Proceedings of the Fields–MITACS Industrial Problems Workshop, 2006

# Nonlinear Dimension Reduction for Microarray Data (Small $n$ and Large $p$)

**Problem Presenter:** Christopher Bowman (National Research Council of Canada)

**Academic Participants:** Dhavide Aruliah (University of Ontario Institute of Technology), Guangzhe Fan (University of Waterloo), Roderick Melnik (Wilfred Laurier University), Suzanne Shontz (Pennsylvania State University), Steven Wang (York University), Jiaping Zhu (University of Waterloo)

**Report prepared by:** Suzanne Shontz[1]

## 1 Introduction

Over the last decade or so, researchers have developed techniques for measuring the expression level of many genes in an organism simultaneously. One such technique is the cDNA microarray [13, 11]. Such techniques generate a torrent of data that can be used to then learn more about gene functions, response to stimuli, and interactions.

A cDNA microarray is a glass slide on which many (usually thousands) segments of DNA (often genes, but not always) are attached in distinct spots. Messenger RNA is then extracted from two different populations of cells (for example, cancer and normal tissue) and reverse transcribed to complementary DNA (cDNA). Each of the two sets of cDNA is tagged with a molecule of fluorescent dye; usually they are red and green, respectively. The cDNA solutions are then washed over the glass slide and hybridized with the genetic material spotted onto the slide. When a molecule of cDNA matches the DNA spotted onto the slide, it reacts and binds to it, bringing along the fluorescent dye molecule. The greater the number of copies of the appropriate piece of cDNA present in the sample, the greater the number of dye molecules which will bind to that particular spot, creating a stronger signal. If red and green dyes are used, the spots will appear to fluoresce with varying intensities of red, green, and yellow (when cDNA from both samples bind to the spot). These intensities can be measured by a scanner to determine the relative expression level of each gene in each of the two cell populations.

Microarray experiments thus typically have thousands of variables explaining each individual sample in the experiment and typically only a handful (a few hundred at most, often

[1]`shontz@cse.psu.edu`

many fewer) distinct samples. Furthermore, the thousands of genes in an organism are not independent entities, and each reacts to the activation level of other genes in a complicated and not well-understood way. This raises a mathematical challenge. Each experimental sample can be viewed as a point in a space of dimension equal to the number of genes being measured. The question is, can one find a lower-dimensional space in which to work? Or, stated more precisely, given a set of $n$ points in an $p$-dimensional linear space, drawn from some unknown distribution $V$, find a $d$-dimensional (possibly nonlinear, $d \ll p$) manifold that approximates the points well?

There are many measures for determining whether or not a lower-dimensional manifold approximates the points well. The simplest is that the orthogonal distance from each point to the manifold should be minimised, but this is by no means the only choice, and other options will be discussed below. It is important to note, however, that any notion of goodness-of-fit should apply not just to the data that has already been collected, but also to future data points drawn from the distribution $V$. To ensure this, a cross-validated estimate of error must be used, for example, the leave-one-out error described below.

## 2 Filtering the data

**2.1 Motivation for filtering.** Typical microarray data have quite high dimensionality due to the number of genes involved. For example, the simplest biological model, yeast, has more than 6000 genes. In 2003, estimates from gene-prediction programs suggested there might be as many as 24,500 protein-coding genes [12]. The Ensemble genome-annotation system estimates their number at 23,299. Therefore, the dimensionality is very high. On the other hand, the number of observations that is available is usually very low due to fact the microarray experiments are too expensive to produce many replications. This is known as the problem of "*large p and small n*".

Although there are many human genes, often medical researchers are only concerned with a dozen or fewer genes if they are interested in one particular disease. Therefore it is not necessary to consider all the genes in the analysis of microarray data. Furthermore, the information or "signal" for the genes of interest could be overwhelmed by the genes that are not relevant to the current analysis. Dimensionality reduction is necessary given the fact that there are not many observations that are scattered in very high-dimensional space.

Furthermore, the filtering is crucial for any data mining technique to work. For example, the clustering procedure is often applied to microarray data to divide the large-dimensional space into subspaces such that the subspaces are much more manageable than the whole space. However, most clustering algorithms would rely on a proper choice of distance function. There are many distance functions proposed in the literature. We will demonstrate our argument by using the most commonly used distance function, *i.e.*, Euclidean distance. To make our argument more transparent, we suppose that there are $p$ random variables that are independent and identically distributed with a standard normal distribution, *i.e.*

$$X_1, X_2, \cdots, X_p \sim N(0,1).$$

Let us further assume that only $Y_1$ and $Y_2$ are important or relevant to us. Furthermore, we assume that

$$Y_1 \sim N(10,1) \quad and \quad Y_2 \sim N(20,1)$$

for the observations from the treatment group and $Y_1, Y_2 \sim N(0,1)$ for the control group.

However, the vector $(Y_1, Y_2, X_1, X_2, \cdots, X_p)$ will not be informative if the Euclidean distance is used. Let $O_k = (Y_{k1}, Y_{k2}, X_{k1}, X_{k2}, \cdots, X_{kp})$ and $O_j = (Y_{j1}, Y_{j2}, X_{j1}, X_{j2}, \cdots, X_{jp})$

represent two observations in a microarray data set. Note that $Y_{k1} \sim N(10,1) \quad and \quad Y_{k2} \sim N(20,1)$.

It can be verified that

$$dist(O_j - O_k)^2 = \sqrt{(Y_{k1} - Y_{j1})^2 + (Y_{k2} - Y_{j2})^2} + \chi^2(p)$$

where $\chi^2(p)$ denotes the central $\chi^2$ distribution with mean $p$ and variance $2p$.

It can be verified that

$$\frac{(Y_{k1} - Y_{j1})^2 + (Y_{k2} - Y_{j2})^2 + \chi^2(p)}{\chi^2(p)} \xrightarrow{P} 1, \quad as \;\; p \to \infty. \tag{2.1}$$

This implies that the Euclidean distance function will be dominated by those variables that are pure noise in general. Although another distance function might be a better choice, they all suffer from the same problem but to a lesser degree.

Therefore, filtering is very important to quickly reduce the genes of interest and avoid the aforementioned problems introduced by those noisy variables which contain no information at all.

**2.2 Example of preprocessing.** Various methods of preprocessing have been proposed in the literature. These methods mainly deal with problems having class labels. For example, Golub et al. [3] proposed a univariate ranking criterion for each gene in a two-class situation. The criterion is can be defined as the ratio of the absolute mean difference of gene expression levels of the two classes with respect to the sum of standard deviations of the two classes for each gene considered. Higher ratio indicate higher ranking of the gene. Some other author used the ratio of between-class sum of squares to within-class sum of squares of each gene for multi-class problems. Later, Tibshirani et al. [14] used the shrunken centroids method for gene classification. Shrinkage and gene selection are integrated into a naive Bayes classifier. The univariate gene selection is based on t-statistics for each gene of each class.

Another method is the random forest procedure [1]. Random forest is an ensemble tree approach in data mining. It has been used as a popular approach in gene selection. Basically, random forest build tree classifiers. A tree classifier recursively partition the data to classify. The tree is built by greedily searching locally optimal split rules recursively. Instead of using all variables (genes) to build the trees, random forest uses a random sample of variables (genes) during the tree construction. The randomness causes different trees built even following the same procedure each time. In this sense, we can get many (usually more than 50) different trees using the same data set. These trees are used as an ensemble classifier via voting. Random forest can have high accuracy in classification. It also effectively estimates the performance of these randomly selected variables (genes) and provides an overall measure of variable (gene) importance. Different ways of evaluating the variables (genes) can be found in the random forest manual. These importance measures will consider interactions among the variables (genes) due to the nature of the tree classifiers.

As an example, let us look at the famous Leukemia data. This data set has 38 training samples and 34 test samples of two types of acute leukemias, acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL) [3]. Each sample is related to 7129 genes.

Due to the large number of genes, we propose a combination of univariate ranking and random forest. First, univariate ranking is performed to select the best 200 genes. Then the random forest procedure is performed on these 200 genes to obtain their measures of importance. Below is a figure showing the importance measure of the 200 pre-selected genes.

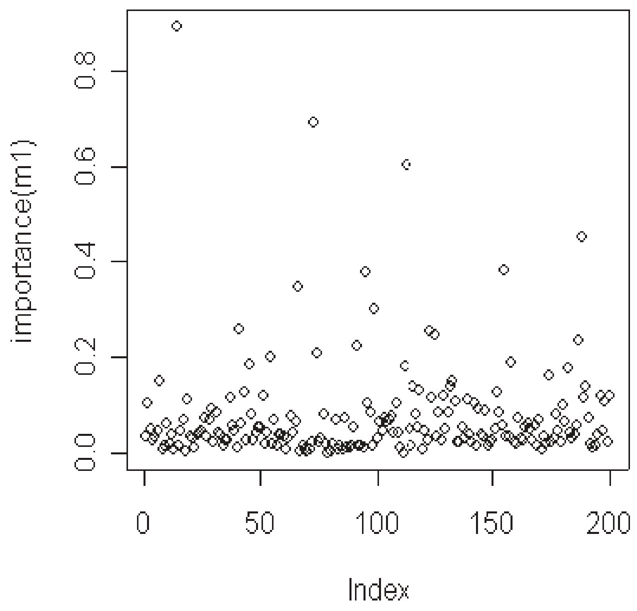

**Figure 1** Variable (Gene) Importance using Random Forest for 200 Pre-selected Genes

We see that the variable importance using random forest is not the same as that using univariate ranking when interactions are considered. We can select a number of genes based on the variable importance for our needs.

**2.3 Future avenues.** As seen in the previous example, univariate ranking methods implicitly assume weak or no interactions or correlations among genes, which may not be true in practice of microarray analysis. The ideal situation would be to have a method which considers all variables together for selection. We now introduce the support vector machine classifier. The support vector machine basically searches for the optimal hyperplane that separates the data. Here an optimal hyperplane is the one that maximizes the geometric margin (the closest distance of the observations to the hyperplane). For a nonlinear pattern, the original space can be mapped to a high-dimensional space using kernel functions and the so-called Reproducing Kernel Hilbert Space (RKHS). So support vector machines can learn very well for a general problem.

In particular, for gene expression data, Guyon et al. [4] proposed a recursive feature eliminating method for gene selection. First, all the variables (genes) are used in model fitting. Then a large number of variables (genes) which are not significant in the model are removed. The model is then rebuilt on the rest of the variables (genes), and we can recursively repeat the procedure until only a small number of genes is left in the model. Zhu and Hastie used penalized logisitic regression with similar ideas of gene selection [16].

For example, in the Golub 1999 data set just described in the previous section, Zhu and Hastie report 26 genes selected with two cross-validated errors on the training set and one error on the test set. The support vector machine selects 31 genes with similar performances.

In the future, we could try to combine random forest with support vector machine to perform gene selection.

## 3 Nonlinear dimension reduction

While filtering the data requires the methodologies to be able to identify in some sense redundant information in the already existing data, the ultimate purpose of dimensionality reduction is to reduce the data to a low-dimensional manifold in such a way that unsupervised learning is possible on new data. In other words, we have to discover the main trend in the existing data in order to be able to deal with newly acquired data in a similar way. Along with the (probabilistic) density estimation techniques, dimensionality reduction is a key methodology in developing algorithms for unsupervised learning [10]. Although these two methodologies can be applied simultaneously, nonlinear reduction can also be developed in a non-probabilistic framework.

Historically, first developed techniques for dimensionality reduction were based on linear versions of principal component analysis (PCA) (described below) and other eigenvalue-based methods such as variants of centre manifold reduction techniques. Due to difficulties with the mapping of the higher-dimensional data (even if represented by clusters) into a single coordinate system of lower dimensionality in applications of such techniques, most recent developments were centered around nonlinear dimensionality reduction methods. These techniques, similarly to traditional linear methods, are essentially based on *spectral embeddings*, but with a key new feature now of being able to generate nonlinear embeddings.

### 3.1 Spectral embedding methodologies.

3.1.1 *Generic setting for the spectral embedding.* Nonlinear procedures of interest can be cast in the following generic setting:

1. Given the input space, compute neighbourhoods;
2. Construct the cost function to determine the weight matrix as a result of an optimisation procedure (e.g., minimising the generalized error function);
3. Based on the eigenvectors of the above matrix (can be shown with the Rayleigh-Ritz ansatz), calculate the spectral embedding.

3.1.2 *The nearest neighbour parameter as a key to success.* The starting point of spectral embedding methodologies is computing neighbourhoods. In all the algorithms currently available to the National Research Council (NRC) of Canada, our industrial partner, we have one of the two situations:

- either the number of nearest neighbours is predefined by a certain value, denoted further by $K$ (as it is the case, for example, in the Locally Linear Embedding (LLE)),
- or the input information requires the neighbourhood radius, denoted further by $\epsilon$ (as is the case, for example, in Isomap).

This may bring difficulties in some of the practical situations where a nonlinear dimensionality reduction algorithm is applied to a set of new data. Indeed, if we choose an estimate for $K$ in the LLE such that $K$ is very small compared to the real situation, a neighbourhood can falsely divide the underlying manifold. On the other hand, if we choose an estimate for $K$ in the LLE such that $K$ is large, the resulting manifold will be excessively smoothed and important small-scale features will be completely missing. Similar difficulties arise in the choice of $\epsilon$.

3.1.3 *Description of some algorithms.* **Locally Linear Embedding (LLE):** The Local Linear Embedding (or LLE) Algorithm is a spectral embedding method for nonlinear dimension reduction developed by Roweis and Saul [8]. This algorithm takes as input $X$, a $p \times n$ matrix (whose columns contain the $n$ data points in $\mathbf{R}^p$) and outputs $Y$, a $d \times n$

matrix, where $d < p$ is the dimensionality of the embedding of the input. The idea behind this algorithm is to characterize the local geometry of patches by linear coefficients that reconstruct each data point from its nearest neighbours when determining the underlying lower-dimensional manifold.

There are three major steps in the LLE algorithm. The first is to determine the neighbours in $X$-space. The second step is to solve for reconstruction weights $W_{ij}$ which allow each point $\mathbf{X}_i$ to be reconstructed from its neighbours $\mathbf{X}_j$. The final step is to compute the embedding coordinates $Y$ using the reconstruction weights $W$. We now describe these three steps in more detail.

The first step is to determine the neighbours for each data point. This can be done in various ways described above. For our purposes, we compute the $K$-nearest neighbours for each data point.

The second step is to determine the reconstruction weights. To that end, we let $W_{ij}$ denote the contribution of the $j^{th}$ data point to the $i^{th}$ reconstruction. Then, the weights $W_{ij}$ are computed that minimise the following cost function:

$$E(W) = \sum_i \|\mathbf{X}_i - \sum_j W_{ij}\mathbf{X}_j\|^2. \tag{3.1}$$

which is known as the reconstruction error. The minimisation is performed subject to two constraints. The first constraint is that $\mathbf{X}_i$ is reconstructed only from its $K$ nearest neighbours. Thus, we set $W_{ij}$ to 0 if $\mathbf{X}_j$ is not a neighbour of $\mathbf{X}_i$. The second constraint is that each set of local weights must sum to 1. Determining the optimal weights is a least squares optimisation problem that is described in further detail in Appendix A of [9].

The final step is to compute the embedding coordinates $Y$ using the weights $W$. This is done by choosing the $Y_i$ that minimise:

$$\Phi(Y) = \sum_i \|\mathbf{Y}_i - \sum_j W_{ij}\mathbf{Y}_j\|^2. \tag{3.2}$$

This specifies a quadratic form in $Y$ which can be minimised by solving a sparse $N \times N$ eigenvector problem. See Appendix B in [9] for more details.

**Isomap:** The Isomap algorithm consists of three primary steps:

1. Construct the neighbourhood graph $G$.
   As with the LLE method, the first step of the Isomap method is to determine which points are neighbours based on Euclidean distances between points in the input data in $\mathbf{R}^p$. The neighbours are obtained using one of the two basic approaches for finding nearest neighbours outlined above. The information about the neighbourhoods is collected into a weighted neighbourhood graph $G$ that has a node for each data point in the original input. The nodes of $G$ are connected iff they are neighbours and the weights on the edges connecting nodes are the corresponding Euclidean distances between neighbouring data points in $\mathbf{R}^p$. The graph $G$ is particularly easy to construct when the number of data points $n$ is smaller than the dimension $p$ of the space in which the data is embedded.
2. Construct the matrix $D$ containing the shortest paths between all pairs of points in the graph $G$.
   Isomap estimates the geodesic distances between all pairs of points on the manifold. This is achieved by computing the shortest path distances between vertices in the

weighted graph $G$ constructed in the first step. Dijkstra's algorithm is known to be a good algorithm to find a shortest path in a weighted graph.

3. Apply Multi-Dimensional Scaling to the matrix $D$ to determine the $d$-dimensional embedding.
   The final step of Isomap applies classical Multi-Dimensional Scaling (MDS) to the matrix of $D$ of graph distances as computed in the second step. The MDS algorithm constructs an embedding of the data in a $d$-dimensional Euclidean space that best preserves the intrinsic geometry of the manifold as determined by the relative distances of the points. The particular embedding found results from minimising a particular measure of error; the solution of this optimisation problem reduces to an eigenvalue problem.

Some more comments are in order concerning the final step of the Isomap algorithm. Given a set of $n$ input vectors $\{\mathbf{X}_1, \ldots, \mathbf{X}_n\} \subset \mathbf{R}^p$, the Isomap algorithm returns a set of $n$ vectors $\{\mathbf{Y}_1, \ldots, \mathbf{Y}_n\} \subset \mathbf{R}^d$ where $d < p$ is prescribed. Let

$$X = [\mathbf{X}_1, \ldots, \mathbf{X}_n] \in \mathbf{R}^{p\times n} \text{ and } Y = [\mathbf{Y}_1, \ldots, \mathbf{Y}_n] \in \mathbf{R}^{d\times n}$$

be rectangular matrices with the input and output column vectors stacked in sequence. The pre-images $Y$ of the input data $X$ are found as the minimisers of a cost function

$$E(Y) = \|\tau(D) - \tau(D_Y)\|_F, \tag{3.3}$$

where $\|A\|_F = \left[\sum_{i,j} |A_{i,j}|^2\right]^{1/2}$ is the usual Frobenius matrix norm. Further, in the definition of the cost function in (3.3), $D_Y$ denotes the matrix of Euclidean distances between all the columns of $Y$ taken pairwise, i.e.,

$$(D_Y)_{i,j} = \|\mathbf{Y}_i - \mathbf{Y}_j\| \quad (i, j = 1, \ldots, n).$$

The operator $\tau$ in (3.3) is defined as

$$\tau(A) := -\frac{1}{2} H(A \cdot A)H \tag{3.4a}$$

where $A \cdot A$ is the Hadamard (entrywise) product of $A$ with itself and $H$ is a centering matrix; explicitly,

$$(A \cdot A)_{i,j} \quad := \quad A^2_{i,j}, \tag{3.4b}$$

$$H_{i,j} \quad := \quad \delta_{i,j} - \frac{1}{n} \tag{3.4c}$$

where $n$ is the number of data points and $delta_{i,j}$ is the usual Kronecker delta. The operator $\tau$ expresses the (Frobenius) distance between matrices using matrix products and thus makes the minimisation of $E(Y)$ easier.

3.1.4 *Optimal number of nearest neighbours.* In what follows, we focus on the problem of optimal choice of nearest neighbour numbers.

As proposed by Kouropteva et al. (with modifications recently suggested by Samko et al.) [6], we choose a set of values of K from $[K_{min}, K_{max}]$. The simplest choice of $K_{min}$ in our case is 1.

Next, for each $K \in [K_{min}, K_{max}]$, we calculate the cost function:

$$E = \|\tau(\bar{D}) - \tau(D_Y)\|, \tag{3.5}$$

where

$$D_Y = \{d_{\mathbf{y}}(i,j) = \|\mathbf{Y}_i - \mathbf{Y}_j\| = \sqrt{\sum_{k=1}^{d} (\mathbf{Y}_i^k - \mathbf{Y}_j^k)^2}\} \tag{3.6}$$

is the matrix of Euclidean distances in the output space, while $\bar{D}$ is different for different spectral embedding algorithms. We focus on a generalization of LLE where we have

$$\bar{D} \equiv D_X = \{d_{\mathbf{x}}(i,j) = \|\mathbf{X}_i - \mathbf{X}_j\| = \sqrt{\sum_{k=1}^{p} (\mathbf{X}_i^k - \mathbf{X}_j^k)^2}\} \tag{3.7}$$

As usual, the operator $\tau$ converts distances to inner products (to simplify the optimisation procedure).

Then all $K$ values where minima of $E(K)$ are achieved will form the set $S_K$ of initial candidates for the optimum value of $K$.

Finally, the nonlinear reduction algorithm should be run for each $K \in S_K$. And $K_{\text{opt}}$ is determined by the following formula based on minimising the residual variance:

$$K_{\text{opt}} = \arg\min_K (1 - \rho^2_{D_X D_Y}), \tag{3.8}$$

where $\rho$ is the linear correlation coefficient taken over all entries of the matrices $D_X$ and $D_Y$ which contain Euclidean distances between pairs of points in the input (dimension p) and output (dimension d) spaces, respectively.

3.1.5 *Choice of the cost function and how to avoid ill-posedness.* The above choice of the cost function in the form of (3.5) is not the only possible one. In the generalized LLE we worked on, the cost function is taken as a measure of the reconstruction error:

$$E(W) = \sum_i \|\mathbf{X}_i - \sum_j W_{ij}\mathbf{X}_j\|^2. \tag{3.9}$$

To avoid ill-posedness it is essential to add constraints to this optimisation problem. The most natural are related to the weights, and probably the simplest one is

$$\sum_j W_{ij} = 1. \tag{3.10}$$

Further, as we observed in our experiments (described below), it might be essential to precondition the Gram matrix by a regularization procedure.

3.1.6 *Further improvements.* The search for the neighbours can be improved further if it is carried out with respect to the geodesic distance, rather than the Euclidean distance as it is commonly done. Only a slight modification of the LLE algorithm is required in this case [15]. In order to eliminate the necessity to estimate geodesic distances between faraway inputs on the manifold, and hence to improve the efficiency, we can apply a semidefinite embedding as recently proposed by Weinberger et al.

Finally, further modifications of the spectral embedding algorithms, described here, can be introduced with the stochastic neighbour embedding which could be useful for relatively noisy data.

**Potential gaps in existing literature:** There is a lot of work and experience to draw upon from the machine learning community to help with the problem of dimensionality reduction for microarray data. However, there are some features of algorithms for nonlinear dimensionality reduction described in the existing literature that complicate the present study rather than making matters more clear. We describe some of these issues with the hope that they can be resolved in later studies.

A number of nonlinear dimensionality reduction algorithms are based on the assumption that there exists a nonlinear manifold embedded in $\mathbf{R}^p$ that underlies the given set of data. The input of the algorithms usually consists of a set of $n$ data points in $\mathbf{R}^p$ and possibly the dimension $d$ of the manifold sought. The output of the algorithms typically consists of a set of $n$ vectors in $\mathbf{R}^d$ with $d < p$ that would be the pre-images of the input data vectors in a presumably lower-dimensional space. However, manifolds consist of uncountably many coordinate charts (smooth mappings from $\mathbf{R}^d$ into $\mathbf{R}^p$ whose ranges are contained in the points on the manifold) in an atlas that cover the whole manifold. It is not typical for a single coordinate chart to cover the whole manifold. Moreover, when the output from a dimensionality reduction algorithm consists of the pre-images of the data under a particular coordinate chart, it is not obvious which coordinate chart is being used. Some charts have greater utility than others in different regions of the manifold.

To clarify the preceding discussion, assume that the manifold from which the input data is sampled is the unit sphere in $\mathbf{R}^3$ (i.e., $p = 3$ and $d = 2$). Consider the mappings $\psi_1 : (0, \pi) \times (\pi, \pi) \to \mathbf{R}^3$ and $\psi_2 : \{(x, y) \in \mathbf{R}^2 : x^2 + y^2 < 1\} \to \mathbf{R}^3$ that are defined by

$$\psi_1(\phi, \theta) := (\cos\theta \sin\phi, \sin\theta \sin\phi, \cos\phi) \qquad (\phi, \theta) \in (0, \pi) \times (\pi, \pi)$$

$$\psi_2(x, y) := (x, y, \sqrt{1 - x^2 - y^2}) \qquad (x, y) \in \{(x, y) \in \mathbf{R}^2 : x^2 + y^2 < 1\}.$$

Both of these charts cover portions of the unit sphere in $\mathbf{R}^3$. However, while $\psi_1$ covers the region near the point $(0, 0, 1)$ relatively poorly due to a coordinate singularity in $\psi_1$ near $\phi = 0$, the chart $\psi_2$ can be used near that region without difficulty. Similarly, the mapping $\psi_2$ encounters difficulty near the boundary of the unit disk in $\mathbf{R}^2$ for exactly the same reason whereas $\psi_1$ has quite well-behaved derivatives near the region where $\phi = \pi/2$. As such, the output of algorithms such as LLE or Isomap consist of vectors in $\mathbf{R}^d$ , but it is in no way obvious which chart has been selected and whether it is one that is appropriate in the region of the manifold being sampled.

Another problem shared by many algorithms is the number of heuristic parameters inherent even in deterministic algorithms. For instance, in LLE, the dimension $d$ of the lower-dimensional manifold on which the sampled data lies is an input parameter of the algorithm. (Admittedly, the Isomap algorithm does not share this particular shortcoming in that it starts from $d = 1$ and increments $d$ until a suitable value of $d$ is determined.) Other heuristic choices in the development of the algorithms include the number of nearest neighbours to choose, the method by which nearest neighbours are measured, and the choice of metric in the objective function to minimise in finding the reconstruction weights.

The most perplexing difficulty arises when trying to compare the performance of distinct algorithms. If the output of algorithm A is a set of pre-images of the data under one coordinate chart and the output of algorithm B is a similar set of pre-images, does it follow that the outputs can be compared? This is a vexing issue for assessing the numerical accuracy and the asymptotic complexity of dimensionality reduction algorithms. Convergence properties of, say, numerical approximation schemes for partial differential equations, can be estimated by numerical experiments where the exact solution is known even when convergence proofs are unattainable. Such numerical experiments are invaluable when new schemes are suggested for comparison to existing frameworks. It does not seem that the literature on nonlinear dimension reduction algorithms has analogous criteria for comparison of algorithms.

## 3.2 Kernel Principle Component Analysis (KPCA).

3.2.1 *Description of the PCA method.* Principal component analysis (PCA) is one of the statistical methods to extract the patterns in the data and to represent the original data in another way based on their similarity and dissimilarity. PCA is not only widely-used for pattern extraction but also for dimensionality reduction and data visualization. Once the patterns hidden in the data are identified, we can project the data into lower dimension by selecting several most important patterns and without losing too much information. As one might expect, it is nontrivial to identify these patterns in the high-dimensional data.

PCA is essentially a basis transformation. Suppose the data points are given by $\{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_n\}$, where $\mathbf{X}_i \in R^p$ and are centered, *i.e.*, such that $\sum_{i=1}^n \mathbf{X}_i = \mathbf{0}$. The covariance matrix is then defined by

$$C = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}_i \mathbf{X}_i^T.$$

Since the orthonormal eigenvectors of the covariance matrix form a basis in space, we can express the data in terms of the eigenvector basis, instead of Cartesian coordinates. Actually, the eigenvectors show the directions of variance in the data. In addition, the corresponding eigenvalues indicate the proportions of the variances. The eigenvector corresponding to the largest eigenvalue is called the principal component.

It will be easier to analyse the data if their dimension is much smaller. Therefore, we can project the data onto a lower-dimensional space. Then the data are approximated by the linear combination of the selected $d$ eigenvectors ($d < p$) and truncating the tails of the vectors creating vectors of length $d$. The value of $d$ may be decided by the specific need, such as a value of two or three for visualization, or by minimising the difference between the original data and its approximation.

3.2.2 *Description of the Kernel PCA method.* The principle component analysis has a very long history and is known to to very powerful for the linear case. However, the sample space that many research problems are facing, especially the sample space of mircoarray data, are considered nonlinear in nature. One reason might be that the interaction of the genes are not completely understood. Many biological pathways are still beyond human comprehension. It is then quite naive to assume that the genes should be connected in a linear fashion.

To handle nonlinear spaces, a natural idea is to make a suitable transformation that tries to make the transformed space linear. Although this idea has been mentioned in the literature many times, the breakthrough did not come until the last 20 years during which time the computational issue has been resolved.

In order to capture nonlinear patterns, it is often useful to consider a nonlinear transformation of the original variables. For example, given two random variables, we might consider only the linear combination, *i.e.,* $a_1x_1 + a_2x_2$. To capture any nonlinear relationship, we might want to consider the ensemble of

$$\mathcal{E} = \{X_1, X_2, X_1^2, X_2^2, X_1X_2, X_1^3, X_2^3, X_1X_2^2, X_1^2, X_2, \cdots\}.$$

Although this can be done, the computational burden associated with the expansion into a higher-dimensional space is very costly. Given the fact the microarray data already has a very high dimensionality to begin with, this does not appear to be feasible.

To be more specific, we consider a mapping:

$$\mathcal{K} : \mathcal{X} \longrightarrow \mathcal{F} \tag{3.11}$$

where $\mathcal{X}$ and $\mathcal{F}$ are the sample and output spaces, respectively, and $K$ is the kernel function.

However, the Kernel PCA method does exactly this seemingly impossible task. The key element of the Kernel PCA is that the original space is transformed into an output space through kernel functions. Kernel functions are designed to capture the nonlinear nature of the original space by expanding the basis functions into a much higher-dimensional space. However, any computation after the transformation can be done using the kernel function and the inner product of the original space. In other words, no actual transformation is necessary, and the results are be obtained without significant computational cost.

Given $f \in \mathcal{F}$ and $g \in \mathcal{F}$, we then have

$$< f, g >= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \mathcal{K}(x_i, x_j). \tag{3.12}$$

Detailed discussions of the Kernel PCA can be found in [12].

3.2.3 *Choice of kernel functions and future research.* There are many kernel functions that have been proposed in the literature. Gaussian functions are commonly used. However, this is no guarantee that a Gaussian function would be applicable all the time. One possible approach is to use the idea of model averaging. To be more specific, one could use an array of kernel functions and evaluate each kernel function for its effectiveness using some loss function, for example, the mean squared error (MSE).

## 4 Numerical experiments and results

**4.1 Available datasets.** We now describe the two types of datasets of interest to us: a microarray dataset from the NRC and some synthetic datasets which we have designed.

4.1.1 *Microarray dataset.* The NRC provided us with an AML microarray dataset of genetic data which sampled 7129 genes in 72 patients; this corresponds to 72 vectors of data in $\mathbf{R}^{7129}$. The patterns in the microarray data are nonlinear and are thus quite complicated. In addition, the data are noisy due to the nature of the microarray experiment.

4.1.2 *Synthetic datasets.* The papers of Roweis and Tannenbaum make extensive use of test data sets for their algorithms. These include an S-shaped ribbon (a two-dimensional manifold embedded in $\mathbf{R}^3$, the standard unit sphere $S^2 \subset \mathbf{R}^3$, and a number of variants involving translations of a fixed image. These examples support the strength of these algorithms in the event that the number $n$ of samples available is larger than the dimension $p$ of the space from which the data are sampled. Unfortunately, this is not the case with cell microarray data.

We advocate generating synthetic test data known to lie on a manifold with known structure of arbitrary dimensions as a means of testing prospective algorithms. The procedure mentioned here was developed using random sampling on the unit sphere $S^d \subset \mathbf{R}^{d+1}$. This does in fact require some care; randomly sampling points on the unit hypersphere needs to be done in a way to ensure that samples are not clustered near poles. Fortunately, a very simple framework for doing so is provided by Knuth.

1. Generate random vectors $\mathbf{Y}_1, \ldots, \mathbf{Y}_n \in \mathbf{R}^{d+1}$, each component being observations of a random variable with Gaussian distribution with mean at 0.
2. $\mathbf{Y}_k \mapsto \mathbf{Y}_k / \|\mathbf{Y}_k\|_2$ $(k = 1, \ldots, n)$ generates set of $n$ random vectors uniformly-distributed on the unit sphere in $\mathbf{R}^{d+1}$.
3. Embed vectors $\mathbf{Y}_k$ into column vectors $\mathbf{X}_k \in \mathbf{R}^p$ by padding with zeros.

The vectors generated thusly lie on the unit sphere $S^d \subset \mathbf{R}^p$. This procedure can be adapted to make data points from the manifolds $S^{d_1} \times S^{d_2}$ embedded in $\mathbf{R}^p$ or any similar Cartesian product manifold. To obscure the obvious manifold structure of a set of vectors in $\mathbf{R}^p$ with zeros in most of the components, a number of strategies can be used.

1. Make the substitutions $\mathbf{X}_k \mapsto P\mathbf{X}_k$ where $P$ is a random $p \times p$ permutation matrix.
2. Make the substitutions $\mathbf{X}_k \mapsto Q\mathbf{X}_k$ where $Q = I - 2\mathbf{u}\mathbf{u}^T$ is a random orthogonal Householder reflection ($\|\mathbf{u}\|_2 = 1$).
3. Make the substitutions $\mathbf{X}_k \mapsto \mathbf{X}_k + \mathbf{a}$ where $\mathbf{a} \in \mathbf{R}^p$ is a random translation.
4. Add Gaussian noise to all of the components of each vector.

**4.2 Numerical experiments.** For our numerical experiments, we tested the LLE algorithm on the NRC microarray dataset described above. In particular, we performed a leave-one-out cross-validation experiment and measured the reconstruction error for several combinations of $d$ and $K$.

Leave-one-out is a cross-validation technique in which the data is divided into $n$ subsets each corresponding to one data point. Training on the data is performed $h$ times, each time using only the omitted subset to compute the error criterion of interest. The following pseudo-code demonstrates the use of the leave-one-out technique for the LLE algorithm:

```
for i = 1:n
  1.  Remove X_i from X, i.e., set Xhat_i = X\{X_i}.
  2.  Compute the manifold on Xhat_i.
  3.  Project X_i onto Xhat_i.
  4.  Compute the reconstruction error for X_i.
end
```

Then, the reconstruction error is the sum of the reconstruction errors for each of the $n$ data points.

We repeated this experiment on the filtered dataset (using the result from the Random Forest algorithm described above). Before discussing our numerical results, we describe three main metrics for analyzing the error in the nonlinear dimension reduction process.

**4.3 Measures for error analysis.** There are three main measures for the error in the nonlinear dimension reduction algorithms: the distortion of the distances, the residual variance, and the reconstruction error.

To measure the distortion of the distances, we compute $E(W) = \sum_{i<j} W_{ij}(D_{ij} - \Delta_{ij})^2$, where $D_{ij}$ is the distance between the points in $\mathbf{R}^p$, and $\Delta_{ij}$ is the corresponding distance in $\mathbf{R}^d$. Although an interesting error metric, we have not studied this metric in favor of pursuing others.

The measures for characterizing the quality of the nonlinear dimension reduction procedure described here are based on the following two choices:

- Minimisation of the generalized reconstruction error:

$$E_{\mathbf{y}} = \frac{1}{n} \sum_i \|\mathbf{X}_i - P\mathbf{X}_i\|^2, \tag{4.1}$$

  where P is the projection operator, $P^2 = P$ such that $\mathbf{Y}_i = P\mathbf{X}_i$. Note that we go from a space of dimensionality $p$ to a space of dimensionality $d$ where for the linear pieces we have

$$\mathrm{var}(\mathbf{Y}) = \mathrm{Tr}(PCP^T), \quad P = \sum_{\alpha=1}^{d} \mathbf{e}_\alpha \mathbf{e}_\alpha{}^T, \tag{4.2}$$

$$C = \frac{1}{n}\sum_i \mathbf{X}_i \mathbf{X}_i^T = \sum_{\alpha=1}^{p} \lambda_\alpha \mathbf{e}_\alpha \mathbf{e}_\alpha{}^T, \tag{4.3}$$

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p. \tag{4.4}$$

- Minimisation of the residual variance:

$$1 - \rho^2_{D_X D_Y} \tag{4.5}$$

  is as discussed above (see (3.8)).

Determining the error of the reduction with (4.5) was proposed for the original LLE algorithm [8]. Note, however, that since all the available algorithms require computing spectral characteristics of the underlying data in one form or another, the computational dichotomy of spectra may represent a non-trivial problem in practice [7]. Nevertheless, recent applications of spectral embedding non-linear reduction techniques, such as LLE and Isomap, to high-density microarray data sets have demonstrated their robustness [5, 2]. Finally, note that in the linear case, the approach based on (4.5) is the standard maximisation of variance subspace:

$$\mathrm{var}(\mathbf{Y}) = \frac{1}{n}\sum_i \|P\mathbf{X}_i\|^2, \tag{4.6}$$

and is equivalent to the procedure (4.1) based on the minimum reconstruction error.

**4.4 Numerical results.** Now that we have described three possible error metrics, we return to a description of the results from obtained from the LLE algorithm by running the leave-one-out cross-validation experiment on the NRC microarray dataset. We will also describe the results from running leave-one-out on the corresponding filtered dataset.

The following two figures show the results from the leave-one-out cross-validation experiment using LLE on the microarray and filtered microarray datasets. We see from both figures that the best results are obtained for roughly $K = 12$ nearest neighbours. This result is independent of the choice of $d$. When a greater number of nearest neighbours is used, the LLE algorithm is more expensive, and there is little to no additional benefit, *i*.e., the reconstruction error decreases little. The figures also demonstrate that the reconstruction error is minimised for low-dimensional manifolds of smaller dimension. This result is also independent of whether or not filtering was applied. As was expected, the amount of reconstruction error decreased when filtering was applied to the microarray dataset before the leave-one-out cross validation experiment was performed; this demonstrates the success of the filtering process.

## 5 Discussion and recommendations

As discussed above, support vector machines may be useful for gene selection when combined with the random forest algorithm. There are other issues to explore within the filtering context such as filtering time trends.

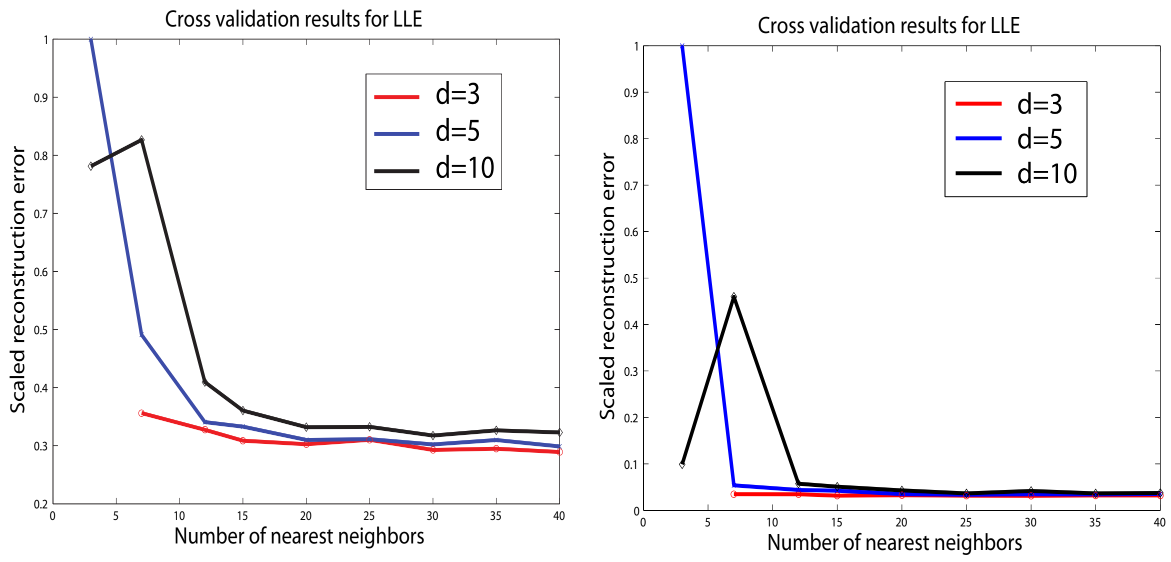

**Figure 2** Leave-one-out cross validation results for the LLE algorithm on the NRC microarray (left) and filtered microarray (right) datasets. These figures demonstrate that 12 is a good choice for the number of nearest neighbours in this algorithm. In addition, the reconstruction error is smaller for lower-dimensional manifolds and for the filtered microarray dataset.

Within Kernel PCA, the choice of kernel function should be investigated to find the most useful type of kernel for the microarray and synthetic datasets.

There are several avenues to pursue within the spectral embedding framework for nonlinear dimensionality reduction. First, we would like to experiment with the Isomap algorithm and compare the results of that algorithm with the LLE experiments. Comparisons should also be made with the other spectral embedding algorithms. Thus far, our results indicate that the Random Forest and LLE algorithms were useful for filtering the genes and nonlinear dimensional reduction as tested on the NRC microarray dataset.

A second avenue to explore is the choice of the nearest neighbours in LLE and other spectral-embedding algorithms. There are many options for choosing the neighbours within LLE such as the using the $K$-nearest neighbours, the points within a ball of a specified radius, or using an adaptive local distance metric to choose the neighbours flexibly within various regions. It is expected that an adaptive choice for the neighbours will produce improved results. Above it was discussed how to pursue an optimal number of neighbours.

A third avenue that needs to be investigated is the choice of cost function. Our experiments measured the reconstruction error because this metric was of interest to the NRC, our industrial partner. It is not clear which error metric would be the most useful for the general case, as we have not run any experiments using the distortion of the distances or the residual variance as our error metric. We have not run any experiments on the synthetic datasets which we have designed; tests will need to be run on this dataset for us to be able to understand how these algorithms perform on additional datasets.

The final issue we have identified for investigation is the choice of cross-validation technique. Our experiments used the leave-one-out cross-validation technique. This can be generalized to the leave-v-out technique, which is a more complicated version of leave-one-out in which all possible subsets of v data points are left out of the training set. As the choice of cost function changes, the most successful cross-validation technique may change as well.

Numerous experiments need to be performed on the microarray and synthetic datasets in order for us to better understand the performance of the algorithms described in this report on filtering of genes and the nonlinear dimensionality reduction problem.

## References

[1] L. Breiman. (2001), Random forests. *Machine Learning* 45:5–32.

[2] K. Dawson, R. L. Rodriguez, and W. Malyj (2005), Sample phenotype clusters in high-density oligonucleotide microarray data sets are revealed using Isomap, a nonlinear algorithm. *BMC Bioinformatics* 6(195):1–17.

[3] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, and M. Caliguiuri (1999), Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science* 286:531536–531536.

[4] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik (2002), Gene selection for cancer classification using support vector machines. *Machine Learning* 46:389–422.

[5] B. W. Higgs, J. Weller, and J. L. Solka (2006). Spectral embedding finds meaningful (relevant) structure in image and microarray data. *BMC Bioinformatics* 7(74):1–13.

[6] O. Kouropteva, O. Okun, and M. Pietikainen (2002), Selection of the optimal parameter value for the linear embedding algorithm. In *Proc. of the 1st International Conference on Fuzzy Systems and Knowledge Discovery* pages 359–363.

[7] R. V. N. Melnik (2000), Topological analysis of eigenvalues in engineering computations. *Engineering Computations* 17(4):386–416.

[8] S. T. Roweis and L. K. Saul (2000), Nonlinear dimensionality reduction by locally linear embedding. *Science* 290:2323–2326.

[9] L. Saul and S. Roweis (2001), An introduction to locally linear embedding. Paper located at http://www.cs.toronto.edu/˜roweis/lle/publications.html

[10] L. K. Saul and S. T. Roweis (2003), Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research* 4:119–155.

[11] M. Schena, D. Shalon, R. Heller, A. Chai, P. O. Brown, and R. W. Davis 1996), Parallel human genome analysis: Microarray-based expression monitoring of 1000 genes. In *Proc. Nat. Acad. Sci.* Volume 93, pages 10615–10619. National Academy of Sciences.

[12] J. Shawe-Taylor and N. Cristianini (2006), *Kernel Methods for Pattern Analysis.* Cambridge University Press.

[13] M. Shena, D. Shalon, R. W. Davis, and P. O. Brown (1995), Quantitative monitoring of gene expression patterns with complimentary DNA microarray. *Science* 270:467–470.

[14] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu (2002), Diagnosis of multiple cancer types by shrunken centroids of gene expression. In *Proc. Nat. Acad. Sci.* volume 99, pages 6567–6572. National Academy of Sciences.

[15] C. Varini, A. Degenhard, and T. W. Nattkemper (2006), ISOLLE: LLE with geodesic distance. *Neurocomputing* 69:1768–1771.

[16] J. Zhu and T. Hastie (2004), Classification of gene microarrays by penalized logistic regression. *Information Processing Systems* 14:427–443.