DYNAMIC ANALYSIS OF HIGH DIMENSIONAL MICROARRAY TIME SERIES DATA USING VARIOUS DIMENSIONAL REDUCTION METHODS

By

BANAFSHEH SAMAREH ABOLHASANI

Bachelor of Science in Mechanical Engineering

University of Pune

Pune, India

2011

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2013

DYNAMIC ANALYSIS OF HIGH DIMENSIONAL

MICROARRAY TIME SERIES DATA

USING VARIOUS DIMENSIONAL

REDUCTION METHODS

Thesis Approved:

Dr. Zhenyu (James) Kong

(Thesis Adviser)

Dr. Satish T S Bukkapatnam

(Committee Chair)

Dr. Baski Balasundaram

(Committee Member)

ACKNOWLEDGEMENTS

I would like to gratefully and sincerely thank Dr. Zhenyu (James) Kong for his guidance, understanding and patience. I am pretty sure that not all the graduate students are given the opportunity to perform research with such independence. I am really thankful that Dr. Kong introduced me to the statistics, and allowed me to strengthen my knowledge in this part. Also I want to extend my sincere gratitude to Dr. Satish T. S. Bukkapatnam and Dr. Baski Balasundaram for giving me thoughtful technical advices and valuable suggestion.

Finally and most importantly, I would like to thank my mother Farzaneh. Her support, encouragement, and unwavering love were undeniably the bedrock upon which the past ten years of my life have been built. I thank my brother Bardia for his continuous kindness, support and his faith in me, and compelled me not to give up.

Also I want to thank my uncles Farzad, Farzam, and Jamshid and my friend Tania for their continued support and encouragement, love and the confidence that they have placed in me. I grew determined to follow my father's path to make a difference in this world. Therefore I dedicate my dissertation to the loving memory of my father, a true role model for me.

Name: BANAFSHEH SAMAREH ABOLHASANI

Date of Degree: DECEMBER, 2013

Title of Study: DYNAMIC ANALYSIS OF HIGH DIMENSIONAL MICROARRAY TIME SERIES DATAUSING VARIOUS DIMENSIONAL REDUCTION METHODS

Major Field: INDUSTRIAL ENGINEERING AND MANAGEMENT

Abstract: This dissertation focuses on dynamic analysis of reduced dimension models of two microarray time series datasets. Underlying research achieves two main objectives; namely, (1) various dimension reduction techniques used on time series microarray data, and (2) estimating autoregressive coefficients using several penalized regression methods like ridge, SCAD, and lasso.

The research methodology includes two research tasks. Firstly, applying several dimension reduction methods on two microarray data sets, and modeling comparisons based on accuracy and computation cost. Secondly, applying the sparse vector autoregressive (SVAR) model to estimate gene regulatory network based on gene expression profile from time series microarray experiment on two datasets and the autoregressive coefficients estimation were calculated using several penalized regression methods, and then performing comparisons among various regression methods for each dimension reduction model.

Study results show that the dimension reduction methods producing orthogonal independent variables are performing better because orthogonality leads to reasonable coefficient estimation with low standard errors. On the other hand, regarding dynamic analysis, it could be seen that factor analysis (FA) outperformed the rest of dimension reduction methods with regards to goodness of fit after applying several penalized regression methods on each model. The reason behind this is due to using varimax rotation in FA, in which most of the coordinates are set closer to zero, and in turn makes the data sparser. Hence inducing additional sparsity subject to maintaining a certain goodness of fit.

TABLE OF CONTENTS

LIST OF TABLES

Tables Page

LIST OF FIGURES

Figures Page

CHAPTER I

INTRODUCTION

This chapter emphasizes the need for dimension reduction in high dimensional datasets with referring to microarray time series data and also the importance of dynamic analysis of such data.

1.1 Motivation

1.1.1 Significance of dimensional reduction techniques in microarray data

The major goal of unsupervised learning is to recover the latent structure to represent properties of objects or data points being modeled that have not been directly observed or to represent hidden causes that explain the observed data. Unsupervised learning algorithms can face challenges like determining the amount of latent structure, number of clusters, and dimensions for better presentation of structures in data.

Choosing the model with the best dimensionality can result in having a good performance, which can characterize the properties of the observed objects. We will give details of such models with different dimensional reduction techniques and possible future work with an application on microarray time series data. We will summarize recent works on exploring the extension of these

nonparametric approaches to models in which objects are represented as an unknown number of latent features.

While dealing with high dimensional data, i.e., data being described by a large number of features, it is often beneficial to reduce their dimension. Dimensionality reduction is an interesting alternative to feature selection. Like feature selection, reduced dimension data yields a low dimensional representation, which helps us to build lower capacity predictors in order to improve generalization.

In fact, if the data is laid on a low-dimensional manifold, it may preserve almost all of the original information while representing it in a way that eases learning. On the other hand, these techniques are purely unsupervised, since the projected data gives us better visualization.

Dimension reduction techniques often can be considered as a pre-processing step prior to data analysis, which simplifies the data. The advantage of working with low-dimensional data is that the data will be more accurate plus the computational cost will significantly reduce.

Statisticians desire that the number of objects in training sets n, e.g., a sample from the data with the correct classification results already assigned exceed the number of features p; however, in practice this is not the case since high dimension data occupy only a manifold in the space so the implicit dimension of the data n will be less than the number of features p. Figure 1.1 shows the desired image of the number of objects and features from a practical and statistical point of view.

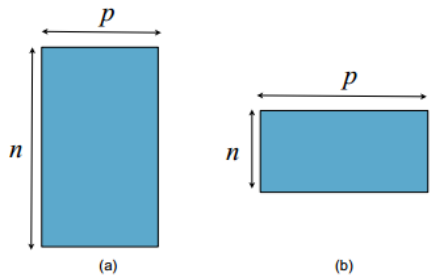


Figure 1. 1 Comparison between numbers of features with the number of objects in a large data set: (a) This figure shows the desired image from statistical point of view, (b) This is the image from practical point of view (Padraig adraig Cunningham, 2007)

There are many reasons behind reducing the dimension of microarray data. For one, identifying genes that are responsible for a specific disease plays an important role in drug selection in order to contend with such diseases. Also, it is important to understand the development stages in many genetic diseases; therefore using time series microarray data to identify genes that play important roles at different developmental stages is essential.

The main purpose of dimension reduction in a time series microarray is to obtain insight regarding the data distribution. Realizing the patterns hidden in a high dimensional data like gene expression offers a tremendous opportunity for a better understanding of genomics; however, there many challenges, such as the complexity of genetic networks, which result in complex or sometimes impossible interpretations of such massive data sets.

1.1.2 Significance of dynamic analysis in microarray time series data

Time series microarray data are context-dependent and they can go under systematic rewiring and not be invariant over time. Researchers are interested in monitoring a gene's behavior at multiple points over time. After selecting the best dimension reduction methods with respect to

computation cost and quality, the main purpose is to perform a dynamic analysis of the selected methods. This dynamic analysis consists of three steps: preprocessing, time series model selection, and forecasting beyond the end of time series.

1.2 Challenges in analyzing time series microarray data

Time series microarray is popular for studying biological systems. Dimension reduction is a preferred technique in data mining before performing data analysis. A preferred dimension reduction technique is one that depends on prior knowledge as little as possible. For example, it is preferable to choose a dimension reduction technique that doesn't require a pre-determination process for the number of features.

Usually microarray data are noisy; hence we require an algorithm able to extract useful information from such noisy data. While analyzing time series microarray data, one challenge would be computational cost, especially in cases of high dimensional data. Another issue is to select an efficient model for such data.

Computation cost consideration is a key characteristic for good dimension reduction techniques, but high quality dimension reduction techniques will judge our model's performance. It should be noted that in such data, performing preprocessing steps such as alignment and normalization are necessary to remove variability in a given data.

Working with multivariate time series data is challenging since data has multiple dimensions to consider, e.g., auto-correlation and cross-correlations, whereas in short time series data conventional time series modeling is easier. Once we used the same dependent variable and the same estimation period for each model, we can then use a number of statistical methods like r-squared to the forecasting models. Such comparisons will validate our selected model.

1.3 Research objectives

Typical microarray data contains 10^3 to 10^4 genes and this number is expected to reach to the order of 10^6; however, the number of samples involved in a microarray experiment is generally less than 100.

The reason behind using dimension reduction techniques is that the current thinking in molecular biology holds that only a small subset of genes participates in any cellular process of interest and that a cellular process takes place only in a subset of the samples. This belief calls for the subspace reduction and then we have to perform a time series analysis on the selected model.

Having explained the challenges one can face during dimensional reduction in a time series microarray data analysis, there are two primary objectives for this research. The first objective is to apply different dimensional reduction techniques on the time series microarray data with a predetermined number of features via singular value decomposition (SVD) selection method and a comparison of algorithms by their computation cost and quality of clustering.

The second objective is to perform a time series analysis of the given reduced dimension microarray data, estimate time series parameters, further assess the adequacy of the fitted model, and perform forecasting. Finally, all the models will be compared with regard to their dynamic performance.

1.4 Organization of the thesis

In this section, the organization of the thesis will be discussed. In chapter two, background information and a review of literature in clustering techniques and dynamic analysis are discussed. A brief introduction to dimension reduction techniques, such as factor analysis, principal component analysis, etc., and the importance of dynamic analysis of microarray data are

reviewed. In chapter five, dimension reduction techniques used in this thesis are introduced in detail.

While the best dimension reduction method with respect to computation cost and quality of a clustering perspective are selected in chapter three, we shall also start the dynamic analysis. Once the time series model is selected, the goodness of fit will be checked. Having checked the goodness of fit of the selected time series model, we will then compare different models with regards to their performance.

CHAPTER II

BACKGROUND AND REVIEW OF LITERATURE

This chapter begins with a literature review regarding analyzing time series microarray data and the relevant research on dimension reduction methods.

2.1 Analyzing microarray data

Microarray analysis entails monitoring the expression level of genes under some particular condition. Microarray technology is a great tool for many biologists to monitor expression levels of genes in a given organism. Microarrays use a glass on which molecules are fixed and focused on specific spots, i.e., features. These features are printed on a glass by robot (M. Madan Babu, et al., 2004). Each microarray contains thousands of features where each feature has a few million copies of common DNA molecules that uniquely represent a gene.

As seen in Figure 2.1, after RNA is extracted from the cells it will be labeled with different colors, e.g., green and red, and then DNA is hybridized on the glass slides, where each DNA is a representation of a gene. After that, DNA will be bound to the spot, and the expression level for each gene can then be presented as an image.

The relative expression level for each gene, i.e., population of RNA in the two samples can be stored as an image. It is important to understand how data is actually being extracted from these images, as this represents the primary data collection step and forms the basis for any further analysis. The relative expression level for a gene is defined as the amount of light (red or green) emitted after excitation.

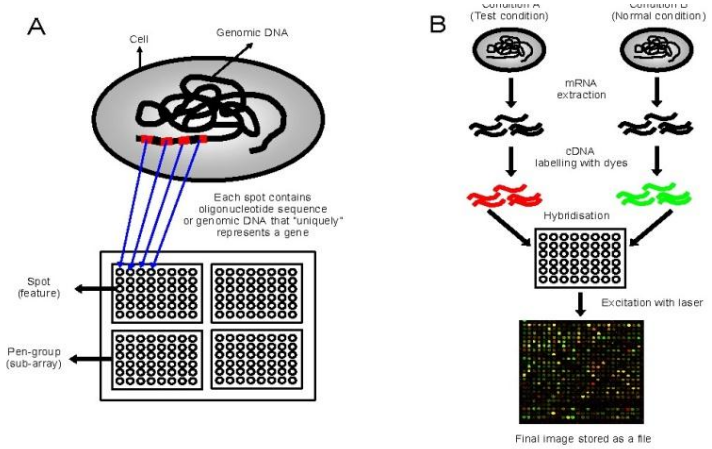
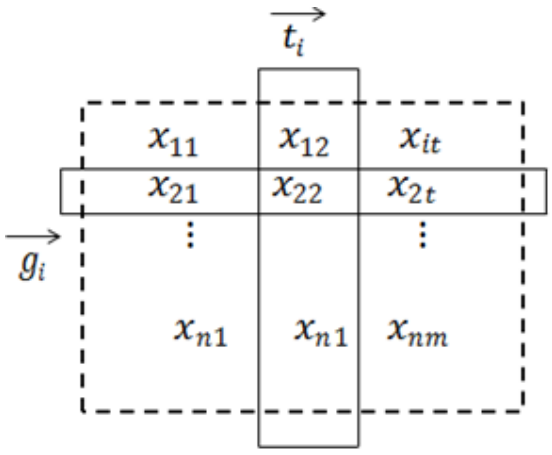


Figure 2. 1 (A) and (B) shows schematic of the experimental protocol to study differential expression of genes. (M. Madan Babu, et al., 2004)

One of the reasons to carry out a microarray experiment is to monitor the expression level of genes at a genome scale, like given time. The processed data, after the normalization procedure, can then be represented in the form of a matrix, often called a gene expression matrix. The expression level for a gene at different experimental conditions is called a sample expression profile. Gene expression matrices contain rows representing genes and columns representing particular conditions, such as time points.

In gene expression matrices, each row represents data for one gene and each column is a developmental time point. A gene expression data set from a microarray experiment can be presented by a real-valued expression matrix $X = \{x_{ij} | 1 \leq i \leq m, 1 \leq j \leq n\}$. Figure 2.2 (a)

shows the gene expression matrix, where the rows $G = \{g_1, \dots, g_n\}$ form the expression pattern of genes and the columns $T = \{t_1, \dots, t_n\}$ represent the expression level of gene *i* in time *j*. Figure 2.2 (b) includes the notation used in this chapter.



m	Number of genes
n	Number of times
X	Gene expression matrix
x_{ij}	Each cell in a gene expression matrix
g_i	A gene
t_i	A time

Figure 2. 2 (a) A gene expression matrix. (b) Notation in this thesis

2.2 Time series microarray data description

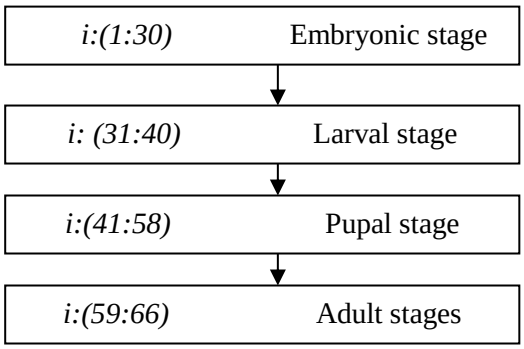
A microarray experiment uses large number of genes under multiple conditions like time series during a biological process. In this thesis we will focus on two actual gene expression data. One would be the RNA expression levels of 4028 genes for nearly one-third of all Drosophila genes during a complete time course of development as documented in Arbeitman et al. (2002) as our input data. In such an experiment, the expression levels of 4028 genes were examined during 66 sequential time periods.

The 66 time points were chosen during the development cycle across four different stages, beginning at fertilization and spanning the embryonic, larval, and pupal periods, i.e., embryonic (1–30 time point), larval (31–40 time point), pupal (41–58 time points) and adult stages (59–66

time points). Early stages change rapidly, so overlapping 1-hour periods were sampled; on the other hand, adults were sampled at multiday intervals.

The second data set used is a Human cancer cell Line (HeLa) cell-cycle gene expression dataset collected by Whitfield et al. (2002). Gene expression was measured using microarrays manufactured in the Stanford Microarray Facility. The data contains 681 genes and 48 time points with one-hour intervals and one reading at each time point. The data can be downloaded from http://microarray.omrf.org/publications/2004/knowlton/MDAT.zip. The four basic stages are described in Table 2.1, where *i* is time in days.

Table 2. 1 Four stages involving in the life cycle of Drosophila



2.3 Dimension reduction techniques

2.3.1 Eigen Laplacian

Introduction to graph theory used in eigen Laplacian

In this section the concepts of these graphs are discussed in detail and a brief review of graph Laplacian is introduced (Luxburg 2007).

Definition 1: Let G= (V, E) be an undirected graph with non-empty and finite vertex set of $V= \{ v_1, .., v_n \}$, if $(v_i, v_j) \in E$, we denote this edge by e_{ij}. If E is symmetric that is, $e_{ij} \in E \Leftrightarrow e_{ji} \in E$ for all edges, we can call *G* an undirected graph.

Definition 2: Let *G'= (V, E')* be a graph, by replacing *E'* by $E=E' \times R_0^+$, which lets each edge $e_{ij} \in E'$ between two vertices v_i, and v_j carries a non-negative weight $w_{ij} \geq 0$.

Definition 3: Considering *G= (V, E)* an undirected and weighted graph with weights w_{ij}. If $(v_i, v_j) \notin E$, then the weighted adjacency matrix of the graph is $W=(w_{ij})_{i,j} = 1, \dots, n$ is called adjacency matrix of *G* .if we define $w_{ij} = 0$, this means that vertices v_i and v_j are not connected, now that we mentioned that $w_{ij} = w_{ji}$, the degree of a vertex

Definition 4: Considering *G= (V, E)* an undirected and weighted graph with weightsw_{ij}. Then the degree of vertex $v_i \in V$ is defined as $d_{i=\sum_{j=1}^{n} w_{ij}}$ for some $i \epsilon \{1, \dots, n\}$.

Definition 5: Considering *G= (V, E)* an undirected and weighted graph with weights w_{ij}. Given a subset of vertices $A \subset V$, if there exists a path between any two vertices in *A* that lies completely in *A*, i.e. all points of this path are in *A*, and there is no connection between *A* and its complement $A^c = V/A$, *A* is called connected component.

Definition 6: Given a subset of vertices $A \subset V$ for an undirected and weighted graph with weights w_{ij} we define indicator vector of A by $1_A = (f_1, \dots, f_n)' \in R^n$ as the vector with entries $f_i = 1 \; if \; v_i \in A$ and $f_i = 0$ otherwise.

Introduction to graph Laplacian and its properties

Graph Laplacian are matrices that represent graphs. The field discussing the study of such matrices is called graph theory; since these graphs hold useful information, graph theory uses such matrices. In the following section, the graph Laplacian is introduced with its properties. It should be mentioned that there is no unique convention for calling a matrix "graph Laplacian"; as it was mentioned in graph section, *G* should be considered as an undirected and weighted graph with degree metric *D* and adjacency matrix *W* (Xu et al, 2008).

Definition 7: Un-normalized Laplacian graph, i.e., suggests that we can normalized it as well, and is the ideal case where data which are in one cluster are not related to those in others, nonzero *L=W- D***.**

The following properties are the summary of the most important facts that are needed for Eigen decomposition using un-normalized algorithm.

1. For every vecor $f \in R^n$ we have $f^{'}Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$
2. *L* is symmetric, since *W* is symmetric and *D* is symmetric.
3. The smallest Eigen value of *L* is 0, a corresponding eigenvector is the constant one vector.
4. *L* has n non-negative, real-values Eigen values $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$

Definition 8: Normalized Laplacian are defined as, $L_{rw} = D^{-1}L$ and $L_{sym} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$. We call L_{sym} as a symmetric matrix, and L_{rw} as it is closely connected to a random walk. In the following section, the properties of L_{sym}, and L_{rw} are summarized. In order to get the inverse of *D* we should assume that *det (D)* ≠0, we know that, *det (D):* $\prod \lambda 1m = \prod d_i \neq 0$, we can assume that $w_{ii} > 0$ for all *i* that is in general $d_i \geq w_{ii} \geq 0$.

The following properties are the summary of the most important facts needed for clustering using normalized algorithm:

1. For every $f \in R^n$ *we have* $f^{'}L_{sym}f = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_i}{\sqrt{d_j}}\right)^2$
2. λ is an eigenvalue for $L_{rw;}$ with Eigen vector v if and only if $\boldsymbol{\lambda}$ is an eigenvalue of L_{sym} with eigenvector $w=D^{\frac{1}{2}}v$
3. λ is an eigenvalue of $L_{rw;}$ with eigenvector of v if and only if $\boldsymbol{\lambda}$ and v solve the generalized $Lv=\boldsymbol{\lambda}Dv$
4. 0 is an eigenvalue of $L_{rw;}$ with the constant one vetor 1 as eigenvector, 0 is an eigenvalue of L_{sym} with Eigen vetor $D^{\frac{1}{2}}1$
5. L_{sym}, and $L_{rw;}$ are positive-semi definite and have n non-negative real-valued eigenvalue $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$

2.3.2 Introduction to Nystrom method

Wen-Yen Chen et al. (2011) proposed the Nystrom method, an efficient technique that speeds up the computation by generating low-rank approximation. It is usually applied to large-scale matrices. It is assumed that a matrix can be approximated by using even a small subset of columns. Hence it can be considered to be a good solution when working with high dimensional data sets.

It should be noted that one assumption to this method is that the original matrix can be approximated by saving few columns; the evidence can then be shown in practical applications in singular value decomposition (SVD) and principal component analysis (PCA). This is an

interesting case since it reduces the computation time; it should also be noted that the sampling must be done in a way that generates exclusive information even from small subsets of columns.

2.3.3 Factor analysis (FA)

Factor analysis is used to show variability among observed and correlated variables $x_1, x_2, \ldots x_n$, in terms of a number of common factors which have lower number of unobserved variables plus a factor which is unique to each variable (Loehlin, 1998). Simply speaking, it is natural to have variation among observed variables because of variation with fewer unobserved variables. Observed variables are then modeled as a linear combination of the factors plus an error term.

The common factor (also known as latent variable) explains the correlation among variables because correlated variables have such factors in common. The reason behind using factor analysis is that we can summarize measurements using a smaller number of factors without losing too much information. Factor analysis can be considered as a data-reduction technique that reduces the number of variables. A good factor solution is both simple and interpretable.

2.3.4 Principal component analysis (PCA)

Principal component analysis is a dimension reduction technique that describes orthogonal directions of maximum variance in the original data, and it maps the data into a low-dimension manifold. While considering the independence of the observations along the rows, we are assuming that some values are correlated within a column vector (Ma, S., et al. (2006).

The objective of PCA is to reduce dimensions by a set of linear combination of given attributes that have the most importance in explaining variation in the given data. The first principal component corresponds to the maximum variation in the given data; as a result, the second

principal component is the next largest variation. It should be noted that each row in the data set always corresponds to the same type of observation.

Generally, once the eigenvector are found from the covariance matrix, we have to order them by eigenvalue from highest to the lowest which gives the component in order of significance. Then we have to form the feature vector (also known as matrix of vectors).

It can be constructed by the eigenvectors that we decided to keep and forming a matrix with these eigenvectors in the column. The feature vector is formed by taking the transpose of the vector and multiplies it on the left of the original data set. FinalData=RowFeatureVector × RowDataAdjust. In terms of our dimensionality reduction, we removed Eigen vectors out, and we kept the vectors that we wanted, hence data has been changed from being in terms of axes in to eigenvectors.

2.3.5 Indian buffet process (IBP) method

Latent or hidden variables are important components of many statistical models. The role of these latent variables may be to represent properties of the objects or data points being modeled that have not been directly observed, or to represent hidden causes that explain the observed data. In all of these representations, the difficult problem is deciding which features an object should possess.

The set of features possessed by a set of objects can be expressed in the form of a binary matrix, where each row is an object, each column is a feature, and an entry of "1" indicates that a particular object possesses a particular feature. Thus, we focus on the problem of defining a distribution on infinite sparse binary matrices.

Figure 2.3 shows the Chinese restaurant process (CRP) in which each data point belongs to a cluster; hence, each cluster is exclusive. In the Chinese restaurant process, we have a number of

customers (as data points) and an infinite number of tables (as clusters). Each customer will enter and will be placed at exclusive tables; however in grouping data, each data point is contained in many clusters at once and we can see clusters overlapping in this case; (co-occurring hidden features in a set of observations) (Wang and Blei, 2009).

To encode this Thomas L. Griffiths et al. (2005) define probability distribution over an equivalent class of sparse binary matrix which they have a finite number of rows and an infinite no of columns.

This approach is based on a probability distribution over equivalence classes of binary matrices with a finite number of rows, corresponding to the data points, and an unbounded number of columns, corresponding to the latent variables. Each data point can be associated with a subset of the possible latent variables, which we refer to as the latent features of that data point. Each object can possess multiple features (What distinguish CRP with IBP).

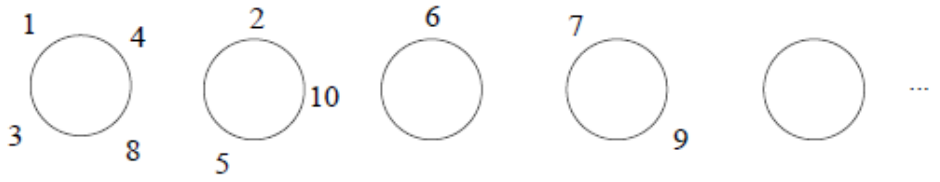


Figure 2. 3 Chinese restaurant overviews, numbers are costumers, and circles are features (Wang and Blei, 2009).

Here we are considering that we have *N* objects, represented by an $N \times D$ matrix *X* where the *ith* row of this matrix, *xi*, consists of measurements of *D* observable properties of the *ith* object, and *K* features, and the *ith* object will possess *Kth* feature by the binary variable z_{ik} which is a random variable. This binary variable is from a feature matrix *Z* of $N \times K$, where *N* is the number

of customers (corresponding to a finite number of rows in Matrix *Z*) and *K* are features (corresponding to an infinite no of columns in matrix *Z*) (Thomas L. Griffiths et al, 2011).

To indicate the latent feature values for all *N* objects, the model is identified by two things prior over features, *p (F),* which indicates the number of features, their probability, the distribution over values associated with each feature, a distribution over observed property matrices conditioned on those features, and *p (X|F)* which determines how these features relate to the properties of objects.

A prior *on F* can be defined by specifying priors for *Z* and *A* separately, with *p (F) = P (Z) p (A).* We will focus on defining a prior on *Z,* since the effective dimensionality of a latent feature model is determined by Z. This prior allows matrix having unbounded number of columns. Non-zero columns follow a *Poisson (α),* hence IBP is controlled by the hyper parameter α. The IBP has a single hyper parameter (α), which controls both the number of feature per object and the total number of features.

Assuming that Z is sparse, we can define a prior for infinite latent feature models by defining a distribution over infinite binary matrices. We have two desiderata for such a distribution: objects should be exchangeable, and inference should be tractable. Indian buffet process with linear likelihood is given by a linear-Gaussian model. Fig 2.5 describes this model where X is data matrix (application dependent) N×D matrix of observation and Z is a feature matrix.

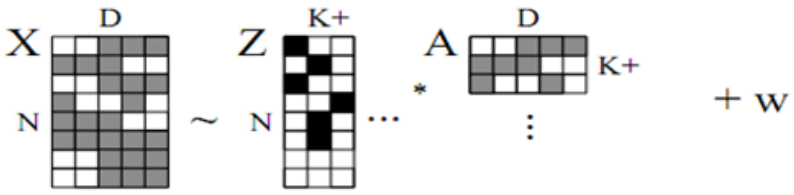


Figure 2. 4 Latent feature variable model (Griffiths et al 2011).

$N \times K$ binary matrix of feature assignments, where each binary Z_{nk}, denotes whether k is present in observation. A is the value of feature k; A is a parameter which is required to generate X given feature matrix Z. Each row of A can be interpreted as a vector characterizing some latent feature in the environment. $K \times D$ matrix of features: each A_{kd} indicates the value of feature k along dimension and ϵ and is Gaussian white noise. ϵ Observation noise in our data which is independent of Z and A and uncorrelated across observation with zero mean Gaussian with noise variance σ_n^2.

2.4 Comparison of dimension reduction methods using clustering algorithms

As mentioned before, we introduced several dimension reduction methods. For the sake of comparison of all these models, we applied all of these methods to the clustering algorithms, checked the accuracy of the clustering, and computed the computational cost for each method.

Clustering is one the most used techniques for exploratory data analysis, with possible applications in statistics, biology, psychology and etc. It can be considered the most important unsupervised learning problem, as it refers to the problem of finding hidden structure in unlabeled data.

Clustering and any other classifying methods detect complex structures in data sets. The simple definition of clustering is grouping similar elements in data points; data points in the same cluster should be similar to each other and those who are dissimilar should be in different clusters.

If we use similarity scores between any two elements, data points that are very similar to each other will have high scores; on the other hand, if we use distance scores between two objects, data points in the same cluster should be closer to each other. Simply speaking, clustering is unsupervised classification of data into groups, while supervised learning classifies data into known groups, where grouping of the data gives us useful information retrieval from the data.

There is no unique clustering result for a given data point; hence, there are different types of clustering algorithms. Mathematically, given a finite set $X \subseteq R^d$ of N distinct data points x_i, $i = 1, \dots, N$, and if we classify each data point in one cluster, we can define clustering as: if there are N_{cl} clusters and we seek S_j , $j = 1,2, \dots, N_{cl}$, then $S_j \cap S_k = \emptyset$, $\forall_j \neq k$ and $x_i \in S_j$ for one j (Lee and Daniels, 2005).

2.4.1 Categories of clustering algorithm

Many clustering algorithms exist, therefore it is difficult to categorize them completely; as a result, and only a few possible taxonomy of clustering approached have been mentioned here: hierarchical, model-based, and partitional techniques (Xu, and Wunsch, 2010).

Hierarchical approach

Figure 2.5 shows the agglomerative and divisive classification. Agglomerative clustering starts from *xi* as an individual cluster and combines the two similar subsets iteratively while in divisive approach we start with a cluster and divide it in to two clusters iteratively (Al-Akwaa, 2012).

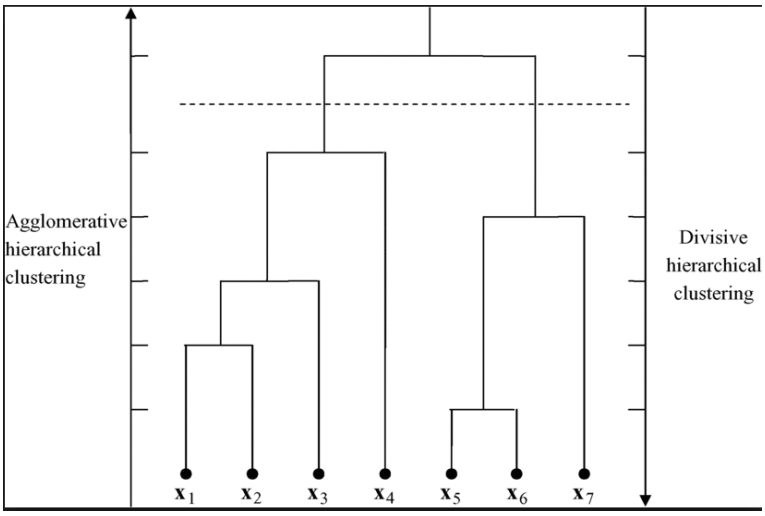


Figure 2. 5 Agglomerative and divisive classification (Al-Akwaa, 2012)

Partitional approach

In this approach, we partition the data set in to N_{cl} clusters and then data sets are randomly assigned to the clusters to minimize or maximize a criterion such as in Figure 2.1.

$$\min \sum_{j=1}^{N_{cl}} \sum_{CL_{j(i)}} \left|\left|x_i - m_j\right|\right|^2 \qquad (2.1)$$

The mean of the *j*th cluster and $CL_{j(i)}$ represents that x_i is in the *jth* cluster and the goal is to find the means and clusters minimizing. *K-mean* algorithm is an example of such approach (Krishnapuram, Raghuram, and Keller, 1993). Figure 2.6 shows the k-mean algorithm, an unsupervised, non-hierarchal, non-deterministic and iterative method.

We have to calculate the distances of each data point to the clusters for each data point, then data point are classified to the closest cluster. The process should be repeated until the clusters reach stability, which means that there won't be any movement of data points to clusters.

The main advantage of using *k*-mean is that while dealing with large number of variables, *k*-mean has low computational cost compared with hierarchical clustering; however, the weak point of such a method is distinct in comparison to the outcome of such clusters. For example, results obtained from *k*-mean can vary for different values of *k*.

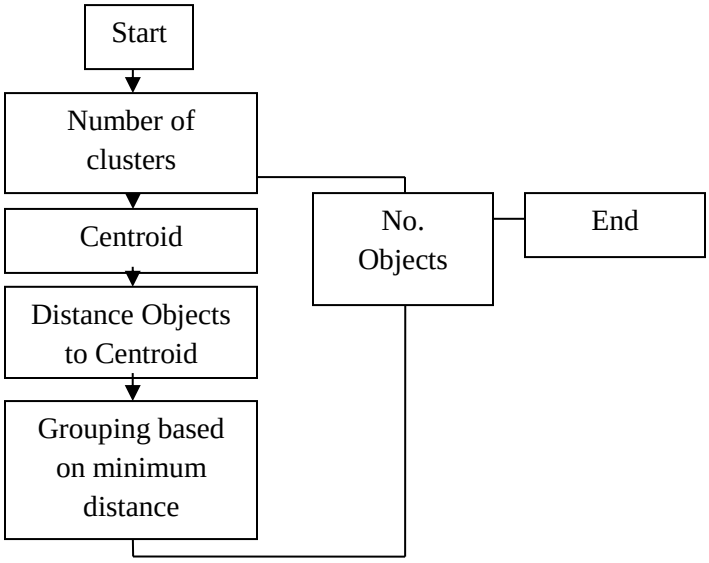


Figure 2. 6 k-mean clustering overview

Model-based approach

Zhiqiang et al. (2008) utilizes a hypothesized model and assumes that data are generated by a mixture of probability distributions, and then they find that the distribution that best approximates to the model to fit the data in to the model, and finally they perform clustering on this approximation.

2.5 Dynamic analysis for time series microarray data

Monitoring changes in a time series microarray data provides useful information regarding the gene expression pattern. Microarray arrays are collections of DNA spots attached to glass slides

in which RNA expression level would be measured (Brockwell, 2002). This can be regarded as high-throughput data collection.

Studying microarray data dynamics provides a deep understanding of complex biological systems. Static microarray data studies mostly focus on the measurement of microarray data in different samples while in time series microarray data studies a temporal process is mainly measured and static microarray data from a sample are *iid* (independent identically distributed); however, in a time series microarray, data have autocorrelation among objects.

A microarray experiment is a collection of different genes in a row and the columns represent the time at different stage of expression development and each cell w_{ij} is the measured expression level of gene *i* in time *j*. Figure 2.7 is a description of microarray matrix.

Steps to follow in microarray time series analysis

Firstly, the time series data is analyzed and displayed, and then the properties of time series model is computed and displayed. These functions are then combined to fit to data and checked that the proprieties of the model match with those data in a suitable sense. Having found the appropriate model, we can then use it in conjunction with the data to forecast future values of the series.

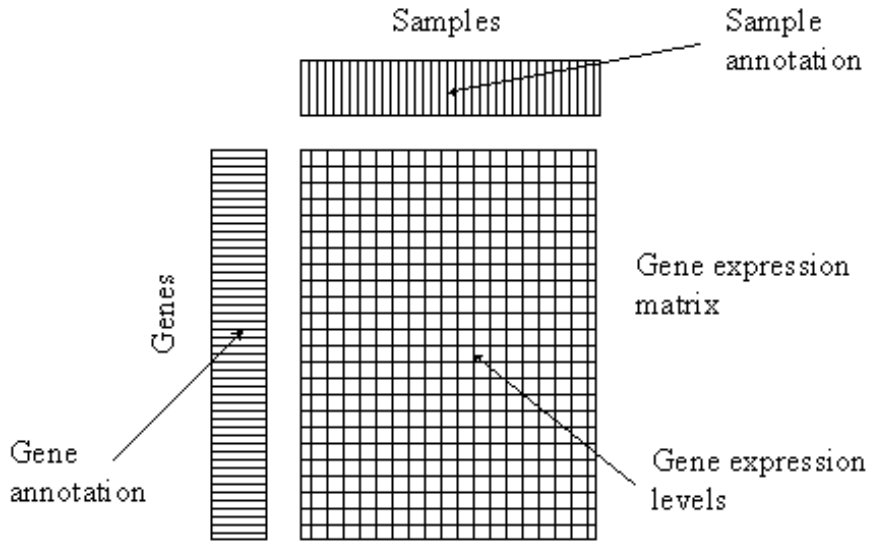


Figure 2. 7 Time series gene expression matrix overview (R Xu, DC Wunsch, 2010)

Time series are best considered as objects of some vector valued (multivariate) time series $\{X_t\}$ having not only serial dependence within each component series $\{X_{ti}\}$, but also interdependence between the different component series $\{X_{ti}\}$ and $\{X_{tj}\}$ (Brockwell 2002). In multivariate time series, any component series $\{Xt1\}$, $\{Xt2\}$, etc. could be studied independently as univariate time series.

Each can be characterized by its own mean and auto covariance function. Such an approach, however, fails to take into account possible dependence between any component series, and such cross-dependence may be of great importance in predicting future values of the any component series. By applying suitable dimension reduction methods, we can derive variables that are nearly uncorrelated or orthogonal. Thus, the problem for multicollinearity among the variables can be solved by using such variables.

CHAPTER III

OVERALL RESEARH METHODOLOGY

This chapter begins with the overall methodology used for the current research, starting with description of the data used for the research, applying different dimensional reduction methods, and finally employing a brief description of the dynamic analysis used for the data. A schematic of the methodology of this research is shown in Figure 3.1 and its five modules can be divided as follows: (1) data preprocessing techniques, (2) dimensional reduction methods on the processed data, (3) fitting time series model and assessing the quality of fitted model, and (4) forecasting to check for the goodness of the fit for the proposed model.

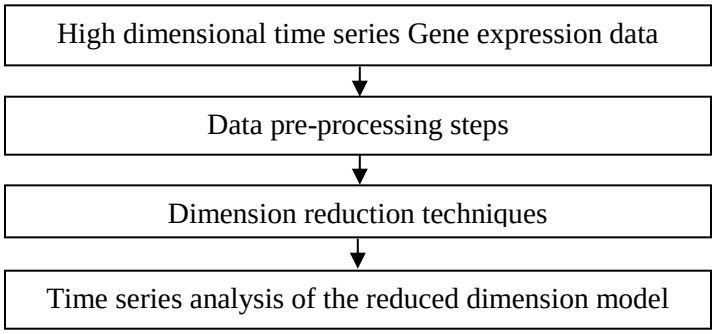


Figure 3. 1 A schematic of the methodology of this research

3.1 Microarray data preprocessing

The preprocessing steps used in this research include normalization, scale transformation, and management of missing values. Without performing such steps we cannot perform any pattern analysis. The processed data set can be further be used for any analysis (J.Herrero, et al., 2002). In sequencing data and microarray data, normalization is considered as an important step in gene expression analysis as it removes sources of variation.

The first step is to find missing values. The excess of such missing values can cause problems in analysis. To bypass this issue, the missing values were filled with zeros (Wang, X. et al., 2006), in which we removed the unknown ratios.

In sequencing data and gene expression data, normalization is considered an important step in gene expression analysis in which sources of variation are removed. Knowlton, N., et al (2004) proposed a method in which these estimators are determined as follows. After calculating the mean and standard deviation (SD), raw data beyond +2 SD above the mean was cut and again mean and SD was calculated and data beyond -2 SD below the mean was cut.

The trimmed data was then subjected to a nonlinear curve fitting procedure using MDAT v1.0, a Matlab toolbox that it can be downloaded at http://alumni.cs.ucsb.edu/~wychen/sc.html. We took expressions at or below a user-selected percentile of data to eliminate highly expressed values.

This data was curve fitted as follows. Using standard deviation and mean, a Gaussian CDF was generated and compared to the actual data's CDF and then the user was presented by a 10 bar histogram superimposed by a Gaussian distribution corresponding to the optimized estimators. After a good fit was agreed upon from users, the estimates were outputted. After the best fit was selected, the data was *Z*-transformed [$Z = (x - \mu)/\sigma$] yielding mean = 0 SD = 1.

Many variables in biology have log-normal distributions; meaning that after log-transformation, the values are normally distributed and data points will be spread more uniformly in the graph, Log transformations make positively skewed distribution more normal. Hence, the data were log-transformed with negative values substituted with the log of the minimum positive values (Keene, 1995).

3.2 Dimension reduction techniques

After performing preprocessing steps on the raw data, we can now apply different dimensional reduction methods in order to search for the best techniques from the computation cost perspective.

Working with high dimensional data using *n* points, the similarity graph has *n*-by-*n* dimension, which can cause time and memory issues; therefore, dimension reduction is a useful task to perform. On the other hand, while working with eigen Laplacian, or Nystrom method, it is necessary that we search for the similarity graph that overcomes such difficulties.

For computational advantages we can try to make the matrix sparse; this sparse representation of the data allows the matrix to store more efficiently rather than storing $n\text{^}2$ entries individually. Based on the computation cost performance and sparse structure, a *t*-nearest graph was selected.

For the sake of comparison we applied different clustering algorithms on all of the reduced dimension methods. We used normalized clustering according to Jordan and Weiss (2002), and Nystrom method (Fowlkes, et al., 2004). Comparing the results with FA, PCA, and IBP show that Nystrom method has lowest computation cost; however, it has a very poor clustering accuracy. On the other hand, Factor analysis shows a good performance and with lower computation cost compared to the other methods.

3.3 Dynamic analysis of high dimension data

To estimate gene regulatory network from time-series data there are several problems that need to be solved. The most important issue that we have to address is to identify a large network from a large number of genes originating from smaller number of experiments (times). This is a challenging task, since the system is underdetermined.

The goal for this part is to fit time series model on several reduced-dimension techniques and check the performance of each techniques. We have applied the sparse vector autoregressive (SVAR) model to estimate gene regulatory network based on gene expression profile from the time-series microarray experiment on two data sets. Autoregressive coefficients were estimated using several penalized regression methods like ridge, SCAD, and lasso.

After the 1-step ahead forecast was used and the goodness of fit was checked using the coefficient of determination, denoted by R^2, the results were compared with regards to performance.

3.4 Summary

After applying several dimension reduction techniques (DRT), we compared all the reduced dimension models with regard to their performance and computation cost. Then we started our major focus of this research: the dynamic analysis on the reduced models.

After estimating the autoregressive coefficients, 1-step ahead forecast was used and the goodness of fit was checked using coefficient of determination denoted by R^2 and the results from the above mentioned metrics were compared with regards to performance.

CHAPTER IV

APPLICATION OF VARIOUS DIMENSION REDUCTION TECHNIQUES IN TIME SERIES MICROARRAY DATASETS

In this chapter, five dimension reduction techniques (DRT) were conducted on two microarray data sets. The research starts with eigen Laplacian method and Nystrom method. In both methods, selecting the best similarity matrix is searched. All the above DRT's were further applied to a selected clustering algorithm for the sake of comparison for accuracy purposes. It shown that for the sake of comparison in accuracy and computation cost, factor analysis (FA) outperforms the rest of the DRT methods.

4.1 Research methodology

In this chapter, before applying any method we used preprocessing techniques such as data normalization and transformation. After the data was prepared we proceeded to the next step, which is to apply various dimension reduction techniques on both data sets.

Among dimension reduction methods used in this thesis, eigen Laplacian, and Nystrom methods have many factors in common. The major step in both methods is to search for similarity graphs. Building a similarity matrix on high dimensional data has time complexity; in order to overcome

this issue we used different similarity graphs, and selected the one with the lowest computation cost. Although Nystrom method shows improvement on the time of performance, the resulted clustering has low accuracy compared with the other DRT's. The reason behind this is due to the generation of low-rank approximation on the similarity graph instead of using the complete similarity graph.

Although using the complete similarity graph may face complexity in time of performance, but it can maintain its quality. On the other hand, factor analysis results in the best accuracy. Figure 4.1 shows the process used in dimension reduction methods used in this research

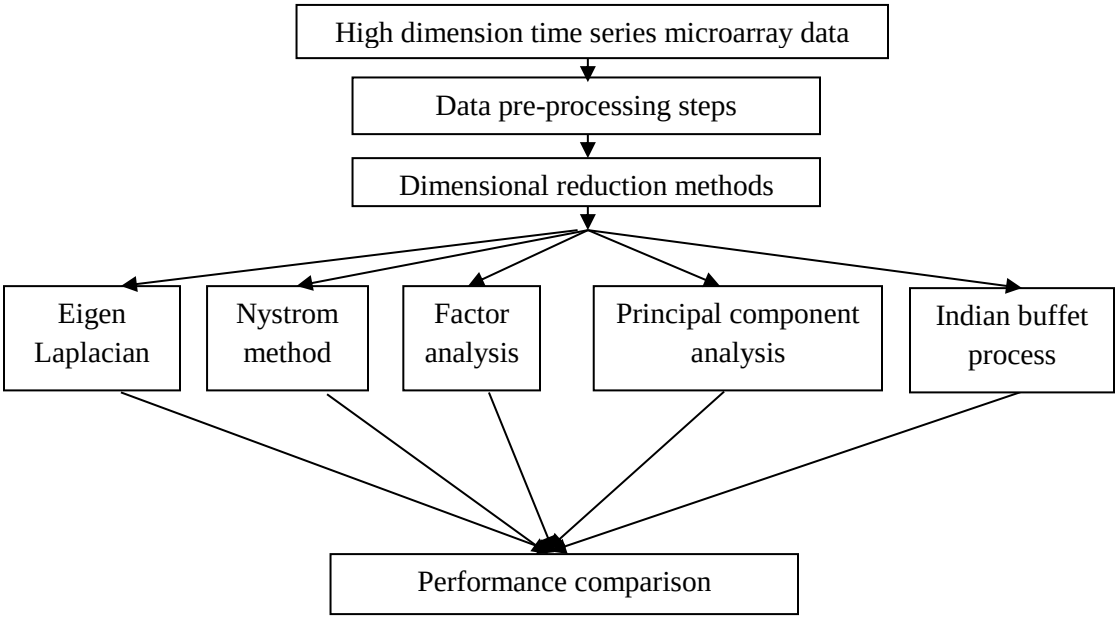


Figure 4. 1 Procedures used for dimension reduction and comparison

4.2 Pre-processing steps

The original microarray data contains noise, missing values, and systematic variations arising from experimental procedure. Without performing such steps, we cannot perform any pattern analysis. The processed data set can be further be used for any analysis (J.Herrero et al, 2002). The preprocessing steps used in this research include normalization, transformation, and management of missing values.

4.2.1 Normalization method applied to the reduced dimension data

The first step is to find missing values. The excess of such missing values can cause problems in analysis, specifically in the case of applying *k*-mean algorithm on reduced dimension data. We may face problems because all the methods in clustering techniques rely on a distance matrix; if these patterns do not have enough points in common, the clustering can fail automatically. To bypass this issue, the missing values were filled with zeros (Wang, X., et al. 2006) in which we removed the unknown ratios.

In sequencing data and gene expression data, normalization is considered an important step in gene expression analysis, during which sources of variation are removed. Knowlton, N., et al (2004) proposed a method in which these estimators are determined as follows: after calculating the mean and standard deviation (SD), raw data beyond +2 SD above the mean was cut and again mean and SD was calculated and data beyond -2 SD below the mean was cut.

The trimmed data was then subjected to a nonlinear curve fitting procedure using MDAT v1.0. We took expressions at or below a user selected percentile of data to eliminate highly expressed values.

This data was curve fitted as follows. Using standard deviation and mean, a Gaussian CDF was generated and compared to the actual data's CDF. The user was then presented with a 10 bar histogram superimposed by a Gaussian distribution corresponding to the optimized estimators; after a good fit was agreed from user, the estimates were outputted. After the best fit was selected, the data was Z-transformed [$Z = (x - \mu)/\sigma$] yielding mean = 0 SD = 1.

Many variables in biology have log-normal distributions, meaning that after log-transformation, the values are normally distributed and data points will be spread more uniformly in the graph, Log transformations make positively skewed distribution more normal. Hence, the data were log-transformed with negative values substituted with the log of the minimum positive values (Keene 1995).

4.3 Dimension reduction techniques

4.3.1 Eigen Laplacian

4.3.1.1 Similarity graph selection

The first step in eigen Laplacian is to choose the best similarity graph. Similarity graphs describe the pair wise similarity of the points in a given dataset. Using pair relationships between the data point, i.e., similarities s_{ij} to describe similarity of x_i and x_j for all the given data, and using data points as vertices and weighting each edge with the corresponding similarity, i.e.$w_{ij} = s_{ij}$, G is called the similarity graph of this dataset.

By constructing the similarity graph, we model the local neighborhood effectiveness which can be found in many areas such as computer science (e.g., Y.Weiss et al 2002, Fowlkes, et al. ,2004); however, working with high dimensional data using *n* data points, the similarity graph has *n*-by-*n* dimension, hence eigen decomposition can face time and memory difficulties, especially

in finding and storing *k* eigenvectors of a Laplacian matrix; therefore, we have to construct such similarity matrices that overcome such difficulties.

In fact, Liu et al. (2004) state that the majority of time is actually spent on constructing the pair wise distance and affinity matrices. For computational advantages, we can try to make the matrix sparse; this sparse representation of the data allows the matrix to store more efficiently rather than storing n^2 entries individually and in which we zero out elements in similarity matrix (i.e. sparsify the matrix).

This is the most common approach to remove such difficulties because spare representation can handle the memory bottleneck; from the obtained similarity matrix we can find the Laplacian matrix. In the following section we try three different similarity matrices. Figure 4.2 shows possible approximation techniques to avoid a dense similarity matrix (Chen, Wen-Yen, et al 2008).

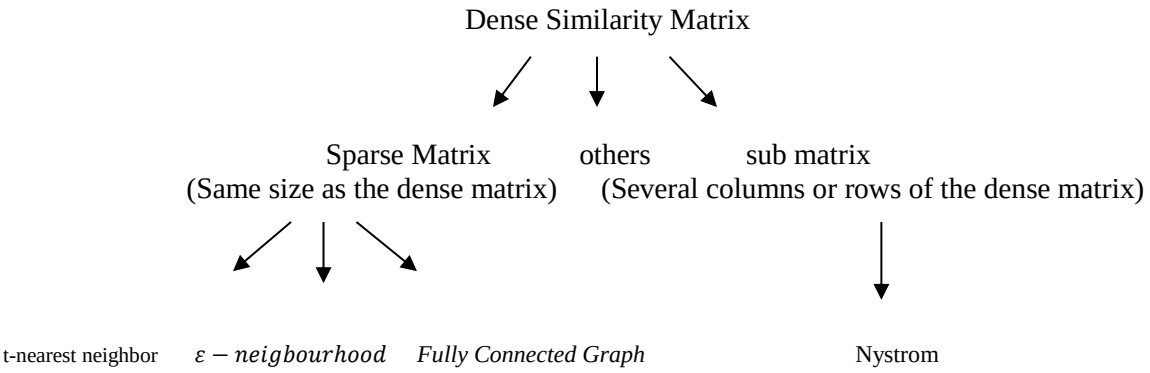


Figure 4. 2 Approximation techniques to avoid storing the dense similarity matrix (Chen, Wen-Yen, et al 2008).

Fully connected similarity graph

In such a similarity graph, all the vertices $v_i \; and \; v_j$ with positive similarity will be connected with each other assuming that weight of all the edges are $s_{ij} \neq 0$; as mentioned earlier, by constructing the similarity graph we model the local neighborhood relationships between the data point by using the similarity function which encodes mainly local neighborhood. Kernel methods (KMs) are class of algorithms; we can call these algorithms as the generalization of inner product (Inderjit S., Yuqiang Guan, and Brian Kulis, 2004).

The approach used in KMs is done by mapping the data in a high dimensional space in which each coordinate correspond to a feature for that particular data (e.g. transforming the data into a set of points in a Euclidean space. Choosing the appropriate kernel and tuning the parameter depends largely on the problem dealing with. We have different kernel functions at hand.

Polynomial Kernel

The polynomial kernel is a non-stationary kernel that allows modeling of all features up to the order of the polynomial where alpha is the slope, the constant term is c, and the polynomial (Inderjit S., Yuqiang Guan, and Kulis, 2004).

$$k(x, y) = (ax^T y + c)^d \tag{4.1}$$

Linear Kernel

Which is the simplest kernel function, which is nothing but the inner product xy plus a constant term (Shawe-Taylor, N., and A. Kandola 2002).

$$k(x, y) = x^T y + c \tag{4.2}$$

Gaussian Kernel

Gaussian Kernel is the example of radial kernel function (Shawe-Taylor, N., and A. Kandola 2002).

$$s_{ij} = exp\left(-\frac{d(x_i, x_j)^2}{2\sigma^2}\right) \tag{4.3}$$

The adjustable parameter σ plays an important role in performance of the kernel. The choice of the kernel function is straightforward based on the information we are expecting to obtain from the data. S.Sathiya et al. (2003) explained the behavior of radial basis function (RBF), and reasons why RBF is the most popular kernel used for the model selection.

The linear kernel is a special case of RBF kernel, since the linear kernel with penalty parameter C has the same performance as the RBF kernel with parameters (C,γ). Also, as mentioned above, the numerical difficulties due to existing of large number of parameters in case of polynomial kernel make RBF kernel the best choice.

The standard approach would be to use a Gaussian similarity function standard approach for fully connected graphs; the similarity of pattern $x_i \; and \; x_j$ is Eq. (4.3) where σ is the Gaussian function parameter. Eq. (4.3), has the following properties:

1. Similarity measurement is symmetrical $s_{ij} = s_{ji}$

2. The Gaussian similarity measurement is non-negative and $0 \leq \; s_{ij} \leq 1$

3. Objects having higher distance scores between them have smaller similarity scores, and vice versa.$\| x_i , x_j \| > \left\| x_i , x_j \right\|$, then $s_{ij} < s_{ji}$.

The σ is a parameter that controls the size of these neighborhoods. And *d* is the metric. Since we searched the whole parameter space in this method, it has a high computational cost. Kernel methods are used in pattern recognition in data mining. In general, in pattern analysis we would like to study the notion of similarities of our inputs in order to find relationships in the data. The kernel function of x_i and x_j in sample space is the same as Eq. (4.3).

As mentioned above, the greater the parameter σ in the similarity measure shows better similarity between $x_i \; and \; x_j$, which means all the objects are too similar; on the other hand, small values for such a parameter makes the pattern in the feature space break apart from each other; as a result, parameter selection is necessary in this process. It should be mentioned that parameter selection obtained from Gaussian similarity measurement by using a Gaussian kernel function that used a kernel method is reasonable.

Gaussian Parameter Selection using Cross-Validation Method

Referring to Eq. (4.3), we can realize the following results (Lee and Daniels, 2005): the greater the parameter σ in the similarity measure shows better similarity between $x_i \; and \; x_j$ which means that all the objects are too similar; on the other hand, small values for such a parameter indicate possible difficulties in clustering due to high grades of dissimilarity. Hence this will lead us in parameter selection for Gaussian function. Before choosing any method for a parameter selection, there is a preprocessing step in which SVM requires the normalization of input data (which brings them to the scale (0, 1)).

The advantage of normalization is to transfer non-numeric values to numeric values; by transferring the values between zero and one, assures that the numeric values are on the same scale so that those with large original scales won't bias the output. Generally, it avoids numerical

difficulties, e.g. the kernel values of polynomial or linear kernel largely depend on their inner product. Therefore, normalization is a good solution to bypass these numerical difficulties Kaibo Duan et al. (2003).

The main method for parameter selection is cross-validation (Duan et al. 2003). Cross-validation is a method that tries to select the best model from a whole parameter space and obtain the best parameter by a comparison of results; hence it has a high computational cost, especially in case of high dimensional data. There are basically two variables for RBF kernel based SVMs: Kernel's model parameter; namely, the width of kernel σ and penalty parameter. Identifying the best parameter should be tuned for better performance.

The process in the *n*-fold cross-validation is to divide the data in to *n* subsets with equal sizes. One subset is tested by using the train classifier training on the remaining *n-1* subsets for different values of γ, then we have to evaluate each trained classifier to compute the error rate, and then select the γ with the lowest error rate. Cross-validation accuracy can be called as the percentage of data that has been classified correctly with the minimum error rate.

One way to select such parameter space is "grid-search", where by choosing parameter space of $[2^{-10}, 2^{4}]$, and the penalty coefficient with parameter space of $[2^{-2}, 2^{10}]$, we can further use each combination of the above parameters to be checked by cross-validation. The one with the best accuracy is selected. Then the selected parameters will be used for the final model.

Grid-search can be considered a very simple method for such selection and there are some advanced methods that can be used. Many advanced methods are iterative processes with a high computation cost. Each parameter can be selected independently; hence grid-search can be easily parallelized.

***K*-nearest neighbors' similarity graph**

Here we connect the vertex v_i with vertex v_j if v_j is among the *k*-nearest neighbors of v_i. This leads us to the directed graph since $w_{ij} \neq w_{ji}$ for all the weights; hence neighborhood relationship is not symmetric. Figure 4.3 shows that the k-nearest neighbor of v_3 is v_2, but the *k*-nearest neighbor of $v_2, is\ v_1$, in which $k = 1$, hence the t-nearest neighbor graph is directed. There are two approaches to convert it to undirected graph (Beyer, et al. 1999).

1. Normal *k*-nearest Neighbors: we connect the vertex v_i with vertex v_j with an undirected edge by ignoring the direction of the edges, if either of them is among *k*-nearest neighbor of the other one.
2. Mutual *k*-nearest neighbor graph: we connect the vertex v_i with vertex v_j if v_i is among the *k*-nearest neighbor of v_j and also v_j is among the *k*-nearest neighbor of v_i

Figure 4. 3 The k-nearest neighbor of vertex $\boldsymbol{v_3}$ is $\boldsymbol{v_2}$ (Beyer, et al. 1999)

Selecting number of neighbors using cross-validation method

While applying *k*-nearest neighbor, i.e., KNN, we have to select the optimal value for *k*. In general, for large value of k the effect of noise in classification reduces but makes the boundaries between classes less distinct. It is important to mention that the accuracy of *K*-NN method can severely be damaged by presence of noise.

The most popular way to search such optimal value is cross-validation. Cross-validation is an automatic parameter selection; when we test a set of *k* values between k_{min} and k_{max}, we have to compute the average error rate or sum of square of error,

$$CV = \sum_{v=1}^{V} \frac{e_v}{v} \tag{4.4}$$

Where e_v is an error rate with $X = v$; then we have to select the optimal *k* as:

$$K = \arg\{min\widehat{CV}: \widehat{k_{min}} \leq k \leq k_{max}\} \tag{4.5}$$

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its *k*- nearest neighbors measured by a distance function-nearest neighbor in a very simple algorithm in which all the vertices will be classified to a class with the most common amongst them.

ε −Neighborhood similarity graph

Although several methods are available to sparse the similarity matrix (Barnard, 2005), one way to have a sparse similarity matrix is to zero out S_{ij} that are smaller than a threshold. This technique will conquer the memory problem. We connect the vertex v_i with vertex v_j if their pair wise distance is smaller than ε; d $(x_i, x_j) < \varepsilon$. This graph is considered to be unweighted since all the connected points are the same or at most ε.

Parameter selection using Pearson-correlation coefficient (PCC)

It is a typical approach to use the Pearson correlation coefficient as a similarity measure. The basic idea behind a similarity measure between two items is to identify a set if users which are related to the objects and then apply the computation techniques. There are several similarity

measuring techniques; however, the most popular one is the Pearson coefficient measure. Pearson correlation coefficient evaluates the deviation from the mean rating of objects:

$$sim(i,j) = PCC(i,j) = \frac{\sum_k (r_{ki} - \overline{r_i})(r_{kj} - \overline{r_j})}{\sqrt{\sum_k (r_{ki} - \overline{r_i})^2 \sum_k (r_{kj} - \overline{r_j})^2}} \quad (4.6)$$

where $\overline{r_i}$ is the mean rating of the object i and k sums over users that have both rated items i and j. The basic idea in correlation thresholding is to set a minimum correlation weight which a neighbor must have in order to accept into user's neighborhood. Therefore, we have to find PCC for different threshold values and compare them by mean absolute error (MAE).

4.3.1.2 Determining about the number of clusters for comparison reasons

There are a variety of methods for such selection (Still, S. and Bialek, W. 2004 and S. Ghemawat, H. Gobioff, and S.T. Leung 2003). From the current research, I chose singular value decomposition (SVD) for selecting the number of clusters because it has the ability to eliminate large portion of data.

Eigenvectors points to the direction in which data points exhibit the maximum variability. Mathematically speaking there exist large number of eigenvectors, of which only few will point in directions of such maximum variation.

4.3.2 Nystrom method

One main disadvantage of the eigen Laplacian using ε −neighborhood similarity graph is that is uses the complete similarity matrix, unlike Nystrom method which uses the dense sub-matrix of similarity matrix instead of computing the similarity between all data points. On the other hand, the eigen Laplacian method using ε −neighborhood similarity graph is worth attempting since the quality of clustering is high (Shawe-Taylor and Cristianini 2004).

In the Nystrom method, the algorithm uses a low-rank approximation to an $n \times n$ Gram matrix G; considering s a similarity matrix $n \times n$, and randomly dividing the similarity matrix to l point, l<n and B is an $l \times (n - l)$ matrix of similarities between l sample points and *(n-l)* remaining points. W is a $n \times l$ matrix consisting of A and B^T and C contains similarities between *(n-l)* remaining points we have,

$$s = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}_{n*n} \quad \text{And } W = \begin{bmatrix} A \\ B^T \end{bmatrix} \tag{4.15}$$

Where, $A \in R^{l \times l}, B \in (n - l), and\ C \in R^{(n-l)\times(n-l)}$, and C contains the similarities between all *(n-l)* remaining points and W is a $n \times l$ matrix consisting of A and B^T.

$$S = WA^{-1}W^T = \begin{bmatrix} A & B \\ B^T & B^T A^{-1} B \end{bmatrix} \tag{4.16}$$

That is, C is replaced by $B^T A^{-1}$B. The normalized Laplacian is $L = D - S = I - D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$, where D is a diagonal matrix with $D_{ii=\sum_{j=1}^{n} S_{ij}}$. Since the degree matrix D is diagonal, its reciprocal square root, $D^{-\frac{1}{2}}$, is simply defined as a diagonal matrix.

In chapter two, section 2.3.1, from definition 8. Properties 1, we can easily prove that the normalized Laplacian matrix is symmetric and positive-semi definite. Eigenvalues and eigenvectors together they provide eigen-decomposition of a matrix which annuluses the structure of a matrix.

Now that our Laplacian matrix is positive-semi definite it means eigen-decomposition of such matrices are always exist, and that their most important properties are that its eigenvalues are always positive or null and its eigenvector are pair wise orthogonal when their eigenvalues are different.

Laplacian matrices are *M*-matrices, i.e., one major property of such matrices is that the diagonal elements of them must be positive. When data in one cluster are not similar to those from other clusters, then non-zero element of *S* (and hence *L)* occur in a block diagram as such,

$$L = \begin{bmatrix} L_1 & & \\ & \ddots & \\ & & L_k \end{bmatrix}$$

From definition 8, properties 5 of chapter two, section 2.3.1, it shows that *L* have *n* non-negative real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ then their corresponding eigenvector $V \in R^{n \times k}$ is, $V = \left[v_1, v_{1,}, \ldots . v_{k,}\right] = D^{\frac{1}{2}}E$, where *vi* for *i=1,...,k* and,

$$E = \begin{bmatrix} e_1 & & \\ & \ddots & \\ & & e_k \end{bmatrix}$$

Where e_i; *i=1..., K* (in different length) are vectors of all ones. As $D^{\frac{1}{2}}E$ has the same structure as E. As was mentioned for the sake of comparison, clustering algorithms such as *k*-means which can easily cluster the *n* rows of *V* into *k*-groups, thus what one needs to do is to find the first *k* eigenvectors of L i.e., eigenvectors corresponding to the *k* smallest eigenvalues, and the eigenvectors are obtained in the form of $V = D^{\frac{1}{2}}EQ$, where *Q* is an orthogonal matrix, recalling Eq. (4.16) where is C is replaced by $B^T * A^{-1}$B.

Assuming that the eigen decomposition of *A* has the form A=$V_A \sum_A V_A^T$ where $\sum_A$ contains the eigenvalues of A and V_A are the corresponding eigenvectors. Fowlkes, et al. (2004) implies that *S* has the eigen decomposition of $S = V\Sigma V^T$ where the approximate eigenvalues (Σ) and eigenvectors (V) of sd are,

$$\Sigma=\left(\frac{n}{n}\right)\sum A \text{ , V}=\sqrt{\frac{1}{n}}\text{W}V_A\sum_A^{-1} \tag{4.17}$$

Then we find the k eigen vector of L and conduct k-mean clustering to the data. To find the degree matrix, we compute the row sums of S. If we want to find S directly, we have to compute matrix-matrix product of C=B^T*A^{-1}B which is expensive: O($L(n-L)^2$). Fowlkes et at. (2004) propose a procedure.

$$S_1=\begin{bmatrix} A1_l + B1_{n-l} \\ B^T1_l + B^TA^{-1}B1_{n-l} \end{bmatrix}=\begin{bmatrix} a+b \\ b_2 + B^T(A^{-1}b_{1)} \end{bmatrix} \tag{4.18}$$

Where a,b_1 and b_2 represent the row sums of A, B, and B^T respectively, and 1 is a column vector of one's using Eq. (4.10) the computation cost becomes $O(L(n-L))$, and we simply have,

$$D^{-\frac{1}{2}}\tilde{S}D^{-\frac{1}{2}} = \begin{bmatrix} \bar{A} & \bar{A} \\ \bar{B}^T & \bar{B}^T\bar{A}^{-1}\bar{B} \end{bmatrix} \tag{4.19}$$

Where $\bar{A} = D_{1:l,1:l}^{-\frac{1}{2}}AD_{1:l,1:l}^{-\frac{1}{2}}$ and $\bar{B} = D_{1:l,1:l}^{-1/2}BD_{1+l:n,l+1:n}^{-1/2}$. We know that $L = D - S = I - D^{-\frac{1}{2}}SD^{-\frac{1}{2}}$ since L and $D^{-\frac{1}{2}}SD^{-\frac{1}{2}}$ have the same eigen space, we can obtain k eigen vector of L with eigen decomposition of $\bar{A} = D_{1:l,1:l}^{-\frac{1}{2}}AD_{1:l,1:l}^{-\frac{1}{2}}$, hence we have

$$\tilde{\Sigma}=\left(\frac{n}{l}\right)\overline{\Sigma_A} \text{ ,}\tilde{V} = \sqrt{\frac{l}{n}}[\frac{\bar{A}}{\bar{B}^T}]\,(\bar{V}_A)_{:,1:k}\left(\overline{\Sigma_A^{-1}}\right)_{1:k,1:k} \tag{4.20}$$

After computing the similarity matrix the following algorithm was used.

Input: Data points $x_1, \dots, x_n$; *l:*number of samples; *k:*number of desired clusters; $l > k$.

1. Construct $A \in R^{l \times l}$ and $B \in R^{l \times (n-l)}$ so that $[A \quad B]$ contains the similarity between $x_1, \dots, x_l$ and $x_1, \dots, x_n$
2. Calculate a=A$1_l, b_1 = B1_{n-1}, b_2 = B^T 1_l$ and $D = diag\left(\begin{bmatrix} a + b_1 \\ b_2 + B^T A^{-1} b_1 \end{bmatrix}\right)$. Here 1 represents a column vector of ones
3. Calculate $\bar{A} = D_{1:l,1:l}^{-\frac{1}{2}} A D_{1:l,1:l}^{-\frac{1}{2}}$ and $\bar{B} = D_{1:l,1:l}^{-1/2} B$ as $D^{-\frac{1}{2}} \tilde{S} D^{-\frac{1}{2}}$
4. Construct $R = \bar{A} + \bar{A}^{-\frac{1}{2}} \bar{B} \bar{B}^T \bar{A}^{1/2}$
5. Calculate eigen decomposition of R, R= $U_R \, \Lambda_R \, U_R^T$
6. Calculate $\bar{V}_A = \begin{bmatrix} \bar{A} \\ \bar{B}^T \end{bmatrix} \bar{A}^{1/2} U_R \Lambda_R^{-1/2}$, as the first *k* eigenvector $\hat{L}$
7. Compute the normalized matrix $\tilde{U}$ of $\tilde{V}$
8. Use *k*-means algorithm to cluster *n* rows of $\tilde{U}$ into *k* groups

Figure 4. 4 Clustering algorithm using Nystrom method (Fowlkes, et al. ,2004)

4.4 Accuracy comparison using clustering algorithm

Now that we are assuming give a data point $x_1, \dots, x_n$, and the selected similarities $s_{ij} = s(x_i, x_j)$ measured, we can use the following most common clustering algorithms. Figure 4.5 shows the unnormalized clustering algorithm

Let S$\in R^{n \times n}$ be the similarity matrix for the n points $x_1 \dots, x_n$, where s_{ij} describes the similarity between x_i and x_j. Let *k* be the desired number of clusters

Construt a similarity graph with adjacency matrix W as discussed before.

Compute the unnormalized graph Laplacian *L=D-W*

Compute the first *k* eigenvectors $u_1 \dots, u_k$ of *L*.

Let $U \in R^{n \times k}$ be the matrix xontaining the vectors $u_1 \dots, u_k$ as columns

For *i=1,...,n* let $y_{i\,i=1}^{n}$ into clusters $C_1, \dots, C_k$ using the *k*-means algorithm

Retrieve clusters $A_1, \dots, A_k$ by $A_i = \{j | y_j \in C_i\}$

Figure 4. 5 Unnormalized clustering (Luxburg, 2007)

It should be mentioned that this algorithm uses the eigen vectors of L, refering to proporteis of normalized Laplacian graph equal to the eigenvector of L_{rw} , which is the eigenvector of the normalized Laplacian L_{rw}; hence, it is called normlaized clustering. The following algorithm also used the normlaized Laplacian with matrix L_{sym}.

Figure 4.6 shows the normalized clustering according to Shi and Malik. Figure 4.7 shows the normalized spectral clustering according to Jordan and Weiss.

Let S∈ $R^{n\times n}$ be the similarity matrix for the n points $x_1 \ldots, x_n$, where s_{ij} describes the similarity between x_i and x_j. Let k be the desired number of clusters.

1. Construt a similarity graph with adjacency matrix W as discussed before
2. Compute the unnormalized graph Laplacian $L=D-W$
3. Compute the first k eigenvectors $u_1 \ldots, u_k$ of the normalized graph Laplacian L_{rw}
4. Let $U \in R^{n\times k}$ be the matrix xontaining the vectors $u_1 \ldots, u_k$ as columns
5. For $i=1,\ldots n$ let $y_i \in R^k$ be vector corresponding to the i-th row of U
6. Cluster thenpoints $(y_i)_{i=1}^{n}$ into clusters $C_1, \ldots, C_k$ using the k-means algorithm
   Retrve clusters $A_1, \ldots, A_k$ by $A_i = \{j | y_j \in C_i\}$

Figure 4. 6 Normalized clustering according to Shi and Malik (2000)

Let S∈ $R^{n\times n}$ be the similarity matrix for the n points $x_1 \ldots, x_n$, where s_{ij} describes the similarity between x_i and x_j. Let k be the desired number of clusters.

1. Construt a similarity graph with adjacency matrix W as discussed before
2. Compute the unnormalized graph Laplacian $L=D-W$
3. Compute the first k eigenvectors $u_1 \ldots, u_k$ of the normalized graph Laplacian L_{sym}
4. Let $U \in R^{n\times k}$ be the matrix xontaining the vectors $u_1 \ldots, u_k$ as columns
5. Form the matrix $T \in R^{n\times k}$ from U by normalizing the rows, i.e. se

$$T = t_{ij}{}_{i,j=1}^{n} \; with \; t_{ij} = \frac{u_{ij}}{(\sum_k u_{jk}^2)^{\frac{1}{2}}},$$

6. For $i=1,\ldots,n$ let $y_i \in R^k$ be the vector corresponding to the i-th row of T
7. Cluster points $(y_i)_{i=1}^{n}$ into clusters $C_1, \ldots, C_k$ using the k-means algorithm

Retrve clusters $A_1, \ldots, A_k \; by \; A_i = \{j | y_j \in C_i\}$

Figure 4. 7 Normalized spectral clustering according to Jordan and Weiss (1999)

Algorithm selection based on the three different types of Laplacian graph

There have always been debates regarding which algorithm should be used in clustering (Luxburg, 2007). There are recent papers that use a normalized Laplacian graph, such as Jordan et al (2000) and Shi et al (2000); Barnard et al (2005) uses unnormalized algorithm. Weiss (1999), found reasonable evidence regarding the preference of normalized over unnormalized.

One of the most important aspects of clustering is whether the selected algorithm that is implemented in which it constructs partition on a finite set of objects will converge to useful information or not. Clustering techniques basically partition sets of data into groups based on their similarities. Although clustering lacks the criteria of "goodness", it is possible to study these algorithms from a theatrical point of view to determine the quality of partitioning.

One important property of clustering is to produce proper partitions using a sufficient number of data points; choosing a wrong algorithm can change the partition drastically and add points to data set and basically destroy the whole idea of identifying a small-sample performance. However, normalized clustering is a reliable algorithm. Figure 4.8 shows the affinity matrix built based on ε −nearest neighborhood.

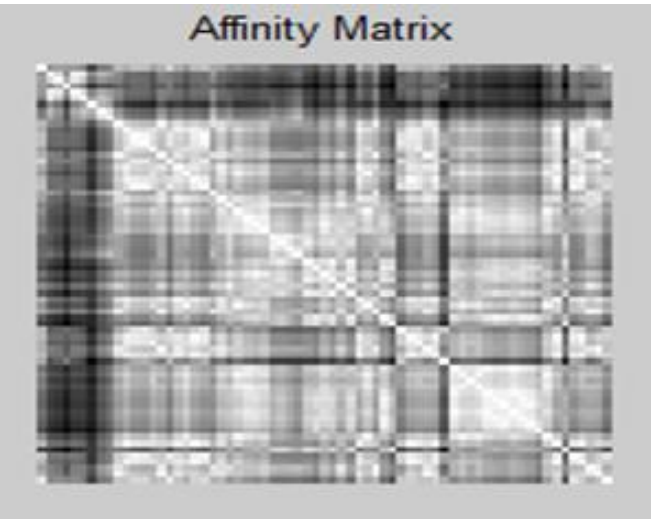


Figure 4. 8 Plot of affinity matrix

As mentioned before, the purpose of applying *k*-mean on all the algorithms is for the sake of comparing each dimension reduction techniques based on accuracy and quality of clustering. The clustering accuracy is measured using MATLAB to conduct this experiment (located at http://alumni.cs.ucsb.edu/~wychen/sc.html).

Normalized Mutual Information

The NMI between category label and cluster label is defined (Estévez, Pablo A., et al 2009).

$$NIM(CAT; CLS) = \frac{I(CAT;CLS)}{\sqrt{H(CAT)H(CLS)}} \tag{4.21}$$

Where $I(CAT;CLS)$, the mutual information between CAT and CLS, and the entropies are $H(CAT)H(CLS)$ and used to normalize the mutual information and set them in the range of [0,1]. The higher the NMI score, the better the clustering quality.

Clustering Accuracy

Cluster accuracy is defined as:

$$\text{Accuracy: } \frac{\sum_{i=1}^{n} \delta(y_i, map(c_i))}{n} \tag{4.14}$$

Where *n* is the number of data and $y_i \; and \; (c_i)$ are true category label and obtained cluster label. Below, the table shows the NMI and accuracy value for different cluster numbers for the algorithm and Nystrom approximation (Huang, et al. 2005).

4.5 Case Study

As discussed in the research methodology section, after the preprocessing steps but before applying clustering methods we have to select the best similarity graph based on the computation cost. Three different similarity graphs are considered: fully connected similarity graph, *k*-nearest neighbor's similarity graph, and ε −neighborhood similarity graph; however, before applying any of these methods, we have to find their selected parameter.

Using LIBSVM, integrated software for support vector classification following results were found (Chang and Lin, 2011): results of 10-fold cross validation SVM kernel with the best parameter σ for the fully connected similarity graph.

Table 4. 1 Parameter selection using cross-validation method in fully connected similarity graph

10 Fold Cross-validation accuracy	Selected parameter
85.07%	7.4

The *k*-nearest neighbors which are measured by similarity measure, e.g. distance function, in which the most common distance metric was used which is Euclidian distance. The results from the optimal selection of neighbors for *k*-nearest neighbor by means of cross-validations using the Classification toolbox for MATLAB, a percentage of error rate for different values of *k*, is below.

Table 4. 2 KNN selected parameter using cross-validation method

K	10	9	8	7	6	5	4	3	2	1
Error rate	98.71	97.10	96.98	95.38	95.35	87.11	89.40	81.89	82.40	80.18

As can be seen from the results above, *k*=1 is the best choice, this is a special case of the *k*-nearest neighbor algorithm which demonstrates that the object is assigned to the class of the nearest neighbor.

We used the Pearson coefficent measure for parameter selection $\varepsilon -$ neighborhood similarity graph. Table 4.3 has the following results. Based on computation performances shown in Table 4.3, we chose e-neighborhood Similarity graph with $\varepsilon = 0.4$ which has a low computation time and MAE.

Table 4. 3 Parameter selection ε-neighborhood similarity graph in using Pearson-based approach

Epsilon value	Elapsed time (Using Euclidian distance)	MAE
0	12.39 seconds	0.91
0.1	14.81 seconds	0.90
0.2	12.52 seconds	0.88
0.3	14.61 seconds	0.83
0.4	**12.99 seconds**	**0.80**
0.5	15.43 seconds	0.83
0.6	14.14 seconds	0.87
0.7	15.34 seconds	0.89
0.8	14.02 seconds	0.87
0.9	13.67 seconds	0.87
1	14.93 seconds	0.87

Similarity search on a given graph can be challenging. After solving the issue regarding the parameter and metric selection for a given graph based on a different similarity algorithm, we have to come to a decision to select the fast similarity graph.

Although the full similarity graph has the lowest computation timing among the others, it has the disadvantage that it lacks sparse structure, which is not recommended for large data sets.

It is a general recommendation to use k-nearest neighbor as the first choice. It is easy to work with, which results in sparse adjacency matrix *w*, and in my research it shows less vulnerability for parameter selection compared with other graphs.

Considering *n* as the number of data and *d* the dimensionality of data and by keeping the max heap with size t, the complexity of generating sparse matrix is O $(n^2d) + O(n^2logt)$ time and *O(nt)* memory. The following results are from processor of 8 GB RAM Core i5 2.3 GHz CPU and 64 bit operating machine.

From current research, we chose singular value decomposition (SVD) for selecting the number of clusters because it has the ability to eliminate large portion of data. Although results show total variation caused in the data is acceptable, in the following section all the dimension reduction methods were applied on the selected number of clusters; that is, k= 1000, 500, 250, and 100.

Table 4. 4 Selecting number of clusters using SVD analysis

Number of Clusters (k)	Total data Variation
1000	**100%**
500	**100%**
250	**98%**
100	**97%**
80	95%
70	95%
60	93%
50	91%
45	89%

As mentioned earlier, we applied normalized clustering algorithm, on all the reduced dimension methods, and compare them with regards to accuracy and quality of clustering. Tables 4.5 and 4.6 shows NMI and clustering accuracy for all the methods.

Table 4. 5 NMI comparison of *k*-mean algorithm applied on all dimension reduction methods

K	Nystrom approximation	Eigen Laplacian	FA	PCA	IBP
1000	0.50	0.75	0.78	0.35	0.30
500	0.43	0.74	0.75	0.34	0.43
250	0.45	0.74	0.74	0.34	0.40
100	0.43	0.73	0.74	0.30	0.41

Table 4. 6 Clustering accuracy comparison of *k*-mean algorithm applied on all dimension reduction methods

K	Nystrom approximation	Eigen Laplacian	FA	PCA	IBP
1000	0.52	0.64	0.50	0.48	0.20
500	0.38	0.64	0.56	0.45	0.15
250	0.38	0.63	0.53	0.46	0.13
100	0.35	0.63	0.56	0.45	0.12

One naive way to compare each model is to find out the computation cost for each method. Although the results from their running cost is a data dependent procedure and a detailed

evaluation is beyond the scope of this article, we applied it on our two data sets to check whether the expected result is driven from out data or not.

We are expecting that the Nystrom method and factor analysis to have the lowest running cost. The following results are from processor of 8 GB RAM Core i5 2.3 GHz CPU and 64 bit operating machine.

Table 4. 7 Analysis of time complexity

Algorithms	Nystrom Method	t-nearest neighbor sparse matrix
Obtaining the similarity matrix	O(nld)	$O(n^2d + n^2logt)$
Finding first k eigenvectors	$O(l^3)$ +O(nlk)	$(o(m^3) + \big(O(nm) + O(nt) \times O(m-k)\big)$

In our case studies we generated eigen Laplacian using t-nearest-neighbor, normalized algorithm. Table 4.8 is the computation cost for all the dimension reduction methods on an average over ten runs and it is in second

Table 4. 8 Computation cost for all dimension reduction methods

Dimension reduction method	Number of genes	k-mean	Total Time
PCA	1000	25.16	62.32
	500	24.21	40.23
	250	23.32	35.55
Eigen Laplacian	1000	18.15	209.73
	500	9.54	189.86
	250	15.08	185.08
Factor analysis	1000	15.32	32.12
	500	13.45	25.20
	250	12.01	20.00
Nystrom method	1000	20.23	50.00
	500	19.36	48.88
	250	18.00	46.41
Indian buffet process	1000	323.32	1062
	500	269.03	1000
	250	250.00	789

We can reach to the following results by comparison of among dimension reduction methods.

1. The above results show that the quality of clustering, accuracy of clustering for eigen Laplacian and factor analysis are high. Although the NMI, and accuracy value for Nystrom method, and Indian buffet restaurant are very low
2. From computational cost perspective, Nystrom method, factor analysis, and PCA show better performance than the rest of methods
3. From the results obtained after an average run of ten times, that as the number of genes increases the quality of clustering also increases.

4.6 Summary

Different numbers of dimension reduction methods were used. It can be seen from the above results that the Nystrom method and factor analysis have the lowest running time compared to the other methods. The reason behind this is because the Nystrom method is an approximation method; as such it uses the approximated graph instead of using the whole similarity graph.

On the other hand, comparing the quality of clustering for different methods show that the factor analysis and eigen Laplacian have the highest value among the rest of methods. In addition to all the above results, it can be seen that as the number of genes are increasing, the quality of clustering also increases.

CHAPTER V

DYNAMIC ANALYSIS OF VARIOUS DIMENSION REDUCTION METHODS

In this chapter, we began with applying preprocessing steps used in the previous chapter on all the reduced dimension data derived from various dimension reduction methods. After checking the stationarity of the reduced dimension data, the multivariate autoregressive coefficents were estimated using several penalized regression methods.

The adequacy of the fitted model was checked. Then 1-step ahead forecast was generated and the goodness of fit was checked using coefficient of determination, denoted R^2. Finally, the results from the above mentioned metrics were compared with regards to performance. Figure 5.1 shows the procedure used in this chapter.

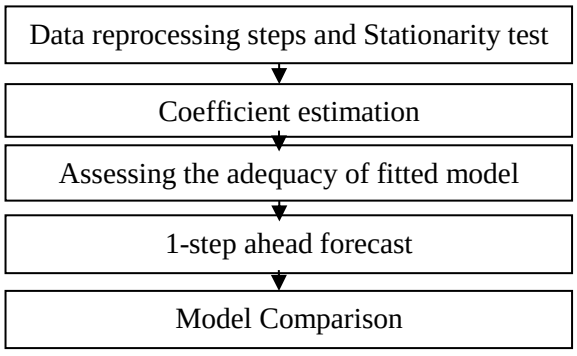


Figure 5. 1 Procedure for time series analysis of the reduced dimension model

5.1 Research methodology

To estimate gene regulatory network from time-series data, there are several problems that need to be solved. First, trying to identify a large network from a large number of genes originating from smaller number of experiments (times) is challenging since the system is underdetermined. Second, due to the high correlation between genes, the resulted least squares estimators will have large variances. Hence, it may be useful to reduce the dimension of the model to eliminate multicollinearity problems among large number of variables.

We used several dimension reduction methods, as described in chapter 4. We applied all of the dimension reduction methods on RNA expression levels of 4028 genes for nearly one-third of all Drosophila genes during a complete time course of development, and Human cancer cell Line (HeLa) cell-cycle gene expression datasets. As our input data for this chapter, we used both the original time series data sets and the reduced dimension data sets.

The goal for this chapter is to fit a time series model on several reduced-dimension techniques and check the performance of each technique. We used preprocessing steps such as data trimming, normalization, and transformation. The stationarity of the reduced dimension data was checked to observe the existence of any trend in the model and then the differencing of one lag was applied in order to remove the existed trend.

We have applied the sparse vector autoregressive (SVAR) model to estimate the gene regulatory network based on the gene expression profile from time-series microarray experiment on two data sets. Autoregressive coefficients were estimated using several penalized regression methods like ridge, SCAD, and lasso.

After that, the 1-step ahead forecast was used and the goodness of fit was checked using coefficient of determination, denoted by R^2. Finally, the results from the above mentioned metrics were compared with regards to performance.

It could be seen that explanatory power under factor analysis is better than the multiple linear regression for both data sets and that basically any method produces orthogonal dependent variables. If we have regressors that have high collinearity, and in such cases the least squares estimators have large variances, inferences based on the regression model can be misleading. To offset this problem, we can use techniques that can remove orthogonality before applying regression.

The reason behind that factor analysis has better performance compared with other methods is because the common factor model rarely produces interpretable solutions, and a rotation to the solution should be applied. Hence, it can be seen that using FA makes most of the coordinates closer to zero, points on the axis become more interpretable, and being close to the axis makes it sparser, which, in turn, induces additional sparsity subject to maintaining a certain goodness of fit.

5.2 Pre-processing steps

5.2.1 Normalization method applied on the reduced dimension data

The reduced dimension data sets should be normalized again. The normalization process used in section 4.2 is applied on the reduced dimension data sets. Then the data is set to be checked for stationarity.

5.2.2 Test of stationary for multivariate time series

Intuitively, a time series is defined to be stationary if the statistical properties of the time series, i.e., the mean and the correlation coefficients, do not change over time. Hence before applying any correlation-based data analysis we firstly need to compute the stationarity test on the original data, i.e., whether the correlation is stable or not.

Mathematically, if a gene expression data set from a microarray experiment can be presented by a real-valued expression matrix such *X* i.e., $X = \{x_i(t)\}_{m\times t}$, consists of time-series of expression of m genes, taken over t time points where $x_i(t)$ denotes the expression of gene i at time *t.* Suppose a vector of gene expression at time t is denoted by row vector $x^t = x_i(t)_{i=1}^m$, and $z^t = x^{t-1}$ which is the vector of gene expressions at previous time-points, then the matrix representation would be $X_{m\times t}$=$[x_2, \dots, x_t]^T$, and $Z_{m\times t} = [z_1, \dots, z_{t-1}]^T$.

If *X* is stationary then the correlation coefficient matrices of *X* and *Z,* i.e., $Corr(X)$, and $Corr(Z)$ would not be statistically different; however if *X* is non-stationery then $Corr(X)$, and $Corr(Z)$ would be statistically different. Hence if the MTS data turns out to be non-stationery we need to stationarize the data before applying any correlation based analysis. Recall that *X* is a univariate time series (UTS) when *m* is equal to 1, and it is a MTS when *m* is equal or greater than 2.

Kiyoung, Y., et al (2005), proposed an approach for testing the stationarity of a MTS data performing Johansen's Co-integration test (1995). It should be noted that we do not use Dickey Fuller (ADF) (Chen, Wen-Yen, et al 2008) test for each UTS, since our interest is not the stationarity of each UTS in an MTS. Johansen's Co-integration test is a procedure for testing co-integration of several multiple time series. This test permits more than one co-integrating relationship.

Using algorithm 1, a number of co-integrating relationships,γ where $1 < \gamma < m$ were extracted. Recall that if coefficient matrices have a full rank, all variables in the system are stationary; hence, if the number of co-integrating relationships are equal to number of variables then we consider MTS to be stationary. Figure 5.2 shows an algorithm for stationarity of an MTS data set.

Algorithm 1 Determine the stationarity of an MTS data set

Require: MTS data set, N {the number of data in the data set}, n {the number of variables in an MTS data}

1: **for** $i = 1$ to N **do**
2: $res \leftarrow$ Johansen's test on the ith MTS data;
3: $\gamma \leftarrow$ extract the number of co-integrating relationships from res;
4: **if** $\gamma = n$ **then**
5: $H(i) \leftarrow 0$; {stationary}
6: **else**
7: $H(i) \leftarrow 1$; {non-stationary}
8: **end if**
9: **end for**
10: **if** $sum(H) > ceil(N/2)$ **then**
11: Stationarize the given MTS data set;
12: **end if**

Figure 5. 2 Algorithm for stationarity of an MTS data set (Kiyoung, Y., et al., 2005)

There are different techniques to eliminate the trend and seasonality; the most popular one is differencing. The idea is simply to consider the differences between pairs of observations with appropriate time separations. This can be achieved by taking first difference of time series *x (t) − x (t − 1).* It is basically differencing time series until it becomes stationary.

For non-seasonal data, first-order differencing is reasonable to obtain stationarity, hence the original matrix *x (1)... x (t)* will be differenced by first-order differences *z (t)* of a time series *x (t)* are described as *z (t) = x (t) −x (t-1).* To fit any autoregressive (AR) model to a data, the sample

mean of it should be small. Once the trend is removed, we can subtract the mean of the transformed data from each observation from each observation (Dickey and Fuller 1979).

5.2.3 Visualizing time series data

Visualizing can be accomplished by producing a standard time series plot from the data as we can see in the Figure 5.3 (a) and (b) which are the standard plots of time series data for both Drosophila and HeLa datasets. However, for multiple time series data the amount of screen space required to build the plot is restricted.

We used mvtsplot function for plotting multivariate time series data in R (Ben-Hur, A., Elisseeff, A., and Guyon, I., 2002). Figure 5.4 (a) and (b) shows the time series plot for both Drosophila and HeLa data. The values of each time series are categorized in three colors: purple as low values, grey as medium values, and green as high values.

On the right hand side panel are box plot of data in each time series and bottom panel are median values across all the time series for each time series point (Peng, 2008).

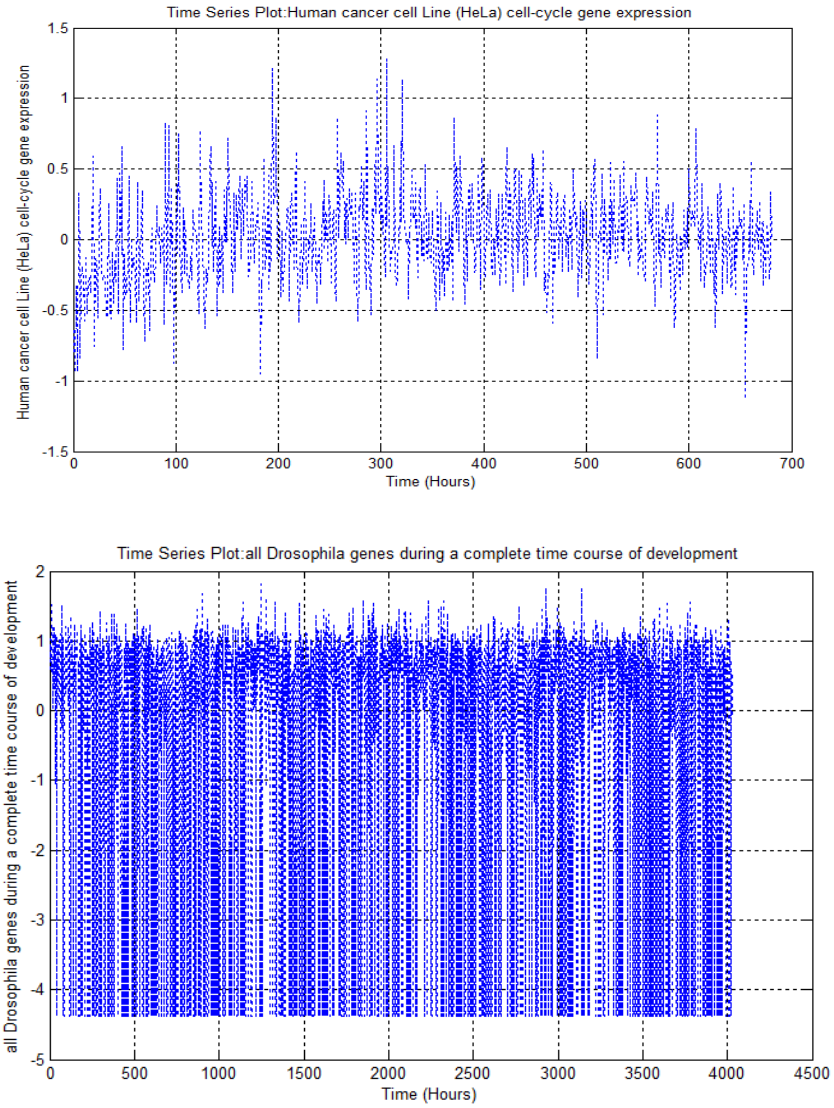
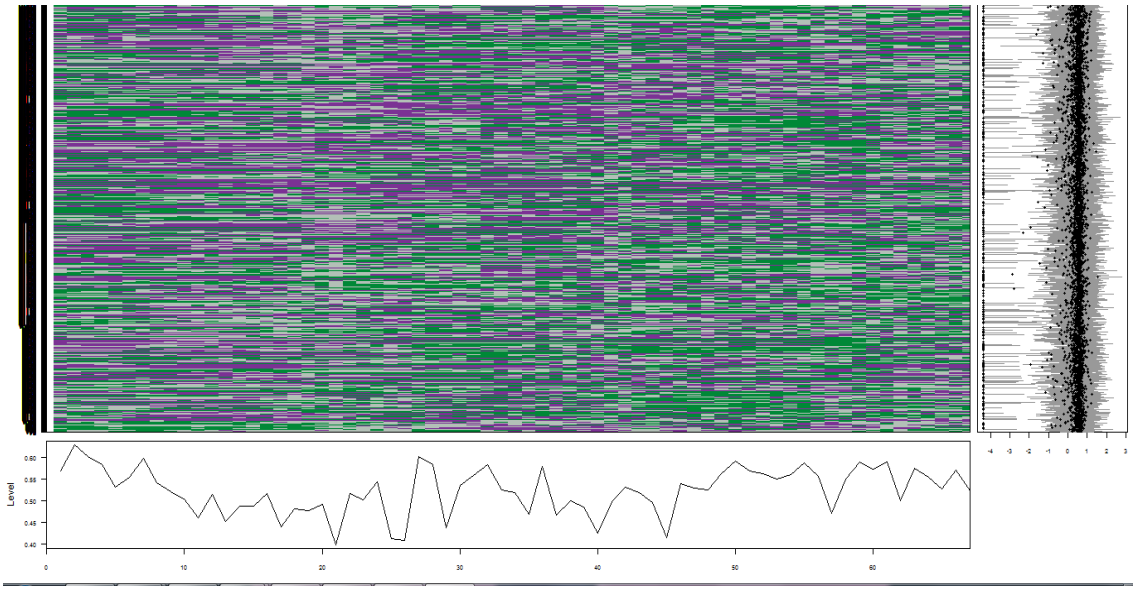


Figure 5. 3 (a) Standard Time series plot of Drosophila genes, and (b) Human cancer cell Line



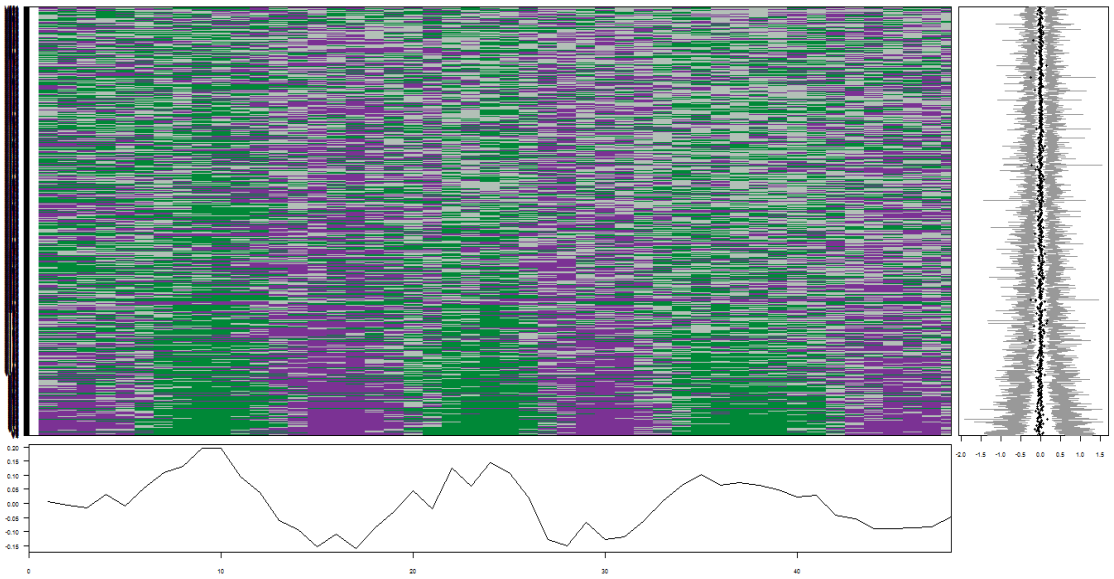


Figure 5. 4 (a) visualizing time series plot of Drosophila genes, and (b) Human cancer cell Line

5.3 Estimating parameters of time series model

Model selection aims to find the most parsimonious model that describes the data. This model should simplify estimation, forecasting, and interpretation. Initially, we have to model the MTS, and its parameter should be estimated. Then the adequacy of the model is checked. In the previous section, we checked the stationary of the model; hence, we confirmed that our variables do not change over time.

5.3.2 Estimating parameter of an autoregressive model (AR)

In order to understand the molecular mechanisms underlying biological processes we need to use some statistical techniques to estimate gene regulatory network from time-series microarray data. However, several problems need to be overcome. For one, we usually try to identify large networks in which a number of genes are larger than the microarray experiments which, in return, implies large number of parameters that need to be estimated.

It is reasonable to consider complexity reduction to fit a time series model for high dimensional data. One solution to this problem would be using multivariate sparse vector autoregressive (MSVAR). There are basically two directions used for the purpose of complexity reduction. One would be reducing the number of parameters and the second direction involved in reducing the dimension of the original series and fit a model to a lower dimension. The focus of this thesis would be on both cases under context of MSVAR (Valdés-Sosa, et al 2005).

5.3.3 Fitting regression models on the original series

As mentioned before, one way to reduce the complexity of time series models is to reduce the number of parameters via variable selection. Multivariate sparse vector autoregressive (MSVAR)

model is used as an example to review variable selection methods for time series models (Rajapakse 2011).

Given the original matrix, i.e., the multivariate time series $X = \{x_i\ (t)\}_{m\times t}$ which consists of time series of *m* genes over *t* time points, where vector $x_i(t)$ denotes the gene *i* at time *t*, the multivariate vector autoregressive model (MVAR) with order *p* is,

$$x(t) = \sum_{p=1}^{p} Ax(t-p) + \varepsilon(t), \tag{5.1}$$

where $x(t)$ is the vector of gene expression, $x(t-p)$ denotes the gene expression at previous time-points, p is the known order of the model, ε(t) denotes residuals that are assumed to be *i.i.d.* and zero mean Gaussian. *A* is the unknown matrix of autoregressive coefficents which represents the interaction between genes *i* and *j*. In general, the model has n^2p coeffiencets. The first order equation is simplified as,

$$x(t) = Ax(t-1) + \varepsilon(t), \tag{5.2}$$

We used VAR(1), i.e., vector autoregressive model with lag 1 as the basis for our approach for convenience. It should be noted that each gene may depend not only on its own past values, but also on the past values of the other genes. If x_{it} denotes the *i*th element in$x_i(t)$, the *i*th row yields

$$x_{it} = a_{i1}x_{1,t-1} + \cdots, a_{im}x_{m,t-1} + \varepsilon_{it} \quad i = 1, \ldots m \tag{5.3}$$

An autoregressive model is a regression model in which *x (t)* is regressed against its own lagged values. Given a vector of gene expression at time *t* is denoted by row vector $x^t = x_i(t)_{i=1}^m$, and $z^t = x^{t-1}$ which is the vector of gene expressions at previous time-point i.e., $X_{m\times t}$=$[x_2, \ldots, x_t]^T$, and $Z_{m\times t} = [z_1, \ldots, z_{t-1}]^T$.

To estimate VAR parameters we can cast the AR model in terms of ordinary regression model. The multivariate model can be written in standard multivariate vector autoregressive (MVAR) form of,

$$x^t = z^t A + \varepsilon^t, \tag{5.4}$$

This model can be estimated by ordinary least squares (OLS) by regressing each variable on the lags of itself and other variables. Considering the multiple linear regression model in (5.4), we can write the matrix notation as,

$$X=Z\,A+E, \qquad E_i \sim N(0,\Sigma),\ i=1,\ldots,m \tag{5.5}$$

Where, *X* is the response variable, and *Z* is the explanatory variable, and *A* is matrix of regression coefficients. We define *T=t-1;* hence, we have $X_{T\times m} = [x_2, \ldots, x_t],^T$ and $Z_{T\times m} = [z_1, \ldots, z_{t-1}],^T$ and $E_{T\times m} = [e_2, \ldots, e_t],^T A_{m\times m} = A_1^T$. For the purpose of system identification, we compare three methods that are least-norm problem, least square, and penalized least square.

Least square problem (OLS)

We need to obtain the least squares estimates for the vector autoregressive model, VAR (1). As mentioned earlier, if we rewrite the (5.2) in the regression form we have (5.5). OLS estimation basically minimizes the sum of squared of residuals (Stock, 1987). The explicit solution for OLS estimator is to choose *A* that minimizes $\|Z\hat{A} - x\|$.

The particular solution is $\hat{A}=(Z^TZ)^{-1}Z^Tx$. Where we can carry out separate regression analyses for each gene. It means we can separately estimate each column of A_i of *A*, where $A_i = (Z^TZ)^{-1}Zx_i$.

The OLS solution works in the case that $m<t$; that is the number of genes is smaller than the observations. Also OLS doesn't ensure sparse connectivity pattern for A, therefore we must turn to regression methods specifically designed to ensure sparsity.

Least-norm problem

It is a $L1$ norm problem which attempts to closley approximate a set of data in which it deals with an underdetermined linear equation that the number of genes are larger than number of samples that is $m < n$ [56]. If A_{ln} is the least norm solution for x^t=z^tA, then the least norm estimate of A, should minimizes the ||A||, so $Z(A - A_{ln}) = 0$. The particular solution is A_{ln}=$Z^T(ZZ^T)^{-1}x$.

Unlike least squares regression, least absolute deviations regression does not have an analytical solving method. Therefore, an iterative approach is required. The method of iteratively least squares (LS) is an approach to solve least norm problems. Usual time-series methods rely on maximum likelihood (ML) estimation which is equivalent to $\hat{A} = arg_A min\ ||X - ZA||_{\Sigma}^2$. The explicit solution, which is the OLS estimator, is $A_i = (Z^T Z)^{-1} Z x_i$.

Penalized Regression

The most serious concern with microarray data analysis is high dimensionality. However, classical logistic regression does not work for microarrays because generally there will be far more variables than observations.

Hence penalized regression could be used instead. Unlike traditional selection methods, the regularized method will simultaneously perform variable selection by setting some estimated coefficients (in A) zero and estimate other coefficients using shrinkage method in the sense of ridge regression (Heckman and Ramsay (2000).

Penalized regression amounts to performing least squares but with some additional constraints on the coefficients. In numerical analysis (inverse problems), this is known as regularization and is mostly applied when the optimization problem to be solved is under-determined.

The penalized regression approaches that can be used are either least square-penalty (LS) or least absolute deviation-penalty (LAD) (Ra Rosen, et al., 1996). In LS, the penalized estimator $\hat{A}$ of A is defined as the constrained minimize of the sum of squared residuals (RSS):

$$\hat{A} = arg_A min\ ||X - ZA||_{\Sigma}^2\ \ s.t.P(A) \leq \lambda \tag{5.6}$$

Where $\| \ . \ \|$ is the L_2-norm, and $\boldsymbol{\lambda}$ is a specified constant. The minimizing above equation is equivalent to minimize the so called penalized RSS with respect to A

$$\hat{A} = arg_A min\ ||X - ZA||_{\Sigma}^2\ + \lambda P(A) \tag{5.7}$$

In case of LS-lasso, L_1-norm is used on its penalty term, then we have,

$$\hat{A} = arg_B min\ ||X - ZA||_{\Sigma}^2 +\ \lambda\alpha|A| \tag{5.8}$$

The results on the LS estimator make use of the nice properties of a least squares loss function in which the convenient geometry associated with L_2-norm. On the other hand, the LAD estimator is defined as the value that minimizes the criterion:

$$\hat{A} = arg_A min\ |X - ZA|\ + \lambda P(A) \tag{5.9}$$

As we can observe, both loss function and penalty function in the LAD are L_1-norm. The choice of any of the above approaches certainly influences the results. Both LAD and LS estimators get a sparse solution.

LAD will simplify the computation compared with LS since both loss function and penalty function are in terms of L_1-norm and is it robust to outliers; however, it is more difficult to deal with LAD loss function since it is not differentiable at zero. Graphically, the *L1* ball has "pointy" corners and thus bias solutions for A towards these corners; in higher dimension this ball will have even more corners. Hence *L2* approach is chosen (Heckman and Ramsay, 2000).

Hence, the trick would be using least squares but with some additional constraints on the coefficients. Figure 5.5 shows the major difference of *L*1 and *L*2 minimization in the sense of number of solutions.

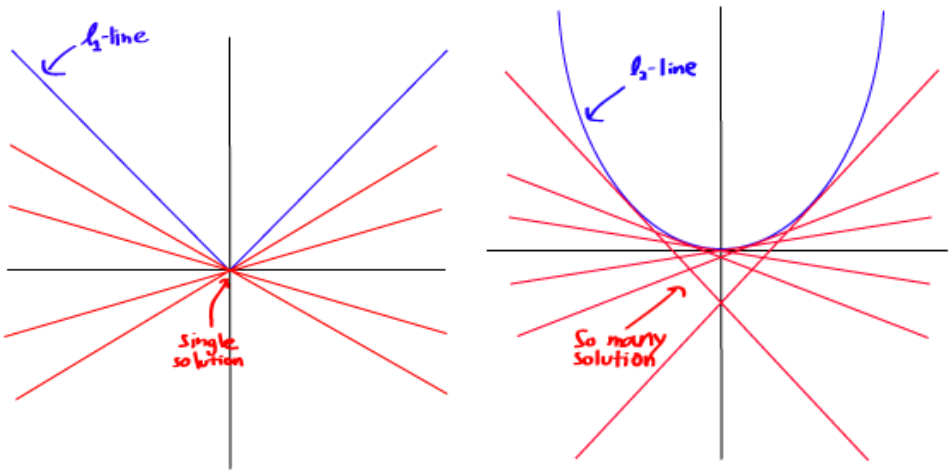


Figure 5. 5 Different solutions of L1 and L2 minimization

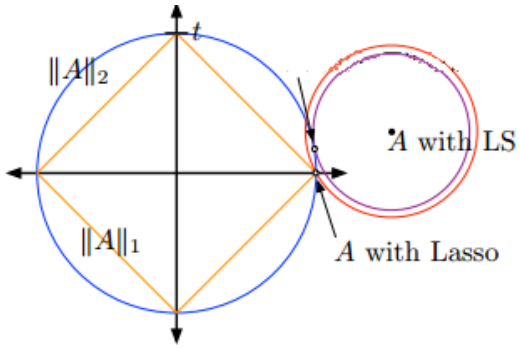


Figure 5. 6 Radius t ball under L1 and L2, and the results of Lasso and LS

Figure 5.6 shows radius *t* ball under L1 and L2. In this figure the center of the red and purple circles are non-regularized OLS estimates. Each circle represents set of weights that result in the same square error. As *t* becomes smaller, it is more and more likely that the solution to *A* is found on a higher-degree corner. Also, the solution found becomes further and further away from the least squares solution. If we set $t = \infty$ then the Lasso solution is the least squares solution.

Throughout this chapter, we will use OLS estimator. As mentioned before, the OLS estimator $\hat{A}=(Z^TZ)^{-1}Z^Tx$ is obtained by minimizing $||X-ZA||_\Sigma^2$ without imposing any constraints. The three penalties used are ridge regression, lasso and SCAD. The following will explain the difference between these three methods.

Tikhonov regularization (Ridge Regression)

Ridge regression penalizes the Euclidean norm of the coefficents (Tibshirani, R., 1996).

$$\hat{A}^{ridge} = arg_B min\ ||X-ZA||_\Sigma^2 + \lambda^2||A||^2 \tag{5.10}$$

where $||A||_2=\sqrt{\sum_{i=1}^{p} A_i^2}$ is the Euclidean norm. This is equivalent to solving

$$arg_A min\ ||X-ZA||_\Sigma^2, \qquad \text{s.t}\ ||A||_2 \ll \lambda \tag{5.11}$$

This is known as ridge regression. This can be solved as $A=(Z^TZ+\ \lambda)^{-1}Z^Tx$. The λ is the tuning parameter which conrols the effective degree of freedom.

Least absolute shrinkage and selection operator (lasso)

By penalizing (or equivalently constraining the sum of the absolute values of the estimates) we end up in a situation where some of the parameter estimates may be exactly zero. The larger the penalty applied, the further estimates are shrunk towards zero. Tibashirani (1996), proposed a new shrinkage method names least absolute shrinkage and selection operator known as lasso. In lasso, some coefficients are shrinks while setting others exactly to zero.

The lasso method shrinks the AR coefficients towards zero by minimizing a target function, the sum of a loss function and a $L1$ penalty on the AR coefficients, and it has the advantage of performing model selection and parameter estimation simultaneously (Tibshirani, R. 1996). The strategy behind multivariate sparse vector autoregressive involves reducing the nonzero AR coefficients and reducing the dimension of the original series and modeling the lower dimension. The alternative approach is to use lasso in which L_1 norm on the penalty instead of L_2 norm in ridge regression. The followings are the properties of LASSO

$$\hat{A} = arg_A min\ ||X - ZA||_{\Sigma}^2 + \ \lambda|A|_1 \tag{5.12}$$

Smoothly clipped absolute deviation penalty (*SCAD*)

Using the smoothly clipped absolute deviation method of Fan & Li (2001) estimates A by minimizing the penalized least squares function:

$$\hat{A} = arg_A \min\{\frac{1}{2n}\ ||X - ZA||_{\Sigma}^2 + \ \sum_{j=1}^{n} p\lambda\ (|A_j|)\} \tag{5.13}$$

where $p\lambda\ (.)$ is the smoothly absolute deviation penalty with a tuning parameter λ.

Generalized cross-validation criterion for prediction purposes

The major goal behind the time series analysis is to know how well AR (1) model predicts, and how accurately this predictive model will perform in practice. Once the prediction is confirmed, we can start to assess how well this model performs via fit measure like R^2. However using R^2 for assessing the model in standard regression is biased by over fitting.

In regression problem, prediction error refers to the difference between the predicted values actual values, and because of over fitting these errors will be biased. Hence one way to address this bias is to use model validation techniques likes cross validation method.

One way to evaluate our model is to use cross validation by looking at its prediction error. In which it doesn't use the entire data set when building a model. In the first round, once the model has been built using training set, the testing set can be used to test the performance of the model on the "unseen" data (i.e. the testing set). Then multiple rounds are performed using different partitions, and then the results are averaged for all rounds.

We used 10-fold cross-validation, in which the original data was randomly portioned into 10 equal sizes of subsamples. A one subsample is trained as validation data for testing the model, and the remaining 9 subsamples were used as training data. Then the procedure was repeated 10 times, and all the 10 results were averaged.

The GCV is a modified form of 10-fold cross-validation. If $H=(Z^TZ+\lambda I)^{-1}Z^Tx$ is the matrix in ridge regression we have,

$$GCV(\lambda)=\frac{||X-Z\hat{A}||_{\Sigma}^{2}}{(tr(I-H))^2} \tag{5.14}$$

In summary, the regularization in linear regression is a means to estimate coefficents A by minimizing the regularized RSS.

5.3.4 Fitting regression models to a lower dimension manifold

For a multiple-response problem, one can initially perfom separate linear regressions on each gene by ignoring the possible intercorellation between response variables; in other words, it is possible to seprately estimate each column A_i of A. $A_i = (Z^T Z)^{-1} Z x_i$, where x_i is the ith column of X.

In applying a penalized regression, the MVAR model is reformulated as a linear regression problem. Such a formulation ignores the temporal dependence in the time series. Song and Bickel give a theoretical discussion on the consequences of applying lasso directly in MVAR models without taking into account the temporal dependence between the response and the explanatory variables (Song, and Bickel, 2011).

The most serious concern with microarray data analysis is high dimensionality and collinearity. Multicollinearity is the undesirable situation where the correlations among the independent variables are strong. As a first step toward parameter estimation, it is beneficial to search for any dependencies among predictors and selecting important predictors.

One way is dimension reduction; that is, to combine the predictor variables into fewer features which can be explained as latent factors that drive the variation in the multiple response variables.

Multicollinearity doesn't affect the properties of OLS estimators. Although the resulted estimator remains unbiased with the presence of multicollinearity, OLS estimators will be estimated imprecisely. If the goal is to predict the dependent variable from set of independent variables then existence of multicollinearity is not a problem and the prediction is still accurate. However if the

goal is to check how various independent variables impact the dependent variable then multicollinearity can cause problems (Farrar and Glauber, 1967).

When collinearity is severe, it leads to unreasonable coefficient estimates and large standard errors. Multicollinearity increases the standard errors of the coefficients. Increased standard errors, in turn, mean that coefficients for some independent variables may not be found to be significantly different from zero, whereas without multicollinearity and with lower standard errors these same coefficients might have been found to be significant.

Therefore strong multicollinearity between variables results in large variances and covariance for the least square estimator of regression coefficients. In our study, this problem is partly solved by reducing genes with little variations before the analysis.

It should be expected that dimension reduction methods producing orthogonal independent variables have better performance with regards to their goodness of fit. The basic assumption with regression models is that no correlation should be between independent variables; i.e., they are orthogonal. Unfortunately, in most applications of regression, the regressors are not orthogonal.

Sometimes the lack of orthogonality is not serious. However, in some situations the regressors are nearly perfectly linearly related. In such cases, the least squares estimators have large variances; hence, inferences based on the regression model can be misleading. To offset this problem we can use techniques that can remove orthogonality before applying regression.

Recall that principal components try to re-express the data as a sum of uncorrelated components, i.e., the goal of PCA is to reduce the measured variables to a smaller set of composite components

and to account for as much of the total variance in the variables as possible with as few components as possible. Each principal component is uncorrelated and orthogonal.

On the other hand, in FA the original variables are defined as linear combinations of the factors. Factor scores can be derived so that they are nearly uncorrelated or orthogonal. Thus, the problem for multicollinearity among the variables can be solved by using coefficients.

The performance of different dimension reduction techniques using various regression methods is data dependent and a detailed evaluation is beyond the scope of this article (Rossi-Doria, et al., 2003). However, comprehensive studies and data analysis will be needed to draw more definitive conclusions, but based on previous work on simple linear regression models using dimension reduction techniques that are concerned with orthogonal variables, it is expected that the relative performances of such models should be better with regards to low variance estimators because orthogonality leads to reasonable coefficient estimation with low standard errors.

We used two different time-series data: the overall performance for each regression method under different conditions was measured by means of the coefficient of determination, denoted by R^2. It should be noted that if the model fits the data well, the overall r-squared value would be high. It can be seen from Table 5.3 and 5.4 that explanatory power under factor analysis and principle component analysis are two examples of dimension reduction techniques concerned with orthogonal variables are better than the multiple linear regression for both data sets (Ma, S., et al. 2006).

Factor analysis first extracts a set a factors from a data set. These factors are almost always orthogonal and are ordered according to the proportion of the variance of the original data that

these factors explain. Generally, we kept a small subset of factors for regression and the remaining factors are considered as either irrelevant or nonexistent.

In an unrotated solution, variables are associated with factors based on high loadings. The first factor describes most of the variability. Ideally, we want to spread variability more evenly among factors and make factors interpretable and present a simple structure.

The complexity of a variable in a factor pattern refers to the number of nonzero elements in the corresponding row of the factor matrix. A variable with complexity one will be referred to as a perfect indicator. The benefit of the simple structure is that usually using factoring finds the groups of variables, but the extraction process is trying to reproduce variance. In factor rotation we change the viewing angle of the factor space, which is the major approach to provide a simple structure.

Because the common factor model only rarely produces interpretable solutions, a rotation to the solution should be applied. All rotation criteria to be considered are expressed as complexity functions to be minimized to yield a simple pattern of loadings. If the factor axes are rotated, the loading of a variable on one factor is maximized while its loading on the other factors is minimized thereby making the factor structure easier to interpret.

By rotation it can be seen that FA makes most of the coordinates to be closer to zero so that points on the axis become more interpretable and being close to the axis makes it sparser, which in turn induces additional sparsity subject to maintaining a certain goodness of fit.

5.4 Assessing the adequacy of a fitted model

Before the fitted model is used for analysis, it should be assessed for adequacy, and adequacy of such models are assessed whether the statistic of the residuals are consistent with assumptions of AR (1) model or not:

$$\hat{\varepsilon}_v = u_v - \widehat{w} - \sum_{l=1}^{p} \hat{A}_l \, u_{v-l} \quad \text{for} \quad v = 1, \dots N \tag{5.15}$$

One of the major assumprions made in the AR model is that the noise vector should be uncorrelated. In particular, we are interested in finding whether the residuals of our model are white noise, that is, they follow process that shows no serial correlation.

5.4.2 Quantitative analysis

Ljung-Box (LB) test is a quantitative approach to test the adequacy of the fitted model. In this option we built a test statistic to test the null hypothesis that the residuals are independent up to a lag *p* (Ljung and Box 1978). These tests should be applied in order to check for nonlinearity in mean and also for nonlinearity in variance. Let $x(t)$ be a time series with the lag-*p* autocorrelation:

$$\rho_1(p) = \frac{cov(x_t, x_{t-p})}{var(x_t)} \,, \quad p = 1,2, \dots. \tag{5.16}$$

The lag-*p* sample autocorrelation of x (t) is,

$$\rho_1(p) = \frac{\sum_{t=k+1}^{T} (x_t - \bar{x}_T)(x_{t-p} - \bar{x}_T)}{\sum_{t=1}^{T} (x_t - \bar{x}_T)^2} \tag{5.17}$$

where *T* denotes the sample size and $\bar{x}_T$ is the sample mean of x_t. The hypotheses consist of *p* autocorrelations are, $H_0 = \rho_1(1) = \rho_1(2) = \cdots = \rho_1(p) = 0$, and $H_1 = \rho_1(k) \neq 0$ for some *k=1, 2... p.* For these hypotheses, the Q_{LB} test statistic is of the form,

$$Q_{LB}(p) = T(T+2)\sum_{k=1}^{p}\frac{\hat{\rho}^2(k)}{T-k} \quad (5.18)$$

This statistic is used to check the whiteness of $x(t)$, i.e.,$\rho_1(p) = 0$.

5.5 Forecasting the selected model

The main purpose of modeling a time series is to make forecasts which are then are used directly for making decisions. A natural starting point for a forecasting model is to use past values of *x (t)*; this simple forecasting technique is called the naive method. A naive forecast for any period simply projects the previous period's actual value. Although this technique may seem too simplistic, its advantages are low cost but its major weakness is its inability to make highly accurate forecasts.

Forecasting for AR models is obtained using an expression for the desired future values in terms of the *x (t)* and the $\varepsilon(t)$ and then by replacing all the unknown terms by their optimal forecasts (Ljung and Box 1978). For one-step predication in terms of AR(1) model, i.e., $x_{t+1} = Ax_t + \varepsilon_{t+1}$. The optimal forecast of ε_{t+1} is zero since ε_{t+1} is uncorrelated with all present and past values ofx_t; hence, the optimum forecast of x_{t+1} is$f_{t,1} = Ax_t$. The error in a forecast is the difference between the realization and the forecast, i.e., $e_{t,1} = x_{t+1} - f_{t,1} = \varepsilon_{t+1}$.

It must be remembered that the purpose of the simulations was to generate time-series from which both gene-gene network and module-module network. A number of indices were calculated to evaluate the performance of different penalized (Reinsel and Ahn. 1992). Overall performance for each regression method under different conditions was measured by means of the coefficient of determination, denoted by R^2.

5.6 Case study

As mentioned before, there are basically two directions used for the purpose of complexity reduction. One would be reducing the number of parameters and the second direction involved in reducing the dimension of the original series and fit a model to a lower dimension. The focus of this thesis would be on both cases under context of MSVAR.

In this section, autoregressive coefficients were estimated using several penalized regression methods like ridge, SCAD, and lasso on both original data sets and the reduced dimension models using various dimension reduction methods. All the above-mentioned models were then compared with regards to performance.

As mentioned before, the number of variables set to zero in any of the methods described will depend on the value of the tuning parameter with higher values of tuning parameter fewer variables are selected.

To measure the goodness of fit is to the coefficient of determination (R2). Tables 5.1, and 5.2 shows results of R-squared value for different techniques for HeLa, and Drosophila data sets simultaneously.

Table 5.1 R-squared value for different dimension reduction techniques for and HeLa data

Dimension reduction method	Number of dimension	Method	R-squared
PCA	500	Lasso	0.53
		SCAD	0.53
		Ridge	0.45
	250	Lasso	0.46
		SCAD	0.46
		Ridge	0.43
Eigen Laplacian	500	Lasso	0.51
		SCAD	0.50
		Ridge	0.48
	250	Lasso	0.39
		SCAD	0.35
		Ridge	0.35
Factor Analysis	500	Lasso	0.80
		SCAD	0.78
		Ridge	0.80
	250	Lasso	0.72
		SCAD	0.72
		Ridge	0.73

Dimension reduction method	Number of dimension	Method	R-squared
Nystrom Method	500	Lasso	0.35
		SCAD	0.32
		Ridge	0.32
	250	Lasso	0.30
		SCAD	0.30
		Ridge	0.30
Indian Buffet Process	500	Lasso	0.17
		SCAD	0.20
		Ridge	0.21
	250	Lasso	0.17
		SCAD	0.17
		Ridge	0.16
Original Data set	500	Lasso	0.58
		SCAD	0.55
		Ridge	0.53
	250	Lasso	0.50
		SCAD	0.50
		Ridge	0.41

Table 5.2 R-squared value for different dimension reduction techniques for Drosophila data

Dimension reduction method	Number of dimension	Method	R-squared
Nystrom Method	1000	Lasso	0.50
		SCAD	0.30
		Ridge	0.21
	500	Lasso	0.42
		SCAD	0.35
		Ridge	0.25
	250	Lasso	0.44
		SCAD	0.44
		Ridge	0.21
Indian Buffet Process	1000	Lasso	0.23
		SCAD	0.17
		Ridge	0.16
	500	Lasso	0.20
		SCAD	0.12
		Ridge	0.15
	250	Lasso	0.22
		SCAD	0.20
		Ridge	0.13
Original Data set	1000	Lasso	0.49
		SCAD	0.44
		Ridge	0.45
	500	Lasso	0.50
		SCAD	0.53
		Ridge	0.55
	250	Lasso	0.60
		SCAD	0.62
		Ridge	0.63

Dimension reduction method	Number of dimension	Method	R-squared
PCA	1000	Lasso	0.39
		SCAD	0.35
		Ridge	0.40
	500	Lasso	0.54
		SCAD	0.45
		Ridge	0.55
	250	Lasso	0.53
		SCAD	0.40
		Ridge	0.41
Eigen Laplacian	1000	Lasso	0.52
		SCAD	0.50
		Ridge	0.50
	500	Lasso	0.50
		SCAD	0.51
		Ridge	0.49
	250	Lasso	0.49
		SCAD	0.42
		Ridge	0.47
Factor Analysis	1000	Lasso	**0.77**
		SCAD	0.78
		Ridge	0.73
	500	Lasso	0.63
		SCAD	0.53
		Ridge	0.52
	250	Lasso	0.53
		SCAD	0.60
		Ridge	0.47

5.7 Interpretation of case study

Using two different time series data, the overall performance for each regression method under different conditions was measured by means of the coefficient of determination, denoted by R^2. It should be noted that if the model fits the data well, the overall r-squared value would be high. Any value above 0.6 would be considered as a good r-squared for this comparison.

The basic idea behind selecting the number of dimension was capturing at or above 98% of variation. The number of dimensions used 1000, 500 and 250 are accounted for 100%, 100%, and 98% of the total variance. For the sake of comparison for the original data sets we selected the first 1000, 500 and 250 variables.

It could be seen from both data sets that the overall r-squared value for those dimension reduction methods that produce orthogonal variables are having higher r-squared values. It can be seen from Table 5.3, and 5.4 that explanatory power under factor analysis, principle component analysis and eigen Laplacian are better than the multiple linear regression for both data sets and basically any method producing orthogonal dependent variables.

The coefficient of determination for factor analysis ranges between (0.72-0.8) and (0.48-0.77) for HeLa and Drosophila data sets respectively. As mentioned earlier that the basic assumption with regression models is that no correlation should be between independent variables; i.e., they are orthogonal. On the other hand, applying varimax rotation in factor analysis makes most of the coordinates to be closer to zero, hence applying penalized regression in turn, induces additional sparsity subject to maintaining a certain goodness of fit, because it reduced the number of inputs and therefore decreased the model complexity.

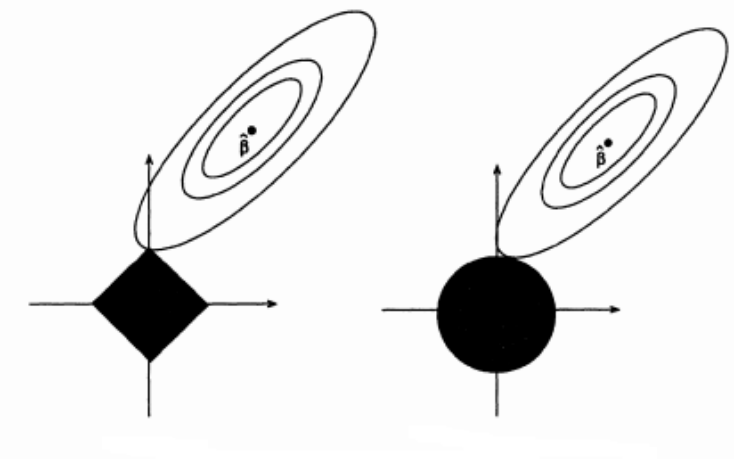
Nystrom method and eigen Laplacian method uses the same algorithm for dimension reduction. However, in Nystrom method it is assumed that a similarity matrix can be approximated by using even a small subset of columns. Hence due to this approximation the quality of dimension reduction techniques using Nystrom method can severely be reduced as it can be seen from the results.

From the above demonstration it is expected that the performance of eigen Laplacian based on goodness of fit compared with Nystrom method should be better. It is observed that for the same number of dimension e.g., 500 for Hela data and Drosophila data set the r-squared value for Nystrom method and eigen Laplacian using lasso are 0.35 and 0.5 and 0.42 and 0.51 respectively.

It is obvious that as the number of selected dimension is higher we observe better performance. Except in the original Drosophila data set we can see that for the number of dimension about 250 the performance is around 0.6 which is better than using higher number of dimension. This is due to the fact that the sampling interval for the early stages is higher (one hour sampling interval), hence as we increase the sampling intervals, the variability gradually decreases and we hence have better performance.

It can be seen that for most of the models the performance of lasso is better than ridge due to the fact that lasso end up estimating sparse coefficient. The main difference between a lasso and a ridge regression is the use of a *L1* instead of an *L2* penalty.

This deference turns out to be important because an *L2* penalty only shrinks coefficients to zero but never sets them to zero exactly. In contrast, an *L1* penalty can set an estimate to zero, thereby excluding the corresponding variable from the active set. Lasso thus performs shrinkage and variable selection simultaneously. Sparsity is closely related to variable selection.



(a) (b)

Figure 5. 7 Estimation picture for lasso (a) and ridge regression (b) (Hasties, Tibshirani, and Freidman, 2003)

To learn that why lasso gives a sparse solution we could take a look at Figure 5.5. This figure is a graphical representation of for lasso and ridge regression. The center of the ellipse, i.e., $\hat{A}$ is the non-regularized OLS estimate. Each elliptical counter is the weight that results in the same square error. In lasso, the goal is to minimize the squared error subject to the constraint that the sum of the absolute value of the weights is less than some constant.

This constrain is defined by a square around the original; i.e., the black square on the left. On the other hand, in ridge regression the goal is that the squares of the weights to be less than some constant; hence, we get a sphere around the origin. Lasso is seeking for the first elliptical that interest the black square, which is highly happen at the corners of the square, and corners are places that weights are zeroed out. Hence it is expected that the lasso's performance be greater than the ridge.

5.8 Conclusion

In this chapter we mainly focused on estimating the coefficients of autoregressive time series model using two-data set. We used penalized regression techniques to estimate coefficients in which we performed least squares but with some additional constraints on the coefficients. We chose the loss function OLS estimator instead of LAD loss function, since LAD loss function since it is not differentiable at zero, and also graphically the *L1* ball has "pointy" corners (Kaiser 1958).

We applied ridge, SCAD, and lasso as three penalized regression techniques on both original data set and the reduced dimension data sets. Performance for each regression method under different conditions was measured by means of the coefficient of determination, denoted by R^2. It could be seen that the dimension reduction methods producing orthogonal independent variables are having higher r-squared value since because orthogonality leads to reasonable coefficient estimation with low standard errors.

On the other it could be seen that factor analysis outperformed the rest of methods, due to using varimax rotation in FA. Hence it can be seen that using FA makes most of the coordinates to be closer to zero so that points on the axis become more interpretable. Being close to the axis makes it sparser, which, in turn, induces additional sparsity subject to maintaining a certain goodness of fit, because it reduced the number of inputs and therefore decreased the model complexity.

Finally, it is seen that lasso outperformed the rest of methods used in this thesis. Because using L1 penalty encourage sparsity, but the L2 penalties in some sense discourage sparsity.

CHAPTER VI

CONCLUSION AND FUTURE WORK

Advance techniques in data collection have led to data overloading in many fields, such as biology and genetics. Such data sets present many challenges in data analysis; for example working with such data sets is computationally expensive. This dissertation focuses on dynamic analysis of two high dimensional microarrays. The major conclusions and future work recommendations are as follows.

6.1 Conclusions

By focusing on the need of dimension reduction techniques used in machine learning, the helpfulness of dimension reduction in microarray data can be seen. There are many reasons behind reducing the dimension of microarray data sets. For one thing, identifying the genes that play important role at different development stages is essential.

Different dimension reduction techniques were used, including FA, PCA, IBP, eigen Laplacian, and Nystrom method on two microarray data sets. For the sake of comparison, we then applied *k*-mean algorithm on them. It could be seen that factor analysis and eigen Laplacian were better when compared with other methods from the perspective of accuracy.

Although computing running time is a dependent act, it could be proof to our expectation for the Nystrom method. Since the Nystrom method only relies on its approximation of similarity matrix instead of computing the whole similarity matrix, the resulted reduced dimension data was derived with lower running time compared with the rest of dimensional reduction methods used in this thesis.

On the other hand, there are several challenges for a dynamic analysis of microarray data. First, trying to identify a large network from a large number of genes originating from smaller number of experiments (times) is challenging since the system is underdetermined. Second, due to the high correlation between genes, the resulted least squares estimators will have large variances. Therefore, it may be useful to reduce the dimension of the model to eliminate multicollinearity problems among large number of variables.

To overcome this issue, we then applied several penalized regression method on the reduced dimension data and on the original data set. The results show that explanatory power under factor analysis and principle component analysis is better than the multiple linear regression for both data sets and basically any method producing orthogonal dependent variables.

It can be seen that factor analysis has the highest r-squared value among all the of the dimension reduction techniques used. Lastly, it can be inferred among all the regression methods, lasso has a better performance.

In conclusion, we could see that factor analysis performed well in both accuracy and performance. The reason behind this is that using FA makes the most of the coordinates to be closer to zero so that points on the axis become more interpretable. Being closer to the axis makes

it sparser, which, in turn, induces additional sparsity subject to maintaining a certain goodness of fit.

6.2 Future work

It could be seen from the conclusion that factor analysis was working perfectly with respect to performance and accuracy. The reason behind this is because FA induces additional sparsity subject to maintaining a certain goodness of fit. The degree of sparsity should be investigated; further the degree to which sparsity is helping this issue should be interrogated.

Although dimension reduction has the benefit of reducing storage space and time, it can cause data distortion due to feature extraction, which reduces the recognition rate. Due to the defect of dimension reduction methods, we have to search for image processing methods to solve the defect of the losing characteristic of the original data after reduction.

REFRENCES

Al-Akwaa, Fadhl M. *Analysis of Gene Expression Data Using Biclustering Algorithms*.Arbeitman et al, *Gene Expression During the Life Cycle of Drosophila melanogaster, Science* 297, 2270 (2002).

Babu, M. Madan *an Introduction to Microarray Data Analysis.* p.225-249.

Ziv Bar-Joseph, *Analyzing time series gene expression data*, April 19, 2004, Vol. 20 no. 16 2004, pages 2493–2503.

Stephan.T.Barnard, *A Algorithm for envelop reduction of sparse matrices*, Numerical Linear Algebra with Applications, 8 July 2005.

Ben-Hur, A., Elissee, A., and Guyon, I. (2002). *A stability based method for discovering structure in clustered data.* In Pacific Symposium on Biocomputing (pp. 6 – 17).

Beyer, Kevin, et al. "When is "nearest neighbor" meaningful?." *Database Theory—ICDT'99*. Springer Berlin Heidelberg, 1999. 217-235.

Pete J.Brockwell, *Introduction to time series and forecasting*, 2nd ed. 2002, XIV.

Chen, Wen-Yen, et al. "PSC: parallel spectral clustering." *Software available. http://www. cs. ucsb. edu/~ wychen/sc* (2008).

Cunningham, Padraig adraig. *A Report on Dimension Reduction,* UCD-CSI-2007 August 8th, 2007.

Chong Wang, and David M. Blei, *Variational Inference for the Nested Chinese Restaurant Process,* Advances in Neural Information, 2009.

Dhillon, I. S.. *Co-clustering documents and words using bipartite graph partitioning*. In Proceedings of SIGKDD, pages 269–274, 2001.

Dhillon, Inderjit S., Yuqiang Guan, and Brian Kulis. "Kernel k-means: spectral clustering and normalized cuts." *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004.

D. A. Dickey and W. A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366):427–431, June 1979.

Estévez, Pablo A., et al. "Normalized mutual information feature selection."*Neural Networks, IEEE Transactions on* 20.2 (2009): 189-201.

Fan, J. and R. Li (2001). "Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties." J Am Stat Assoc **96**(456): 1348-1360.

Farrar, D. E. and R. R. Glauber (1967). "Multicollinearity in Regression Analysis: The Problem Revisited." The Review of Economics and Statistics 49(1): 92-107.

C. Fowlkes, S. Belongie, F. Chung, and J. Malik. grouping using the Nyström method. IEEE Transactions on Pattern Analysis.

Gao, X. and T. U. o. Iowa (2008). Penalized Methods for High-dimensional Least Absolute Deviations Regression.

S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. *In Proceedings of SOSP*, pages 29–43, 2003.

Golub, G., et al. (1999). "Tikhonov Regularization and Total Least Squares." SIAM Journal on Matrix Analysis and Applications **21**(1): 185-194.

Griffiths, Thomas Z Ghahramani *Infinite latent feature models and the Indian buffet process*, - 2005.

Thomas L. Griffiths et al, *The Indian Buffet Process: An Introduction and Review*, the Journal of Machine Learning Research archive, volume 12, 2/1/2011, p. 1185.

The Elements of Statisstical Learning: Data Mining, Inference, and Prediction In the Elements of Statistical Learning (30 August 2003), Trevor Hasties, Robert Tibshirani, and Jerome Freidman.

Heckman, N. E. and J. O. Ramsay (2000). "Penalized regression with model-based penalties." Canadian Journal of Statistics **28**(2): 241-258.

J.Herrero et al, *Gene Expression data preprocessing*, September 2002.

Huang, Joshua Zhexue, et al. "Automated variable weighting in k-means type clustering." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*27.5 (2005): 657-668.

Human cell cycle: HeLa cells [http://genome-www.stanford.edu/ Human-CellCycle/HeLa/]

S. Johansen. Likelihood-Based Inference in Cointegrated Vector Autoregressive Models. Oxford University Press, 1995.

Kaibo Duan et al, *Evaluation of simple performance measures for tuning SVM hyperparameters,* Neurocomputing 51 (2003), p.41 – 59.

Kaiser, H. (1958). "The varimax criterion for analytic rotation in factor analysis." Psychometrika **23**(3): 187-200.

Keene, Oliver N. "The log transformation is special." *Statistics in medicine* 14.8 (1995): 811-819.

S. Sathiya Keerthi et al , *Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel,* Neural computation, 2003.

Kiyoung, Y., et al. (2005). "On the Stationarity of Multivariate Time Series for Correlation-Based Data Analysis." IEEE Computer Society : 805-808.

Knowlton, N., et al. (2004). "Microarray Data Analysis Toolbox (MDAT): for normalization, adjustment and analysis of gene expression data." Bioinformatics **20**(18): 3687-3690.

Krishnapuram, Raghuram, and James M. Keller. "A possibilistic approach to clustering." *Fuzzy Systems, IEEE Transactions on* 1.2 (1993): 98-110.

R. Liu and H. Zhang. *Segmentation of 3D meshes through clustering. In Proceedings of Pacific Graphics,* 2004.

Ljung, Greta M., and George EP Box. "On a measure of lack of fit in time series models." *Biometrika* 65.2 (1978): 297-303.

Loehlin, J. C. (1998). Latent variable models: An introduction to factor, path, and structural analysis (3rd Ed.). Mahwah, NJ, US, Lawrence Erlbaum Associates Publishers: xi, 309.

U. Luxburg. *A tutorial on clustering. Statistics and Computing*, 17(4):395–416, 2007.

Ma, S., et al. (2006). "Additive risk models for survival data with high-dimensional covariates." Biometrics 62(1): 202-210.

Peng, Roger D., "A Method for Visualizing Multivariate Time Series Data" (February 2008). *Johns Hopkins University, Dept. of Biostatistics Working Papers.* Working Paper 166. http://biostats.bepress.com/jhubiostat/paper166.

Reinsel, Gregory C., and Sung K. Ahn. "Vector autoregressive models with unit roots and reduced rank structure: estimation. likelihood ratio test, and forecasting." *Journal of Time Series Analysis* 13.4 (1992): 353-375.

Ra Rosen, J., et al. (1996). "Total Least Norm Formulation and Solution for Structured Problems." SIAM Journal on Matrix Analysis and Applications 17(1): 110-126.

Rajapakse JC, Mundra PA: Stability of building gene regulatory networks with sparse autoregressive models. *BMC Bioinformatics* 2011, 12(Suppl 13):S17.

Rossi-Doria, O., et al. (2003). A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem. Practice and Theory of Automated Timetabling IV. E. Burke and P. Causmaecker, Springer Berlin Heidelberg. **2740:** 329-351.

Sei-Hyung Lee and Karen Daniels, *Thesis of Gaussian Kernel Width Selection And Fast Cluster Labelin For Support Vector Clustering,* 2005, p. 10-95.

Shawe-Taylor, N., and A. Kandola. "On kernel target alignment." *Advances in neural information processing systems* 14 (2002): 367.

Shawe-Taylor, John, and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

Jianbo Shi et al, *Normalized Cuts and Image Segmentation*, VOL. 22, NO. 8, August 2000

Song, S. and Bickel, P. J. (2011), Large vector auto regressions," Arxiv preprint arXiv:1106.3915.

Sotak Jr, George E., and Kim L. Boyer. "The Laplacian-of-Gaussian kernel: a formal analysis and design procedure for fast, accurate convolution and full-frame output." *Computer Vision, Graphics, and Image Processing* 48.2 (1989): 147-189.

Still, S. and Bialek, W. (2004*). How many clusters? an information-theoretic perspective.* Neural Comput., 16(12), 2483 – 2506.

Stock, James H. "Asymptotic properties of least squares estimators of cointegrating vectors." *Econometrica: Journal of the Econometric Society*(1987): 1035-1056.

Tibshirani, R. (1996). "Regression Shrinkage and Selection via the Lasso." Journal of the Royal Statistical Society. Series B (Methodological) **58**(1): 267-288.

Valdés-Sosa, Pedro A., et al. "Estimating brain functional connectivity with sparse multivariate autoregression." *Philosophical Transactions of the Royal Society B: Biological Sciences* 360.1457 (2005): 969-981.

Von Luxburg, Ulrike *A Tutorial on Clustering*, Article in Statistics and Computing, 17 (4), 2007

Wang, X., et al. (2006). "Missing value estimation for DNA microarray gene expression data by Support Vector Regression imputation and orthogonal coding scheme." BMC Bioinformatics **7**(1): 1-10.

Y.Weiss et al, On *Clustering: Analysis and an algorithm*, Advances in neural information …, 2002.

Yair Weiss , *Segmentations using Eigen vectors unifying view*, 1999, p. 975 - 982 vol.2.

Wen-Yen Chen et al, *Parallel Clustering in Distributed Systems*, March 2011, p. 568 – 586.

Whitfield, M. L., et al. (2002). "Identification of genes periodically expressed in the human cell cycle and their expression in tumors." Mol Biol Cell 13(6): 1977-2000.

R Xu, DC Wunsch, *Clustering algorithms in biomedical research: a review*, 04 October 2010, p. 120 – 154.

Zhiqiang Xu et al, *Model-based Approach to Attributed Graph Clustering*, 2008

VITA

Banafsheh Samareh Abolhasani

Candidate for the Degree of

Master of Science/Arts

Dissertation: **DYNAMIC ANALYSIS OF HIGH DIMENSIONAL MICROARRAY TIME SERIES DATAUSING VARIOUS DIMENSIONAL REDUCTION METHODS**

Major Field: Industrial Engineering and Management

Biographical:

Education:
Received Bachelor of Science degree in Mechanical engineering from University of Pune 2011; completed Master of Science with a major in Industrial Engineering and Management at Oklahoma State University in December, 2013.

Experience:
Employed by Oklahoma State University, at School of Industrial Engineering and Management as a graduate research assistant and teaching assistant, from August 2012 to December 2013.

Professional Memberships:
- Institute for Industrial Engineering (IIE)
- American Society of Quality (ASQ)
- Institute for Operations Research and the Management Sciences (INFORMS)