Linear and Non-linear Dimension Reduction Applied to Gene Expression Data of Cancer Tissue Samples

Franck Olivier Ndjakou Njeunje

Applied Mathematics, Statistics, and Scientific Computation
University of Maryland - College Park
fndjakou@math.umd.edu

Advisers
Wojtek Czaja
John J. Benedetto

Norbert Wiener Center for Harmonic Analysis
Department of Mathematics
University of Maryland - College Park

December 15, 2014

Abstract

In computational biology and medicine, gene expression data are a very useful and important piece of the puzzle as they are one of the main source from which are derived gene functions and various disease mechanisms. Unfortunately, the analysis, classification and visualization of gene expression data is not an easy task due to the high dimension of the data generated from high-density microarrays. In this project, we will be interested in two methods developed to carry dimension reduction on such data so that they will become better suited for further analysis. It is our belief that a non-linear approach to dimension reduction would perform better than a linear approach in preserving the internal structure of the data. The goal of this project is to be able to demonstrate the effectiveness of non-linear versus linear dimension reduction algorithm in capturing biologically relevant structures in cancer cell expression dataset. In particular we will be working with Laplacian Eigenmaps and Principal Components Analysis as our non-linear and linear dimension reduction methods respectively and the NCI60 as our cancer cell expression dataset.
Contents

1 Background and Motivation .................................................. 4
  1.1 Gene Expression Data .................................................... 4
  1.2 Dimension Reduction ................................................... 4
  1.3 Clustering ................................................................. 5

2 Approach ........................................................................... 5
  2.1 Principal Component Analysis [1] ...................................... 6
    2.1.1 Shifting and Scaling ................................................. 6
    2.1.2 Principal Components .............................................. 8
    2.1.3 Power Iteration [11] .................................................. 8
    2.1.4 Choosing $m$ .......................................................... 9
  2.2 Laplacian Eigenmaps [2] .................................................. 10
  2.3 Hierarchical Clustering ................................................... 10
  2.4 K-means clustering ....................................................... 11

3 The Data ............................................................................. 12
  3.1 Genotype and Phenotype .................................................. 12
  3.2 DNA Microarray ............................................................ 13
  3.3 The Main data: NCI-60 [3, p. 95-96] .................................. 14

4 Implementation and Validation methods .............................. 15
  4.1 Software and hardware .................................................... 15
  4.2 Validation methods ........................................................ 15
  4.3 The rand index [12] ......................................................... 16

5 Test Problem ...................................................................... 17
  5.1 Helix ............................................................................ 17
    5.1.1 Principal Components Analysis ................................ 17
  5.2 Twin Peaks .................................................................... 18
    5.2.1 Principal Components Analysis ................................. 19

6 Analysis on NCI-60 .......................................................... 19
  6.1 Principal Component Analysis ........................................ 19

7 Results .............................................................................. 20

8 Timeline ............................................................................ 21

9 Deliverable ....................................................................... 22
Notation

\( x \) Gene or row of dimension \( M \) in the matrix \( X \).
\( c \) Columns of the matrix \( X \).
\( X \) Gene expression matrix of dimension \( N \times M \).
\( Y \) Reduced gene expression matrix of dimension \( N \times m \).
\( \hat{X} \) Standardized matrix of the matrix \( X \).
\( N \) Number of data points contained in the matrix \( X \).
\( M \) Dimension of each data points \( x \) before dimension reduction.
\( m \) Dimension of each data points \( y \) after dimension reduction.
\( y \) Reduced dimension data of dimension \( m \).
\( \bar{x}_i \) Mean of the vector \( x_i \).
\( \sigma_{ii} \) Variance of the vector \( x_i \).
\( C \) Covariance matrix.
\( \Lambda \) Diagonal matrix containing the eigenvalues, \( \lambda_i \), of the covariance matrix.
\( U \) Matrix containing the eigenvectors, \( u_i \), of the covariance matrix.
\( W \) Weight matrix.
\( L \) Laplacian matrix.
\( D \) Diagonal or degree matrix.
\( u_i \) and \( f_i \) Eigenvectors.
\( \lambda_i \) Eigenvalues.
\( m_j \) Means for \( 1 \leq j \leq k \) where \( k \) is the number of means.
\( S^{(t)}_i \) sets or clusters.
1. Background and Motivation

1.1. Gene Expression Data

Gene expression data are information that numerically represent the expression level of a set of genes due to environmental factors. These environment factors could be of natural cause such as the effect of cancer or any other diseases on a set of genes; or they could be reaction to drugs or medicines taken to fight said diseases. The data are usually given in matrix form, let’s call this matrix $X$, in order to obtain the gene expression matrix $X$, the microarray chip is scanned by a computer to numerically determine the level of expression of a set of genes over multiple samples or observations. The matrix $X$ has dimension $(N \times M)$ where the number of genes is given by the variable $N$ and the number of samples is given by the variable $M$.

Due to the usefulness of the gene expression data, a wide range of algorithms have been developed to study the biological network provided by high-density microarrays. The main ones are classification and clustering techniques. It has been shown that classification of gene expression data could help us distinguish between various cancer classes; while clustering techniques could help separate tumor from healthy normal tissues. Unfortunately, the number of observations or samples, $M$, is in general very high and this makes it difficult to visualize the results from the similarity learning analysis. Therefore in order to determine the structure of those data in the hope of getting more information from them, whether to classify them as genes of the same kind based on their expression or to visually separate healthy ones from unhealthy ones a dimension reduction algorithm is necessary as a pre-processing step.

1.2. Dimension Reduction

By taking a closer look at the data, in figure 1 we notice that within each expression array $x$, across the multiple samples, a lot of redundancy can be found in the data. This will provide us with a platform allowing us to do some pre-processing on the data in order to retain only the most pertinent information. The methods used in this part of the analysis are known as dimension reduction techniques, this will be our focus throughout this year long project. Given an array $x$ of dimension $M$ we would like to be able to reduce this array to an $m$-dimensional array $y$ such that $m$ is very small compare to $M$, while retaining the most important information about the array across all the samples.

There are two classes of dimension reduction techniques: linear (LDR) and non-linear (NDR). The linear techniques assume a linear relationship between the data and does well when that assumption is met. The problem we are facing is that most data that arise from gene expression do not entirely have a linear relationship and so to remedy to this, non-linear methods have been developed. The advantage here is that non-linear methods aim to preserve the intrinsic or natural geometrical structure between the variables or data points. Before and after this step is completed a similarity learning analysis known as clustering is applied to the data in order to get an idea of how much of the structure within the data was preserved.
1.3. Clustering

In order to determine just how well the dimension reduction techniques applied to the data preserve the internal structure of the data, we will first need a way of defining or getting a sense of this structure residing within the data. Now the true and natural structure within the expression data could be a difficult and complex relation to establish if one does not have a solid background in a biological related field. Instead, a clustering analysis will be performed on the data before and after dimension reduction to get a pseudo-structure on the gene expression. The goal of clustering is to group elements of a set in separate subgroups called clusters in such a way that elements in the same cluster are more similar than elements in different clusters in one way or another. In practice, different clustering methods perform differently based on the nature of data they are applied to.

2. Approach

For this project I will be interested in Principal Component Analysis also known as PCA as my linear dimension reduction method, which is the most common linear dimension reduction method used in the analysis of gene expression data. As a contestant, I will also look at Laplacian Eigenmaps abbreviated as LE as my non-linear dimension reduction method. I am interested in how the output from the dimension reduction algorithms listed above compares structurally to the original input. This will be done using similarity learning such as Hierarchical clustering and K-means clustering on the data before and after dimension reduction, then an analysis of the resulting clusters will give us a sense of how much of the structure within the data was preserved. The subsections below gives us a better understanding on how these methods operate mathematically.
2.1. Principal Component Analysis

PCA is a linear dimension reduction algorithm, a statistical technique to handle multivariate data that make utilizes of the Euclidean distance to estimate a lower dimensional data. While this method sometimes fails at preserving the intrinsic structure of the data (given the data have a non-linear structure) it does a good job preserving most of the variability from data. The algorithm for this method can be viewed as three steps:

• **Step 1**: Given the initial matrix $X$ representing the set of data, we will need to construct the standardized matrix $\tilde{X}$ by making sure that each sample column has zero mean and unit variance.

\[
\tilde{X} = \begin{pmatrix} \tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_M \end{pmatrix} = \begin{pmatrix} c_1 - \bar{c}_1 \sqrt{\sigma_{11}}, c_2 - \bar{c}_2 \sqrt{\sigma_{22}}, \ldots, c_M - \bar{c}_M \sqrt{\sigma_{MM}} \end{pmatrix}.
\]

Here, $\bar{c}_1, \bar{c}_2, \ldots, \bar{c}_M$ and $\sigma_{11}, \sigma_{22}, \ldots, \sigma_{MM}$ are respectively the mean values and the variances for corresponding column vectors.

• **Step 2**: Compute the covariance matrix of $\tilde{X}$, then make a spectral decomposition to get the eigenvalues and its corresponding eigenvectors.

\[C = \tilde{X}'\tilde{X} = U\Lambda U^T.\]

Here $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_M)$, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_M$, $U = (u_1, u_2, \ldots, u_M)$. $\lambda_i$ and $u_i$ are respectively the $i$th eigenvalue and the $i$th eigenvector for the covariance matrix $C$.

• **Step 3**: Given that we would like the target lower dimensional space to be of dimension $m$, the $i$th principal component can be computed as $\tilde{X}u_i$, and the reduced dimensional $(N \times m)$ subspace is $\tilde{X}U_m$.

Notice from **Step 3** that each principal components making up the reduced dimensional subspace is just a linear combination of the raw variables.

2.1.1. Shifting and Scaling

The idea of Principal Component Analysis go hand in hand with the idea of projection from a multidimensional space $\mathcal{M}_1$, of dimension $M$ to much lower dimensional space $\mathcal{M}_2$, of dimension $m$; where, $m \ll M$. If we think about our data as being represented inside an $M$-dimensional ellipsoid; and furthermore, assume some of the axis belonging to the ellipsoids are small, this would imply that the variance along those axis is also relatively small. Therefore, by discarding those axis where the variance is not as significant compare to the others axis we only lose a minimal amount of information. In order to find the axis (principal components) making up the ellipsoid, we need to first shift the data to the origin through the operation $c_i - \bar{c}_i$, then compute the covariance matrix (what happen after will be explained in the next section). It is important to note that it is not always necessary to scale
the data by dividing the standard deviation, see equation (1). The principal components analysis method is very sensitive to scaling since the goal is to find the directions or axis of highest magnitude or variance. So while the former shifting operation does not affect our results, the latter scaling operation might be harmful. There is no consensus as to how to scale the data one is working with to obtain the best result when applying PCA. From the figure bellow we could see how scaling the data gives us different results when using PCA (The color coating is not important for this plot and is purely decorative). On the middle and on the far right you have the visual representation of the reduced data when PCA was applied to the data after shifting, and after shifting and scaling respectively. The equations 4 and 5 bellow provide the resulting directional vectors and eigenvalues from the analysis, as expected, their are not the same.

**shifted:**

\[
U = \begin{bmatrix}
1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}; \quad \Lambda = \begin{pmatrix} 4500 & 4500 & 1000 \end{pmatrix}.
\]

**shifted and scaled:**

\[
U = \begin{bmatrix}
1 & 0 & 0 \\
0 & -0.7462 & 0.6657 \\
0 & -0.6657 & -0.7462 \\
\end{bmatrix}; \quad \Lambda = \begin{pmatrix} 19999 & 19999 & 19999 \end{pmatrix}.
\]

Figure 2: PCA applied on \(N = 20000\) data points modeling an Helix with \(M = 3\) and \(m = 2\).

For further consideration on scaling, consider the following scenario. Let say you are working with a 2-dimensional dataset with the first variable representing the temperature in degrees and the second the distance in kilometers. Let us also assume that the variable samples have almost equal variance and are positively correlated. Then a PCA analysis will suggest that the direction of the first principal component will be obtained with almost equal contribution from both variables. Now, consider another data collector gave you the same data for analysis only this time the data were measured in degrees and meters (every second datum in the data is multiplied by a thousand) respectively. In this instance, a PCA analysis
would align the first principal component almost completely with the second variable. This suggests that PCA could be viewed as an arbitrary analysis method whenever the variables have different units.

2.1.2. Principal Components

Given a set of data points, after centering the data around the origin (and scaling if necessary), the first principal component will correspond to the line going through the origin with a direction that minimizes the sum of squares distance of the data points from the line. In this sense it will represent the direction that accounts for most of the variability among the data. Similarly, the second principal component is obtained with the same idea in mind after all correlation with the first principal component has been removed from the data points. Mathematically we are attempting to project the set of \( M \)-dimensional data points \( X \) into a set of \( m \)-dimensional data points in \( Y \). The resulting matrix from this projection is such that \( Y = \tilde{X}U_m \), where \( Y \) is made up of the principal components and \( U_m \) contains the loading vectors \( u_1, u_2, \ldots, u_m \), which contains the contribution needed from each \( x \) to build each \( y \).

Finding the first loading vector \( u_1 \) must be done so that the magnitude of the first principal component is maximized:

\[
\mathbf{u}_1 = \arg \max_{||\mathbf{u}|| = 1} \|Y_1\|^2 \quad (6)
\]

\[
= \arg \max_{||\mathbf{u}|| = 1} \|\tilde{X}\mathbf{u}\|^2 \quad (7)
\]

\[
= \arg \max_{||\mathbf{u}|| = 1} |\mathbf{u}'\tilde{X}'\tilde{X}\mathbf{u}| \quad (8)
\]

The quantity to be maximized is well-known as the Rayleigh quotient. Given that the matrix \( \tilde{X}'\tilde{X} \) is symmetric, a standard solution to this problem is simply the eigenvector corresponding to the eigenvalue of the largest magnitude of the matrix \( \tilde{X}'\tilde{X} \).

To find the remaining loading vectors \( \mathbf{u}_k \) for \( k = 2 \ldots m \) consider applying the same idea to the modified matrix \( \mathbf{X}_k \).

\[
\mathbf{X}_k = \mathbf{X} - \sum_{i=1}^{k-1} \mathbf{X}\mathbf{u}_i\mathbf{u}_i' \quad (9)
\]

where all correlation with the previously found loading vectors has been removed. The \( k^{th} \) loading vector will then be equal to the eigenvector corresponding to the eigenvalue of the largest magnitude of the matrix \( \tilde{X}_k'\tilde{X}_k \).

2.1.3. Power Iteration \[11\]

For a given matrix \( \mathbf{A} \in \mathbb{R}^{n \times n} \), the power iteration method is one of many methods that have been written to estimate one single eigenvector corresponding to the eigenvalue of highest magnitude of \( \mathbf{A} \) under some assumptions. Suppose that the set \( \{\mathbf{u}_i\} \) of unit eigenvectors of
A forms a basis of $\mathbb{R}^n$, with corresponding real eigenvalues $\{\lambda_i\}$ such that $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$. If we consider $u^{(0)}$ with $\|u^{(0)}\| = 1$ to be an approximation to one of the eigenvectors of $A$ then we can write $u^{(0)}$ as a linear combination of the eigenvectors of $A$. Therefore, for some $c_1, c_2, \ldots, c_n \in \mathbb{R}$ we have the following:

$$u^{(0)} = c_1 u_1 + c_2 u_2 + \ldots + c_n u_n. \quad (10)$$

Under the assumption that $c_1 \neq 0$,

$$Au^{(0)} = c_1 \lambda_1 u_1 + c_2 \lambda_2 u_2 + \ldots + c_n \lambda_n u_n \quad (11)$$

$$A^k u^{(0)} = c_1 \lambda_1^k u_1 + c_2 \lambda_2^k u_2 + \ldots + c_n \lambda_n^k u_n \quad (12)$$

$$A^k u^{(0)} = \lambda_1^k (c_1 u_1 + c_2 (\frac{\lambda_2}{\lambda_1})^k u_2 + \ldots + c_n (\frac{\lambda_n}{\lambda_1})^k u_n). \quad (13)$$

So, as $k$ increases we get,

$$u_1 \approx \frac{A^k u^{(0)}}{\|A^k u^{(0)}\|}, \quad (14)$$

with the other terms getting close to zero.

From this derivation we can then write the power iteration algorithm as stated below:

---

**Algorithm: Power Iteration Method**

Pick a starting vector $u^{(0)}$ with $\|u^{(0)}\| = 1$

while $\|u^{(k)} - u^{(k-1)}\| > 10^{-6}$ do

1. Let $w = Au^{(k-1)}$
2. Let $u^{(k)} = \frac{w}{\|w\|}$

---

Although this method works well on large, sparse matrices it has some drawbacks. For instance, the rate of convergence of the algorithm relies on the ratio between the eigenvalues’ magnitude, and convergence is only guaranteed when our assumptions about the eigenvalues (distinct eigenvalues) are met.

2.1.4. Choosing $m$

In most applications the number $m$, which represents the dimension of the reduced data, is computed so that the sum of the eigenvalues corresponding to the eigenvectors making up the $m$ first principal components accounts for 90 percent of the total variability in the data. The total variability in the data is represented by the sum of all the eigenvalues obtained from the covariance matrix $C$. For our particular problem this number could be calculated directly by considering the trace of the matrix $C$ which is available to us, this will prevent us from computing all the eigenvalues in order to get that number.
2.2. Laplacian Eigenmaps [2]

This method has its advantages since it is a non-linear approach to dimension reduction. It aims to preserve the intrinsic or natural geometric structure of the manifold from the high dimension to the lower dimension. This method could also be summarized in three steps:

- **Step 1**: Given a set of $N$ points or nodes $x_1, x_2, \ldots, x_N$ in a high dimensional space $\mathbb{R}^M$, construct a weighted graph with $N$ nodes. Constructing the graph is as simple as putting an edge between nodes that are close enough to each other. In doing this, one might either consider the $\epsilon$-neighborhood technique where two nodes are connected if their square Euclidean distance is less than $\epsilon$, and not connected otherwise. This might sometimes lead to graphs with several connected nodes or even disconnected graphs. An alternative would be to consider the $k$-nearest neighbor where each node is connected to its $k^{th}$ nearest neighbors. Both techniques do yield a symmetric relationship.

- **Step 2**: Choose the weight for the edges and construct the weight matrix $W$. This could be as simple as putting a 1 between two connected nodes and a 0 otherwise (if the node are not connected). One could also consider a weight as a function of the Euclidean distance between two connected nodes and 0 otherwise.

- **Step 3**: For each connected sub-graph(s), solve the following generalized eigenvector problem,

$$L f = \lambda D f,$$

where $D_{ii} = \sum_j W_{ji}$, the diagonal matrix; and $L = D - W$, the Laplacian matrix. Let $f_0, f_1, \ldots, f_{N-1}$ be the solutions of (15) with corresponding $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$ such that, $L f_i = \lambda_i D f_i$ for $i$ going from 0 to $N - 1$ and $0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{N-1}$. Then the $m$-dimensional Euclidean space embedding is given by:

$$x_i \rightarrow y_i = (f_1(i), \ldots, f_m(i)).$$

2.3. Hierarchical Clustering

Next I will consider a couple of clustering methods. Starting with hierarchical clustering (HC), this is a connectivity based algorithm, the idea is that nodes that are closer to each other are more related than those who are father apart. There are two ways of implementing HC; one could either take a bottom-up (order $O(n^3)$) approach; where each data points start as being in its own cluster, and as we move on pairs of clusters are merged together, see figure 3 for a simple illustration. Otherwise, one could consider a top-down (order $O(2^n)$, mostly due to the search algorithm) approach; where we start with one big cluster and splits are performed recursively as we move further down the hierarchy. The later approach will not be considered during application for this project.

In order to proceed we then need to decide on a metric, a way to measure the distance between two pairs of observation and a linkage criteria, a function of the pairwise distances between observations in the sets which has for output the degree of similarity between sets (this function will let us know whether or not two sets could be merged). Here are some commonly used metrics and linkage criteria:
• Examples of metrics:

- Euclidean distance:

\[ \|a - b\|_2 = \sqrt{\sum_i (a_i - b_i)^2} \]  \hspace{1cm} (17)

- Manhattan distance:

\[ \|a - b\|_1 = \sum_i |a_i - b_i| \]  \hspace{1cm} (18)

• Examples of linkage criteria:

- Maximum or CLINK (complete linkage clustering)

\[ \max\{d(a, b) : a \in A, b \in B\} \]  \hspace{1cm} (19)

- Inner squared distance or ward linkage clustering (minimum variance algorithm)

- Mean or average linkage clustering

\[ \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a, b). \]  \hspace{1cm} (20)

![Figure 3: Bottom up Hierarchical clustering illustration [7].](image)

2.4. K-means clustering

The idea here is to randomly select an initial set of \( k \) means, these could be random vectors selected either within your data set or outside your data set. This selection is follow by an assignment step where all individual data points are assigned to the nearest means according to a well-defined metric (square Euclidean distance). After this step is done the mean within each of the clusters formed gets updated to the mean of the data in the cluster. The two previous steps are repeated until no new assignment is made, this means that the clusters remain the same before and after an assignment step. This method is \( NP \)-hard (Non-deterministic Polynomial-time hard) and can be summarized as such:
• Initialized a set of $k$ means $m^{(1)}_1, m^{(1)}_2, \ldots, m^{(1)}_k$.

• Assignment step: Assign each observation $x_p$ to exactly one set $S_i$ containing the nearest mean to $x_p$.

\[
S^{(t)}_i = \{ x_p : \| x_p - m^{(t)}_i \|^2 \leq \| x_p - m^{(t)}_j \|^2 \ \forall j, 1 \leq j \leq k \}. \tag{21}
\]

• Update step: update the mean within each cluster,

\[
m^{t+1}_i = \frac{1}{|S^{(t)}_i|} \sum_{x_j \in S^{(t)}_i} x_j. \tag{22}
\]

• Repeat the two previous steps.

• Stop when no new assignment is made.

See figure 4 for an illustration of those steps.

![Figure 4: K-means clustering illustration](image)

### 3. The Data

#### 3.1. Genotype and Phenotype

The genotype of an organism is the inherited instructions it carries within its genetic code, it is what makes up the genetic code. Not all organisms with the same genotype act or react the same and not all organisms that behave the same have the same genotype.

A phenotype is what makes up an organism observable behavior and characteristics. It is a product of the environment in which the organism find itself. In a more general approach, in *The Extended Phenotype* [10], Richard Dawkins defines phenotype to mean all the effect a gene has on the outside world that may influence its chance of being replicated. These could be effect on the organism in which the gene resides, the environment or other organisms. In a way the phenotype is a function of the genotype, it is the dependent variable. This genotype-phenotype distinction was proposed in 1911 by Wilhelm Johannsen to make clear the difference between an organisms heredity and what that heredity produces [9].

Gene expression is the most fundamental level at which the genotype gives rise to the phenotype. It expresses the level at which a particular gene reacts within the cell or organism it finds itself due to environmental factors. Expression levels of large number of genes is obtained or measured simultaneously by using a microarray or biochip.
3.2. DNA Microarray

DNA (deoxyribonucleic acid) microarray also known as biochip or DNA chip is a small silicon or glass surface the size of a postage stamp that contains a wide collection of spots (in the range of thousands). Each of these spots contain a very small amount of well-known specific DNA sequence known as reporters or probes and will be used as a basis for the known available collection of DNAs. The core principle behind microarray rests on the hybridization between two DNA strands. The target or sample DNA strands will be able to attach themselves to the DNA strands in each spot forming hydrogen bonds. The figure 5 bellow gives a schematic representation of a microarray experiment.

Figure 5: Schematic representation of a microarray experiment [5].

The steps involved in these experiment could be summarized as follow:

- **Step 1:** Take samples from both healthy and unhealthy tissues (Cancerous tissue samples will do). Each tissue contains a great amount of cells in which resides mRNAs (messenger ribonucleic acids) which are to be extracted from both samples.

- **Step 2:** The extracted mRNAs from the cells are then reverse transcribed using an enzyme called *reverse transcriptase* into more stable complementary DNAs known as...
cDNAs. The cDNAs are very numerous and minuscule, so fluorescent tags or labels are added to each sample. The tags will emit a red light for the damaged cDNAs and a green one for the healthy cDNAs when exposed to ultraviolet light from a scanner, then the two cDNA samples are combined. The goal is to obtain an expression representative of the number of cDNAs in the cell.

- **Step 3:** The combined sample of fluorescently labeled cDNAs are then spread on the microarray that contains thousands of spots each corresponding to a cDNA representing each gene. Each labeled cDNA will recognize and attach itself to its complementary sequence of DNA basis on the microarray. They will hybridize or bind tightly to the corresponding gene features found in exactly one of the spots; otherwise, they will not. This will allow us to rinse off the ones that are not attached.

- **Step 4:** After the hybridization step, a laser will then scan each spot on the microarray and activate the florescent dies in the samples cDNAs. The intensity information from each DNA spot is captured by a computer, the computer will then compute the ratio of red and green which is indicative of what genes are expressed on the healthy and unhealthy tissue samples.

It is important to note that this experiment could also be carried strictly using either unaffected or affected tissue samples. For instance one could repeat the experiment for multiple cancer cell lines or samples; in this case, only affected tissue samples would be used. In another experiment one could use the DNA microarray to genotype multiple regions of the genome. The figure provided here gives a step-by-step of the microarray experiment. For more on the subject, I direct the attention for the reader to the Unsolved Mystery of Human Health website.

![Figure 6: Step by Step microarray experiment](image)

### 3.3. The Main data: NCI-60 [3, p. 95-96]

The NCI-60 data I will be working with consists of microarray expressions of 20,002 genes activities within 60 different cancer cell lines. I plan on working with the traditional gene expressions across these 60 cancer cell lines, without presence of drugs. Although there will be not be any drugs present in the samples, the presence of cancer stimulant is enough to make this analysis meaningful and interesting. These data are available to download through the CellMiner database under the NCI (National Cancer Institute) website.
4. Implementation and Validation methods

4.1. Software and hardware

The two dimension reduction algorithms described above will be implemented using Matlab as a mathematical tool. This decision is due to the superior ability of Matlab to deal with matrix operations. Another reason would be the wide range of toolbox available to bring this project to completion in a timely manner, the toolbox will provide us with test data and prior implementation of PCA and LE for validation and bench-marking. I will be using my personal laptop with $8Gb$ of memory to run simulations on smaller data sets and the Norbert Wiener Center lab, clocking at $128Gb$ of memory for larger data set if needed. Clustering algorithms (K-means and Hierarchical) built into the Matlab DRtoolbox will be used as a tool to create various clusters for our data.

4.2. Validation methods

We will take advantage of the DRtoolbox\footnote{Laurens van der Maaten, Delft University of Technology [4]} which contains implementation of the Principal Component Analysis method and the Laplacian Eigenmaps methods describe above. The DRtoolbox also contains a number of well understood data sets in 3-dimensional space with corresponding representation in 2-dimensional space for testing and validating the dimension reduction methods implemented for this project. Some examples of dataset courtesy of the DRtoolbox include the following.

- The Swiss Roll dataset in figure 7

\[ F : (x, y) \to (x \cos(x), y, x \sin(x)) \]  

(23)

Figure 7: 3-dimensional presentation of the Swiss Roll data.
• the Twin Peaks dataset in figure 8

\[ f(x, y) = x^4 + 2x^2 + 4y^2 + 8x \]  

(24)

Figure 8: 3-dimensional presentation of the Twin Peaks data.

Note that unlike the other plots we will be displaying later, the color coating on figure 7 and figure 8 have no particular meaning. The goal of the coating is to make those 3-dimensional images more perceptive to the reader.

4.3. The rand index [12]

Since we would like to demonstrate the effectiveness of Laplacian Eigenmaps over Principal Components Analysis in preserving biologically relevant structures in cancer expression dataset, we will need a means of quantifying how much of the structure was preserved after dimension reduction. The rand index is a measure of agreement between two data clustering and is defined in the following way, given a set of \( n \) elements \( S \) and two partition \( P_1 \) and \( P_2 \) of the set \( S \), the rand index \( r \) is given by:

\[ r = \frac{a + b}{a + b + c + d} \]  

(25)

In this equation:

• \( a \), the number of pairs of elements in \( S \) that are in the same set in \( P_1 \) and in the same set in \( P_2 \).

• \( b \), the number of pairs of elements in \( S \) that are in different sets in \( P_1 \) and in different sets in \( P_2 \).

• \( c \), the number of pairs of elements in \( S \) that are in the same set in \( P_1 \) and in different sets in \( P_2 \).
• $d$, the number of pairs of elements in $S$ that are in different sets in $P_1$ and in the same set in $P_2$.

In a way, the amount $a + b$ will represent the number of agreements between $P_1$ and $P_2$. Similarly, $c + d$ will represent the number of disagreements between $P_1$ and $P_2$. Defined in this manner, if we consider $P_1$ to be the partition generated from a clustering analysis on the original data and $P_2$ to be the partition generated from the same clustering analysis on the reduced data, then the rand index $r$ will give us a fractional quantity that represents how much of the structure between data before and after dimension reduction was preserved. This will give us a way to evaluate the effectiveness of the two dimension reduction methods.

5. Test Problem

In this section we consider a couple of well known data sets from the DRtoolbox. A data set modeling an Helix equation and another modeling the Twin Peaks equation. Those data set have a well understood structure in 3-dimensional space with corresponding representation in 2-dimension space for testing and validating the dimension reduction methods implemented for this project. We will compare the results obtained from the DRtoolbox with the ones obtained from the implemented methods. The color coating on the data will represent the cluster into which each data were assigned. This will give us a visual sense of the data distribution on the higher and lower dimensional space.

5.1. Helix

The matrix $X$ comprises of 2000 data points in a 3-dimensional space governed by the following parametric equation (26). See figure 9 for a visual representation, the colors represent the cluster assignments of each data in the set, the clustering was done using k-means clustering.

$$F : (x, y, z) \rightarrow [(2 + \cos(8t)) \cos(t), (2 + \cos(8t)) \sin(t), \sin(8t)]$$ (26)

5.1.1. Principal Components Analysis

Since there is no unit of measure associated with data in this test case, no scaling has been done on the data. This will prevent any stretching and/or shrinking on the data, which in turn will not affect the original within the data keeping PCA honest. Figure 10 gives us the result from both the DRtoolbox on the left and the implemented PCA on the right. Both algorithms used two principal components which in this case account for about 90% of the variability within the data. In figure 9 you could see the direction of the loading vectors. The first and second loading vectors have corresponding eigenvalues indicating that the resulting principal components each account for 45% of the variability within the data. Last, the third direction not represented here yields a principal component accounting for only 10% of the variability. The loading vectors and corresponding eigenvalues are giving bellow:
Figure 9: 3-dimensional presentation of the Helix data set.

\[
U = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}; \quad \Lambda = \begin{pmatrix}
4500 & 4500 & 1000
\end{pmatrix}.
\] (27)

Figure 10: PCA applied to the Helix data set. DRtoolbox output on the left and implemented PCA output on the right.

5.2. Twin Peaks

The matrix \(X\) comprises of 20000 data points in a 3-dimensional space governed by the following equation (28). See figure 11 for a visual representation, the colors represent the cluster assignments of each data in the set, the clustering was done using k-means clustering.

\[
f(x, y) = x^4 + 2x^2 + 4y^2 + 8x
\] (28)
5.2.1. Principal Components Analysis

Since there is no unit of measure associated with data in this test case, no scaling has been on the data. This will prevent any stretching and/or shrinking on the data, which will not affect the original variance within the data keeping PCA honest. Figure 12 gives us the result from both the DRtoolbox on the left and the implemented PCA on the right. Both algorithms used two principal components which in this case account for about 99% of the variability within the data. In figure 11 you could see the direction of the loading vectors. The first loading vectors, almost completely associated with the \( z \)-axis, has a corresponding eigenvalue indicating that the resulting principal component account for about 98% of the variability within the data. The second and third (the third not represented here) almost completely associated with the \( x \) and \( y \) axis both yield principal components, each accounting for only about 1% of the variability within the data. The loading vectors and corresponding eigenvalues are giving bellow:

\[
\begin{bmatrix}
0.0036 & 0.5832 & 0.8123 \\
0.0010 & 0.8123 & 0.5832 \\
1.0000 & -0.0029 & -0.0024
\end{bmatrix}
\quad \Lambda = \begin{pmatrix} 69547 & 696 & 651 \end{pmatrix}.
\tag{29}
\]

6. Analysis on NCI-60

6.1. Principal Component Analysis

The NCI-60 data were downloaded through the CellMiner database under the NCI (National Cancer Institute) website. The data have been standardized prior to the download by scientists in the field. The original data are of size \( N = 20002 \) and \( M = 60 \), I have applied PCA on the data to reduce the dimension for various value of \( m \) (from 1 to 60), where \( m \) represents the dimension of the reduced data. I ran the experiment using both k-means clustering and hierarchical clustering to determine how much of the structure within the
Figure 12: PCA applied to the Twin Peaks data set. DRtoolbox output on the left and implemented PCA output on the right.

Data was preserved after dimension reduction. The plots below will provide you with some of the results.

The colors represent the cluster assignments of each data in the NCI-60 data set. This will give us an idea of where each data was projected after reduction. In Figure 13, for instance, with $m = 2$, k-means clustering was applied to the data before dimension reduction, the plot on the left shows you where each data was projected on the two-dimensional space. k-means clustering was then applied again on the reduced data and the plot on the right reflects the result. Afterward, similarity analysis based on the rand index was done on the two clustering results and we found out that the cluster assignments were 83% compatible. This agreement between the two clustering outputs is indicative of how much of the structure within the data was preserved after dimension reduction. Figure 14 provides similar results using hierarchical clustering. Here, we get close to 73% compatibility between cluster assignments.

The later experiment was carried over a wide range of dimension reduction values $m$, for $m = 1, 2 \ldots 60$, Figure 15 gives the plot of the rand index $r$ as a function of the reduced dimension $m$. Furthermore, Figure 16 gives us an idea of much of the variability was preserved within the data for each $m$ value, the red (horizontal) line represents the 90% variability cut off and the green (vertical) line shows how many principal components ($m = 44$) we need to use to achieve this result.

7. Results

At the end of this project we expect to see a better performance overall from the Laplacian Eigenmaps method versus Principal Component Analysis. But nothing can be certain at this point.
8. **Timeline**

This timeline will reflect what I have done so far and what is left to do. So far my biggest challenge has been to efficiently implement a method for getting the eigenvalues and vectors that I need for the dimension reduction methods. At this point I have written codes to implement the three steps in the PCA algorithm. I have implemented the Power iteration method and I am working some variants, such are the Inverse iteration method and the Rayleigh Quotient (RQ) iteration method to find the eigenvalues of lowest magnitude. These methods would be used in the implementation of the Laplacian Eigenmaps DR method. They are not fully operational as they involve taking the inverse of a matrix with a large condition number, which makes the resulting eigenvalues inaccurate. I plan on exploring other methods in the remainder of the school year to avoid convergence and matrix inversion issues.

- October - November: The PCA algorithm
  - Power iteration method has been implemented.
  - PCA algorithm has been implemented.
  - Rand index algorithm has been implemented.
  - I have implemented the Inverse iteration method and the Rayleigh Quotient (RQ) iteration method but there are some accuracy issues due to matrix inversion.
  - Testing and validating the PCA algorithm.
Figure 14: PCA analysis on NCI-60 with Hierarchical clustering: 73% compatibility.

- December: Mid-year presentation was done on the second of December.
- January: First semester progress report.
- February - April:
  - Implementation of LE algorithm.
  - Testing and validating.
- April - May:
  - Implementation of k-means clustering algorithm if time permits.
- May: Final report

9. Deliverable

The following materials are expected to be delivered by the end of the academic year:

- Weekly Report
- Self Introduction
- Project Proposal
Figure 15: Rand index computed after clustering of the reduced data for \( m = 1 \cdots 60 \).

- First-Semester Progress Report
- Mid-year Status Report
- Final Report
- Code for Principal Component Analysis implementation
- Code for Laplacian Eigenmaps implementation
- NIC-60 data set.
Figure 16: Variability preserved within the data after dimension reduction for \( m = 1 \cdots 60 \).

References


[3] Vinodh N. Rajapakse (2013). Data Representation for Learning and Information Fusion in Bioinformatics. Digital Repository at the University of Maryland, University of Maryland (College Park, Md.).


