Analysis of Medical Data Using Dimensionality Reduction Techniques

Robert E. Colgan[a], David E. Gutierrez[b], Jugesh Sundram[c] and Gnana Bhaskar Tenali[d]

[a]Computer Science, Columbia University, New York, NY; rec2111@columbia.edu

[b]Computer Sciences, Florida State University, Tallahassee, Florida; deg10e@my.fsu.edu

[c]Mechanical Engineering, Florida Institute of Technology, Melbourne, Florida; jsundram2012@my.fit.edu

[d]Electrical & Computer Engineering, Florida Institute of Technology, Melbourne, Florida; gtenali@fit.edu

ABSTRACT

High-dimensional data can be difficult to analyze, almost impossible to visualize, and expensive to process and store. In many cases, the high-dimensional data points may all lie on or close to a much lower-dimensional surface, or manifold, implying the intrinsic dimensionality of the data is much lower. In that case, the data could be described with fewer dimensions, allowing us to mitigate the curse of dimensionality. Transforming the high-dimensional representation of the data to a lower-dimensional one without losing important information is the central problem of dimensionality reduction. Many methods of dimensionality reduction have been developed, including classical techniques like Principal Component Analysis (PCA) and newer methods such as Diffusion Maps (DM). Most of these methods often perform well on some types of data but poorly on others. We apply different dimensionality reduction methods to medical data, including breast tissue tumor data and kidney proteomics data, in order to determine which methods and parameters work best on on the different types of data. To evaluate the performance of the reduction method, we also classify the data in the reduced dimension using standard classification algorithms and evaluate the accuracy.

Keywords: Classification, Diffusion Maps, Dimensionality Reduction, High-dimensional Data, Independent Component Analysis, Locally Linear Embedding, Kernel Principal Component Analysis, Mapping, Nonlinear Dimensionality Reduction, Principal Component Analysis, K-Nearest Neighbors

1. INTRODUCTION

Dimensionality reduction is the mapping of high dimensional to a more meaningful, lower dimensional space. DR is in some cases an essential preliminary step towards building models.[2] In a medical context, dimensionality reduction is often particularly necessary, as raw medical data (*e.g.*mass-spectra based proteomic data, human gene distributions) is often of very high dimensionality. By reducing the dimensions, we can mitigate this problem and possibly reduce computational time for analysis and further processing (*e.g.*clustering, regression, etc), visualization and storing.

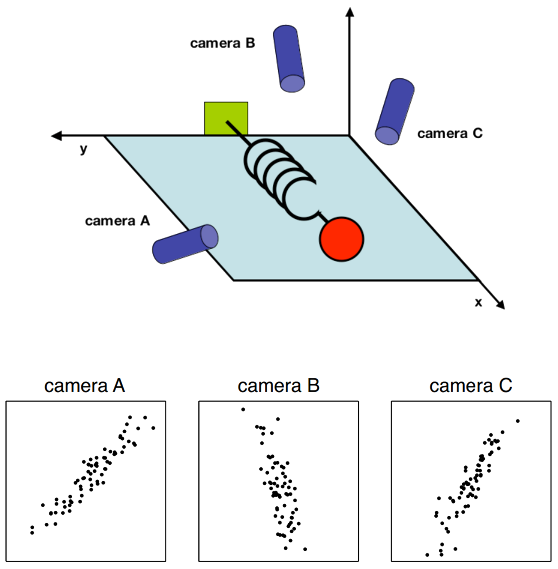
Reducing the dimensions of high-dimensional data also allows us to acquire a better understanding of the underlying structure of the data, and bring to light a more meaningful representation of what is sometimes difficult to interpret and impossible to visualize. Dimensionality reduction can be achieved either by feature selection or feature transformation. Feature selection is the selection of relevant and non-redundant features from certain data, thus reducing the dimensions. Feature transformation reduces dimensions by transforming the data into a lower dimensional space while still maintaining the underlying structure from the feature space (*i.e.*the mapped low-dimensional data through feature transformation should preserve similar characteristics of the high-dimensional data in the feature space, such as the local mutual distances). After performing feature transformation, the features of the reduced data do not correspond directly to original features. Since the mapping of feature transformation expresses the relationship between the initial features, feature transformation have more potential for exposing the difference in content than feature selection.[2] This report is focused on a few methods for feature transformation and their application on certain types of medical data. In particular, we considered the following dimensionality reduction methods: PCA, Independent Component Analysis (ICA),

AMALTHEA REU Technical Report No. 2013-1; available at www.amalthea-reu.org. Please send your correspondence to Georgios C. Anagnostopoulos. Copyright © 2013 The AMALTHEA REU Program.

Locally Linear Embedding (LLE), and DM. Feature transformation can be further subcategorized into two groups: linear dimensionality reduction methods and nonlinear dimensionality reduction (NLDR) methods. PCA and ICA are both linear dimensionality reduction methods, while LLE and DM are NLDR methods.

1.1 Example of Dimensionality Reduction

As an example of the utility of dimensionality reduction, consider the following example, adapted from.[20] Consider the study of motion of a ball attached to a spring, without anything about its motion or the best way to represent its position. Let three cameras pointing at the ball from different angles capture the motion of the ball. Over N frames, each camera captures an x and y coordinate representing the ball's position in its field of view (Figure 1a). Thus, we obtain N data points, each with six dimensions (Figure 1b). Clearly, this is not the most efficent representation of the data; it also is difficult to interpret the results and understand the underlying process when the data is in this form. If the data could be represented well in only one dimension, a dimensionality reduction technique could be applied to reduce the N points from the original six-dimensional space to N points in one-dimensional space—*i.e.*a single value for every instance (Figure 1c). (Choosing the number of dimensions to reduce the data to is often a more difficult problem than this, as the true intrinsic dimensionality of the data is rarely as obvious as in this example.) This reduced data better reflects the motion of the ball, and also is easier to store and process.



(a) Three cameras capture the motion of a ball attached to a spring.

$$\mathbf{X} = \begin{bmatrix} x_A^{(1)} & y_A^{(1)} & x_B^{(1)} & y_B^{(1)} & x_C^{(1)} & y_C^{(1)} \\ x_A^{(2)} & y_A^{(2)} & x_B^{(2)} & y_B^{(2)} & x_C^{(2)} & y_C^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_A^{(N)} & y_A^{(N)} & x_B^{(N)} & y_B^{(N)} & x_C^{(N)} & y_C^{(N)} \end{bmatrix}$$

(b) Original data matrix $\mathbf{X}$, consisting of N six-dimensional data points. $x_A^{(t)}$ refers to the x-coordinate collected by camera A in frame t.

$$\mathbf{R} = \begin{bmatrix} r^{(1)} \\ r^{(2)} \\ \vdots \\ r^{(N)} \end{bmatrix}$$

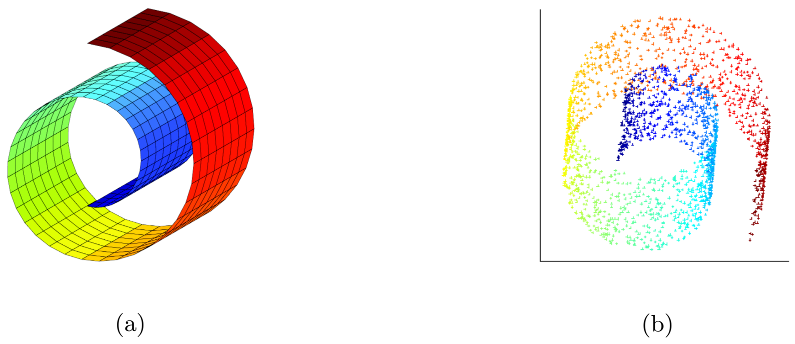
(c) Reduced data matrix $\mathbf{R}$, consisting of N one-dimensional data points. $r^{(i)}$ is the reduced version of row i of $\mathbf{X}$.

Figure 1: An example illustrating the utility of dimensionality reduction.

1.2 Linear vs. Nonlinear

Dimensionality reduction has traditionally been performed using linear methods such as PCA. However, the intrinsic geometry of data points lying on a nonlinear manifold is better captured by applying nonlinear dimensionality reduction methods. This point is highlighted by applying PCA (linear DR method) and LLE

(NLDR method) on the "Swiss roll" dataset, which consists of points lying on a two-dimensional manifold that has been nonlinearly embedded in three-dimensional space (Figure 2a).



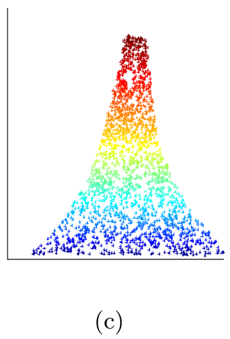


Figure 2: (a)Original Swiss roll in 3-D; (b) reduced to 2-D with PCA; (c), reduced to 2-D with LLE

Linear methods fail to capture the underlying manifold of the data. Consider the PCA mapping of the "Swiss roll" in Figure 2b. PCA maps far away points on the manifold to nearby points on the 2-D plane. In contrast to the traditional linear methods, nonlinear methods such as LLE can correctly determine the underlying structure and map the red points far from the blue points (Figure 2c).

1.3 Notations

We use the following notational conventions in this report. A dataset consisting of m data points each with n dimensions is represented as an $m \times n$ matrix $\mathbf{X}$. Each row of the data matrix $\mathbf{X}$ contains a single data point, which we also refer to as an instance, and which is represented as a row vector $\mathbf{x}$, or $\mathbf{x}_i$ to indicate the ith data point (ith row of $\mathbf{X}$). Each data point $\mathbf{x}$ contains n elements, corresponding to the columns of $\mathbf{X}$. We refer to the columns of $\mathbf{X}$ as dimensions, features, variables, components, or parameters.

1.4 Organization of the Report

This report is organized as follows: an introduction on a few classification methods are discussed in Section 2. Basic aspects, procedures and some examples of various dimensionality reduction algorithms are presented in the following sections: PCA in Section 3, ICA in Section 4, LLE in Section 5, Kernel Principal Component Analysis (KPCA) in Section 6, and DM in Section 7. We then discuss the two datasets we worked with, the Wisconsin Diagnostic Breast Cancer (WDBC) dataset(Section 8) and the African American Study of Kidney Disease and Hypertension (AASK) dataset (Section 9), and the results we obtained for them. Finally, Section 10 encompasses the conclusion of this report.

2. CLASSIFICATION METHODS

We made use of several classification methods We tested several classification methods on the data before applying dimensionality reduction techniques in order to get a baseline accuracy level to which we could compare the accuracy of classification after dimensionality reduction. After dimensionality reduction, we mainly used k-nearest neighbors (KNN) to classify the results, because the choice of reduction method is usually more important than the classification algorithm;[8] we hoped to achieve results that would be easy to classify using any method, and KNN is straightforward and well-tested.

2.1 K-Nearest Neighbors

KNN classifies data points based on the known classification of their nearest neighbors. We simply classify a test point $\mathbf{x}$ according to the classes of the points closest to it by Euclidean distance in whatever dimension the points lie. If a majority of $\mathbf{x}$'s k nearest neighbors belongs to class 1, we classify $\mathbf{x}$ as belonging to class 1; if a majority belongs to class 2, we classify it as belonging to class 2. k is taken to be an odd number to avoid ties; we used values of k between 1 and 19, depending on the data.

We can also perform KNN after normalizing the dataset by subtracting the mean and dividing each feature by the standard deviation of that feature. This has the advantage of not over-emphasizing features because they have a greater magnitude.

2.2 Z-score Based Methods

We used several methods based on the z-score, or standard score, of the data points. The z-score of a data point x is the difference between that point and the mean of the population, divided by the standard deviation of the population:[13]

$$z(s) = \frac{x - \bar{x}}{s} \tag{1}$$

Where $\bar{x}$ is the mean and s is the standard deviation. In other words, it measures how many standard deviations away from the mean x is. A drawback of these methods is that (unlike KNN) they assume the data points are normally distributed. To classify a data point based on the z-score, we compute the mean of each feature in the training data, separately for each class. We also compute the overall standard deviation of each feature. To classify a test point, for each of its features, we compute the difference between that feature and the mean of the corresponding feature in class 1, and also the difference between that feature and the mean of the corresponding feature in class 2. We then divide both these differences by the overall standard deviation of that feature, getting a z-score for that feature for each class. We then take the average of these z-scores across all features, and classify the test point as belonging to whichever class for which the average z-score of that point is smaller.

One variation on this method we used made use of feature selection to improve accuracy. After computing the mean of each feature in the training data for each class, we compute the difference between these means for each feature and normalize by the overall standard deviation of that feature. This gives a rudimentary score of the "importance" of each feature based on how many standard deviations apart the means of that feature in the two classes are. This can be thresholded, or features can be ranked according to this scale to select a certain number of most important features.

2.3 Cross-validation

We used repeated random sub-sampling validation to cross-validate our results for all classification methods we used. If $\mathbf{X}$ is the dataset on which we want to test a classification algorithm, we randomly divide the data points in $\mathbf{X}$ into two groups: $\mathbf{X}_{tr}$ (training) and $\mathbf{X}_{te}$ (testing), with an 80%-20% split between groups. We use $\mathbf{X}_{tr}$ as the training data for whatever classification algorithm we are using, then attempt to classify each point in $\mathbf{X}_{te}$ with that classification algorithm, and record the accuracy of the classifications. We repeat this process for some number of trials (usually at least 1000), with a new random division between testing and training data for each trial, and take the average of the accuracy over all trials.

3. PRINCIPAL COMPONENT ANALYSIS

3.1 Overview

PCA is one of the most commonly used methods of dimensionality reduction. The goal is to linearly transform the data to a lower-dimensional representation while preserving as much of the variance of the original data as possible. After applying PCA, each data point from the original dataset is re-expressed in terms of a new set of variables, known as the principal components. The principal components are linear combinations of the original features, and are ordered such that the first principal component accounts for the most variance, the second accounts for the second most variance, and so on. Ideally, nearly all of the variance in our data might be expressed by only the first few principal components, allowing us to discard the rest of the components without losing much important information. For example, after applying PCA to the data collected from the ball on a spring example discussed previously, the first principal component would correspond to the ball's position along the axis of the spring. This would account for all of the variance in the data collected about the movement of the ball, except what results from noise. After discarding the other components, the ball's position would then be expressed using only one variable, instead of the original six.

3.2 Procedure

Assume we have a dataset represented by $\mathbf{X}$, an $m \times n$ matrix where rows represent data points and columns represent variables/features. The first step is to subtract the mean. We compute the mean of each of the n variables (columns) across all m rows, then subtract the result from each row. The data points are now centered at the origin. Now we compute the covariance matrix $\mathbf{C}$:

$$\mathbf{C} = \frac{1}{m}\mathbf{X}^T\mathbf{X} \tag{2}$$

$\mathbf{C}$ is a symmetric $n \times n$ matrix which has been normalized by the number of data points. The entries on the diagonal of $\mathbf{C}$ represent the variance of each feature, and the other entries (i, j) represent the covariance between feature i and feature j. Then we calculate the eigenvector decomposition of $\mathbf{C}$:

$$\mathbf{C} = \mathbf{\Phi}\mathbf{\Delta}\mathbf{\Phi}^{-1} \tag{3}$$

where $\mathbf{\Phi}$ is a square $n \times n$ matrix whose columns contain the normalized eigenvectors of $\mathbf{C}$, and $\mathbf{\Delta}$ is a diagonal matrix containing the eigenvalues corresponding to each eigenvector. Because $\mathbf{C}$ is a symmetric matrix, $\mathbf{\Phi}^{-1} = \mathbf{\Phi}^T$. The vectors in $\mathbf{\Phi}$ represent an orthonormal basis for our data, and the eigenvalues in $\mathbf{\Delta}$ represent the variance explained by each of the eigenvectors. If we sort the eigenvectors in $\mathbf{\Phi}$ by their eigenvalues and save this matrix as $\mathbf{P}$, then the eigenvector in the first column of $\mathbf{P}$ is the principal component. $\mathbf{P}$ is a projection matrix we can use to transform our original data:

$$\mathbf{Y} = \mathbf{X}\mathbf{P} \tag{4}$$

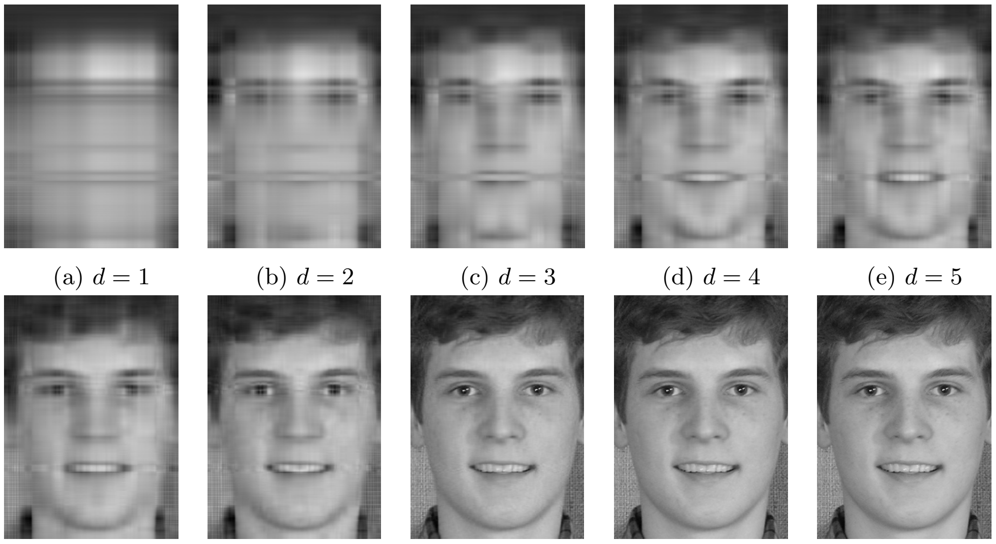
Each row of the transformed data in $\mathbf{Y}$ corresponds to the same data point as the corresponding row in $\mathbf{X}$, but the columns correspond to the components of the new basis, in order from most to least important. We can thus take only the first d columns of $\mathbf{Y}$, where d is the number of dimensions (components) to which we want to reduce the data. (Or we could have taken only the first d columns of $\mathbf{P}$ instead of the full $\mathbf{P}$ in Eq. (4).)

3.3 Example: Image Compression

If we consider an $m \times n$ pixel grayscale image as an $m \times n$ matrix of pixel values $\mathbf{X}$, we can perform PCA on $\mathbf{X}$ to represent it as an $n \times d$ principal component matrix $\mathbf{P}$ and a $m \times d$ data matrix $\mathbf{Y}$, saving storage space. We run Algorithm 1 with $\mathbf{X}$ as input, saving $\mathbf{P}$ and the output $\mathbf{Y}$. To view the compressed image $\mathbf{X}'$, we convert back to the original basis by calculating $\mathbf{X}' = \mathbf{Y} \times \mathbf{P}^T$.[16]

Algorithm 1 Dimensionality Reduction by PCA

```
Input: Data $\mathbf{X} \in R^{m \times n}$, $d$ ($m$ = No. of data points, $n$ = No. of dimensions, $d$ = No. of dimensions to reduce to)
Output: $\mathbf{Y} \in R^{m \times d}$
1. colMeans ← mean of each column of $\mathbf{X}$
2. For all rows $\mathbf{x_i}$ in $\mathbf{X}$:
3.     $\mathbf{x_i} \leftarrow \mathbf{x_i} - \mathbf{colMeans}$
4. End For
5. $\mathbf{C} \leftarrow \frac{1}{m}\mathbf{X}^T\mathbf{X}$
6. $\mathbf{\Phi} \leftarrow$ eigenvectors of $\mathbf{C}$
7. $\lambda \leftarrow$ eigenvalues of $\mathbf{C}$
8. $\mathbf{P} \leftarrow$ columns of $\mathbf{\Phi}$ sorted by corresponding eigenvalue in $\lambda$
9. $\mathbf{P} \leftarrow$ first $d$ columns of $\mathbf{P}$
10. $\mathbf{Y} \leftarrow \mathbf{X} \times \mathbf{P}$
```



(a) $d = 1$ (b) $d = 2$ (c) $d = 3$ (d) $d = 4$ (e) $d = 5$

(f) $d = 6$ (g) $d = 11$ (h) $d = 41$ (i) $d = 81$ (j) Original

Figure 3: Image compression by PCA for different values of d, the number of principal components used.

4. INDEPENDENT COMPONENT ANALYSIS

4.1 Overview

To illustrate the motivation behind the method of Independent Component Analysis and to better explain the algorithm, consider the famous cocktail-party problem. Consider a room where two people are talking. Suppose two microphones located at different locations around the room. Each microphone captures a mixed signal from the two voices of the people talking. We would like to get the independent voice signals (*i.e.* each individual voice) from the mixed signals captured by both microphones. An important note to consider is that if N sources are present, at least N observations (*e.g.* microphones) are needed to recover the original signals.

These mixed signals may be represented as the following linear system:

$$x_1(t) = \alpha_{11}s_1 + \alpha_{11}s_2 \quad (5)$$

$$x_2(t) = \alpha_{21}s_1 + \alpha_{22}s_2 \quad (6)$$

The idea is to separate the individual signals $s_1(t)$ and $s_2(t)$ using only the recorded signals $x_1(t)$ and $x_2(t)$. A key assumption that is made in ICA, is that at each time instant t, $s_1(t)$ and $s_2(t)$ are statistically independent.

This allows for a reasonable estimation of the α_{ij}. Once the α_{ij} have been estimated, the independent signals $s_1(t)$ and $s_2(t)$ can be extracted from the mixed signals $x_1(t)$ and $x_2(t)$.[10]

Let $\mathbf{A}$ be the matrix with elements α_{ij}.

$$\texttt{Let} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \text{and} \quad \mathbf{s} = \begin{bmatrix} s_1 \\ \vdots \\ s_n \end{bmatrix}$$

The ICA model can then be written as

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{7}$$

All that is observed is the vector $\mathbf{x}$, and both $\mathbf{A}$ and $\mathbf{s}$ must be estimated using it. After estimating the matrix $\mathbf{A}$, its inverse, say $\mathbf{W}$, can then be computed. The independent signals can then be computed by:

$$\mathbf{s} = \mathbf{W}\mathbf{x} \tag{8}$$

4.2 Independence

Two scalar-valued random variables x_1 and x_2 are said to be independent if information on the value of x_1 does not convey any information on the value of x_2 and vice versa. As a consequence, receiving information about one of the variables does not change the probability distribution of the other. In technical terms, independence between two random variables is defined as the joint probability density. Let $p(x_1, x_2)$ denote the probability density function of x_1 and x_2. Let $p(x_1)$ be the pdf of x_1 when it is considered alone:

$$p_1(x_1) = \int p(x_1, x_2) dx_2 \tag{9}$$

$p_2(x_2)$ is defined similarly. Then, random variables x_1 and x_2 are independent if and only if:

$$p_1(x_1, x_2) = p_1(x_1) p_2(x_2) \tag{10}$$

If two variables x_1 and x_2 are independent, then they are uncorrelated. However, if two variables x_1 and x_2 are uncorrelated, they are not necessarily independent. Uncorrelatedness is a weaker form of independence. Two variables x_1 and x_2 are said to be uncorrelated if their covariance is zero:

$$E\{x_1 x_2\} E\{x_1\} E\{x_2\} = 0 \tag{11}$$

Many ICA methods give uncorrelated estimates of the independent components, reducing the number of free parameters and simplifying the problem.[10] For ICA to be possible, independent components must be nongaussian.

4.3 Measures of Nongaussianity

A quantitative measure of nongaussianity of a random variable, say x, is needed to use nongaussianity in ICA estimation. Let us assume that x is centered and has variance equal to one. Two methods will be discussed in this section, Kurtosis and Negentropy.

4.3.1 Kurtosis

The kurtosis of x is defined by

$$kurt(x) = E\{x^4\} - 3(E\{x^2\})^2 \tag{12}$$

Since x is assumed to be of unit variance, the right-hand side equals $E\{y^4\} - 3$. Kurtosis is zero for a gaussian random variable and nonzero for most nongaussian random variables.[10]

Other methods for measuring nongaussianity might be better in some cases because kurtosis is very sensitive to outliers.[9]

4.3.2 Negentropy

The entropy of a random variable x with density function $p(x)$ is defined as:

$$H(x) = -\int p(x) \log p(x) dx = -E\{\log p_i(x)\} \tag{13}$$

The negentropy can be approximated by:

$$J(x) \approx \sum_{i=1}^{p} k_i \left[E\{G_i(x)\} - E\{G_i(g)\}\right]^2 \tag{14}$$

where k_i are some positive constants, and g is a Gaussian variable with zero mean and unit variance. G_i are some non-quadratic functions such as:

$$G_1(u) = \frac{1}{a_1} \log\cosh a_1 u, G_2(u) = -\exp(\frac{-u^2}{2}) \tag{15}$$

where $1 \leq a_1 \leq 2$ is some suitable constant.

4.3.3 Preprocessing for ICA

It is convenient to preprocess data before applying ICA. Two techniques will be discussed to make ICA estimation simpler and reduce the complexity of the problem.

4.3.4 Centering

Typical ICA algorithms use centering as preprocessing step to simplify the implementation. To center $\mathbf{x}$, subtract its mean vector $\mathbf{m} = E\{x\}$ to create a zero mean variable. Centering $\mathbf{x}$ implies that $\mathbf{s}$ is zero mean vector as well, as can be seen by taking the expected value on both sides of Eq. (7).

4.3.5 Whitening

After centering the data, another useful preprocessing technique is whitening. In whitening, the vector $\mathbf{x}$ is linearly transformed to obtain a new vector $\tilde{\mathbf{x}}$ where its components are uncorrelated and their variances equal unity, *i.e.* the covariance matrix of $\tilde{\mathbf{x}}$ equals the identity matrix:

$$E\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T\} = \mathbf{I} \tag{16}$$

4.4 FastICA Algorithm

4.4.1 Single-unit version

The following is a one-unit version of FastICA. "one unit" refers to a single computational unit. The FastICA essentially finds a unit vector $\mathbf{w}$ such taht the projection $\mathbf{w}^T\mathbf{w}$ maximizes nongaussianity as measured by the approximation of negentropy $J(\mathbf{w}^T\mathbf{w})$ given in (14). The basic form of the FastICA algorithm is as follows:

1. Choose an initial(*e.g.*random) weight vector $\mathbf{w}$
2. Let $\mathbf{w}^+ = E\{\mathbf{x}g(\mathbf{w}^T\mathbf{x})\} - E\{g'(\mathbf{w}^T\mathbf{x})\}\mathbf{w}$
3. Let $\mathbf{w} = \frac{\mathbf{w}^+}{\|\mathbf{w}^+\|}$
4. If not converged, go back to 2.

Convergence means that the dot product of the old and new $\mathbf{w}$ is almost equal to 1. g corresponds to the derivatives of the nonquadratic functions used in (15). The derivatives are:

$$g_1(u) = \tanh(a_1 u), g_2(u) = u \exp\frac{-u^2}{2} \tag{17}$$

4.4.2 Multiple-units version

The one-unit FastICA algorithm needs to be run using several units with weight vectors $\mathbf{w}_1, ...\mathbf{w}_n$. Vectors need to be prevented from converging to the same maxima, so we must decorrelate the outputs $\mathbf{w}_1^T x, ..., \mathbf{w}_n^T x$ after every iteration.

It may be desired to use a symmetric decorrelation in some applications, thus the following algorithm assures that no vectors are privileged over others[12] by the classical method involving matrix square roots:

$$\texttt{Let }\ \mathbf{W} = (\mathbf{W}\mathbf{W}^T)^{-\frac{1}{2}}\mathbf{W} \tag{18}$$

Another simple way of achieving decorrelation is with the following iterative algorithm:[10]

$$\begin{aligned} &1.\ \texttt{Let }\ \mathbf{W} = \frac{\mathbf{W}}{\sqrt{\|\mathbf{W}\mathbf{W}^T\|}} \\ &\quad\texttt{Repeat 2. until convergence:} \\ &2.\ \texttt{Let }\ \mathbf{W} = \frac{3}{2}\mathbf{W} - \frac{1}{2}\mathbf{W}\mathbf{W}^T\mathbf{W} \end{aligned} \tag{19}$$

4.5 Examples

Two images are selected from a "see-through" data set.[1] The problem at hand consists of separating two images from two mixed images. Imagine two pictures have been printed on opposite sides of a very thin sheet paper. When the images are scanned, the image in printed in the back shows due to the transparency of the paper. This problem is suitable for ICA because from the mixed pictures, we would like to obtain each original image separately. The data used is from a severe case of this problem, in which the paper used was "onion skin" (i.e. semi-transparent paper often used in professional drawing).

As seen in Figure 4, images (e) and (f) were successfully separated from the mixed images (c) and (d). The extracted images are not quite as perfect as the originals, but compared to the mixed images, the distinction between the two independent images is quite clear. At first, only one of the independent pictures was clear after the ICA algorithm was applied. The other seemed to be a negative image(*i.e.* lightest areas appear darkest and the darkest areas appear lightest). After readjusting and scaling the picture, the picture became clear. ICA does not guarantee that the separated signals will be of the right scale (even the signs might be changed).[10]

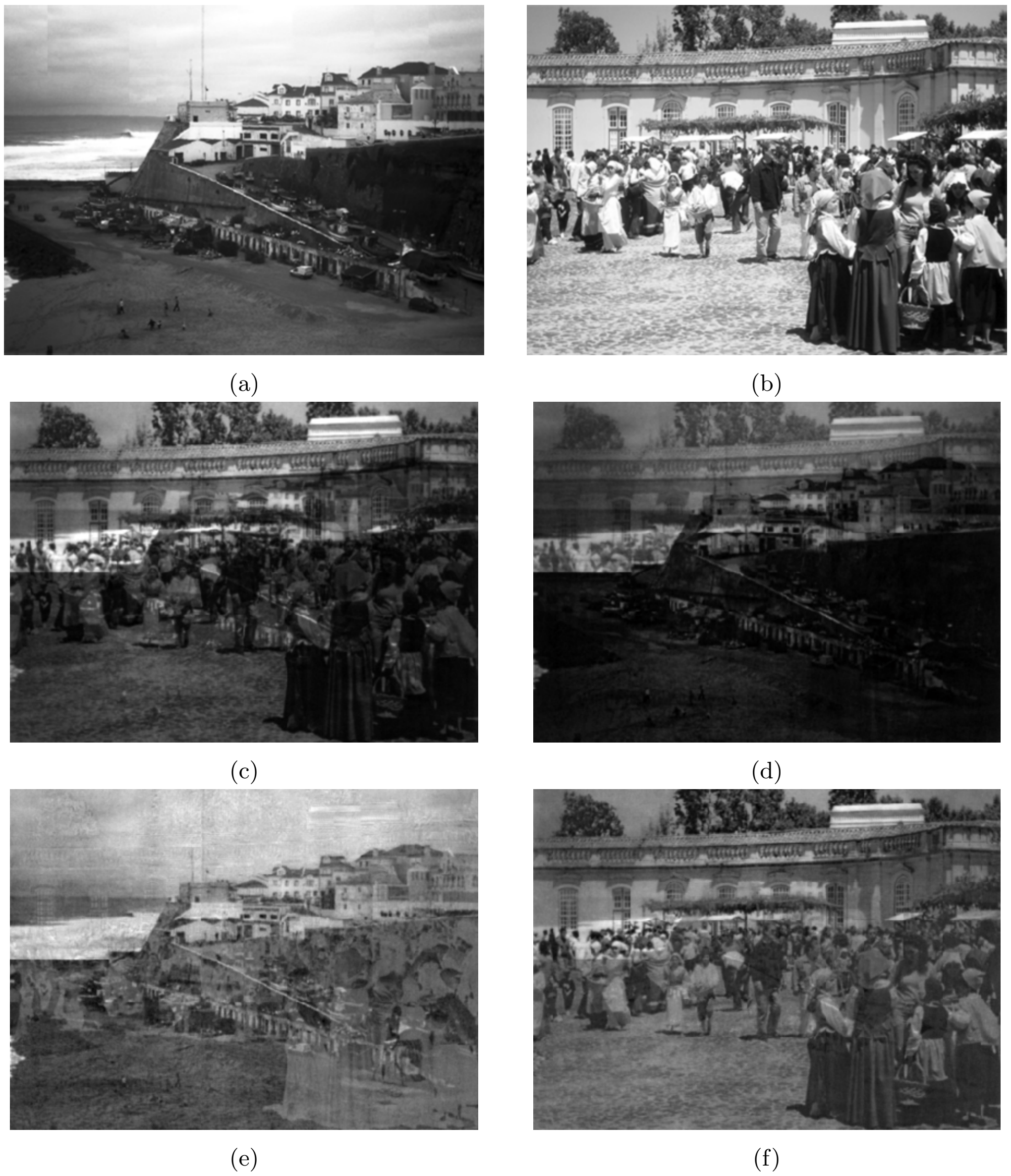


Figure 4: Original images (a) and (b). Mixed images (c) and (d). ICA was used to separate the individual images (e) and (f) from the mixed images.

Algorithm 2 Dimensionality Reduction by LLE

```
Input: Data $\mathbf{X} \in R^{m \times n}$, $d$ ($m =$ No. of data points, $n =$ No. of dimensions, $d =$ No. of dimensions to reduce to), $K =$ No. of neighbors
Output: $\mathbf{Y} \in R^{m \times d}$
1. Compute the K neighbors of each data point, $\mathbf{X}_i$.
2. Compute the weights $\mathbf{W}_{ij}$ that best reconstructs each data point $\mathbf{X}_i$ from its $K$ neighbors, minimizing the cost in Eq. (20) by constrained linear fits.
3. Compute the vectors $\mathbf{Y}_i$ best reconstructed by the weights $\mathbf{W}_{ij}$, minimizing the quadratic formula in Eq. (21) by its bottom nonzero eigenvectors.
```

5. LOCALLY LINEAR EMBEDDING

5.1 Overview

Locally Linear Embedding (LLE) is a nonlinear, unsupervised learning algorithm that takes advantage of local symmetries of linear reconstruction to embed high-dimensional data into a lower dimensional space while preserving local properties of the data. Although the data itself is assumed to lie on a nonlinear manifold, LLE assumes that each point and its k nearest neighbors lie on or close to a linear manifold at a local level. Since LLE preserves similarities at a local level, it is less sensitive to outliers than other nonlinear algorithms[17] because only a small amount of local properties are affected by outliers. The simplified idea is that LLE reconstructs each data point from a linear combination of its k nearest neighbors.

5.2 Procedure

The reconstruction errors are measured by the cost function:

$$\mathcal{E}(\mathbf{W}) = \sum_i \left| \mathbf{x}_i - \sum\nolimits_j \mathbf{W}_{ij}\mathbf{x}_j \right|^2 \tag{20}$$

Where the weights $\mathbf{W}_{ij}$ correspond to the contribution of the jth data point $\mathbf{x}_j$, to the ith transformed point $\mathbf{x}_i$. There are two constraints to finding the weights $\mathbf{W}_{ij}$: the weight for a particular data point can only be calculated using that point's k nearest neighbors (*i.e.* $w_{ij} = 0$ if $\mathbf{x}_j$ is not a neighbor or $\mathbf{x}_i$); and second, the sum of the rows of the weight matrix equal one (*i.e.* $\sum_j w_{ij} = 1$). A key point in LLE is that the reconstruction weights w_{ij} are invariant to transformations such as translation, rotation and scaling. This is important because we can then expect the weights $\mathbf{W}_{ij}$ that reconstruct the data in a high-dimensional space, to also maintain the transformation in a mapping to a lower dimensional space. By minimizing the embedding cost function:

$$\Phi(\mathbf{Y}) = \sum_i \left| \mathbf{y}_i - \sum\nolimits_j \mathbf{W}_{ij}\mathbf{y}_j \right|^2, \tag{21}$$

the d-dimensional coordinates $\mathbf{y}_i$ are chosen. Here, the weights $\mathbf{W}_{ij}$ are held fixed while the coordinates $\mathbf{y}_i$ are optimized. Roweis and Saul showed[17] that by computing the smallest d nonzero eigenvalues of the inproduct $(\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})$, the low-dimensional representations $\mathbf{Y}_j$ can be found efficiently giving substantial computational savings for large values of N, where N is the number of data points in the data and I is the $N \times N$ identity matrix.

5.3 Examples

Image processing is an area where LLE seems to produce promising results. For example, consider a dataset that contains images from a video of a subject making different facial expressions. Each image has 40 x 53 pixels which can be represented as a vector of 2120 pixel values. LLE was applied, using $k = 7$, revealing an interesting underlying structure on a 3-dimensional space. It is expected that data points close to each other in the lower space should have similar characteristics in the feature space. As seen in Figure 5, the green and red path on the graph are close to each other and correspond to the green and red facial frames. In other words, both red and green color-coded frames represent the movement of the head to the left. Since both facial movements are very similar to each other, it would be expected that corresponding frames would be mapped close to each other on the low dimensional space.

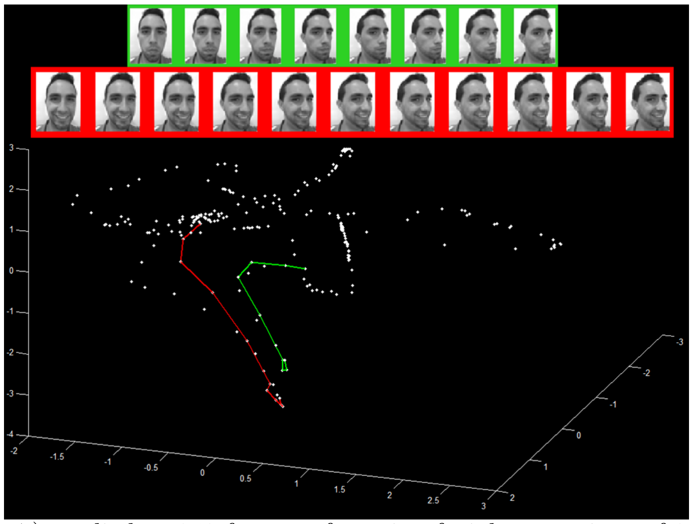


Figure 5: LLE ($K = 7$) applied to 950 frames of varying facial expressions of 2120 (40 x 53) pixels.

LLE has been found to lack in certain areas like in biomedical datasets.[14] One explanation for this might be that LLE does not do a good job when the manifold on the feature space is broken[17] (*i.e.* with holes, like the broken swiss-roll), thus giving an inaccurate mapping on the lower-dimensional space. One area in which LLE might have potential is in the facial expression recognition area.

6. KERNEL PRINCIPAL COMPONENT ANALYSIS

6.1 Overview

Kernel Principal Component Analysis (KPCA) is an nonlinear dimensionality reduction (NLDR) technique that extends Principal Component Analysis (PCA) using kernel functions. The kernel function is used to nonlinearly map the data into a higher-dimensional feature space F with a mapping Φ. Standard linear PCA is then performed on the data in the higher-dimensional feature space.[18] Instead of performing eigendecomposition on the covariance matrix and finding its principal eigenvectors, as in PCA, KPCA decomposes the kernel matrix and finds its principal eigenvectors.

Mapping the data nonlinearly to a higher-dimensional space allows the linear method of PCA to perform nonlinear mappings of the data.[23] Actually computing the mapping Φ can be difficult or even impossible because of the possibly high dimensionality of the kernel space. But in order for PCA to do the eigendecomposition of the data mapped by Φ to F, all we need are the dot products of pairs of mapped data points $(\Phi(x) \cdot \Phi(y))$, and for certain kernel functions $k(x, y)$ and mappings Φ it has been shown that $k(x, y) = (\Phi(x) \cdot \Phi(y))$.[18][19]

KPCA with a linear kernel makes it equivalent to traditional PCA; other possible kernels include Gaussian and polynomial.[23] Determination of the best kernel function for a particular problem is an open question.

6.2 Procedure

Assume we have a dataset $\mathbf{X}$, an $m \times n$ matrix where rows represent data points and columns represent variables/features. Using a chosen kernel function $k(x, y)$, *e.g.*the Gaussian kernel $k(x, y) = exp(-\|x - y\|^2 / 2\sigma^2)$, we compute the $m \times m$ kernel matrix $\mathbf{K}$:[18]

$$\mathbf{K}_{ij} = k(x_i, x_j) \tag{22}$$

Before proceeding to the decomposition of $\mathbf{K}$, we need to center the data, but since it lies in the feature space F this is not as straightforward as in PCA since we do not have the explicit form of the data. However, it can be done by performing

$$k_{ij} = -\frac{1}{2}(k_{ij} - \frac{1}{n}\sum_{l} k_{il} - \frac{1}{n}\sum_{l} k_{jl} + \frac{1}{n^2}\sum_{lm} k_{lm}) \tag{23}$$

for all entries k_{ij} in $\mathbf{K}$, where n is the number of data points.[23] We then calculate the eigenvector decomposition of $\mathbf{K}$ similarly to PCA:

$$\mathbf{K} = \mathbf{\Phi}\mathbf{\Delta}\mathbf{\Phi}^{-1} \tag{24}$$

where $\mathbf{\Phi}$ is a square $n \times n$ matrix whose columns contain the eigenvectors v_i of the modified $\mathbf{K}$, and $\mathbf{\Delta}$ is a diagonal matrix containing the eigenvalues λ_i corresponding to each eigenvector. The eigenvectors a_i of the covariance matrix in the feature space F can be computed from the v_is:[23]

$$a_i = \frac{1}{\sqrt{\lambda_i}} v_i \tag{25}$$

Finally, to compute the reduced version y_i of each original data point x_i from $\mathbf{X}$, we project the x_is on to the eigenvectors a_i of the covariance matrix:[23]

$$y_i = \left[\sum_{j=1}^{n} a_1^{(j)} k(x_j, x_i) \quad \sum_{j=1}^{n} a_2^{(j)} k(x_j, x_i) \quad \cdots \quad \sum_{j=1}^{n} a_d^{(j)} k(x_j, x_i) \right] \tag{26}$$

where a_l is the lth eigenvector of the covariance matrix and $a_l^{(j)}$ is the jth element of that eigenvector.

7. DIFFUSION MAPS

7.1 Overview

Diffusion Maps (DM) is a recently developed technique of nonlinear dimensionality reduction.[3] It attempts to discover the underlying structure of the data by finding the "diffusion distances" between points, and transform data points from their original high-dimensional space into a lower-dimensional space based on their diffusion distances. The underlying idea is that even if two points do not lie nearby according to Euclidean distance in their original high-dimensional data space, they may lie nearby along the surface of a low-dimensional manifold embedded in the high-dimensional space. If so, then if we take a random walk starting at the first point and successively jumping to nearby points, the probability that our path will lead along the surface of the manifold from the first point to the other is relatively high. The diffusion distance between two points is calculated based on the probability of random walks taking these paths connecting the two points. We consider all possible paths of a certain length and the probability of each of those paths occurring to determine the diffusion distance.

Take the Swiss Roll example shown previously (Figure 2), and consider three points: one of the dark red points, one of the orange points, and one of the light blue pointa. The Euclidean distance between the dark red point and the orange point is high, and the Euclidean distance between the dark red point and the light blue point is low. But along the surface of the manifold, the orange point is much closer to the red point than the light blue point is. DM attempts to discover this kind of relationship by considering paths that run from point to point along the surface of the manifold, connecting the red points to the orange points, then the green points, then the blue points, in that order. Although the distance between the red point and the blue point is small,

Algorithm 3 Dimensionality Reduction by KPCA

```
Input: Data $\mathbf{X} \in R^{m\times n}$, $d$ ($m$ = No. of data points, $n$ = No. of dimensions, $d$ = No. of dimensions to reduce to), $k(x, y)$ (kernel function)
Output: $\mathbf{Y} \in R^{m\times d}$
1. Initialize $m \times m$ kernel matrix $\mathbf{K}$
2. For all rows $x_i$ in $\mathbf{X}$:
3.    For all rows $x_j$ in $\mathbf{X}$:
4.       $\mathbf{K}_{ij} \leftarrow k(x_i, x_j)$
5.    End For
6. End For
7. For all entries $k_{ij}$ in $\mathbf{K}$:
8.    $k_{ij} \leftarrow -\frac{1}{2}(k_{ij} - \frac{1}{n}\sum_l k_{il} - \frac{1}{n}\sum_l k_{jl} + \frac{1}{n^2}\sum_{lm} k_{lm})$
9. Calculate eigendecomposition of $\mathbf{K}$: $\mathbf{K} = \mathbf{\Phi\Delta\Phi}^{-1}$ (eigenvectors $v_i$ in $\mathbf{\Phi}$, eigenvalues $\lambda_i$ on diagonal of $\mathbf{\Delta}$)
10. For all columns $v_i$ of $\mathbf{\Phi}$:
11.    $a_i \leftarrow \frac{1}{\sqrt{\lambda_i}} v_i$
12. End For
13. Initialize $m \times d$ reduced data matrix $\mathbf{Y}$
14. For all rows $y_i$ of $\mathbf{Y}$:
15.    $y_i \leftarrow \left[\sum_{j=1}^{n} a_1^{(j)} k(x_j, x_i) \quad \sum_{j=1}^{n} a_2^{(j)} k(x_j, x_i) \quad \cdots \quad \sum_{j=1}^{n} a_d^{(j)} k(x_j, x_i)\right]$
16. End For
```

the distance from the red point to another red point is much smaller, and the probability of a single jump from a red point to another red point is much higher than the probability of a jump from the red point to the blue point. By considering random walks made up of a series of these jumps from point to nearby point, DM can evaluate the structure of the manifold and determine that in the new representation, the orange point should be placed closer than the blue point to the red point.

Assume we have a dataset represented by $\boldsymbol{X}$, an $m \times n$ matrix where rows represent data points and columns represent variables/features. First we normalize the data by subtracting the minimum entry from $\boldsymbol{X}$ and dividing by the maximum entry, so values range from 0 to 1. Then we calculate the connectivities between points.

7.1.1 Connectivity

The probability of making any jump from one point to another in a single step of the random walk is called the connectivity of the two points. Determining the connectivity between each pair of points is the first step in calculating the diffusion distance. The connectivity between two points $\boldsymbol{x}$ and $\boldsymbol{y}$ is calculated based on a kernel function $k(\boldsymbol{x}, \boldsymbol{y})$. There are many possible kernel functions, some of which are based on Euclidean distance. As with KPCA, the choice of kernel function can significantly affect the quality of the results, but how to determine the best kernel function for a particular type of data remains an open question.[18][11] A few are presented below, based on those used by:[11]

$$\text{Gaussian kernel: } k(\boldsymbol{x}, \boldsymbol{y}) = exp(-\left\|\boldsymbol{x} - \boldsymbol{y}\right\|^2 / 2\sigma^2) \tag{27}$$

$$\text{Laplacian kernel: } k(\boldsymbol{x}, \boldsymbol{y}) = \frac{exp(\left|\left\|\boldsymbol{x} - \boldsymbol{y}\right\| - \mu\right| / b)}{2b} \tag{28}$$

$$\text{Rayleigh kernel: } k(\boldsymbol{x}, \boldsymbol{y}) = \frac{\left\|\boldsymbol{x} - \boldsymbol{y}\right\|^2 exp(-(\left\|\boldsymbol{x} - y\right\|^2)^2 / 2\sigma^2)}{\sigma^2} \tag{29}$$

$$\text{Polynomial kernel: } k(\boldsymbol{x}, \boldsymbol{y}) = (1 + \langle \boldsymbol{x}, \boldsymbol{y} \rangle)^d \tag{30}$$

$\boldsymbol{x}$ and $\boldsymbol{y}$ represent data points (rows) of the original $m \times n$ data matrix $\boldsymbol{X}$. All of these kernels satisfy the necessary conditions of symmetry and positivity preservation. The results of the kernel function applied to each pair of points is stored in an $m \times m$ kernel matrix $\boldsymbol{K}$ such that $\boldsymbol{K}_{ij} = k(\boldsymbol{x_i}, \boldsymbol{x_j})$. The actual connectivity between a pair of points, $p(\boldsymbol{x}, \boldsymbol{y})$, represents a probability, so the output of the kernel function must be normalized by the sum of all the outputs for each data point:

$$p(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{d_x} k(\boldsymbol{x}, \boldsymbol{y}) \tag{31}$$

where d_x is the sum of the row of $\boldsymbol{K}$ containing point $\boldsymbol{x}$. These probabilities are stored in the $m \times m$ matrix $\boldsymbol{P}$.

7.1.2 Diffusion Process

$\boldsymbol{P}$ is a Markov matrix containing the probabilities for a single transition from $\boldsymbol{x_i}$ to $\boldsymbol{x_j}$. We want to consider random walks consisting of multiple transitions, or timesteps. If t is the number of timesteps, then $\boldsymbol{P}^t$ is the Markov matrix containing the probabilities $p_t(\boldsymbol{x_i}, \boldsymbol{x_j})$ of a transition from $\boldsymbol{x_i}$ to $\boldsymbol{x_j}$ in a random walk of length t. Larger values of t allow us to consider larger neighborhoods of data points. By increasing the number of steps in the random walk, the probability of going from $\boldsymbol{x_i}$ to $\boldsymbol{x_j}$ increases if there are many high-probability paths from $\boldsymbol{x_i}$ to $\boldsymbol{x_j}$ along the surface of the manifold on which the data lies.

7.1.3 Diffusion Distance

The diffusion distance is a metric to measure the similarity of two points, defined based on the Markov matrix $\boldsymbol{P}$. It is defined such that if there are many high-probability paths of t steps between point $\boldsymbol{x}$ and point $\boldsymbol{y}$, the diffusion distance between them will be small, reflecting that they are "nearby" according to this metric.[5]

$$\begin{aligned} \boldsymbol{D}_t(\boldsymbol{x}, \boldsymbol{y})^2 &= \sum_{u \in \boldsymbol{X}} \left| p_t(\boldsymbol{X}_i, u) - p_t(\boldsymbol{X}_j, u) \right|^2 & (32) \\ &= \sum_k \left| \boldsymbol{P}^t_{ik} - \boldsymbol{P}^t_{kj} \right|^2 & (33) \end{aligned}$$

7.1.4 Diffusion Map

We would now like to transform our dataset according to the diffusion distances between points such that the Euclidean distance between two points is equal to their diffusion distance. The way we defined diffusion distance allows us to simply use the probability matrix $\boldsymbol{P}^t$ for this. Consider the ith row of $\boldsymbol{P}$, which contains the mapped point $\boldsymbol{x'_i}$ corresponding to the original point $\boldsymbol{x_i}$ from $\boldsymbol{X}$:

$$\boldsymbol{P}^t_i = \boldsymbol{x'_i} = \begin{bmatrix} p_t(\boldsymbol{x_i}, \boldsymbol{x_1}) & p_t(\boldsymbol{x_i}, \boldsymbol{x_2}) & \cdots & p_t(\boldsymbol{x_i}, \boldsymbol{x_m}) \end{bmatrix} \tag{34}$$

and similarly the jth row of $\boldsymbol{P}$, which contains the mapped point $\boldsymbol{x'_j}$ corresponding to the original point $\boldsymbol{x_j}$. The Euclidean distance between $\boldsymbol{x'_i}$ and $\boldsymbol{x'_j}$ is:

$$\begin{aligned} \left\| \boldsymbol{x'_i} - \boldsymbol{x'_j} \right\|^2_E &= \sum_{\boldsymbol{u} \in \boldsymbol{X}} \left| p_t(\boldsymbol{x_i}, \boldsymbol{u}) - p_t(\boldsymbol{x_j}, \boldsymbol{u}) \right|^2 & (35) \\ &= \sum_k \left| \boldsymbol{P}^t_{ik} - \boldsymbol{P}^t_{kj} \right|^2 & (36) \\ &= \boldsymbol{D}_t(\boldsymbol{x_i}, \boldsymbol{x_j}) & (37) \end{aligned}$$

Looking back to Eq. (32), we can see that this is the diffusion distance between $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$. $\boldsymbol{P}$ has m dimensions.

7.1.5 Reduction

Dimensionality reduction is now possible by neglecting some dimensions of the diffusion space. By performing an eigendecomposition of $\boldsymbol{P}$, we can calculate a lower-dimensional representation of the data that preserves the diffusion distances between points as well as possible. If we sort the eigenvectors ϕ and eigenvalues λ of $\boldsymbol{P}$ by eigenvalue (ignoring the trivial first eigenvector, whose eigenvalue will be 1), the mapped point $\boldsymbol{x'_j}$ can be expressed in reduced form as $\boldsymbol{x''_i}$ by the first d eigenvectors and eigenvalues:

$$\boldsymbol{x''_i} = \begin{bmatrix} \lambda_1^t \boldsymbol{\phi_1}^{(i)} & \lambda_2^t \boldsymbol{\phi_2}^{(i)} & \cdots & \lambda_d^t \boldsymbol{\phi_d}^{(i)} \end{bmatrix} \tag{38}$$

where $\boldsymbol{\phi_j}$ is the jth eigenvector of $\boldsymbol{P}$ and $\boldsymbol{\phi_j}^{(i)}$ is the ith element of that eigenvector. This also allows us to avoid calculating the full diffusion matrix $\boldsymbol{P}^t$, which would be computationally expensive. Instead, we just decompose $\boldsymbol{P}$ and raise its eigenvalues to t.

7.2 Procedure

Assume we have a dataset represented by $\boldsymbol{X}$, an $m \times n$ matrix where rows represent data points and columns represent variables/features. The first step is to normalize $\boldsymbol{X}$ by subtracting by the minimum entry and dividing by the maximum entry, so entries now vary between 0 and 1. We then create the kernel matrix $\boldsymbol{K}$ by applying the chosen kernel function to each pair of data points $(\boldsymbol{x_i}, \boldsymbol{x_j})$ and storing the result in $\boldsymbol{K}_{ij}$. We normalize $\boldsymbol{K}$ so the rows sum to 1 by dividing each row by its sum to arrive at the probability matrix $\boldsymbol{P}$. We then calculate the eigendecomposition of $\boldsymbol{P}$ (for a more detailed proof of why $\boldsymbol{P}$ can always be eigendecomposed and how to find its eigenvectors and eigenvectors, see section 8 of[5]) and store the eigenvectors in $\boldsymbol{\Phi}$ and the eigenvalues in $\boldsymbol{\Delta}$. As in PCA, we sort the eigenvectors in $\boldsymbol{\Phi}$ by their corresponding eigenvalues, then take only the first d eigenvectors and eigenvalues and discard the rest. We calculate the rows of the reduced dataset in the diffusion space, $\boldsymbol{Y}$, with Eq. (39):

$$\boldsymbol{Y} = \begin{bmatrix} \lambda_1^t \phi_1^{(1)} & \lambda_2^t \phi_2^{(1)} & \cdots & \lambda_d^t \phi_d^{(1)} \\ \lambda_1^t \phi_1^{(2)} & \lambda_2^t \phi_2^{(2)} & \cdots & \lambda_d^t \phi_d^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ \lambda_1^t \phi_1^{(m)} & \lambda_2^t \phi_2^{(m)} & \cdots & \lambda_d^t \phi_d^{(m)} \end{bmatrix} \tag{39}$$

7.3 Examples

DM was applied to the same data set that was used in LLE. The data set consists of images from a video of a subject making different facial expressions. Each image has 40 x 53 pixels which can be represented as a vector of 2120 pixel values. DM was applied, using $\sigma = 6$ and $t = 2$, and reduced to 3 dimensions. Although DM mapping seemed to produce a different structure from that of LLE, the characteristics on the embedding space seem to remain the same (*i.e.*points close together in the embedded space correspond to similar facial expressions). In the case of DM (Figure 6), the green and red path correspond to the head of the subject turning in opposite directions.

Another data set that was used to test DM consists of 500 discs (Figure 7), of 255 x 255 pixels each which can be represented as a vector of 65025 pixel values, rotated at different angles. DM was applied to this data set using $\sigma = 5$ and $t = 15$ and reduced to 2 dimensions. As seen in Figure 8, discs with similar rotation angles are mapped close to each other in the reduced diffusion space.

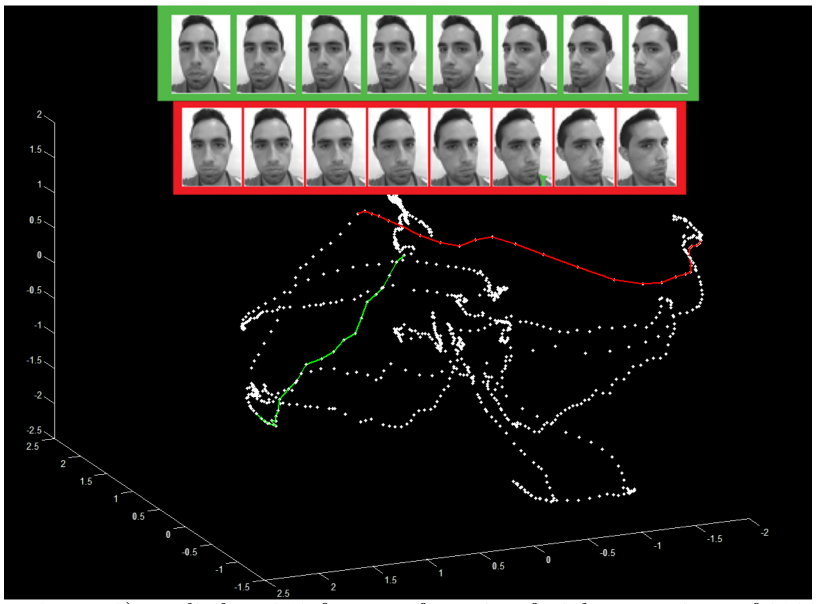


Figure 6: DM ($\sigma = 6$, $t = 2$) applied to 950 frames of varying facial expressions of 2120 (40 x 53) pixels.

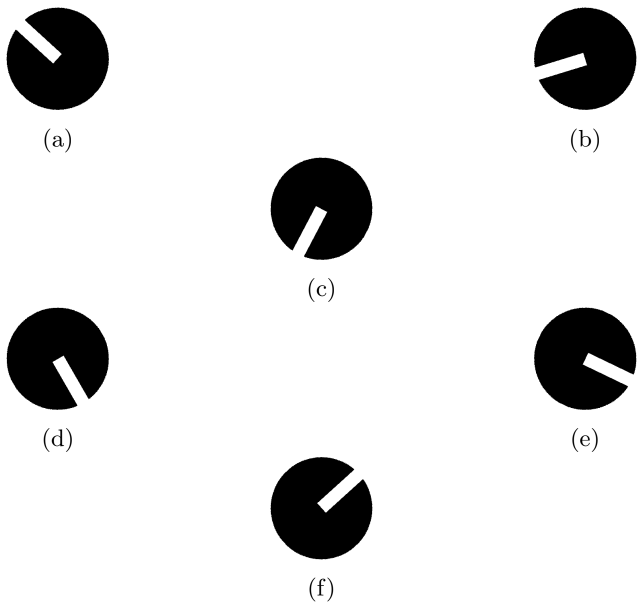


Figure 7: A few discs from the data set rotated at different angles.

Algorithm 4 Dimensionality Reduction by DM

```
Input: Data $\boldsymbol{X} \in R^{m\times n}$, $d$ ($m$ = No. of data points, $n$ = No. of dimensions, $d$ = No. of dimensions to reduce to), $k(\boldsymbol{x}, \boldsymbol{y})$ (kernel function)
Output: $\boldsymbol{Y} \in R^{m\times d}$
1. $min \leftarrow$ smallest entry in $\boldsymbol{X}$
2. $max \leftarrow$ largest entry in $\boldsymbol{X}$
3. For all entries $x_{ij}$ in $\boldsymbol{X}$:
4.    $x_{ij} \leftarrow (x_{ij} - min)/max$
5. End For
6. Initialize $m \times m$ kernel matrix $\boldsymbol{K}$
7. For all rows $\boldsymbol{x_i}$ in $\boldsymbol{X}$:
8.    For all rows $\boldsymbol{x_j}$ in $\boldsymbol{X}$:
9.       $\boldsymbol{K}_{ij} \leftarrow k(\boldsymbol{x_i}, \boldsymbol{x_j})$
10.   End For
11. End For
12. $\boldsymbol{rowsums} \leftarrow$ sum of each row in $\boldsymbol{K}$
13. Initialize $m \times m$ probability matrix $\boldsymbol{P}$
14. For all rows $\boldsymbol{p_i}$ in $\boldsymbol{P}$
15.    $p_i \leftarrow \boldsymbol{p_i} - \boldsymbol{rowsums}_i$
16. End For
17. $\boldsymbol{\Phi} \leftarrow$ eigenvectors of $\boldsymbol{P}$ (as column vectors), sorted by magnitude of corresponding eigenvalue
18. $\boldsymbol{\lambda} \leftarrow$ eigenvalues of $\boldsymbol{P}$ sorted by magnitude
19. $\boldsymbol{\Phi} \leftarrow$ first $d$ columns of $\boldsymbol{\Phi}$
20. $\boldsymbol{\lambda} \leftarrow$ first $d$ entries of $\Lambda$
21. Initialize $m \times d$ reduced data matrix $\boldsymbol{Y}$
21. For all entries $y_{ij}$ of $\boldsymbol{Y}$:
22.    $y_{ij} = \boldsymbol{\lambda}_j^t * \boldsymbol{\Phi}_{ij}$
23. End For
```

8. ANALYSIS OF DIAGNOSTIC BREAST CANCER DATA

The Wisconsin Diagnostic Breast Cancer (WDBC) dataset consists of 569 data points classified as either malignant or benign. Data was computed from a digitized image of a fine needle aspirate (FNA) of a breast mass.[25] Each instance contains 30 features describing different characteristics of the cell nuclei present in the image. The characteristics of the nucleus described are:[25]

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- concave points (number of concave portions of the contour)
- symmetry
- concavity (severity of concave portions of the contour)
- fractal dimension (“coastline approximation” - 1)
- compactness ($perimeter^2/area - 1.0$)

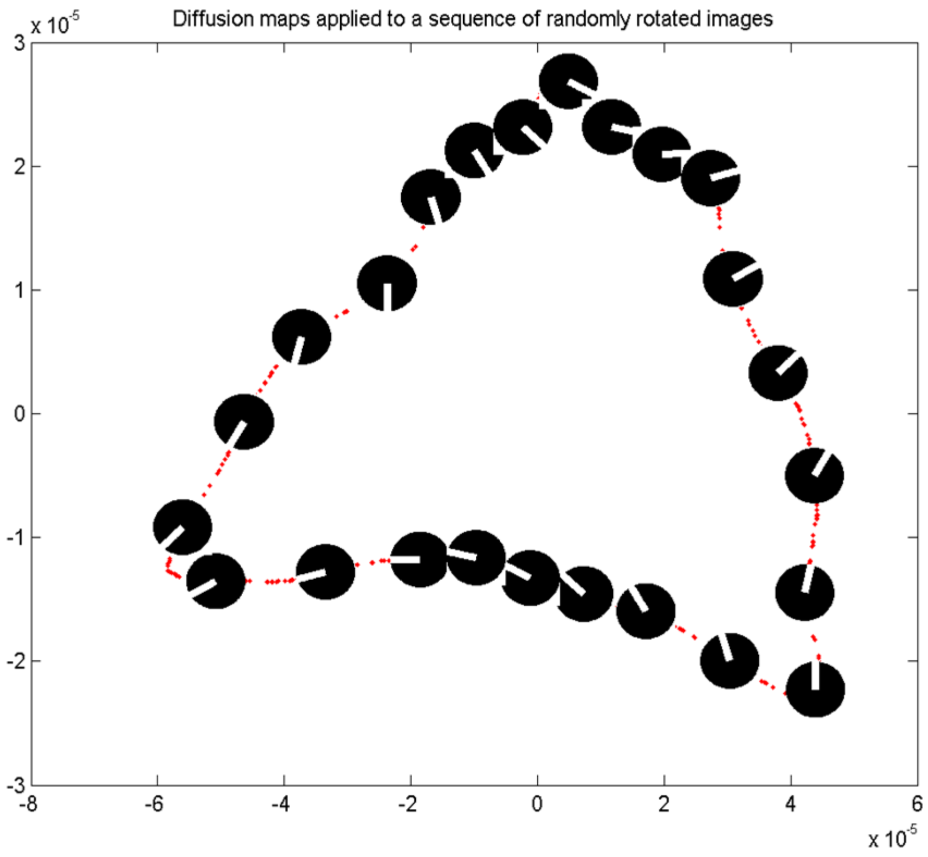


Figure 8: DM ($\sigma = 15$, $t = 5$) applied to 500 images of discs rotated at different angles. Images contain 65025 (255 x 255) pixels. This data was reduced to 2 dimensions.

For each of the above ten characteristics, the mean, standard error, and worst (largest) value is included, resulting in 30 features for every data point.

8.1 Results

We tested the classification accuracy of k-nearest neighbors (KNN) classification on the original data and the data after reducing it to 3-D, 2-D, and 1-D with diffusion maps.

Dimensions	**Accuracy**	**Sensitivity**	**Specificity**
30 (original)	0.933	0.874	0.968
30 to 3	0.928	0.852	0.973
30 to 2	0.913	0.837	0.959
30 to 1	0.908	0.835	0.952

Table 1: Classification accuracy, sensitivity, and specificity of KNN applied to original 30-dimensional WDBC data and WDBC data reduced with DM to 3, 2, and 1 dimensions.

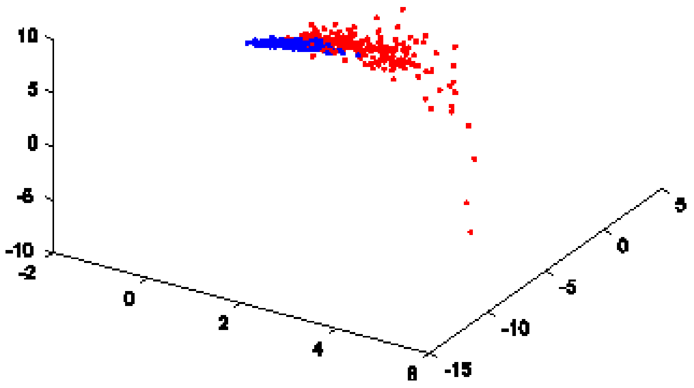


Figure 9: Data reduced to 3-D using DM with t=3, σ=1

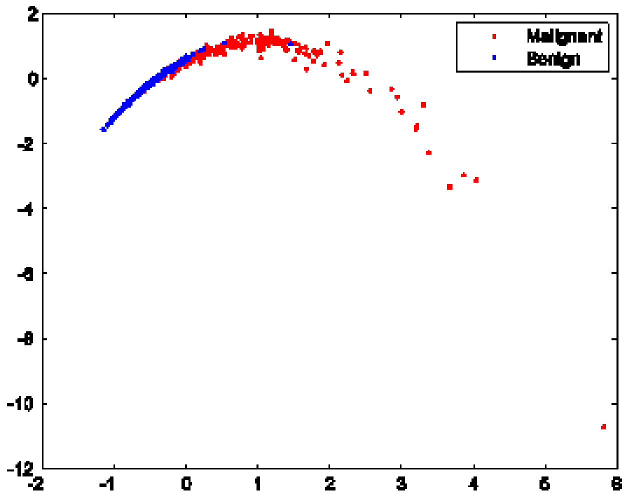


Figure 10: Data reduced to 2-D using DM with t=3, σ=1

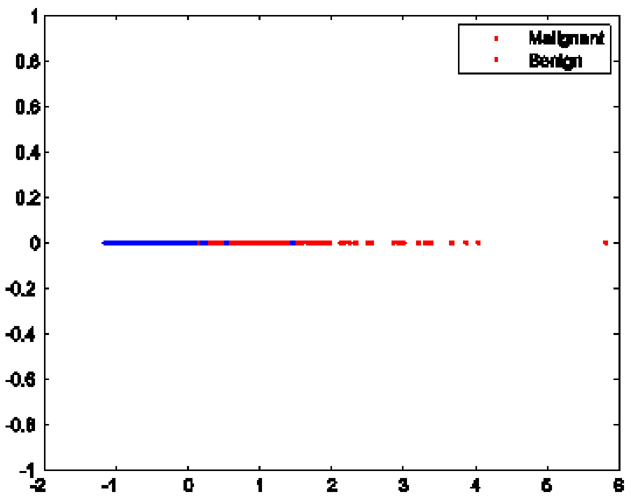


Figure 11: Data reduced to 1-D using DM with t=3, σ=1

8.2 Approaches and Conclusions

KNN was used to classify the data before and after applying DM. Before reducing the dimensionality of the data, KNN produced an accuracy of around 93%. As can be noted in Table 1, the accuracy drops as the number of dimensions reduced to decreases. One explanation could be that as the features are transformed, some instrinsic valued is lost in the mapping process. Therefore, it can be concluded that there is some tradeoff between accuracy and the amount of space and computation required to process the data at hand. In other words, by reducing the dimensionality of the data with DM, the accuracy of KNN decreases, but the amount of space required to process the data is fewer (recall the curse of dimensionality[8]).

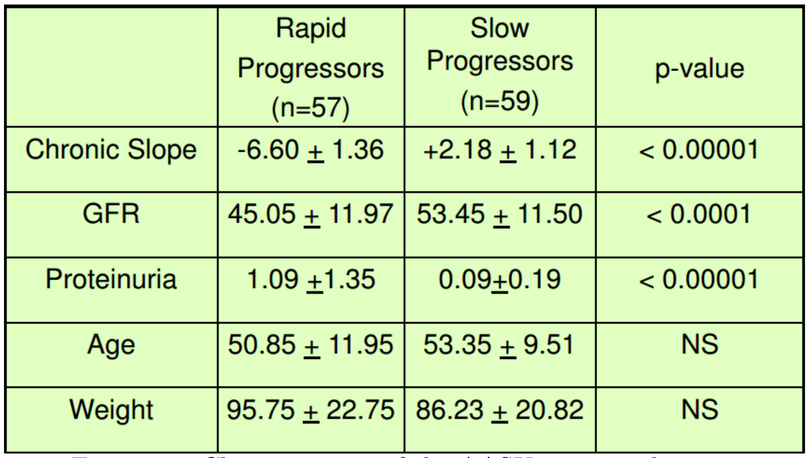
9. ANALYSIS OF KIDNEY PROTEOMIC DATA

The prevalence of Renal Disease in the USA exceeds 7 million people who suffer from stages 3-5 of Chronic Kidney Disease (CKD)[4] in the US. CKD is a progressive loss in renal function over a period of months or years. The severity of CKD is classified in five stages, with stage 1 being the mildest with usually few symptoms, to stage 5 being a severe illness with poor life expectancy. There is a high incidence of kidney disease in African Americans with hypertension.

We studied a dataset from the African American Study of Kidney Disease and Hypertension (AASK), consisting of 116 instances of 5251 features (courtesy: Dr. M. Lipkowitz of Georgetown University Hospital and Dr. M. Subasi of Florida Institue of Technology). Features correspond to serum proteomic levels (peaks extracted from raw SELDI-tof mass spectrometry data). All patients in the dataset suffer from CKD and are classified as either slow or fast progressors. The classification of slow and fast progressors (see Figure 12) is based on the rate of decline of Glomerular Filtration Rate (GFR), a measurement of kidney function.

9.1 Approaches

Compared to the WDBC dataset, this dataset was of much higher dimensionality and considerably more difficult to classify using standard methods. Standard KNN ($k = 3$) on the raw data produced a classification accuracy of about 57.8%, with sensitivity 55.1% and specificity 61.6%. We hoped to find a method of dimensionality reduction that would reduce the data to a more manageable size while also improving the performance of classification methods. We tried several reduction algorithms, including PCA, KPCA, LLE, and DM.



	Rapid Progressors (n=57)	Slow Progressors (n=59)	p-value
Chronic Slope	-6.60 ± 1.36	+2.18 ± 1.12	< 0.00001
GFR	45.05 ± 11.97	53.45 ± 11.50	< 0.0001
Proteinuria	1.09 ±1.35	0.09±0.19	< 0.00001
Age	50.85 ± 11.95	53.35 ± 9.51	NS
Weight	95.75 ± 22.75	86.23 ± 20.82	NS

Figure 12: Characteristics of the AASK patients data set

9.2 Results

Of the methods we applied, only PCA and DM were able to successfully reduce this very high-dimensional data; the algorithms we used for LLE and KPCA did not produce reliable results. The LLE algorithm returned fewer data points than were supplied; we found that the results returned by the KPCA method were strongly influenced by the order in which the data points were supplied, which we speculate may have been caused by the compounding of rounding error or other imprecisions in calculation. Between DM and PCA, classification with KNN on data reduced by {DM produced superior results, achieving accuracy up to approximately 64% depending on the dimension reduced to as well as the chosen kernel function and parameters. KNN after PCA achieved accuracy up to 58.4%, only marginally higher than KNN on the unreduced data. Tables showing results from DM using various kernels and parameters are shown below. Although the Gaussian kernel is commonly used for DM, we found that it was consistently outperformed by the Laplacian kernel for this dataset, at least for the parameters we tested. We were only able to test a few values for each parameter because of limited time and computational power; we believe, given more time, it would be possible to improve on these results by fine-tuning the parameters.

9.2.1 All Kernels

Dimensions	**Gaussian**	**Laplacian**	**Polynomial**
	$t = 10, \sigma = 1$	$t = 10$, $\mu = 10$, $b = 2$	$t = 6$, $d = 10$
5251 to 2	Acc: 0.454	**Acc: 0.635**	Acc: 0.502
	Se: 0.468	**Se: 0.648**	Se: 0.505
	Sp: 0.457	**Sp: 0.632**	Sp: 0.514
5251 to 3	Acc: 0.528	Acc: 0.564	Acc: 0.508
	Se: 0.557	Se: 0.544	Se: 0.534
	Sp: 0.512	Sp: 0.595	Sp: 0.497
5251 to 4	**Acc: 0.570**	Acc: 0.591	Acc: 0.536
	Se: 0.593	Se: 0.572	Se: 0.520
	Sp: 0.558	Sp: 0.623	Sp: 0.561
5251 to 5	Acc: 0.533	Acc: 0.576	Acc: 0.580
	Se: 0.556	Se: 0.523	Se: 0.559
	Sp: 0.522	Sp: 0.640	Sp: 0.611
5251 to 6	Acc: 0.526	Acc: 0.523	Acc: 0.584
	Se: 0.586	Se: 0.476	Se: 0.547
	Sp: 0.479	Sp: 0.581	Sp: 0.634
5251 to 7	Acc: 0.451	Acc: 0.534	Acc: 0.597
	Se: 0.485	Se: 0.514	Se: 0.586
	Sp: 0.431	Sp: 0.567	Sp: 0.617
5251 to 8	Acc: 0.468	Acc: 0.570	**Acc: 0.606**
	Se: 0.501	Se: 0.547	**Se: 0.614**
	Sp: 0.448	Sp: 0.606	**Sp: 0.610**

Table 2: Classification accuracy, sensitivity, and specificity of KNN applied to AASK data reduced by DM with various kernels. Each kernel is shown with the parameters that produce the best classification accuracy in any dimension, and the best results achieved by each kernel are bolded.

9.2.2 Gaussian Kernel

Dimensions	$\sigma = 0.1$	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 5$
5251 to 2	Acc: 0.512	Acc: 0.490	Acc: 0.530	Acc: 0.454
	Se: 0.495	Se: 0.464	Se: 0.512	Se: 0.468
	Sp: 0.540	Sp: 0.530	Sp: 0.561	Sp: 0.457
5251 to 3	Acc: 0.505	Acc: 0.438	Acc: 0.481	Acc: 0.528
	Se: 0.484	Se: 0.429	Se: 0.481	Se: 0.557
	Sp: 0.537	Sp: 0.463	Sp: 0.497	Sp: 0.512
5251 to 4	Acc: 0.524	Acc: 0.506	Acc: 0.549	Acc: 0.570
	Se: 0.517	Se: 0.498	Se: 0.564	Se: 0.593
	Sp: 0.542	Sp: 0.529	Sp: 0.549	Sp: 0.558
5251 to 5	Acc: 0.502	Acc: 0.494	Acc: 0.491	Acc: 0.533
	Se: 0.498	Se: 0.477	Se: 0.503	Se: 0.556
	Sp: 0.520	Sp: 0.523	Sp: 0.490	Sp: 0.522
5251 to 6	Acc: 0.505	Acc: 0.506	Acc: 0.491	Acc: 0.526
	Se: 0.501	Se: 0.496	Se: 0.477	Se: 0.586
	Sp: 0.520	Sp: 0.529	Sp: 0.516	Sp: 0.479
5251 to 7	Acc: 0.496	Acc: 0.496	Acc: 0.501	Acc: 0.451
	Se: 0.504	Se: 0.517	Se: 0.504	Se: 0.485
	Sp: 0.499	Sp: 0.482	Sp: 0.509	Sp: 0.431
5251 to 8	Acc: 0.494	Acc: 0.568	Acc: 0.512	Acc: 0.468
	Se: 0.491	Se: 0.569	Se: 0.529	Se: 0.501
	Sp: 0.508	Sp: 0.581	Sp: 0.510	Sp: 0.448

Table 3: Classification accuracy, sensitivity, and specificity of KNN applied to AASK data reduced by DM with the Gaussian kernel for $t = 10$ and various values of the parameter σ.

9.2.3 Laplacian Kernel

Dimensions	$\mu = 0.1,\ b = 0.5$	$\mu = 0.1,\ b = 2$	$\mu = 0.1,\ b = 8$
5251 to 2	Acc: 0.427	Acc: 0.569	Acc: 0.534
	Se: 0.441	Se: 0.608	Se: 0.539
	Sp: 0.428	Sp: 0.541	Sp: 0.540
5251 to 3	Acc: 0.446	Acc: 0.578	Acc: 0.580
	Se: 0.476	Se: 0.624	Se: 0.570
	Sp: 0.429	Sp: 0.545	Sp: 0.603
5251 to 4	Acc: 0.528	Acc: 0.537	Acc: 0.561
	Se: 0.588	Se: 0.565	Se: 0.565
	Sp: 0.482	Sp: 0.523	Sp: 0.569
5251 to 5	Acc: 0.561	Acc: 0.492	Acc: 0.560
	Se: 0.602	Se: 0.550	Se: 0.564
	Sp: 0.532	Sp: 0.450	Sp: 0.565
5251 to 6	Acc: 0.571	Acc: 0.516	Acc: 0.611
	Se: 0.621	Se: 0.585	Se: 0.637
	Sp: 0.531	Sp: 0.459	Sp: 0.596
5251 to 7	Acc: 0.532	Acc: 0.544	Acc: 0.549
	Se: 0.487	Se: 0.538	Se: 0.558
	Sp: 0.588	Sp: 0.560	Sp: 0.554
5251 to 8	Acc: 0.468	Acc: 0.506	Acc: 0.512
	Se: 0.546	Se: 0.554	Se: 0.537
	Sp: 0.406	Sp: 0.470	Sp: 0.502

Table 4: Classification accuracy, sensitivity, and specificity of KNN applied to AASK data reduced by DM with the Laplacian kernel for $t = 10$, $\mu = 0.1$ and various values of the parameter b.

Dimensions	$\mu = 0.5$, $b = 0.5$	$\mu = 0.5$, $b = 2$	$\mu = 0.5$, $b = 8$
5251 to 2	Acc: 0.474	Acc: 0.574	Acc: 0.466
	Se: 0.458	Se: 0.579	Se: 0.518
	Sp: 0.502	Sp: 0.579	Sp: 0.426
5251 to 3	Acc: 0.530	Acc: 0.580	Acc: 0.509
	Se: 0.556	Se: 0.645	Se: 0.515
	Sp: 0.518	Sp: 0.526	Sp: 0.516
5251 to 4	Acc: 0.553	Acc: 0.530	Acc: 0.543
	Se: 0.542	Se: 0.562	Se: 0.544
	Sp: 0.575	Sp: 0.511	Sp: 0.556
5251 to 5	Acc: 0.553	Acc: 0.509	Acc: 0.527
	Se: 0.614	Se: 0.552	Se: 0.535
	Sp: 0.507	Sp: 0.478	Sp: 0.532
5251 to 6	Acc: 0.505	Acc: 0.511	Acc: 0.508
	Se: 0.520	Se: 0.549	Se: 0.498
	Sp: 0.504	Sp: 0.483	Sp: 0.530
5251 to 7	Acc: 0.561	Acc: 0.585	Acc: 0.507
	Se: 0.540	Se: 0.570	Se: 0.526
	Sp: 0.592	Sp: 0.610	Sp: 0.504
5251 to 8	Acc: 0.474	Acc: 0.563	Acc: 0.482
	Se: 0.556	Se: 0.599	Se: 0.455
	Sp: 0.407	Sp: 0.539	Sp: 0.524

Table 5: Classification accuracy, sensitivity, and specificity of KNN applied to AASK data reduced by DM with the Laplacian kernel for $t = 10$, $\mu = 0.5$ and various values of the parameter b.

Dimensions	$\mu = 1$, $b = 0.5$	$\mu = 1$, $b = 2$	$\mu = 1$, $b = 8$
5251 to 2	Acc: 0.445	Acc: 0.547	Acc: 0.515
	Se: 0.428	Se: 0.608	Se: 0.547
	Sp: 0.477	Sp: 0.500	Sp: 0.498
5251 to 3	Acc: 0.529	Acc: 0.513	Acc: 0.467
	Se: 0.509	Se: 0.550	Se: 0.446
	Sp: 0.560	Sp: 0.486	Sp: 0.500
5251 to 4	Acc: 0.555	Acc: 0.551	Acc: 0.470
	Se: 0.543	Se: 0.571	Se: 0.441
	Sp: 0.575	Sp: 0.544	Sp: 0.510
5251 to 5	Acc: 0.508	Acc: 0.506	Acc: 0.518
	Se: 0.519	Se: 0.526	Se: 0.486
	Sp: 0.510	Sp: 0.497	Sp: 0.561
5251 to 6	Acc: 0.496	Acc: 0.532	Acc: 0.540
	Se: 0.552	Se: 0.545	Se: 0.508
	Sp: 0.457	Sp: 0.532	Sp: 0.582
5251 to 7	Acc: 0.515	Acc: 0.573	Acc: 0.556
	Se: 0.558	Se: 0.562	Se: 0.529
	Sp: 0.482	Sp: 0.598	Sp: 0.592
5251 to 8	Acc: 0.501	Acc: 0.577	Acc: 0.543
	Se: 0.492	Se: 0.563	Se: 0.506
	Sp: 0.518	Sp: 0.603	Sp: 0.595

Table 6: Classification accuracy, sensitivity, and specificity of KNN applied to AASK data reduced by DM with the Laplacian kernel for $t = 10$, $\mu = 1$ and various values of the parameter b.

Dimensions	$\mu = 10$, $b = 0.5$	$\mu = 10$, $b = 2$	$\mu = 10$, $b = 8$
5251 to 2	Acc: 0.515	Acc: 0.635	Acc: 0.569
	Se: 0.523	Se: 0.648	Se: 0.562
	Sp: 0.521	Sp: 0.632	Sp: 0.588
5251 to 3	Acc: 0.602	Acc: 0.564	Acc: 0.542
	Se: 0.569	Se: 0.544	Se: 0.518
	Sp: 0.645	Sp: 0.595	Sp: 0.573
5251 to 4	Acc: 0.485	Acc: 0.591	Acc: 0.558
	Se: 0.465	Se: 0.572	Se: 0.519
	Sp: 0.515	Sp: 0.623	Sp: 0.608
5251 to 5	Acc: 0.471	Acc: 0.576	Acc: 0.538
	Se: 0.499	Se: 0.523	Se: 0.562
	Sp: 0.455	Sp: 0.640	Sp: 0.525
5251 to 6	Acc: 0.542	Acc: 0.523	Acc: 0.511
	Se: 0.532	Se: 0.476	Se: 0.543
	Sp: 0.566	Sp: 0.581	Sp: 0.492
5251 to 7	Acc: 0.515	Acc: 0.534	Acc: 0.532
	Se: 0.574	Se: 0.514	Se: 0.535
	Sp: 0.470	Sp: 0.567	Sp: 0.542
5251 to 8	Acc: 0.469	Acc: 0.570	Acc: 0.575
	Se: 0.522	Se: 0.547	Se: 0.572
	Sp: 0.430	Sp: 0.606	Sp: 0.591

Table 7: Classification accuracy, sensitivity, and specificity of KNN applied to AASK data reduced by DM with the Laplacian kernel for $t = 10$, $\mu = 10$ and various values of the parameter b.

9.2.4 Polynomial Kernel

Dimensions	$d = 1$	$d = 2$	$d = 5$	$d = 10$
5251 to 2	Acc: 0.555	Acc: 0.513	Acc: 0.540	Acc: 0.502
	Se: 0.548	Se: 0.533	Se: 0.615	Se: 0.505
	Sp: 0.577	Sp: 0.507	Sp: 0.477	Sp: 0.514
5251 to 3	Acc: 0.473	Acc: 0.480	Acc: 0.494	Acc: 0.508
	Se: 0.471	Se: 0.486	Se: 0.495	Se: 0.534
	Sp: 0.488	Sp: 0.489	Sp: 0.507	Sp: 0.497
5251 to 4	Acc: 0.532	Acc: 0.544	Acc: 0.538	Acc: 0.536
	Se: 0.570	Se: 0.575	Se: 0.529	Se: 0.520
	Sp: 0.508	Sp: 0.530	Sp: 0.558	Sp: 0.561
5251 to 5	Acc: 0.509	Acc: 0.540	Acc: 0.561	Acc: 0.580
	Se: 0.533	Se: 0.563	Se: 0.578	Se: 0.559
	Sp: 0.500	Sp: 0.532	Sp: 0.558	Sp: 0.611
5251 to 6	Acc: 0.531	Acc: 0.555	Acc: 0.547	Acc: 0.584
	Se: 0.562	Se: 0.576	Se: 0.553	Se: 0.547
	Sp: 0.512	Sp: 0.546	Sp: 0.551	Sp: 0.634
5251 to 7	Acc: 0.565	Acc: 0.586	Acc: 0.585	Acc: 0.597
	Se: 0.581	Se: 0.601	Se: 0.577	Se: 0.586
	Sp: 0.560	Sp: 0.585	Sp: 0.608	Sp: 0.617
5251 to 8	Acc: 0.579	Acc: 0.583	Acc: 0.582	Acc: 0.606
	Se: 0.648	Se: 0.596	Se: 0.578	Se: 0.614
	Sp: 0.525	Sp: 0.581	Sp: 0.597	Sp: 0.610

Table 8: Classification accuracy, sensitivity, and specificity of KNN applied to AASK data reduced by DM with the Polynomial kernel for $t = 6$ and various values of the parameter d.

10. CONCLUSIONS

NLDR techniques such as DM can effectively transform highdimensional medical data to as few as one dimension while maintaining its important characteristics, allowing us to classify data points with similar accuracy as on the raw data. This saves computational time and space.

DR methods involve feature transformation. Therefore, in most cases an interpretation of the reduced data may not be possible in terms of the original features. However, certain important biomarkers could be devised based on the reduced data.

Classification of certain data can be complicated due to its intrinsic features. DR methods can improve results of classification algorithms.

Future work: study the DR methods on kidney datasets to evolve suitable biomarkers.

ACKNOWLEDGMENTS

The authors acknowledge support from National Science Foundation (NSF) grant No. 1263011. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

The authors would also like to acknowledge help from Dr. M. Subasi and Dr. M. Lipkowitz for providing the African American Study of Kidney Disease and Hypertension (AASK) dataset. Also, Dr. Koksal, chair of Math department for allowing the use of math lab.

REFERENCES

[1] URL: http://www.lx.it.pt/~lbalmeida/ica/seethrough/index.html#download.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. doi:10.1109/TPAMI.2013.50.

[3] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[4] Josef Coresh, Brad C Astor, Tom Greene, Garabed Eknoyan, and Andrew S Levey. Prevalence of chronic kidney disease and decreased kidney function in the adult us population: Third national health and nutrition examination survey. *American Journal of Kidney Diseases*, 41(1):1–12, 2003.

[5] J de la Porte, BM Herbst, W Hereman, and SJ van der Walt. An introduction to diffusion maps. *Applied Mathematics Division, Department of Mathematical Sciences, University of Stellenbosch, South Africa and Colorado School of Mines, United States of America*, 2008.

[6] Duncan F. Gillies. Lecture 15: Principal component analysis. URL: http://www.doc.ic.ac.uk/~dfg/ProbabilisticInference/IDAPILecture15.pdf.

[7] Jihun Ham, Daniel D Lee, Sebastian Mika, and Bernhard Schlkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the twenty-first international conference on Machine learning*, page 47. ACM, 2004.

[8] Melanie Hilario and Alexandros Kalousis. Approaches to dimensionality reduction in proteomic biomarker studies. *Oxford Journals*, 9:102–118, 2008. doi:10.1093/bib/bbn005.

[9] Peter J Huber. Projection pursuit. *The annals of Statistics*, pages 435–475, 1985.

[10] A. Hyvrinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(45):411 – 430, 2000. URL: http://www.sciencedirect.com/science/article/pii/S0893608000000265, doi:10.1016/S0893-6080(00)00026-5.

[11] J.C. Isaacs. Diffusion map kernel analysis for target classification. In *OCEANS 2009, MTS/IEEE Biloxi - Marine Technology for Our Future: Global and Local Challenges*, pages 1–7, 2009.

[12] J. Karhunen, E. Oja, L. Wang, R. Vigario, and J. Joutsensalo. A class of neural networks for independent component analysis. *Neural Networks, IEEE Transactions on*, 8(3):486–504, 1997. doi:10.1109/72.572090.

[13] Erwin Kreyszig. *Advanced Engineering Mathematics*, chapter Data Analysis. Probability Theory, page 1014. Wiley Press, 2011.

[14] Ik-Soo Lim, P. de Heras Ciechomski, S. Sarni, and D. Thalmann. Planar arrangement of high-dimensional biomedical data sets by isomap coordinates. In *Computer-Based Medical Systems, 2003. Proceedings. 16th IEEE Symposium*, pages 50–55, 2003. doi:10.1109/CBMS.2003.1212766.

[15] Julien Prados, Alexandros Kalousis, and Melanie Hilario. On preprocessing of seldi-ms data and its evaluation. In *Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on*, pages 953–958. IEEE, 2006.

[16] JL Richardson and RJ Bigler. Principal component analysis of prairie pothole soils in north dakota. *Soil Science Society of America Journal*, 48(6):1350–1355, 1984.

[17] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. URL: http://www.sciencemag.org/content/290/5500/2323.abstract, arXiv:http://www.sciencemag.org/content/290/5500/2323.full.pdf, doi:10.1126/science.290.5500.2323.

[18] Bernhard Schlkopf, Alexander Smola, and Klaus-Robert Mller. Kernel principal component analysis. In *Artificial Neural NetworksICANN'97*, pages 583–588. Springer, 1997.

[19] Bernhard Schlkopf, Alexander Smola, and Klaus-Robert Mller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[20] Jonathon Shlens. A tutorial on principal component analysis. *Systems Neurobiology Laboratory, University of California at San Diego*, 2005.

[21] Lindsay I Smith. A tutorial on principal components analysis. *Cornell University, USA*, 51:52, 2002.

[22] K. Thangavel and A. Pethalakshmi. Dimensionality reduction based on rough set theory: A review. *Applied Soft Computing*, 9(1):1 – 12, 2009. URL: http://www.sciencedirect.com/science/article/pii/S1568494608000963, doi:http://dx.doi.org/10.1016/j.asoc.2008.05.006.

[23] LJP Van der Maaten, EO Postma, and HJ Van Den Herik. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10:1–41, 2009.

[24] Juan Flix vila Herrera. *Mixed Integer Linear Programming Based Implementations of Logical Analysis of Data and Its Applications*. PhD thesis, Florida Institute of Technology, 2013.

[25] William H. Wolberg, W. Nick Street, and Olvi L. Mangasarian. Breast cancer Wisconsin (diagnostic) data set. URL: http://archive.ics.uci.edu/ml/.