reduction. They are simple, but lay the foundations for the advanced, more complex methods for nonlinear dimensionality reduction. We will discuss two main methods in this section - the principle component analysis and metric multidimensional scaling.

### 2.1 Principle Component Analysis

It is a linear transformation that chooses a new coordinate system for the data set such that the greatest variance by any projection of the data set comes to lie on the first axis (called the first principal component), the second greatest variance on the second axis, and so on. PCA can be used for reducing dimensionality in a dataset while retaining those characteristics of the dataset that contribute most to its variance, by keeping lower-order principal components and ignoring higher-order ones. In PCA, the input patterns  $x_i \in \mathbb{R}^d$  are projected into the

$m$ -dimensional subspace that minimises the reconstruction error,

$$\epsilon_{PCA} = \sum_i \left\| x_i - \sum_{\alpha=1}^m (x_i \cdot e_\alpha) e_\alpha \right\|^2, \quad (1)$$

where the vectors  $\{e_\alpha\}_{\alpha=1}^m$  define the  $m$  principle components of the given input pattern.

The algorithm of PCA is quite simple. The data is first centered at the origin. Then, a  $d \times d$  covariance matrix is formed, having entries for each dimension of the input space.

$$C = \frac{1}{n} \sum_i x_i \cdot x_i^T, \quad (2)$$

The eigenvectors of this covariance matrix are the principal axes. The eigenvector associated with the largest eigenvalue has the same direction as the first principal component. The eigenvector associated with the second largest eigenvalue determines the direction of the second principal component and so on. The top  $m$  eigenvectors are sufficient to represent the input pattern in  $m$  dimensions. The outputs of PCA are the coordinates of input patterns using the directions specified by the eigenvectors as the principal axis.

## 2.2 Metric multidimensional scaling (MDS)

The main idea behind this method is to preserve the inner products between different input patterns. The outputs of MDS minimise the function

$$\epsilon_{MDS} = \sum_{ij} (x_i \cdot x_j - \psi_i \cdot \psi_j)^2, \quad (3)$$

Let the coordinates of  $n$  points in  $d$  dimensions be denoted by  $x_i, i = 1, \dots, n$ . These can be collected together in a  $n \times d$  matrix  $X$ . The dissimilarities are calculated by  $\delta_{ij} = (x_i - x_j)^T (x_i - x_j)$ . Given these dissimilarities, we construct the matrix  $A$  such that  $a_{ij} = -\frac{1}{2} \delta_{ij}$ , and then set  $B = HAH$ , where  $H$  is the centering matrix  $H = I_n - \frac{1}{n} 11^T$ . This matrix  $B$  is also known as the *Gram matrix*. With  $\delta^2 = (x_i - x_j)^T (x_i - x_j)$ , the construction of  $B$  leads to  $b_{ij} = (x_i - \bar{x})(x_j - \bar{x})$  where  $\bar{x}$  is the mean of the input points. In matrix form, we have  $B = (HX)(HX)^T$ , and  $B$  is real, symmetric and positive semi-definite. Let the eigen-decomposition of  $B$  be  $B = V\Lambda V^T$ , where  $\Lambda$  is a diagonal matrix and  $V$  is a matrix whose columns are eigenvectors of  $B$ . If  $d < n$ , there will be  $n - d$  zero eigenvalues. Then the matrix  $\hat{X}$  of reconstructed coordinates of the points can be obtained as  $\hat{X} = V_d \Lambda_d^{\frac{1}{2}}$  with  $B = \hat{X} \hat{X}^T$ . Refer chapter 2 of [3]

In MDS, we try to approximate the dissimilarity between input patterns  $\delta_{ij}$  such that

$$d_{ij} \approx f(\delta_{ij}) \quad (4)$$

where  $f$  is a specified analytic function. Sometimes the values of these  $\delta$  parameters can also be learnt by obtaining derivative of the error function with respect to the coordinates of the points that define the  $d_{ij}$ 's and use a gradient-based methods to minimise the error. Note that if  $f(\delta_{ij})$  has some adjustable

parameters  $\theta$  and is linear with respect to  $\theta^2$ , then the function  $f$  can also be adapted and the optimal values for the parameters given the current  $d_{ij}$ 's can be obtained by (weighted) least-squared regression.

Though based on a somewhat different geometric intuition, metric MDS gives same outputs as that of PCA; basically a rotation of the inputs followed by a projection into the subspace with the highest variance. The gram matrix of MDS has the same rank and eigenvalues up to a constant factor as the covariance matrix in PCA. What is more interesting, though is the fact that useful nonlinear generalisations of metric MDS are obtained by substituting generalised pairwise distances and inner products in place of Euclidean measurements.

### 3 Graph-based Methods

- PCA and MDS generate faithful output when the high dimensional input patterns are confined to a low dimensional subspace.
- If input patterns are distributed throughout the subspace, the eigenvalue spectra reveal the data set's high dimensionality - in other words the number of underlying modes of variability.
- In such cases, the structure of data will be highly non-linear, and hence linear methods are bound to fail.

Graph-based techniques come in as a powerful tool for analysing data that has been sampled from a low dimensional submanifold. All the graph based algorithms follow a similar kind of pattern [1]. They begin by constructing a sparse graph in which the nodes denote the input patterns and the edges represent neighbourhood relations. The resulting graph can then be viewed as a discretised approximation of the submanifold sampled by the input pattern. These graphs can then be used to construct matrices whose spectral decomposition will reveal the low dimensional structure of the manifold. Though capable of revealing the highly non-linear structure, graph-based methods are based on highly tractable (polynomial time) algorithms such as shortest path problems, least square fits, semidefinite programming, and matrix diagonalisation. We discuss four prominent methods for non-linear manifold learning in the following section.

#### 3.1 Isomap [Tanenbaum et al.,2000]

- Isomap is a method that looks to preserve pairwise distances between input patterns, as measured along the submanifold from which they are sampled.
- It can be viewed as a variant of MDS algorithm, where the Euclidean distances are substituted by the estimates of geodesic distances along the submanifold.
- The algorithm has three major steps. The first step is to compute the k-nearest neighbours of each input pattern and to construct a graph whose vertices represent input patterns and whose (undirected) edges connect the k-nearest neighbours.

- Edges are assigned weights proportional to the Euclidean distance between the neighbours.
- Compute all the pairwise distances  $\Delta_{ij}$  between all nodes (i,j) along shortest paths through the graph. For this the Dijkstra's algorithm is used to find shortest path which is done in  $O(n^2 \log n + n^2 k)$ .
- These pairwise distances are as input to MDS yielding low dimensional outputs  $\psi_i \in \mathbb{R}^m$  for which  $\|\psi_i - \psi_j\|^2 \approx \Delta_{ij}^2$

The key assumption made by isomap [7] is that the quantity of interest, while comparing two points, is the distance along the curve between the two points; if that distance is too large, its to be taken even if in fact the two points are close in  $\mathbb{R}^d$ . refer figure 1 below.

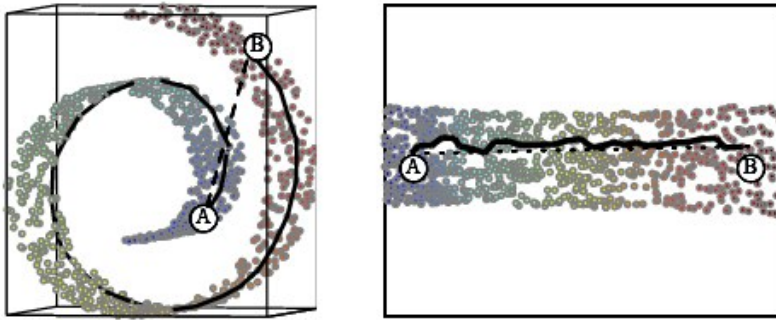


Figure 1: On the left are two points A and B whose euclidean distance is quite small compared to the geodesic distance which spans over the whole spiral. The straight line describes the euclidean distance, whereas  $k = 12$  nearest neighbours are required to approximate the distance between them over the geodesic.

Isomap shares with the other manifold learning techniques the property that it does not provide a direct functional form for mapping  $\mathbb{R}^d \rightarrow \mathbb{R}^m$  that can be simply be applied to new data, so sometimes the computational complexity is an issue for the algorithm. But when it succeeds, Isomap yields a low dimensional representation in which the Euclidean distances between outputs match the geodesic distances between the input patterns on the submanifold from which they are sampled. There are formal guarantees of convergence when the input pattern is sampled from a submanifold that is isometric to a convex subset of Euclidean space.

### 3.2 Maximum variance unfolding [Weinberger and Saul, 2005]

- The method is based on preserving the distances and angles between nearby input patterns.
- Similar to Isomap, it appeals to the notion of isometry to get the low dimensional representation, but unlike it, maximum variance unfolding does not involve "estimation" of geodesic distances.

- The algorithm tries to "unfold" a dataset by pulling the input patterns as far apart as possible subject to distance constraints that ensure that the final transformation from input patterns to outputs looks locally like a rotation plus translation.

The first step of the algorithm [11] is to compute the k-nearest neighbours of each input pattern. Then, a neighbourhood indicator matrix is defined as  $\eta_{ij} = 1$  if and only if the input patterns  $x_i$  and  $x_j$  are k-nearest neighbours or if there exists another input pattern of which both are k-nearest neighbours; otherwise  $\eta_{ij} = 0$ . The constraints to preserve distances and angles between k-nearest neighbours are written as:

$$\|\psi_i - \psi_j\|^2 = \|x_i - x_j\|^2, \quad (5)$$

for all  $(i, j)$  such that  $\eta_{ij} = 1$ . Also, to eliminate the translational degree of freedom, the outputs are constrained to be centered at origin. The algorithm tries to "unfold" the input patterns by maximising the variance of the outputs as ,

$$\text{var}(\psi) = \sum_i \|\psi_i\|^2, \quad (6)$$

while preserving the local distances and angles as in equation (5). Figure 2 shows the relation between maximising the variance and reducing dimensionality. The

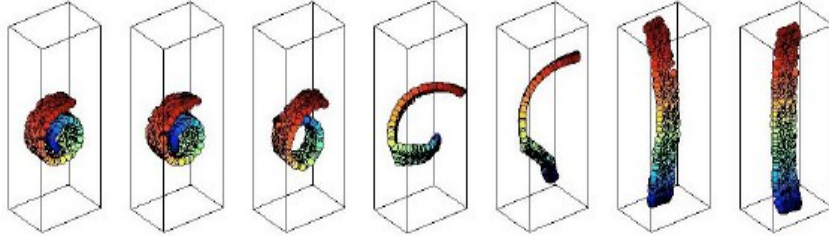


Figure 2: Input patterns sampled from a swiss roll, "unfolded" by the maximum variance method. The intermediate positions are feasible, but non optimal solutions to the semidefinite program. Refer LSaul for further details.

constraints in equations 5, and 6 can be posed as a semidefinite program (Its a linear program with additional constraint that a matrix whose elements are linear in the optimisation variables must be positive semidefinite).

The formulation of the program is done as follows. Let,  $K_{ij} = \psi_i \cdot \psi_j$  denote the Gram matrix of the products. The equations 5, 6 and the centered assumption can be stated as :

**Maximise trace ( $K$ ) subject to:**

- 1)  $K \succeq 0$ .
- 2)  $\sum_{ij} K_{ij} = 0$ .
- 3)  $K_{ij} - 2K_{ij} + K_{jj} = \|x_i - x_j\|^2$  for all  $(i, j)$  such that  $\eta_{ij} = 1$ .

After this optimisation, we obtain the gram matrix  $K$  of the "unfolded" data. This can be considered equivalent to the new distance representations, (or inner products) of the given input patterns.

### 3.3 Locally Linear Embedding [Roweis and Saul, 2000]

Locally linear embedding (LLE) models the manifold by treating it as a union of linear patches, in analogy to using coordinate charts to parameterise a manifold in different geometry. Suppose that each point  $x_i \in \mathbb{R}^d$  has a number of close neighbours indexed by the set  $N(i)$ , and let  $y_i \in \mathbb{R}^d$  be the low dimensional representation of  $x_i$ . The idea is to express each  $x_i$  as a linear combination of its neighbours, and then construct the  $y_i$  so that they can be expressed as the same linear combination of their corresponding neighbours (The later is also indexed by  $N(i)$ ).

The algorithm has three steps. The first step, is to compute the  $k$ -nearest neighbours of each high dimensional input pattern  $x_i$ . However, in LLE we construct a *directed* graph (rather than an undirected graph) whose edges indicate nearest neighbour relations (which may or may not be symmetric). The second step is to attach weights  $W_{ij}$  to each edge. Each node and its  $k$ -nearest neighbours are viewed as a small linear "patch" on the low dimensional submanifold. The weights are chosen in such a way so as to minimise the reconstruction error:

$$\epsilon_W = \sum_i \|x_i - \sum_j W_{ij} x_j\|^2, \quad (7)$$

The whole minimisation is performed subject to two constraints. (i)  $W_{ij} = 0$  if  $x_j$  is not among the  $k$ -nearest neighbours of  $x_i$ ; (ii)  $\sum_j W_{ij} = 1$  for all  $i$ . The weights  $W_{ij}$  thus constitute a sparse matrix  $W$  that encodes local geometric properties of the data set by specifying the relation of each input pattern  $x_i$  to its  $k$ -nearest neighbours.

Thus the final objective function that we wish to minimise is

$$F \equiv \sum_i \left\{ \frac{1}{2} \|x_i - \sum_{j \in N(i)} W_{ij} x_j\|^2 - \lambda_i \left( \sum_{j \in N(i)} W_{ij} - 1 \right) \right\} \quad (8)$$

where the constraints are enforced with Lagrange multipliers  $\lambda_i$  (Burges 2004). Since the sum splits into independent terms, we can minimise each  $F_i$  independently. Thus fixing  $i$  and letting  $x \equiv x_i$ ,  $v \in \mathbb{R}^n$ ,  $v_j \equiv W_{ij}$ , and  $\lambda \equiv \lambda_i$ , and introducing the matrix  $C \in S_n$ ,  $C_{jk} \equiv x_j \cdot x_k$ ,  $j, k \in N(i)$ , and the vector  $b \in \mathbb{R}^n$ ,  $b_j \equiv x \cdot x_j$ ,  $j \in N(i)$ , then requiring that the derivative of  $F_i$  with respect to  $v_j$  vanishes gives  $v = C^{-1}(\lambda e + b)$ . Imposing the constraint  $e'v = 1$  then gives  $\lambda = (1 - e' C^{-1} b) / (e' C^{-1} e)$ . Thus  $W$  can be found by applying this for each  $i$ .

Given the  $W$ 's the second step is to find a set of  $y_i \in \mathbb{R}^d$  that can be expressed in terms of each other in the same manner. We then get similar error function to minimise, but here an extra constraint is seen which is that the  $y$ 's should have unit covariance. The derivative of the minimisation function comes out to be (Burges 2004)

$$(I - W)^T (I - W) Y = \frac{1}{m} Y \Lambda, \quad (9)$$

Choosing the smallest eigenvectors guarantees the  $\psi$  vectors are zero mean and are orthogonal to each other. The  $d$ -dimensional embedding is thus obtained by computing the bottom  $m + 1$  eigenvectors of the matrix  $(I - W)^T(I - W)$ .

There are two distinct things about LLE algorithm which make it very unlike the algorithms discussed previously, firstly, the *bottom* eigenvectors are chosen instead of the top  $m$ . Secondly, there is not a distinct gap between the bottom  $m+1$  and the remaining eigenvalues. As a result of that, the algorithm does not indicate the modes of variability of the input patterns, but instead needs  $m$  as a input. The figure below shows the application of LLE on the swiss roll.

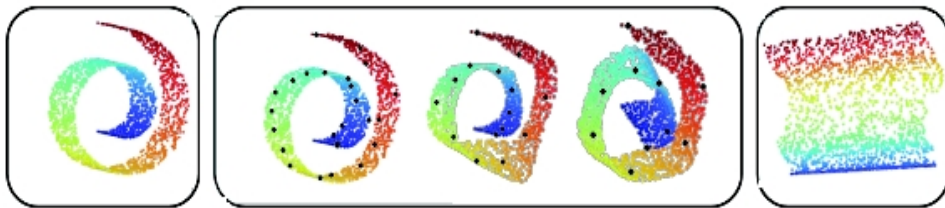


Figure 3: Leftmost panel shows the input patterns of a swiss roll, whereas the rightmost one shows the two dimensional representation as recovered by LLE. The middle panels show  $l < n$  randomly chosen outputs constrained to be equal to their corresponding inputs. Note that in these middle panels, the outputs have the same dimensionality as the inputs. Thus, the goal of the optimisation in the middle panels is not dimensionality reduction; rather, it is locally linear reconstruction of the entire data set from a small sub-sample.

### 3.4 Laplacian Eigenmaps[Belkin and Niyogi, 2003]

This method finds its connections to spectral graph theory. Introduced by Belkin and Niyogi, it requires us to find the laplacian of a graph and then apply a svm-type operator on it. The Laplacian matrix for any weighted, undirected graph is defined by  $\mathbb{L} \equiv D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$ , where  $L_{ij} \equiv D_{ij} - W_{ij}$  and where  $D_{ij} \equiv \delta_{ij}(\sum_k W_{ik})$ . It can be proved that  $\mathbb{L}$  is positive definite. Let  $G$  be a graph and  $m$  be its number of nodes. For  $W_{ij} \in [0, 1]$ , the spectrum of  $G$  (defined as the set of eigenvalues of its Laplacian) characterises its global properties. For a detailed description and proofs regarding laplacian and spectral graph theory, refer Chung, 1997.

The laplacian eigenmaps algorithm uses  $W_{ij} = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$ . Let  $D$  denote the diagonal matrix with elements  $D_{ii} = \sum_j W_{ij}$ . We would like to find outputs  $\psi$  that minimise the cost function ,

$$\epsilon_L = \sum_{ij} \frac{W_{ij} \|\psi_i - \psi_j\|^2}{\sqrt{D_{ii}D_{jj}}}, \quad (10)$$

This minimisation results in simple eigenvalue problem  $Ly = \lambda Dy$  (Belkin and Niyogi, 2003) Thus, the minimum of equation 10 is computed from the bottom  $(m+1)$  eigenvectors of the matrix  $\mathbb{L} = I - D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$ . The matrix  $\mathbb{L}$  is the symmetrized normalised form of the graph Laplacian, given by  $D - W$ . As in

LLE, the bottom eigenvector is discarded, and the remaining eigenvectors (each of size  $n$ ) yield low dimensional outputs  $\psi_i \in \mathbb{R}^m$ .

## 4 Kernel based algorithms

PCA is a linear method, in the sense that the reduced dimension representation is generated by linear projections (although the eigenvectors and eigenvalues depend non-linearly on the data). Several versions of nonlinear PCA have been proposed in the hope of overcoming this problem. In this section we describe a more recent algorithm called kernel PCA (Scholkopf et al. 1998).

### 4.1 Kernel PCA

The kernel PCA is posed as follows. Suppose we are given a real-valued function  $k: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$  with the property that there exists a map  $\Phi: \mathbb{R}^d \rightarrow H$  into a dot product feature space  $H$  such that for all  $x, x' \in \mathbb{R}^d$  we have  $\Phi(x) \cdot \Phi(x') = k(x, x')$ . The kernel function can be viewed as a similarity measure.

Its not immediately obvious that PCA can be applied to kernels, since in PCA the data appears in expectations over products of individual components of vectors, not over dot products between the vectors. However (Scholkopf et al.) show how the problem can indeed be cast entirely in terms of dot products. They make two key observations: first, that the eigenvectors of the covariance matrix in  $\mathbb{F}$  lie in the span of the (centered) mapped data, and second, that therefore no information in the eigenvalue equation is lost if the equation is replaced by  $m$  equations, formed by taking the dot product of each side of the eigenvalue equation with each (centered) mapped data point. The covariance matrix of the mapped data in feature space is

$$C \equiv \frac{1}{m} \sum_{i=1}^m m(\Phi_i - \mu)(\Phi_i - \mu)^T \quad (11)$$

where  $\Phi_i \equiv \Phi(x_i)$  and  $\mu \equiv \frac{1}{m} \sum_i \Phi_i$ . We are looking for solutions  $v$  of

$$Cv = \lambda v \quad (12)$$

Since this can be written  $\frac{1}{m} \sum_{i=1}^m (\Phi_i - \mu)[(\Phi_i - \mu) \cdot v] = \lambda v$ , the eigenvectors  $v$  lie in the span of the  $\Phi_i - \mu$ 's or

$$v = \sum_i \alpha_i (\Phi_i - \mu) \quad (13)$$

for some  $\alpha_i$ . Hence, we can write,

$$(\Phi_i - \mu)^T Cv = \lambda (\Phi_i - \mu)^T v \quad (14)$$

Now, consider the kernel matrix  $K_{ij}$ , the matrix of dot products in  $\mathbb{F}$ :  $K_{ij} \equiv \Phi_i \cdot \Phi_j$ ,  $i, j = 1, \dots, m$ . Using this we create a centered kernel matrix  $K_{ij}^C \equiv (\Phi_i - \mu) \cdot (\Phi_j - \mu)$ . Thus we have a kernel equation such that if there exists a solution to that equation, we have solution for equation 14.

Further simplification, leads to the following procedure for extracting the  $i$ th principle component in  $\mathbb{F}$  using kernel PCA is therefore:

1. Compute the  $i$ 'th principle eigenvector of  $K^C$  with eigenvalue  $\bar{\lambda}$ .
2. Normalise the corresponding eigenvector,  $\alpha$  to have length equal to  $\frac{1}{\sqrt{\bar{\lambda}}}$ .
3. For training point  $x_k$ , the principle component is then just  $(\Phi(x_k) - \mu).v = \bar{\lambda}\alpha_k$
4. For a general test point  $x$ , the principle component is  $(\Phi(x) - \mu).v = \sum_i \alpha_i k(x, x_i) - \frac{1}{m} \sum_{ij} \alpha_i k(x, x_j) - \frac{1}{m} \sum_{ij} \alpha_i k(x_i, x_j) + \frac{1}{m^2} \sum_{i,j,n} \alpha_i k(x_j, x_n)$  where the last two terms can be dropped since they don't depend on  $x$ .

Kernel PCA may be viewed as yet another method for something like feature selection, rather than directly applying it for classification or regression. Kernel PCA followed by linear SVM on a pattern recognition problem has shown to give similar results to using nonlinear SVM. Classical PCA has a significant advantage that it depends only on the first and second moments of the data, but Kernel PCA doesn't. One can define  $k(i, j) = (x_i \cdot x_j + b)^p$  thus containing powers up to order  $2p$ , which is a particularly desirable feature. Kernel PCA has computational limitations of having to compute eigenvectors for square matrices of size  $m$ , but this problem can be reduced by using a subset of the training data.

Kernel PCA is often used for nonlinear dimensionality reduction with polynomial or Gaussian kernels. It is important to realize, however, that these generic kernels are not particularly well suited to manifold learning. This can be seen from the diagram below.

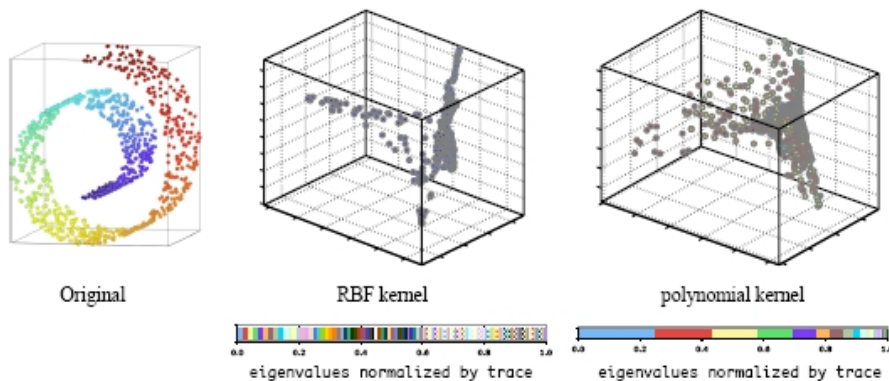


Figure 4: Results of Gaussian and polynomial kernels when applied to input patterns sampled from a swiss roll. The kernels do not lead to a good low dimensional approximation of the Swiss roll.

## 4.2 Graph based kernels

All the algorithms discussed in section 3 can be viewed as instances of kernel PCA, that are derived from sparse weighted graphs rather than from predefined kernel function. Ham et al defines how most of these techniques can be formally converted into kernel functions.

The construction of a kernel matrix is equivalent to mapping data points  $p_1, p_2, \dots, p_m$  in a Hilbert space so that  $K_{ij} = \langle p_i, p_j \rangle$  is a positive definite kernel. For isomap, the kernel is related to commute times; for LLE, the kernel can be associated to specially constructed graph operator. Note that kernel matrix in all these algorithms is defined on training data. Moreover, in contrast to traditional kernels, such as Gaussian kernel, the element  $K_{ij}$  in the kernel matrix not only depends on inputs  $x_i$  and  $x_j$ , but also on other training points. This can be seen in figures where the induced feature distance defined by kernels does not depend simply on the distance in the input space. However for small distances, there appears to be more of a direct relationship indicating the role of local structure in constructing the kernel. For all the three algorithms, the existence of a kernel formulation indicates that the algorithms may be viewed as warping of the input space into a feature space where the manifold is flat.

The LLE and Laplacian eigenmaps algorithms though do not explicitly construct a gram matrix, but matrices that they diagonalize can be related to operators on graphs and interpreted as "inverse" kernel matrices. The above analysis provides some insight into the differences between Isomap, maximum variance unfolding, laplacian eigenmaps and LLE. The metrics introduced Isomap and maximum variance unfolding are related to geodesic and local distances, respectively, on the submanifold on which the input patterns are sampled. On the other hand, the metric induced by the graph laplacian is related to the commute times of Markov Chains; these times involve all connecting paths between two nodes of a graph, not just the shortest one. In many applications, the kernel matrices in isomap and maximum variance unfolding have a distinct gap in their eigen value spectra that reflects the dimensionality of the underlying submanifold from which the data has been sampled. On the other hand, those from the laplacian eigenmaps and LLE do not reflect the geometry of the submanifold in such a manner.

## 5 Conclusions

Here we discuss the advantages and disadvantages of the various methods discussed in the previous sections. What is also evident from the description is that the underlying mathematics; the same Lagrange multiplier trick is used to enforce the constraints. Isomap, LLE and Laplacian eigenmaps share the property that in their original forms, they do not provide any direct functional form for dimension reducing mapping, so their extension to new data requires retraining. Isomap is often called a "global" dimensionality reducing algorithm, because it attempts to preserve all geodesic distances; by contrast LLE, and laplacian eigenmaps are local (for example LLE attempts to preserve local translations, rotations and scaling of data).

The asymptotic convergence of maximum variance unfolding has not been studied in a formal setting. Unlike Isomap, however, the solutions from maximum variance unfolding are guaranteed to preserve distances between nearest neighbors for any finite set of  $n$  input patterns. Maximum variance unfolding also behaves differently than Isomap on data sets whose underlying submanifold is isometric to a connected but not convex subset of Euclidean space. Of the algorithms described in section 3, LLE and Laplacian eigenmaps scale best to moderately large data sets ( $n < 10000$ ), provided that one uses special purpose

eigensolvers that are optimized for sparse matrices. The internal iterations of these eigensolvers rely mainly on matrix-vector multiplications which can be done in  $O(n)$ . The computation time in Isomap tends to be dominated by the calculation of shortest paths. The most computationally intensive algorithm is maximum variance unfolding, due to the expense of solving semidefinite programs over  $n \times n$  matrices. For significantly larger data sets, all of the above algorithms present serious challenges: the bottom eigenvalues of LLE and Laplacian eigenmaps can be tightly spaced, making it difficult to resolve the bottom eigenvectors, and the computational bottlenecks of Isomap and maximum variance unfolding tend to be prohibitive.

## References

- [1] L. K. Saul, K. Q. Weinberger, et al. Spectral Methods for Dimensionality Reduction
- [2] C. J. C. Burges. Geometric methods for feature extraction and dimensional reduction. *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*. Kluwer Academic Publishers, 2005.
- [3] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman and Hall, London, 1994.
- [4] J. Ham, D. D. Lee, S. Mika, and B. Scholkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, Banff, Canada, 2004.
- [5] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 2000.
- [6] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 2003.
- [7] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000.
- [8] C. K. I. Williams. On a connection between kernel PCA and metric multidimensional scaling. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*. MIT Press.
- [9] B. Schölkopf, A. J. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 1998.
- [10] F.R.K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

- [11] K. Q. Weinberger, B. D. Packer, and L. K. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In Proceedings of the Tenth International Workshop on AI and Statistics, 2005.