WORCESTER POLYTECHNIC INSTITUTE

Analysis and Prediction of Community Structure Using Unsupervised Learning

By

Rakesh Biradar

A thesis submitted to the faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Data Science

Jan-2016

APPROVED:

Professor: Randy C Paffenroth, Advisor

____

Professor: Xiangnan Kong, Reader

____

Contents

WORCESTER POLYTECHNIC INSTITUTE

Abstract

Data Science

Master of Science

By Rakesh Biradar

Abstract

In this thesis, we perform analysis and prediction for community structures in graphs using unsupervised learning. The methods we use require the data matrices to be of low rank, and such matrices appear quite often in real world problems across a broad range of domains. Such a modelling assumption is widely considered by classical algorithms such as principal component analysis (PCA), and the same assumption is often used to achieve dimensionality reduction. Dimension reduction, which is a classic method in unsupervised learning, can be leveraged in a wide array of problems, including prediction of strength of connection between communities from unlabeled or partially labeled data. Accordingly, a low rank assumption addresses many real world problems, and a low rank assumption has been used in this thesis to predict the strength of connection between communities in Amazon product data. In particular, we have analyzed real world data across retail and cyber domains, with the focus being on the retail domain.

Herein, our focus is on analyzing the strength of connection between the communities in Amazon product data, where each community represents a group of products, and we are given the strength of connection between the *individual* products but not between the product *communities*. We call the strength of connection between individual products *first order data* and the strength of connection between communities *second order data*. This usage is inspired by [1] where first order time series are used to compute second order covariance matrices where such covariance matrices encode the strength of connection between the time series. In order to find the strength of connection between the communities, we define various metrics to measure this strength, and one of the goals of this thesis is to choose a good metric, which supports effective predictions. However, the main objective is to predict the strength of connection between most of the communities, given measurements of the strength of connection between only a few communities. To address this challenge, we use modern extensions of PCA such as eRPCA that can provide better predictions and can be computationally efficient for large problems. However, the current theory of eRPCA algorithms is not designed to treat problems where the initial data (such as the second order matrix of communities strength) is *both low rank and sparse*. Therefore, we analyze the performance of eRPCA algorithm on such data and modify our approaches for the particular structure of Amazon product communities to perform the necessary predictions.

Acknowledgement

I would like to express my sincere gratitude to my advisor Prof. Randy Paffenroth for the continuous support of my thesis study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my thesis work.

Definitions & Basic Notation

In this Section, we define the following notations, some of which represent parameters used in the eRPCA algorithm. eRPCA takes as one of its inputs a second order data matrix and outputs a low rank matrix and a sparse matrix. The matrix formed by converting the first order matrix of communities and products, into a matrix that represents the relationship between communities is called a second order matrix and is denoted by $\boldsymbol{M}$**.** The number of rows or columns of the second order matrix (M) of Amazon community data, which represents the **number of communities** is denoted by $\boldsymbol{C_s}$ (sometimes in this thesis, we refer to the same idea as community size or length of communities)**.** The total number of entries in a second order matrix M, which is nothing but $C_s \times C_s$, is denoted by $\boldsymbol{N}$. The number of zeros in the second order matrix M is denoted by $\boldsymbol{M_z}$ and this represents the **sparsity** of M. The number of non-zeros in the second order matrix M is denoted by $\boldsymbol{M_{nz}}$.

The low rank matrix that arises from an eRPCA decomposition is denoted by $\boldsymbol{L}$**.** We measure two main components of the low rank matrix. One is the number of zeros in the low rank matrix L and it is denoted by $\boldsymbol{L_s}$**.** The other is the number of non-zeroes in the low rank matrix L and it is denoted by $\boldsymbol{L_{nz}}$. The individual entries in L are denoted by $\boldsymbol{L_{ij}}$. The rank of the low rank matrix is denoted by $\boldsymbol{L_R}$.

The sparse matrix decomposed from eRPCA, is denoted by $\boldsymbol{S}$. We measure two main components in the sparse matrix S. One is the number of zeros in the sparse matrix S and it is denoted by $\boldsymbol{S_z}$**.** The other is the number of non-zeroes in the sparse matrix S and it is denoted by $\boldsymbol{S_{nz}}$**.** The individual entries in S are denoted by $\boldsymbol{S_{ij}}$.

Chapter 1

Introduction

The main challenge addressed in this thesis is the analysis and prediction of the strength of the connection between Amazon products communities [2] using robust dimension reduction techniques [1]. Herein, we begin by tracing back to the roots of solving such problems using Principal Component Analysis (PCA) [3], which is a statistical method that transforms a set of observed data into a group of linearly uncorrelated variables. A typical application of PCA is for dimensionality reduction and the prediction of a set of unknown values of the variables from a set of known values of the variables. There are various dimensionality reduction techniques for performing PCA, and the Singular Value Decomposition (SVD) is the standard technique in this domain [3]. In this thesis, we use three main techniques SVD [4], RPCA [5] and eRPCA [1] on Amazon communities to analyze and predict the strength of connection between the communities. We explain the challenges faced in predicting the strength of connection between the communities and we derive several approaches for making such predictions.

The results obtained in our experiments provide better understanding of performance of the eRPCA algorithm, and the ways it can be used for novel prediction problems. The prediction of entries in second order matrix M can be substantially improved, by a deeper and richer understanding of the parameters of the algorithm, and this analysis has been a novel contribution in our thesis. In addition, the results also show the best way of measuring the errors between the entries of the second order matrices we study.

1.1 Motivation

We are seeing the burgeoning of technological devices and their users every year, therefore there has been an exponential increase in the data collected because of users activity on these devices [6]. The data collected from these activities can be categorized mainly into two types, one is structured data and the other is unstructured data [7]. The growth of unstructured data has been increasing more rapidly with various new forms of unstructured data such as text data, audio data, network data (which can be both structured and unstructured) [8] and video data occurring all the time [9].

Our interest in this thesis has been mainly on analyzing network data. Networks or graphs can be mathematically represented as a set of points or vertices joined in pairs by lines or edges. In practical settings, networks are a natural way to represent social, biological, technological and information systems [10]. There are various types of network data spread across different domains, such as social network data, web network data, electronic circuit data, biochemical network data and many more [10] [11]. Nodes in many such networks organize naturally into densely linked groups that are commonly referred to as *network communities* or *clusters* [12]. There are many reasons why nodes in network organize into densely linked groups. For instance, society is organized into social groups, families and associates. Similarly on the Internet, topic related pages link more closely among themselves. As the amount of data is growing, the communities are becoming bigger in size and the numbers of communities are increasing. With such a rapid increase, prediction of the strength between the communities becomes much more important for various reasons. Today, as we have huge number of social and internet groups, predicting the strength or similarities between the groups is essential to provide actionable information for various socio-economic and business decisions, and providing such information has become a substantial challenge. Therefore, there are prominent methods that help to predict the strength between the large groups by just knowing the strength between a few groups and this thesis demonstrates how to leverage one such method for prediction of communities connection strength.

There are many research papers on community definition and detection [7], [2] , [12] and [10], and our problem leverages that body of literature by assuming that we know the communities either by structural (mathematically calculated) or functional (naturally formed) methods. We have taken a few of the best community definitions from the research done in [12], and having the community definition in hand, we try to predict the strength of connection between many communities given the strength of connection between a few communities. Many real world problems are very similar to the one on which we are working, for example in social network data and protein molecule data, we already know the communities based on different kinds of user interaction and protein molecule reaction behavior. Unfortunately, in many kinds of such network data there are only a few communities whose strength is known and most of them are unknown [12], and this makes our problem pragmatically challenging to find the strength between those unknown communities.

In this thesis we have focused on the Amazon communities data as it represents real world communities or network data, and this data presents interesting opportunities for analysis that are quite similar to many problems in different domains of network data. The Amazon data has co-purchasing frequency of many Amazon products and the product's ground truth communities based upon the categories of the products [12]. The questions that we attempt to answer in this thesis are of practical importance because they assist in marketing by cross selling of products. In addition, for the new products that are launched in the market, it is very useful to predict their best affiliation with present product communities. With such a huge number of communities and products, we can identify the strength between the communities by knowing the strength of just a few communities. Unfortunately, such problems are difficult to solve using PCA or RPCA type techniques because they are simultaneously low rank and sparse. As we will detail in Chapter 2, 3 and 4 such sparse connections between communities leads to problems that are quite difficult, but tractable using the techniques developed here.

Second order matrices are formed by filling in the values of the strength of the connections between the communities. If we look at the structure of the second order matrices of Amazon product communities, the exact structures of interest are ill defined at the onset. However, as we demonstrate here such matrices with large numbers of communities can be analyzed as a dimension reduction problem, which is a classic example of unsupervised learning. To remove the curse of dimensionality we must leverage the fact that such data are intrinsically low dimensional, e.g. that they lie on some low-dimensional subspace [13] [3], are sparse in some basis [13], or lie on some low-dimensional manifold [3]. In addition, as we examine the quality of our data, we note that we have many values that are missing or incorrect, and these aspects of our data make the problem much more complicated. Therefore, we use robust techniques like RPCA, and our own eRPCA, which as we demonstrate can make accurate predictions even in the presence of such complicating factors.

1.2 Background

One of the most important properties of a matrix to be considered in our thesis is its rank. The rank of a matrix is defined as the maximum number of linearly independent columns (or rows) in the matrix. We can look at the rank of matrices in practical settings, where it can be interpreted as the amount of redundancy in a matrix. Lower rank means more redundancy and therefore more predictable entries. In effect, low rank implies that we require fewer entries to predict other entries in the matrix. In the Amazon communities data, a low rank second order matrix allows us to use the strength of only a few communities to predict the strength of many other communities (but not all) in the matrix.

Before we proceed further, we need to understand why the assumption of low rank is so important and reasonable. A low-rank matrix can be understood in many ways. It is a matrix with small number of non-zero singular values in its singular value decomposition (SVD). It is also a matrix with a small number of linearly independent rows (or columns), which produce the remaining rows (or columns) as the linear combination of those few. A matrix always has column rank equals row rank [14] [13]. In visual terms, it means there are a small number of basis rows (columns) that span the range space of the matrix. In many domains, low rank matrices are encountered quite often, so the assumption of low rank is a reasonable one. For an $n \times n$ matrix, a low-rank matrix by definition has rank k, where $k < n$, and this property has various benefits as it helps to reduce computational cost, make prediction of entries and detect anomalies. Accordingly, the low rank property of a matrix is important in many applications [15]. Especially, in our case for making predictions, working with low rank matrices can be extremely advantageous.

PCA is arguably the most widely used statistical tool for data analysis and dimensionality reduction today. Consider a matrix M, whose columns have been normalized to mean zero and each row represents one record of the collected data, and each column represents one field. In PCA analysis, the assumption is that the data all lie near some low-dimensional subspace. More precisely, it means that if we stack all the data points as column vectors of a matrix M, the matrix should have (approximately) low rank. Classical PCA decomposition can be written as $M = L_0 + N_0$, where L_0 is the true low rank component, and N_0 is small noise, and L is the approximation of L_0. Classical PCA [3] seeks the best (in a l^2sense) rank-k estimate of L_0 by solving as showing in equation 1)

$$L = \arg\min_{L} \left|\left|M - L\right|\right|_F^2, \qquad 1)$$
$$s.t. rank(L) \leq k,$$

where the Frobenious norm $\|\quad\|_F$ is defined in Chapter 2. The most common way to compute PCA is by using the SVD. This is a common linear algebra technique used to factorize matrix in to three main components. The equation for the SVD is shown in 2)

$$M = U\Sigma V^T , \qquad 2)$$

where U and V are unitary, so multiplying by their respective (conjugate) transposes yields identity matrices i.e. $UU^T = I \; and \; V^T V = I$. In addition, Σ is a diagonal matrix and the entries of Σ are known as the singular values of M, and the columns of U and V are known as left singular vectors and right singular vectors of M. The relationship between PCA and SVD is explained in detail in Chapter 2.

Our Amazon communities data has two main properties that make its analysis challenging: 1) Many of the entries in Σ are small from the beginning, 2) Changing just a small number of entries in M can have a large effect on Σ (since so many entries in M are already zero).

In addition, PCA is brittle with respect to grossly corrupted observations that can put its validity in jeopardy for making predictions in the Amazon Community Data. For example, a single grossly corrupted entry in M could render the estimated L arbitrarily far from the true L_0. Unfortunately, gross errors are frequent in modern applications such as image processing, web data analysis, and bioinformatics, where some measurements may be arbitrarily corrupted (due to occlusions, malicious tampering, or sensor failures). When one has grossly corrupted entries, one could use many modern techniques, which are the extensions of PCA. As we will discuss such modern extensions of PCA like RPCA [5] can solve the problem when M includes corrupted data and, eRCA technique can solve similar problems for data which has corruptions or uncertainty in all points, and potentially large and varied amounts of it.

The RPCA technique can be used to treat many real world data matrices and recover their low rank component (L_0) and sparse matrix component(S_0). In particular, we use RPCA to decompose M as $M = L + S$, where L is low rank and S is sparse. The low rank matrix represents those parts of the data that can be represented by limited number of basis (vectors) and this is where we can make easy predictions. The sparse matrix cannot be explained by the basis obtained by the low rank approximation but the entries in the sparse matrix represent, perhaps, important effects in the data. There are many applications of RPCA and it has been utilized in a number of different domains [15].

Finally, we explain some high level ideas regarding a formulation of RPCA that combines robustness to noise with partial observations. This method uses entry wise error constraints that allow entries of the matrix to have different noise properties, as opposed to classic approaches of RPCA that apply a single global noise constraint. We call this method eRPCA, and it can be used to decompose $M = L + S + Z$, where L is a low rank matrix, S is a sparse and Z is a dense matrix of small noise terms. Classically, one treats a low rank matrix M, which is incomplete but dense (i.e. not too many zeros), however in our problem we are forced to treat a low rank M, which on one hand has many zeros and on the other is dense with anomalies. Accordingly, a method for analyzing a matrix M, which is incomplete, sparse (many of the observed entries are zero) and low rank (which the Amazon data forces on us) needs to developed. *The development of such a method is a key novelty of our thesis.*

1.3 Experiments & Results

Our focus here is mainly on the predictive capabilities of the eRPCA algorithm, with a practical mindset of analyzing Amazon community data. There are number of features of eRPCA algorithm that would allow us to improve the prediction accuracy and understand the way the parameters need to be changed for better prediction. Our thesis work revolves around understanding all such parameters and we mainly categorize our work into three parts.

The raw data that we get from Amazon product communities is very challenging as there is no direct measurement of the strength between the communities. Therefore, our *first objective* is to form a second order matrix of Amazon communities that gives us a matrix whose rows and columns represent communities, and whose values give us a measurement that represents the strength between the communities. After forming the second order matrix of different measurements of strength of connection between the communities, the *second objective* is to validate the applicability of the second order matrix using algorithms like SVD and eRPCA, and choose the most appropriate definition of community connection strength going forward in our analysis.

In the eRPCA algorithm, there are many parameters to explore but we restrict our analysis to those parameters that would help in prediction, which is our *third objective*. We experiment on various parameters of eRPCA involved in predicting the entries, and analyze various ways to make the better prediction of entries. We have mainly experimented with using the controlling parameter λ to understand the low rank and sparse matrix to make better predictions. We have also found a formula for the empirical relation between $\boldsymbol{\lambda}$ and non-zero entries in the sparse matrix, which would help us to predict the number of non-zeroes in the sparse matrix given a value of $\boldsymbol{\lambda}$. Finally, we have proposed a different way to measure the errors between the entries of the second order matrix and the low rank matrix.

1.4 Our Contribution

In this work, we make several contributions to the current understanding of the problem at hand. As we look into the raw data of the Amazon communities product data [2], there are many ways to measure the strength of communities connection, and our contribution revolves around analyzing various ways to measure the communities strength, and we use SVD to measure the better way of representing the strength of the communities connection for eRPCA algorithm. We leveraged three definitions from the literature [12] and derived one novel definition of our own.

The current theory of the eRPCA algorithm is not designed to deal with a second order matrix M which is sparse (many of the observed entries are zero) [1]. Accordingly, we develop a method for analyzing a matrix M, which is incomplete, sparse and low rank (which the Amazon data forces on us), and this method is a novel contribution of this thesis.

We have experimented on balancing the entries of a second order matrix filling between the low rank and the sparse components. A study has also been made to understand the behavior of the low rank and the sparse matrix by changing the controlling parameter λ, where higher λ values will make putting non-zero entries into the sparse matrix more difficult. In other words, a low λ value will make L lower rank by putting more entries in S. We have also found an empirical relationship between the number of non-zeroes in the sparse matrix and the controlling parameter λ, which would give us the ability to predict the non-zeroes in the sparse matrix for a given value of λ. When dealing with such a second order data matrix, which is sparse, it becomes difficult to understand how to measure the errors in the various entries with usual way of measuring errors. Therefore, we layout a different way of looking into such problems, we have used the percentile values to measure the errors between the entries. Such an analysis gives a clear understanding of the errors of the entries than usual methods.

1.5 Structure of thesis

In this work, we attempt to provide the reader with background information on the problem that is being solved. Chapter 2 describes the theory and mathematical analysis on the problem. Chapter 3 describes the Amazon communities dataset. Chapter 4 focus on the experiments and analysis done in this thesis. Chapter 5 concludes this work and mentions further enhancements.

Chapter 2

Theoretical & Mathematical Derivation

This Section includes the background theory and mathematical analysis of the thesis work and describes the prediction of Amazon community structure using the eRPCA algorithm. We present the basic concepts of dimensionality reduction and principal component analysis, and then extend these concepts by way of the SVD. We explain about singular values that are essential components of our analysis. They are the key parameters for our prediction of the strength of connection between the communities. We then explain the limitation of using PCA on the Amazon data and then derive extensions to PCA, such as Robust PCA and eRPCA, which are the focus of our thesis work.

2.1 Principal Component Analysis

PCA is a statistical method that is used to convert high dimensional data into low dimensional data that does not involve any response (or dependent) variable [16]. It is also defined as a method that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of linearly uncorrelated variables called principal components, where the number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.

PCA can be performed on the covariance matrix or the correlation matrix (in which each variable is scaled to have its sample variance equal to one). For the covariance or correlation matrix, the eigenvectors correspond to principal components (PC) and the eigenvalues to the variance explained by the principal components. Principal component analysis of the correlation matrix provides an orthonormal eigen-basis for the space of the observed data. In this basis, the largest eigenvalues correspond to the principal components that are associated with most of the covariability among the observed data.

Let M be the second order matrix of size $m \times m$, an eigenvector or characteristic vector of a square matrix M is a vector that does not change its direction under the associated linear transformation.

In other words, if v is a vector that is not zero, then it is an eigenvector of a square matrix M if Mv is a scalar multiple of v. This condition can be written as $M.v = \lambda.v$,

where,

M: $m \ x \ m$ *covariance matrix or second order matrix*,
v: $m \ x \ 1$ *non − zero vector or eigen vector*, and
λ: *scalar*.

The resulting vectors are an uncorrelated orthogonal basis set. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is real and symmetric. The value of λ for which this equation has a solution is called the eigenvalue of M and the vector v, which corresponds to this value, is called the eigenvector of M.

How is PCA related to our thesis? When we convert high dimensional data to low dimensional data using PCA, the low dimensional PCs are a faithful representation of the high dimensional data. Accordingly, we can use PCs to do prediction of variables or features. In addition, we can use a small number of PCs or a large number of PCs based on the required percentage of variance to be captured in our dataset (and there for the accuracy of our predictions).

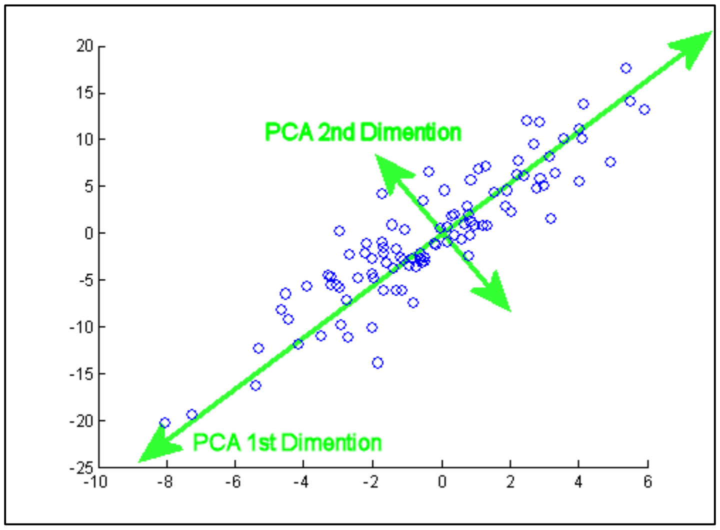


Figure 2: In this figure, the blue dots represent the sample data points, for these data points we draw two principal components that can explain the maximum variance of the data. The principal components we get from PCA are PCA 1st dimension (the direction of largest variance) and PCA 2nd dimension (the second largest direction of variance). As you can see, these two direction are, by definition, orthogonal.

PCA is often used as a tool in exploratory data analysis and for making predictive models. PCA is the simplest of the eigenvector-based methods for multivariate analysis. PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value decomposition of a data matrix, usually after mean centering and normalizing the data matrix for each attribute. Often, its operation can be thought of as revealing the internal linear structure of the data in a way that best explains the variance in the data. If a multivariate dataset is represented as a set of coordinates in a high-dimensional data space (1 axis per variable), PCA can supply the user with a lower-dimensional visualization, i.e. a projection or "shadow" of this object when viewed from its most informative viewpoint. This is done by using only the first few principal components so that the dimensionality of the transformed data is reduced. PCA is sensitive to the relative scaling of the original variables. In the Figure 2, the data points (blue dots) are represented in the original coordinates, and we draw two PCs (PCA 1st and 2nd Dimension) to explain the variance of the data points, where the PCA 1st Dimension captures more variance compared to PCA 2nd Dimension.

2.2 Singular Value Decomposition

The PCA algorithm is usually implemented by computing the eigenvalues and eigenvectors of the covariance matrix M, which is the product XX^T, where X is a normalized data matrix of size $m \times n$. Since the covariance matrix is symmetric, the matrix is diagonalizable, and the eigenvectors can be normalized such that they are orthonormal. The columns of W are the eigen-vectors of XX^T, and D is a diagonal matrix containing the eigen-values of XX^T. So,

$$M = XX^T = WDW^T. \qquad 3)$$

Analogously, applying the SVD to the data matrix XX^T produces the following decomposition:

$$X = U\Sigma V^T, \qquad 4)$$

where,

U is an $m \times m$ orthogonal matrix of the left singular-vectors of M,
V is an $n \times n$ orthogonal matrix of the right singular-vectors of M, and
Σ is an $m \times n$ matrix with non-zero diagonal entries, the diagonal values inside of which are referred to as the "singular-values" of M.

Attempting to construct the covariance matrix from this decomposition gives

$$\begin{aligned} XX^T &= (U\Sigma V^T)(U\Sigma V^T)^T \\ XX^T &= (U\Sigma V^T)(V\Sigma U^T) \end{aligned} \qquad 5)$$

and since V is an orthogonal matrix we have that $V^TV = I$. Therefore the equation 5) becomes

$$M = XX^T = U\Sigma^2U^T \qquad 6)$$

and the correspondence between the two approaches is easily seen (the square roots of the eigenvalues of XX^T are the singular values of X, etc.).

The singular values computed by the SVD allow one to determine what combination of variables is most informative (high variance), and which ones are not useful (low variance), and thereby perform PCA. The way it works is simple. You perform SVD over your training data (call it matrix X), to obtain U, S or Σ and V^T. Then, set to zero all values of S less than a certain arbitrary threshold (e.g. 0.1), call this new matrix S'. Then obtain $X' = US'V^T$ and use X' as low rank matrix as your PCA projection. Some of your features are now set to zero and can be removed, sometimes without a negligible performance penalty (depending on your data and the threshold chosen). *The SVD allows you to predict when information and features are redundant and when some features are linear combination of others, and therefore predictable.*

2.3 Low Rank Matrices

A low rank matrix can be thought of in multiple ways. It is a matrix, which has small number of non-zero singular values in its SVD [16]. It is also a matrix with small number of linearly independent rows (or columns), which means it needs a small number of rows (or columns) to predict the remaining rows (or columns) as linear combination of those independent rows (or column). In a low rank matrix, column rank is always equal to row rank [13].

In addition, low-rank approximation can be thought of as a minimization problem, in which the cost function measures the fit between a given matrix (the data) and an approximating matrix (the optimization variable), subject to a constraint that the approximating matrix has reduced rank. The rank constraint is related to the complexity of a model that fits the data. Let X be a data matrix of size $m \times n$, and k be a positive integer, we would like to find an $m \times n$ matrix X_k of rank at most k, so as to minimize the $Frobenius\ norm$ of the matrix difference

$C = X - X_k$, defined to be

$$\|C\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij}^2}\ . \qquad 7)$$

Therefore, the Frobenius norm of C measures the discrepancy between X and X_k. Our goal is to find a matrix X_k that minimizes this discrepancy, while constraining X_k to have rank at most k. If r is the rank of X then $X = X_r$ and the Frobenius norm of the discrepancy is zero in this case. When k is smaller than r, we refer X_k as low rank approximation of X.

The problem with fit measured by the Frobenius norm is

$$\min_{X_k} \|X - X_k\|_F{}^2, \quad subject\ to\ rank(X_k) \leq r. \qquad 8)$$

Perhaps surprisingly, this optimization has and analytic solution in terms of the singular value decomposition of the data matrix. The result is referred to as the matrix approximation theorem or the Eckart–Young–Mirsky theorem [17].

In brief, the Eckart–Young–Mirsky theorem teaches us that the minimization of this optimization can be computed using the following deterministic procedure. Let X be a second order matrix then

$$X = U\Sigma V^T \in R^{mxn}, m \leq n$$

is the singular value decomposition of X and partition U, V and $\Sigma =: diag(\sigma_1, \ldots., \sigma_m)$

as follows

$$U =: [\,U_1\ \ U_2\,]\,, \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}\ and\ V =: [\,V_1\ \ V_2\,]\,,$$

where Σ_k is $r\,X\,r$, U_k is $m\,X\,r$, and V_k is $n\,X\,r$. Then the rank-r matrix, obtained from the truncated singular value decomposition is

$$X_k = U_k \Sigma_k V_k^T\ ,$$

and we have that

$$\|X - X_k\|_F = \min_{rank(X_k) \leq r} \|X - X_k\|_F = \sqrt{\sigma_{r+1}^2 + \cdots \sigma_m^2}. \qquad 9)$$

The minimizer X_k is unique if and only if $\sigma_{r+1} \neq \sigma_r$ [ ]. This low rank approximation appears in most of the practical situations as explained in Chapter 1 and we use this idea as a key component of our prediction methodology.

2.4 Robust PCA

Traditional PCA recovers a low rank matrix from a high dimensional matrix M in the presence of small perturbation as represented in the below equation,

$$M = L_0 + N_0 \, , \qquad 10)$$

Where

M – high dimensional matrix,

L_0 – Low rank matrix (unobserved) and

N_0 – Small perturbations.

From equation (10), we can optimize the rank of M using the solution given by the SVD, but as mentioned before PCA is highly sensitive to outliers or corrupted data. Small errors in M will make small changes to the singular values of M, and as the result of which the actual singular values with small noise and without small noise will have a similar values which could be neglected in our case as that wouldn't affect the approximate rank of the matrix [5]. However, large errors may make large changes to the singular values, and drastically alter the non-zero singular values. This makes the optimization of the rank of M harder and may give rise to inaccurate results. This disadvantage of PCA is ameliorated by RPCA in order to solve the optimization of the rank of M, when the data is highly corrupted.

Robust Principal Component Analysis (RPCA) is a modification of the widely used statistical procedure PCA that works well with respect to grossly corrupted observations. A number of different approaches exist for Robust PCA, including an idealized version of Robust PCA, which aims to recover an approximate low-rank matrix L from highly corrupted measurements $M = L_0 + S_0$, unlike the small noise term N_0 in classical PCA, the entries in S_0 can have arbitrarily large magnitude, and their support is assumed to be sparse but unknown [5]. The unknown support of the errors makes the problem more difficult than the matrix completion problem that has been recently well studied [18]. This decomposition into low-rank and sparse matrices can be achieved by techniques such as Principal Component Pursuit method (PCP), Stable PCP [19], Quantized PCP [1], Block based PCP [1], and Local PCP [19] under some basic assumptions about the rank and sparsity of L_0 and S_0 [20] and, in reality, these assumptions are often satisfied by real world data.

The RPCA decomposition [21], with unobserved entries [18] is constructed by the following optimization. We are given a matrix $M \, \epsilon \, R^{mxn}, m \leq n$ that is formed by If $M = L_0 + S_0$ and we are given only $P_\Omega(M)$ (defined below), and if certain identifiability, rank, and sparsity conditions on L_0, S_0 and Ω (defined below), and are met; then, with high probability, the decomposition can be recovered by the following *Principal Component Pursuit* (PCP) convex optimization problem

$$\min_{L,S} \|L\|_* + \lambda \|S\|_1 \quad \text{Subject to } P_\Omega(M) = P_\Omega(M+S), \tag{11)}$$

with $\lambda = \sqrt{m/|\Omega|}$, or for the fully observed case which we have $\lambda = \sqrt{1/\max(m,n)}$, exactly recovering the low rank matrix L_0 as well as the entries of the sparse matrix $S_0 = P_\Omega(S_0)$. $\|L\|_*$ is defined as the nuclear norm of a matrix or sum of the singular values, $\|L\|_* = \sum_{k=1}^{n} \sigma_k(L)$, and the one-norm is represented as $\|M\|_1 = \sum_{k=1}^{n} |M_k|$ and is defined as the sum of the magnitudes. We denote by Ω the locations of the observed entries i.e. $(i,j) \epsilon\, \Omega$, if M_{ij} is observed as

$$\Omega\,(i,j) = \begin{cases} 1 & i,j \in \Omega \\ 0 & otherwise \end{cases} \tag{12)}$$

We denote by $P_\Omega(M)$ the projection of the matrix M onto the set of entries indexed by the indices i, j in the Ω as

$$[\,P_\Omega(M)]_{i,j} := \begin{cases} M_{i,j} & i,j \in \Omega \\ 0 & otherwise \end{cases} \tag{13)}$$

We must tease out the underlying low rank matrix L_0, and identify the sparse anomalies introduced by S_0, *without knowing a priori* the true rank of L_0, and without knowing the number or locations of the nonzero entries in S_0. Furthermore, the magnitude of the few nonzero entries in may be of arbitrarily large size. These challenges may be further compounded by failure to observe a subset of entries in M, and by noise, that adds small errors to each of the observed entries. Technically speaking, the second order matrices M in this thesis are all fully observed. However the use of vE_H plays the role of designating the "unobserved" entries in this thesis [1].

2.5 eRPCA

Building upon the RPCA derivation in the previous section, here we present some high level ideas regarding a new formulation of Robust Principal Component Analysis, called eRCPA that combines robustness to noise with partial observations [1]. This method uses point wise error constraints that allow entries of the matrix to have different noise properties, as opposed to the standard Frobenius norm approach that applies a single global noise constraint. This derivation closely follows that in [1].

In [1], we find theorems and algorithms for addressing noisy problems with partial observations, based upon an equivalent problem formulation, which allows solution of the optimization using a standard Alternating Direction Method of Multipliers [1]. Herein, we apply that method to second-order matrices to detect sparsely correlated phenomena in measured data from the SKAION network and make predictions in the Amazon Network Communities dataset.

The RPCA algorithm for performing matrix decomposition into low rank and sparse components has been extended further with addition of small but *dense* noise. To that end, we are interested in recovering M from $M = L_0 + S_0 + Z_0$, where Z_0 is a dense matrix of small noise terms. In this case, the convex program of interest is *Principal Component Pursuit with Frobenius Constraints* given by

$$\min_{L,S} ||L||_* + \lambda||S||_1 \,, \quad subject\ to\ \|M - L - S\|_F \preccurlyeq \delta \,, \tag{14)}$$

Where λ is the tradeoff parameter between the rank of L and sparsity of S and $\delta \coloneqq ||Z_0||_F$. Algorithms for solving the matrix decomposition problem have been presented in [22] and [20] and the *Principal Component Pursuit with Entry-Wise Constraints* [1] is given by equation

$$\min_{L,S} ||L||_* + \lambda||S||_1 \,, \quad subject\ to\ |P_\Omega(M) - P_\Omega(L+S)| \preccurlyeq \epsilon, \tag{15)}$$

where ϵ represents a matrix of entry-wise error bounds. We define ϵ as

$$\epsilon = \begin{cases} \epsilon_{ij}, & i,j \in \Omega \\ \infty\,, & otherwse \end{cases} \tag{16)}$$

However, in practical application, we make $\epsilon_{ij} = vE_H$ in order for the eRPCA algorithm to predict the corresponding entry (i,j) of M. We refer to our algorithm for solving PCP in noisy environments using inequality constraints as eRPCA. In this acronym, the RPCA stands for "Robust Principal Component Analysis", while the 'e' in eRPCA is a reminder that inequality constraints are enforced *point-wise* with matrix epsilon.

There are many attributes of the eRPCA algorithm and we use only a few of them in our thesis. The maximum rank of M to consider for completion, is denoted by $\mathbf{P_{max}}$. The value of the coupling constant between L and S is denoted by $\boldsymbol{\lambda}$. This constant is used to balance the density between the low rank and sparse matrix**.** The number of entries observed in eRPCA is denoted by **K.** The number of entries in M to be predicted is denoted by $\boldsymbol{M_p}$.

The point wise error bounds for each of the entries in M, that is used by eRPCA to make predictions is denoted by $\boldsymbol{vE}$, the values of vE used in our thesis is either High (1e+5 and denoted by $\boldsymbol{vE_H}$) or low (1e-5 and denoted by $\boldsymbol{vE_L}$). In particular, we use vE_H, in order to designate those entries we wish to use the eRPCA algorithm for making predictions. The indices that are observed on the matrix M are denoted by u, v. The decomposed low rank and sparse matrix constructed by eRPCA, where some of the entries are used for prediction by using vE_H, are denoted by $\boldsymbol{L_p}$ and $\boldsymbol{S_p}$. In the analysis of our matrices, if we consider all the entries of L or M is called ***Complete Entries*****.** The entries of $L\ or\ M$ whose corresponding entries of sparse matrix are zero are called **GPE - Good Predictable Entries** (as these entries are always predicted well (with small error).

2.6 What Makes our Second Order Matrix Novel?

Let X be a data matrix of size $n\ x\ p$, where n is the sample size and p is the number of variables (we refer to X as a first order matrix). Let us assume that variables are centered i.e., the column means have been subtracted are now equal to zero. Then the covariance matrix M is formed by taking the dot product of first order matrix and its inverse, it is defined mathematically as $M = X^T X/(n-1)$ which is of size $p\ x\ p$. The three important properties of M that are important for our work are 1) The values of the covariance matrix lies in the range of $-\infty < M < \infty$ 2) In most practical applications, there are a few zeros and many non-zero values ($M_z \ll M_{nz}$) 3) M is a symmetric matrix. This is a typical kind of second order matrix M that has been dealt with by eRPCA in the literature [1].

On the other hand, the formation of the second order matrix of Amazon communities data is different. The second order matrix of Amazon communities data has been formed by taking the edges and nodes between the communities to represent the strength of connection between the communities, where each communities represent the group of nodes (Amazon products) and the details of this formulation is explained in Chapter 3. In short, the typical M represents the correlation, or linear similarity, between the random variables, whereas the M being dealt with here represents a derived definition of similarity well suited for the Amazon communities data.

The three important properties of M that are important for our work on Amazon communities data are 1) The values of the second order matrix lies in the range of $0 \leq M < \infty$ 2) There are many zeros and a few non-zero values ($M_z \gg M_{nz}$) 3) M is a symmetric matrix. *This is the kind of second order matrix M that has never been dealt by eRPCA so far*. In particular, property 2) above is quite challenging for standard techniques and the ability to treat these type of problems is a key novelty of this thesis.

We will illustrate with a diagrammatic representation the two kinds of M discussed in the above paragraphs. In the figure 2.1, the typical second order matrix M does not have many zeroes. In the figure 2.3 and 2.4, the second order matrix M that we are mainly dealing with in our thesis work *has many zeroes* and classic eRPCA is not designed for such matrices.

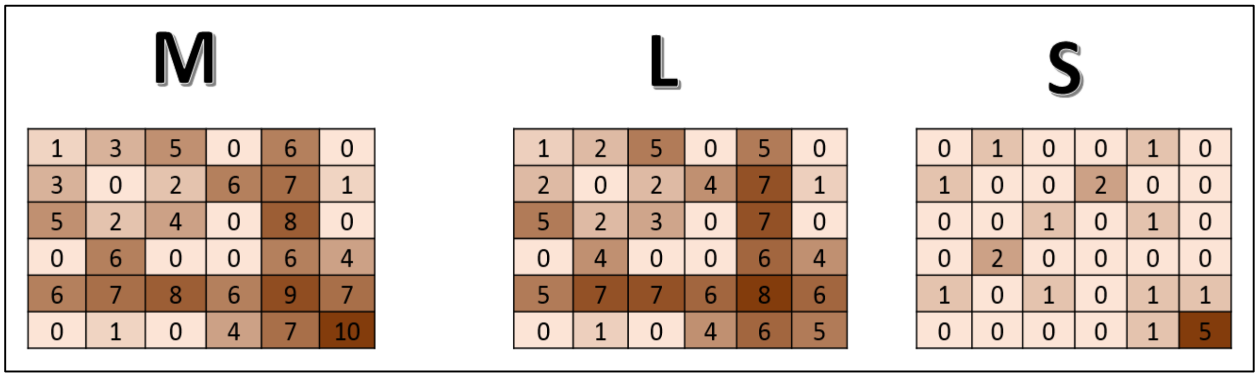


Figure 2.1: In this Figure, it shows a typical second order matrix that is not much spare, and most these matrices are being solved using dimensionality reduction techniques like RPCA or eRPCA, the color with different intensity represents the values inside the matrix.

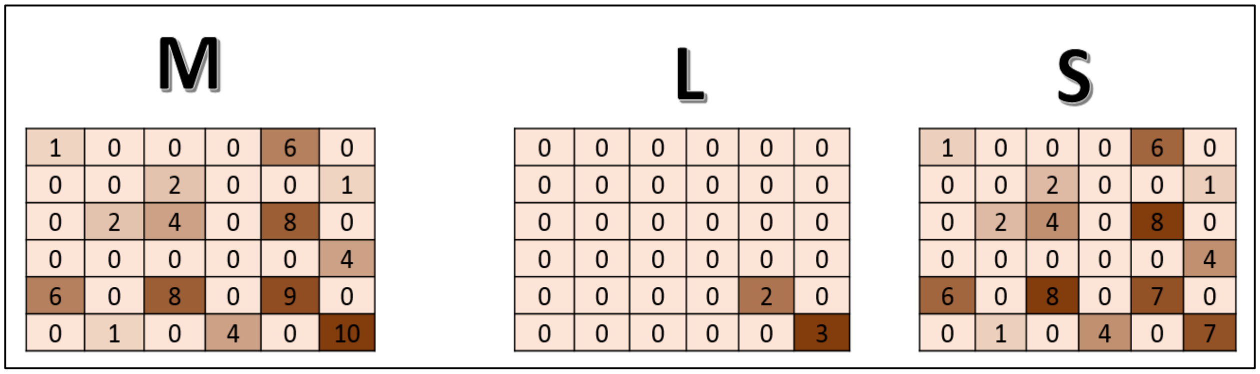
The decomposition of the second order matrix gives us a low rank matrix L and sparse matrix S. We can compute L and S from M as discussed in Sections 2.4 and 2.5. With a decomposition of a typical M using eRPCA, the low rank matrix L in Figure 2.1 has less than 6 linearly independent columns (or rows) (i.e., at least one part of one column can be predicted from the rest). And, the sparse matrix has many values that are zero and a few anomalies which represent a classic example of the decomposition of typical second order matrix into L and S. The above decomposition using eRPCA is performed with λ_0, where $\lambda_0 = 1/\max(m, n)$, in other words $\lambda = l \times \lambda_0$,where $l \epsilon \mathbb{R}$. In this case $l = 1$, however a judicious choice of l is a fundamental aspect of treating problems where is M is low rank and sparse simultaneously.

This problem would have been much more straightforward if the above decomposition had occurred in a similar fashion for our Amazon communities second order matrix but unfortunately, this did not happen. In the first place, the Amazon communities second order matrix is the kind of M where most of the values are zero and a few of the values are non-zero ($M_z \gg M_{nz}$). An example of such a matrix is shown in Figure 2.2 and 2.3, and such matrices are much harder for eRPCA to decompose.

When we use techniques like eRPCA on the Amazon communities data M, we find that all the singular values of L for such a matrix are zero when we use $\lambda_0 \; or \; \lambda = 1 \times \lambda_0$, and this means that no entries in L can be effectively predicted (i.e., every entry in M is viewed as an unpredictable anomaly). This makes our second order matrix special, which is already a low rank with a few outliers as shown in Figure 2.2. In addition, to our surprise, the sparse matrix S is actually not sparse anymore. This means that the sparse matrix S has many non-zero values as compared to the low rank matrix L (i.e.,$L_{nz} \ll S_{nz}$) as seen in the Figure 2.2. In other words, the low rank matrix is acting as the sparse matrix, and the sparse matrix is acting as the low rank matrix in terms of the number of zeroes it contains.

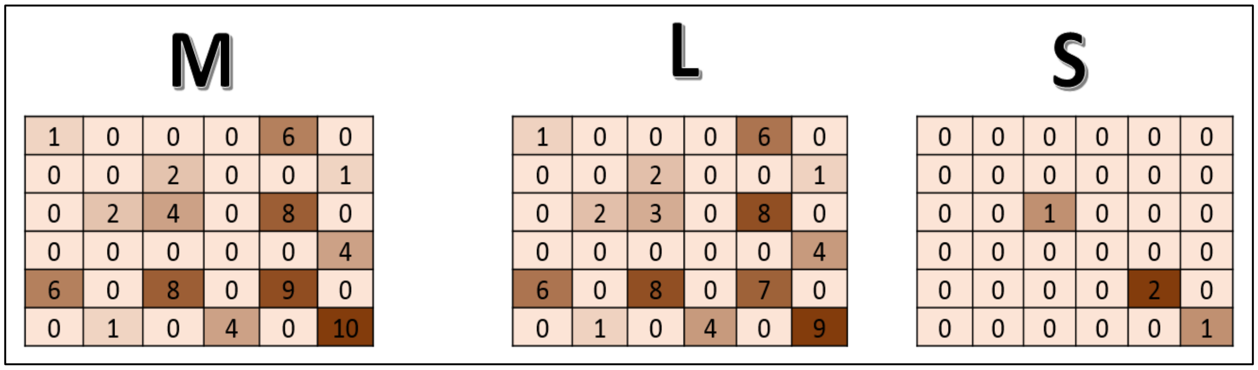
The above observation creates for us new opportunities to explore the controlling parameter λ, to make the sparse matrix much sparser and to optimize the rank of low rank matrix for making better predictions of the second order matrix, which is the scope of our thesis. As we understand the limitation of using eRPCA with default value λ_0 on such sparse second order matrices, we explore the possibilities of using eRPCA on such sparse second order matrix by varying the controlling parameter $\lambda = l \times \lambda_0$.

In the Figure 2.2 and 2.3, we illustrate that by using the eRPCA algorithm with increasing value of λ that the low rank matrix becomes denser and sparse matrix becomes sparser for every increased value of λ. We see that for a high value of λ, as depicted in Figure 2.3, the number of zeroes in this sparse matrix has decreased relative to the low rank matrix. In other words, the number of non-zeroes in L has increased relative to S ($i.e., L_{nz} \gg S_{nz}$). Also by increasing the value of λ_0 to λ, the rank of the low rank matrix increases from 0 to $rank(L) \leq 6$ and, surprisingly, this higher rank (but not too high) helps us to predict entries in M.



$$M = \begin{array}{|c|c|c|c|c|c|}\hline 1 & 0 & 0 & 0 & 6 & 0 \\ \hline 0 & 0 & 2 & 0 & 0 & 1 \\ \hline 0 & 2 & 4 & 0 & 8 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 4 \\ \hline 6 & 0 & 8 & 0 & 9 & 0 \\ \hline 0 & 1 & 0 & 4 & 0 & 10 \\ \hline \end{array} \quad L = \begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 2 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 3 \\ \hline \end{array} \quad S = \begin{array}{|c|c|c|c|c|c|}\hline 1 & 0 & 0 & 0 & 6 & 0 \\ \hline 0 & 0 & 2 & 0 & 0 & 1 \\ \hline 0 & 2 & 4 & 0 & 8 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 4 \\ \hline 6 & 0 & 8 & 0 & 7 & 0 \\ \hline 0 & 1 & 0 & 4 & 0 & 7 \\ \hline \end{array}$$

Figure 2.2: In this figure, we show a second order matrix M that is too sparse for the standard eRPCA theory to apply. The L and the S matrices show a typical decomposition that a classic eRPCA problem would give for the low rank and sparse matrices. In this case, the “sparse’ matrix S is relatively not sparse as compared to the low rank matrix L or the original matrix M. In fact, and low rank matrix L is highly sparse as see from this figure, which is not desirable from the prediction point of view. The color with different intensity represents the values inside the matrix.



$$M = \begin{array}{|c|c|c|c|c|c|}\hline 1 & 0 & 0 & 0 & 6 & 0 \\ \hline 0 & 0 & 2 & 0 & 0 & 1 \\ \hline 0 & 2 & 4 & 0 & 8 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 4 \\ \hline 6 & 0 & 8 & 0 & 9 & 0 \\ \hline 0 & 1 & 0 & 4 & 0 & 10 \\ \hline \end{array} \quad L = \begin{array}{|c|c|c|c|c|c|}\hline 1 & 0 & 0 & 0 & 6 & 0 \\ \hline 0 & 0 & 2 & 0 & 0 & 1 \\ \hline 0 & 2 & 3 & 0 & 8 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 4 \\ \hline 6 & 0 & 8 & 0 & 7 & 0 \\ \hline 0 & 1 & 0 & 4 & 0 & 9 \\ \hline \end{array} \quad S = \begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 2 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

Figure 2.3: In this figure, we show a second order matrix M that is too sparse for the standard eRPCA theory to apply, so we use eRPCA with high value of λ on M. The L and the S matrices show a decomposition that a classic eRPCA problem would give for the low rank and sparse matrices for high value of λ. In this case, the "sparse' matrix S is relatively more sparse as compared to the low rank matrix L or the original matrix M. In fact, and low rank matrix L is dense as compared to L in Figure 2.2. The color with different intensity represents the values inside the matrix.

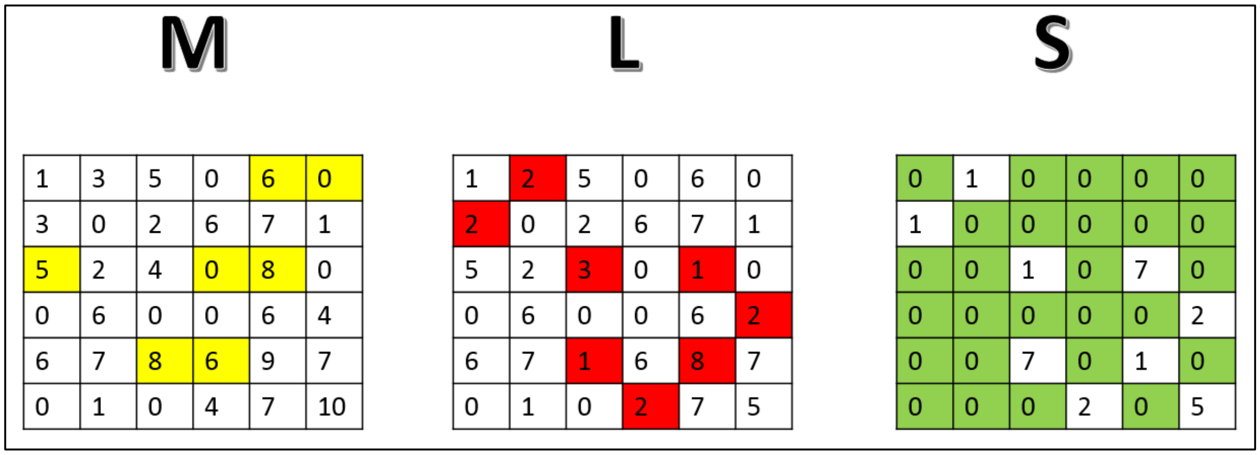
2.7 Various scenarios that exists in eRPCA

For real world data, when converted in to second order form, there could be many unknown entries in the matrix M, and one would like to predict the values of those unknown entries. Therefore, in such cases we observe all the entries of M (in the technical RPCA sense of every entry of Ω being 1).

However, in the eRPCA sense, we make $\epsilon_{ij} = vE_L$ for those entries we feel confident that we know and make $\epsilon_{ij} = vE_H$ for those entries we do not know and wish to predict. So, harkening back to the definition of Ω in Section 2.5, we write

$$\begin{aligned} u, v &\in P_{\Omega}, \\ u_p, v_p &\in vE_H, \end{aligned} \qquad 17)$$

to denote the entries we wish to predict.



1	3	5	0	6	0
3	0	2	6	7	1
5	2	4	0	8	0
0	6	0	0	6	4
6	7	8	6	9	7
0	1	0	4	7	10

1	2	5	0	6	0
2	0	2	6	7	1
5	2	3	0	1	0
0	6	0	0	6	2
6	7	1	6	8	7
0	1	0	2	7	5

0	1	0	0	0	0
1	0	0	0	0	0
0	0	1	0	7	0
0	0	0	0	0	2
0	0	7	0	1	0
0	0	0	2	0	5

Figure 2.4: This shows the decomposition of Second order matrix to low rank matrix (L) and sparse matrix(S), the yellow color in $'M'$ indicates the entries which we need to predict. The red color entries in L means we may not make good prediction relative to other and the entries in L whose corresponding entries are green color in S means we can make better prediction relative to the red color entries.

In the case of a second order matrix M with many unknown entries, we understand from the principles of eRPCA that not all the entries could be predicted well. The errors of each of the entries in the matrix could vary, and so there are entries which could have relatively high errors, and we call them $Type - HE$ errors, and those entries with low errors, which we call $Type - LE$ errors. *As per the principles of eRPCA,* $Type - HE$ *errors are generated by those entries whose corresponding entries in sparse matrix* S *are non-zeroes and have anomalies.* $Type - LE$ *errors are generated by those entries whose corresponding sparse entries are zeroes and have no anomalies*.

Accordingly after the eRPCA algorithm is complete we know which are the entries have $Type - HE$ errors and $Type - LE$ errors merely by examining S. Accordingly, we can write that

$$\begin{aligned} i, j \in S_{ij} = 0 &: Type - LE \\ i, j \in S_{ij} \neq 0 &: Type - HE \end{aligned} \qquad 18)$$

In the Figure 2.4, the red color in the low rank matrix is classifies as those entries which could cause $Type - HE$ errors, and the white color in the low rank matrix is classifies as those entries which could cause $Type - LE$ errors.

From the above ideas, we know how to make use of eRPCA for prediction, and we know the possibilities of different error points. Moreover, having understood the above two ideas, we focus on some of the practical aspects of using eRPCA from a business or an applied point of view. In a real situation, we may need to predict some specific entries, as predicting those entries may involve generating money to the business directly or indirectly, or could have much value from any problem perceptive. In the Figure 2.4, we see that yellow color entries (there are seven entries) represent those entries, which we want to predict for the business needs. Among those seven entries, we can say with a tolerance limit that five entries could fall in $Type - LE$ and two entries would fall under $Type - HE$ error.

In any given situation, from a business prediction perspective, if the business wants a few of the entries that come with $Type - LE$ errors to be predicted more precisely, then we may compute those entries with more accuracy but at the cost of more data (i.e., higher rank of low rank matrix by increasing the value of λ). In addition, we can say that out of M_p entries, how many of the entries may fall under $Type - HE$ or $Type - LE$ errors.

Chapter 3

Description of Datasets

3.1 Overview

In this chapter, we explain the structure and preprocessing of the Amazon product community data set [2] and the SKAION [23] network dataset.

In the SKAION data set, the raw data represents the data packets exchanged between sensor nodes (source and destination IP address and ports). In other words, the raw data has rows labeled by IP addresses and ports, and the columns represent data packets received or sent by the respective sensor node. We have analyzed various ways of converting the first order matrix of raw packet data to a second order matrix for better representation of the similarities between the sensor nodes.

In the Amazon product community data set, the original raw dataset has two representations, one with the connections between individual products and the other listing the communities of Amazon products. We call this raw dataset our first order matrix. We analyze various ways to form the second order matrix of Amazon communities, where the values represent the strength of connection between the communities. We have discussed the formation of second order "similarity" matrices in four different ways by using different definitions of strength of connection between the communities, and have chosen the best among them based upon their effectiveness for prediction.

3.2 SKAION Network data

The raw data or first order matrix (FOM) of the SKAION Network data consists of rows of Source and Destination IP address and their ports, and the column represents Internet packets (P1, P2 and so on until P10000 for this particular dataset). Each of the rows represents all of the Internet packet data for the given sensor node. We convert the first order matrix into a second order matrix by taking the dot product of the normalized first order matrix and its transpose, which gives the covariance matrix between the sensor nodes.

$$M = FOM \times FOM^T, \qquad 19)$$

where,

FOM is of the order of 759×10000,
FOM^T is of the order of $10{,}000 \times 759$.

Sensor Nodes	**P1**	**P2**	**P3**	**P4**	**P5**	**…**	**P10000**
12.100.178.10	1	0	1	0	1	0	….10000
12.100.178.11	0	1	0	1	0	0	….10000
….	0	1	1	0	0	1	….10000
100	1	0	0	0	1	0	….10000
200	0	0	0	0	1	1	….10000
…..							

Table 3.1: *The table shows a small sample of first order matrix of SKAION data, with rows representing the senor nodes of source and destination IP address and ports, and columns represent 10000 internet packets of the sensor nodes.*

The raw data from SKAION Network that comprises the first order matrix has binary values of 0's and 1's, with 10,000 packets of data, a sample of which is shown in the Table 3.1. In effect, each IP Address and each port is thought of as a sensor. Therefore, for example a one in the row labelled 12.100.178.10 means the packet originated at this IP Address. Similarly, a one in the row labelled 100 means the packet originated from port 100. First 225 rows of sensor nodes represent Source IP address, from 226 row to 425 row of sensor nodes represent destination IP address, from 426 row to 610 row of sensor nodes represent source ports and the remaining from 611 row to 759 row represent destination ports.

The first order matrix is converted to a second order matrix by taking the dot product of the FOM and the transpose of the FOM. Then, the size of second order matrix becomes 759×759 and that is what is used for analysis. In the SKAION data, as the first order matrix is mainly categorical (binary) in nature, we normalize the first order matrix by rows (i.e., on each sensor node). We then take the dot product on the normalized data of FOM and its transpose to get the second order matrix (i.e., the data is normalized so the diagonal of the second order matrix is one). In Figure 3.1, the first order matrix of normalized SKAION data is shown, the x-axis represents the sensor nodes and y-axis represents the internet packets. The colors in the plot represent the normalized values inside the first order matrix, the red color indicates the lowest value -1 and black color indicates the highest value$+1$, and the contrast from red to black is the values in the range from -1 to $+1$. In the plot, we see four patterned roughly slanted black lines, the first pattered black line from left represent the source IP address (first 225 sensor nodes) which is transmitting the data and the second black line from left represent the destination IP address (226 to 425 sensor nodes) which is receiving the data. Similarly, the third black line (426 to 610 sensor nodes) and fourth black line (611 to 759 sensor nodes) represent the source and destination ports, the former one is transmitting and later one is receiving the data.

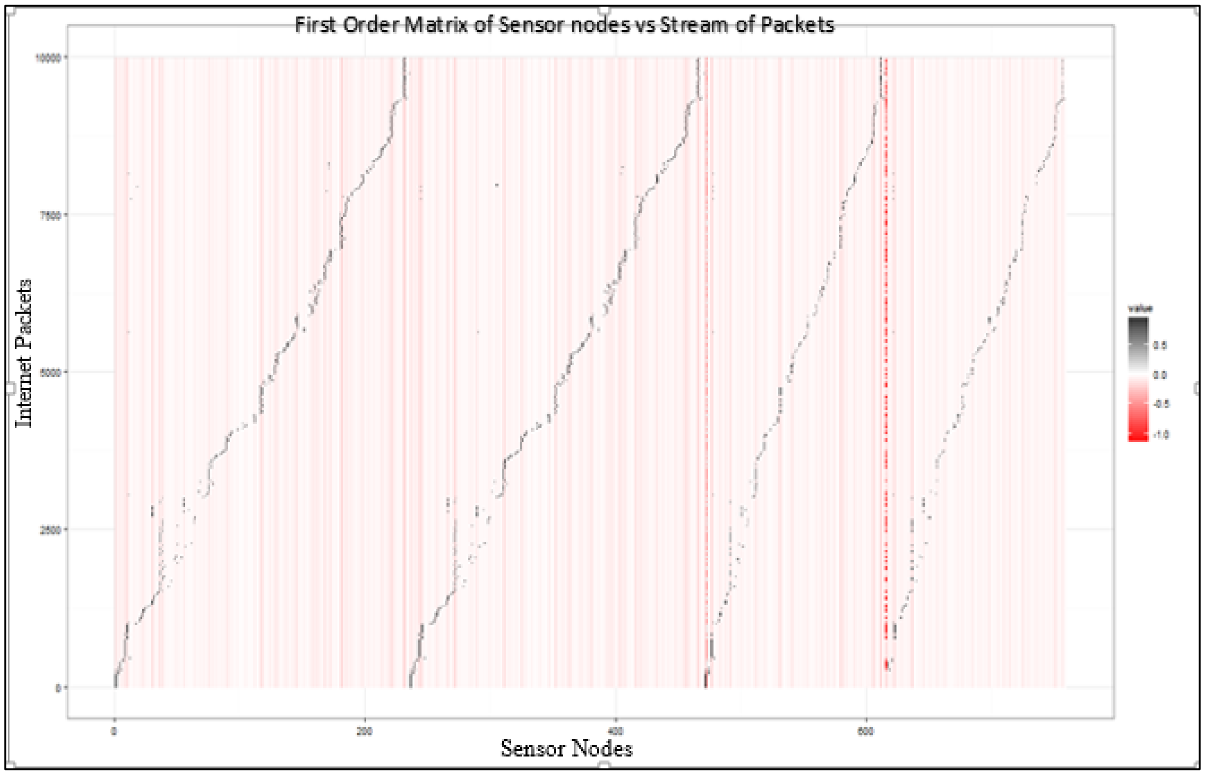


Figure 3.1: In the above graph, x-axis represents sensor nodes and y-axis represents internet packets (which is normalized across each sensor nodes), there are 10,000 packets and 759 sensor nodes, the values in the color bar shows, with red representing the lowest value of -1 and black representing the highest value of +1.

3.2.1 First Order Matrix from SKAION data

Based upon our preliminary analysis we felt that modifying the raw data of SKAION network data based on the needs of different problems would improve our analysis and therefore we augment the first order matrix using three different structures. The three structure differ by modifying the IP address of both source and destination, Table 3.2 shows the original form of IP address and three representations of those IP Addresses.

IP Address	IP Address Distinct	IP Address SUBNET1	IP Address SUBNET2
192.168.5.130	192.168.5.130	192.168.5.x	192.168.x.x
192.168.5.130	192.168.5.131	192.168.6.x	
192.168.5.131	192.168.5.132		
192.168.5.132	192.168.6.134		
192.168.5.132			
192.168.6.134			

Table 3.2: This table shows the example of converting the IP Address of sensor nodes to unique value of IP Address to SUBNET-1 and SUBNET-2.

In building the first order matrix, we take unique values of all the IP address to get the column *IP Address Distinct*. In building the second and third structure of IP Addresses, we start with the column *IP Address Distinct*, and take SUBNET-1 and SUBNET-2 of *IP Address Distinct* to create two additional columns *IP Address SUBNET1* and *IP Address SUBNET2*. For instance in the table 3.2, If we take unique values of the above IP address, we get *192.168.5.130, 192.168.5.131, 192.168.5.132, 192.168.5.132, 192.168.6.134* which is represented in column *IP Address Distinct.* Similarly, we take the unique values of SUBNET-1 *(192.168.5.x, 192.168.6.x, 192.168.5.x, 192.168.6.x)* to get rows *192.168.5.X, 192.168.6.X*, and we take the unique value of SUBNET-2 *(192.168.x.x, 192.168.x.x)* to get row 192.168.X.X.

We construct three forms of the second order matrix of SKAION data from three different structures of the first order matrix. These different forms of second order matrix can be used for various purpose based on the area of interest in cyber network problems. If the distributed attack is made by varying only SUBNET values then using distinct IP Address will unnecessarily increase the computation time, so choosing SUBNET-1 and SUBNET-2 may give better results and therefore, we experiment with the three different structure of formation of the second order matrix.

We analyze the singular values of the three different structures of the second order matrix of SKAION using the SVD. In order to do this experiment, we have taken 1000 rows (sensor nodes) for *IP Address Distinct* from which we get 900 rows for *IP Address SUBNET1* and 850 rows for *IP Address SUBNET2*. Figure 3.2 and 3.3 shows the singular value plot of SUBNET-1 and SUBNET-2, the singular values decreases exponentially and has approximately 318 non-zero singular values, which implies that we can predict (759-318 = 441) sensor nodes from 318 sensor nodes. The singular values in the Figure 3.2 and 3.3 are very similar to Figure 4.1, where 320 sensor nodes have non-zero singular values, which shows that although we are taking the IP Address structures SUBNET1 and SUBNET2 that reduces the number of rows in the matrix and therefore the amount computation. However, we get almost same number of nonzero singular values around 320. We see from the graph of the SVD that it does not make much of a difference by taking unique values of SUBNET and we are not concluding that taking SUBNET values is helpful in this case. However, it may be that for different problems taking SUBNETs would be helpful. In this situation, we are taking the *IP Address Distinct* because we do not want to lose information as the number of rows decreases by taking the SUBNETs.

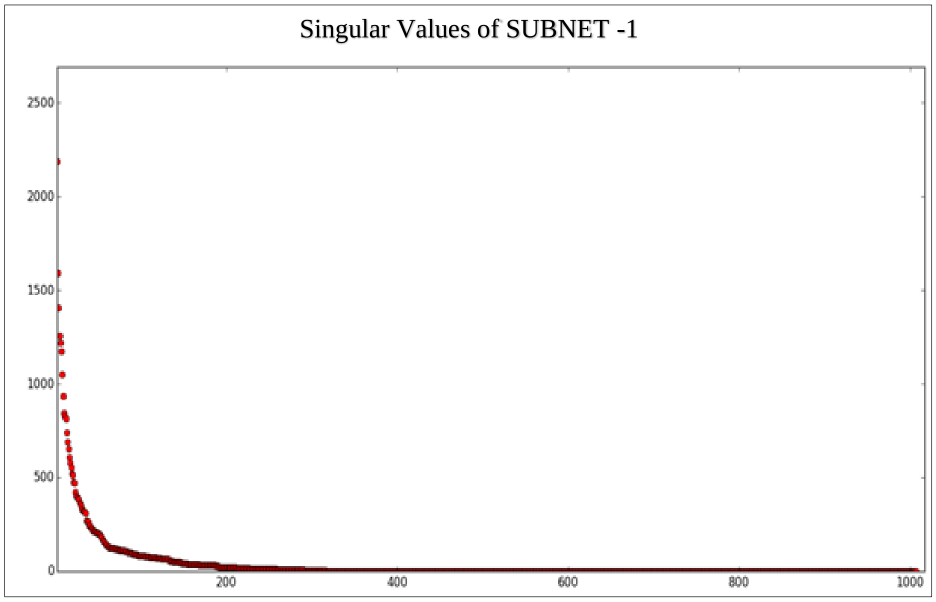


Figure 3.2: In the above graph, the y-axis represents the size of singular values, the x-axis represents the index of the singular values ordered from largest to smallest, and this is from the second order matrix of SKAION data using SUBNET-1

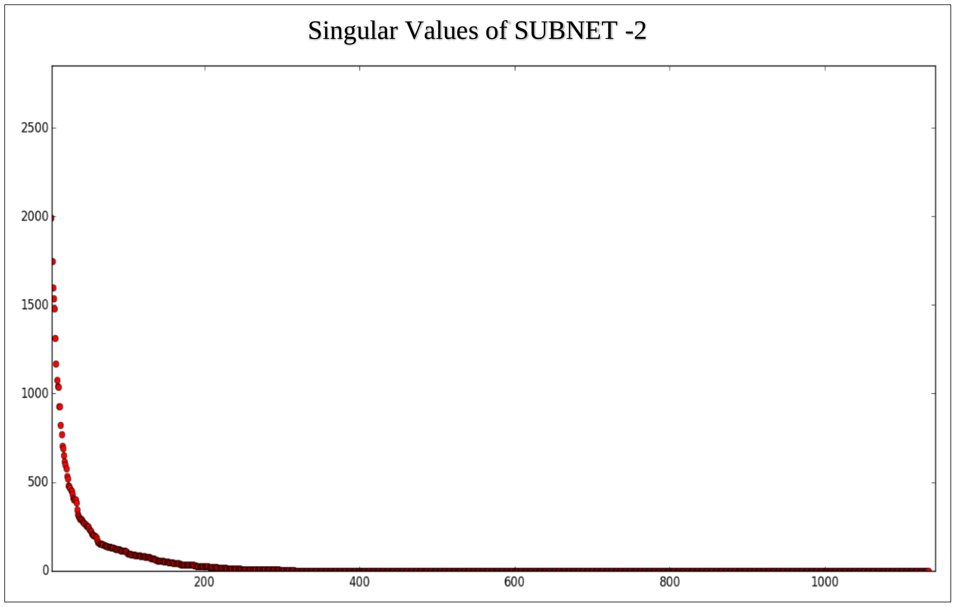


Figure 3.3: In the above graph, the y-axis represents the size of singular values, the x-axis represents the index of the singular values ordered from largest to smallest, and this is from the second order matrix of SKAION data using SUBNET-1

3.3 Amazon Communities Network Data

The second type of network we consider is the Amazon product co-purchasing network. The Amazon Communities Network data set is downloaded from Stanford Large Network Dataset Collection [11]. It is based on the *Customers Who Bought This Item Also Bought* feature of the Amazon website. The nodes of the network represent products and edges link commonly co-purchased products. If a product *i* is frequently co-purchased with product *j*, the graph contains an undirected edge from *i* to *j*. In addition, Amazon defines product categories and calls these categories "ground-truth" communities [12]. Each product (*i.e.*, node) belongs to one or more hierarchically organized product categories and products from the same category define a group that Amazon, and we, view as a ground-truth community [12]. In this case, nodes that belong to a common ground-truth community share a common function or purpose.

The Amazon product co-purchasing network has FromNodeId to ToNodeId connection data, this is an undirected graph where FromNodeId represents the origin node (which represents one Amazon products) and ToNodeId represents the destination node (which represents another Amazon product). A small sample of this data is shown in table 3.3. For instance, in the first row, we can say that product *1* is frequently co-purchased with product *88160.*

FromNodeId	**ToNodeId**
1	88160
1	118052
1	447165
1	500600
2	27133
2	62291
4	16050

Table 3.3: In this table, the two columns represent the nodes that are Amazon products, and the table as a whole represents the relationship between two nodes, which is the Amazon product co-purchasing network

In the Amazon ground truth communities data, a sample of which is shown in the Table 3.4, each row in the community has a different number of nodes depending on the size of the ground truth community. In the table below C1, C2….etc. represent the communities and the values inside represents the NodeId which are the products in that communities. For instance, the community C1 may be bread and the three nodes in C1 may represent three kinds of bread (maybe white, multigrain and wheat). Unfortunately, we do not know exactly the name of the communities nor the name of its products.

C1	164985	225214	232761				
C2	167906	288975	421279	442612	451312		
C3	69880	231953	518997	523128			
C4	135499	160294	304770				
C5	112813	112814	112821	112823			
C6	112813	112814	112821	112823			
C7	199372	399560	447268	471226	522928	439998	280502
C8	179001	391697	412528				
C9	21166	207188	405926	531532	540207		
C10	118948	191846	209822	455700	482725		
C11	55727	78359					
C12	246337	301834	389644				
C13	99505	126694	133115	264885			
C14	75000	156489	207684	278335	533982		
C15	112813	112814	112821	112823	213617		

Table 3.4: In this table of the Amazon ground truth communities data, the first column or the row names represent the communities and the columns after first column represent the nodes for each of the communities in their respective row (or communities) inside each of the communities.

In our analysis, we combined these two datasets to form a second order matrix that gives us the strength between the communities, the rows and columns represents the communities, and the values in the second order matrix gives the strength of connection between the communities. The strength of connection between the communities can be expressed in many ways and we present four different ways and chose the best metric among the four that would be most suitable for our analysis. For instance in the Table 3.5, the strength of connection between communities C1 and C2 is 0 which means that none of the products in C1 are co-purchased with C2. The strength of communication between C1 and C100 is 1 which means that “some” of the products in C1 are co-purchased with the products in C100 (the precise definition of what “some” means will be the focus of the rest of this Section). Our methodology of second order matrix formation enforces that the matrix be symmetric. However, a different sort of data, one could definitely imagine a non-symmetric second order matrix (say, if you knew which product was purchased first). Note, this is contrast to the covariance based SKAION data second order matrix by definition is symmetric.

Communities	**C1**	**C2**	**C3**	**…**	**C100**
C1	2.2	0	0	0	1
C2	0	4.5	0	5.5	0
C3	0	0	6.5	0	0
..	0	5.5	0	7.8	0
C100	1	0	0	8	7.6

Table 3.5: In this table, it shows the sample of second order matrix of Amazon data, formed from Amazon communities' and product data, the values inside the above matrix represents the strength of connection between two communities.

Dataset statistics	
Nodes	334863
Edges	925872
Nodes in largest WCC	334863 (1.000)
Edges in largest WCC	925872 (1.000)
Nodes in largest SCC	334863 (1.000)
Edges in largest SCC	925872 (1.000)
Average clustering coefficient	0.3967
Number of triangles	667129
Fraction of closed triangles	0.07925
Diameter (longest shortest path)	44
90-percentile effective diameter	15

Table 3.6: In this table, we show all of the statistics mentioned in the original data description of the Amazon dataset, which is extracted from Stanford Large Network Dataset

We regard each connected nodes in a product category as a separate ground-truth community. We remove the ground-truth communities, which have less than three nodes. We also provide the top 5,000 communities with highest quality, which are described in [2]. As for the network, we analyze the largest connected component. The metrics in Table 3.6 represents various metrics of nodes and edges of the original data from Stanford Large Network Dataset [11]

3.3.1 Metrics of Communities strength

In order to define precisely, the strength of connection between the communities, we introduce four different methods of representing the community strength of Amazon data and we call them as Community Density-1 (CD-1), Community Density-2 (CD-2), Community Density-3 (CD-3) and

Community Density-4 (CD-4). We referred to the literature in [12] for CD-1, CD-2, CD-3, and we came up with CD-4 from the inspiration of previous three metrics.

Beginning with CD-1, if a community C1 is connected to C2 then the value in the second order matrix between row C1 and column C2 would be defined as the number of edges between these two communities. In Figure 3.6 and equation (20), there are two communities represented by the left and right circles, with dots represents the number of nodes in each of the communities. The edges represent the nodes from each of the communities are connected in the co-purchasing network.

$$CD1 = No\ of\ Edges\ connecting\ between\ the\ communities\ (NEBN) \qquad 20)$$

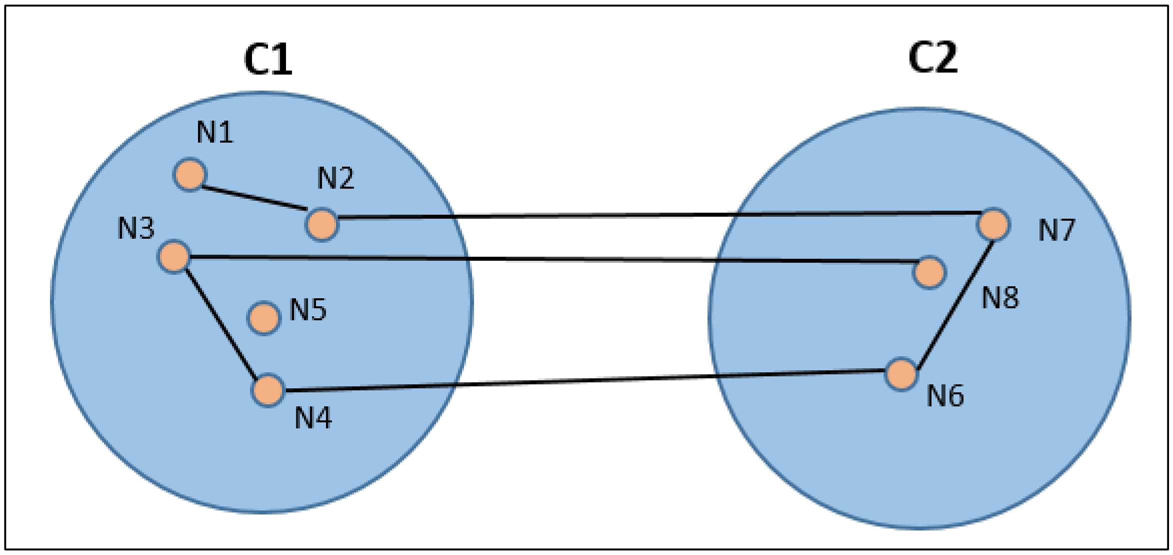


Figure 3.6: In the Figure there are two communities C1 and C2, the nodes are represented by N1, N2….N8. The lines represent the edges connecting between two nodes. C1 has five nodes and C2 has three nodes. In this example CD1 is 3.

The definition of community strength CD1 (for instance in the Figure 3.6 CD1 is 3) does not take into account the number of nodes in each of the communities. This definition is adequate for all communities that have same number of nodes but in our dataset the communities have different number of nodes. For instance, consider two pairs of communities. In the first pair, each community has 4 nodes and there are three edges between the communities. In the second pair each communities has 100 nodes and again there are three edges between them. Clearly, the first pair of communities have a stronger connection then the second pair but CD1 does not recognizes this fact.

So, as it is clear that the number of nodes in the communities should influence the strength of connection between communities, [12] formulates a new metric that would take the number of nodes in each community along with the number of edges. Hence, we call this metric Community

Density – II (CD2) and define it as the number of edges between the communities divided by the product of the number of nodes in the two communities.

$$CD2 = \frac{NEBN}{NC1 * NC2} \tag{21}$$

Thirdly, as we observe from the literature [12], the strength of connection between communities C1 and C2 should be influenced by the strength of connection C1 with other communities apart from C2, so equation (22) includes the relative strength with other communities in the metric. For instance, in the Figure 3.5, we see that C1 and C2 are connected with 3 edges, C1 and C3 are connected with 2 edges, so in total the edges connected from C1 to other communities is (3+2=5). Accordingly, it is desirable to consider these other edges in the definition of the community strength between C1 and C2 or C1 and C3. Therefore, Community Density –III is defined as the edges connecting between C1 and C2 (NEBN) divided by the edges outgoing from C1 (NOEC1).

$$CD3 = \frac{NEBN}{NOEC1} \tag{22}$$

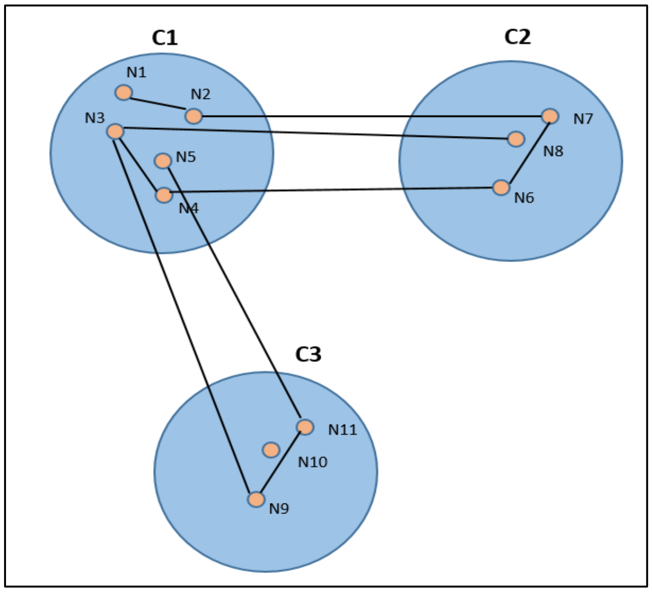


Figure 3.7: in the Figure there are three communities C1, C2 and C3, the nodes are represented by N1, N2….N11. The lines represent the edges connecting between two nodes, essentially when viewed from the communities. C1 has five nodes, C2 and C3 has three nodes.

Finally, inspired by the three different metrics from the literature [12], we wanted to alter the second and third metric to form a new metric that would consider giving importance to the number of nodes in these two communities, and the relative strength between communities C1 with C2 relative to the strength of connection between C1 and other communities. Therefore, the new metric we define as Community Density - IV, which counts the edges connecting C1 and C2 divided by the product of edges outgoing from C1, and the number of nodes in the two communities.

$$CD4 = \frac{NEBN}{NOEC1 * NC1 * NC2} \qquad 23)$$

In the next Section, we provide results that compare and contrast these various definitions.

Chapter 4

Experiments & Results

4.1 Overview of Experiments

In this Chapter, we present the experiments conducted on the SKAION network data and the Amazon communities network data. In Section 4.2 and 4.3, we take the SKAION network data as a reference for analyzing the typical kind of second order matrix that appears in most of the real world problems. In the SKAION network data, we analyze the singular values obtained from the SVD, RPCA and eRPCA techniques. In addition, we understand the behavior of the low rank matrix and the sparse matrix arising from SKIAON's second order matrix. The other analysis involves evaluating various ways of using the first order form of the SKAION network data and finding the anomalies in the low rank and the sparse matrix.

In Section 4.4 and 4.5, we evaluate the four methods of forming a second order matrix from the Amazon communities data. The three main parameters of eRPCA that we observe are *Prediction Error*, $\boldsymbol{\lambda}$ and *number of non-zeroes*. We have mainly divided the prediction experiments into Section 4.7 and 4.8 based on the target matrix used for prediction. In Section 4.7, we do the prediction of one low rank matrix from another low rank matrix by varying λ (i.e. between L_0 and L_p). In Section 4.8, we use the low rank matrix to predict the second order matrix by varying λ $(\text{i.e. between } M - L_0 \text{ and } M - L_p)$. In Section 4.9, we find the empirical relation between λ and the non-zero entry values in a sparse matrix. In Section 4.10, we find the distribution of error entries in prediction of $M - L_p$, and find the best technique to capture the errors.

4.2 Singular values of SKAION data

The second order matrix of SKAION data has the size of 759×759 and is analyzed using the SVD technique. In Figure 4.1, the y-axis represents the size of the singular values and x-axis represents the index of the singular value order from largest to smallest. We observe that the singular values decreases exponentially and it takes approximately 0 at the sensor node of 318, which implies that we can predict the entries of 441 (759-318 = 441) sensor nodes using the entries of only 318 sensor nodes. This happens because of the low rank property which implies that we may not need the entries of all of the 759 (high dimensional) sensor nodes, we may just need the entries of 318 (low dimensional) sensor nodes to predict the entries of remaining 441 sensor nodes. The question here is, can we predict more nodes using a lesser number of nodes? Yes, using techniques like RPCA and eRPCA we can improve our ability to predict many sensor nodes from just a few sensor nodes that we need to measure.

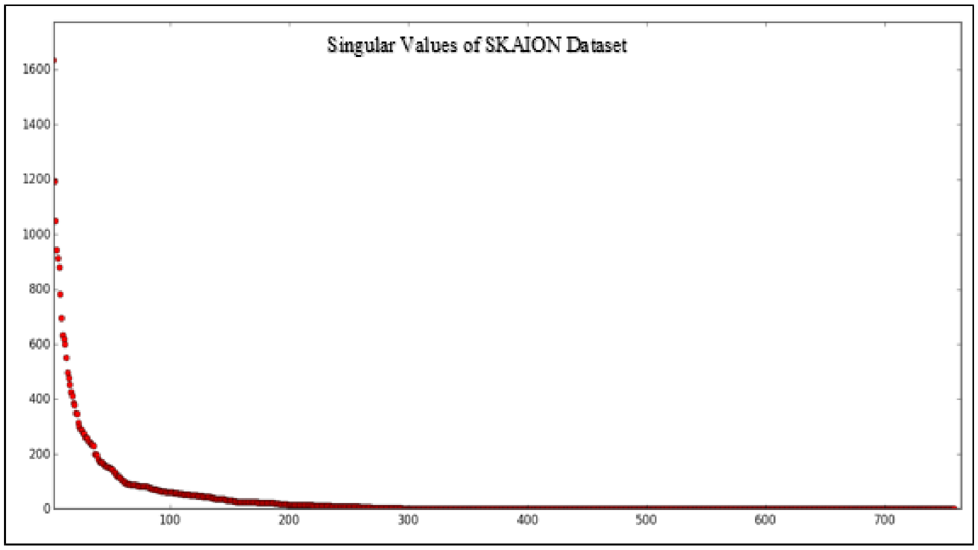


Figure 4.1: In this graph, the y-axis represents the size of singular values, the x-axis represents the index of the singular value order from largest to smallest, the singular values decrease and becomes almost zero and constant after 318 sensor nodes. This implies that given 318 sensor nodes we can predict rest of the other sensor nodes.

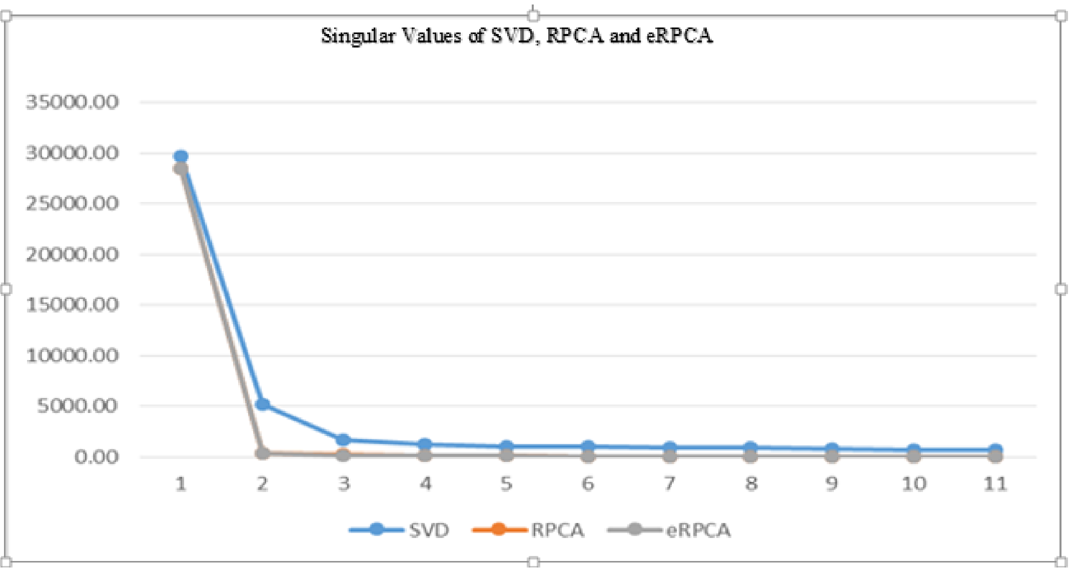


Figure 4.2: In this graph, the y-axis represents the size of singular values, and the x-axis represents the index of the singular value order from smallest to largest. It shows that the singular values of three techniques: SVD, RPCA and eRPCA. We see that SVD produces larger number of non-zero singular values than RPCA and eRPCA. These singular values of RPCA and eRPCA are similar and that is why the lines are coincided.

We use three dimensionality reduction techniques, SVD, RPCA and our own eRPCA, to compare the singular values of the second order matrix of SKAION network data. In Figure 4.2, the y-axis represents the size of the singular values and x-axis represents the index of the singular value order from largest to smallest. We observe that the singular values decrease for all the three techniques. We find that SVD produces larger number of non-zero singular values compared to Robust and point wise error techniques like RPCA and eRPCA. This implies that SVD requires many sensor nodes to predict all of the rest of the entries in other sensor nodes. The small number of non-zero singular values for RPCA and eRPCA imply that these algorithms can predict almost all of the sensor nodes from a tiny fraction of other sensor nodes. The essential idea is that a few sensor nodes are hard to predict and account for the majority of non-zero singular values. However, most of the sensor nodes can be predicted using just a few non-zero singular values. Partitioning the sensor nodes into those that are easy to predict and those that are hard to predict is the essence of RPCA and eRPCA. Therefore, we use RPCA techniques for further analysis in our thesis.

4.3 Understanding of Low Rank and Sparse Matrix of eRPCA

We use eRPCA on SKAION's second order matrix to understand the decomposition of M into a low rank and a sparse matrix, and take this decomposition as typical for eRPCA. As we understand the behavior of the low rank and the sparse matrix using SKAION data, we will be able to foreshadow using eRPCA on Amazon communities data.

The principles of eRPCA algorithm give us the freedom to observe all the entries of the data (or part of it) based on our problem requirements [1]. We observe all the entries except the diagonal entries, following [1], as the diagonal entries represent the auto-correlation of the variables, and our main objective is to find the relationship between different variables. In the eRPCA algorithm, one sets $\epsilon_{ij} = vE_L$ for all the entries we know and sets $\epsilon_{ij} = vE_H$ for all entries we wish to predict. In effect, setting $\epsilon_{ij} = vE_L$ gives the algorithm no freedom in optimizing that entry, since the user measured that entry. On the other hand, setting $\epsilon_{ij} = vE_H$ gives the algorithm the freedom to optimize that entry, and the optimal entry is the predicted value.

There are many input parameters to consider in the eRPCA algorithm, and the following are the parameters that are important for our analysis. The parameters u, v represent the indices that are observed from the matrix M (In this thesis, we actually observe all of the entries in M and we use $\epsilon_{ij} = vE_L$to label entries we wish to predict). vE is point wise error bounds which takes on values vE_L or vE_H . ρ_{max} denotes maximum rank of M to consider for completion. λ is the value of the coupling constant between L and S (and λ is a key focus of our work) . The output parameters from the decomposition using eRPCA algorithm are U, E, VT and S. The U, E and VT components represent the SVD components of the matrix L. The U denotes the left singular vectors of the matrix L. The E denotes the diagonal matrix of singular values of L. The VT denotes the right singular vectors of the matrix L. The S denotes the sparse matrix. The output components are used to construct the low rank matrix by multiplying as $L = U * Diagonal\ (E)\ * VT$. The low rank matrix L and the sparse matrix S have the same dimensions.

The key observation of this thesis is that the values of M whose corresponding values of S are zero can be predicted from the low rank matrix L with small error. These are precisely the entries we called $Type - LE$ in Chapter 2! Similarly, the values of M whose corresponding values of S are non-zero can be predicted from the low rank matrix L with large error. These are precisely the entries we called $Type - HE$ in Chapter 2. Using PCA, differentiating $Type - HE$ and $Type - LE$ is extremely difficult. However, using our techniques such differentiation is quite easy.

In the Figure 4.3, the plot on the left side is the low rank matrix and the plot on the right side is the sparse matrix. As we can see from the graph, the low rank matrix L represents the classical L which is much lower rank than M, where the striped lines in L are normal for a very low rank matrix. We also observe from the graph that most of the values in low rank matrix are not zero i.e., only a few values are zero.

The right graph in the Figure 4.3 represents a sparse matrix, where there are many zero entries and there are a few non-zero entries. The non-zeros entries are called anomalies as per the principles of eRPCA [1], and the corresponding entries in the low rank matrix with these anomalies comes are of $Type - HE$. We can predict with high accuracy (low error) of all the entries of the low rank matrix whose corresponding sparse matrix entries are zero (i.e., green in color), and these entries are of $Type - LE$.

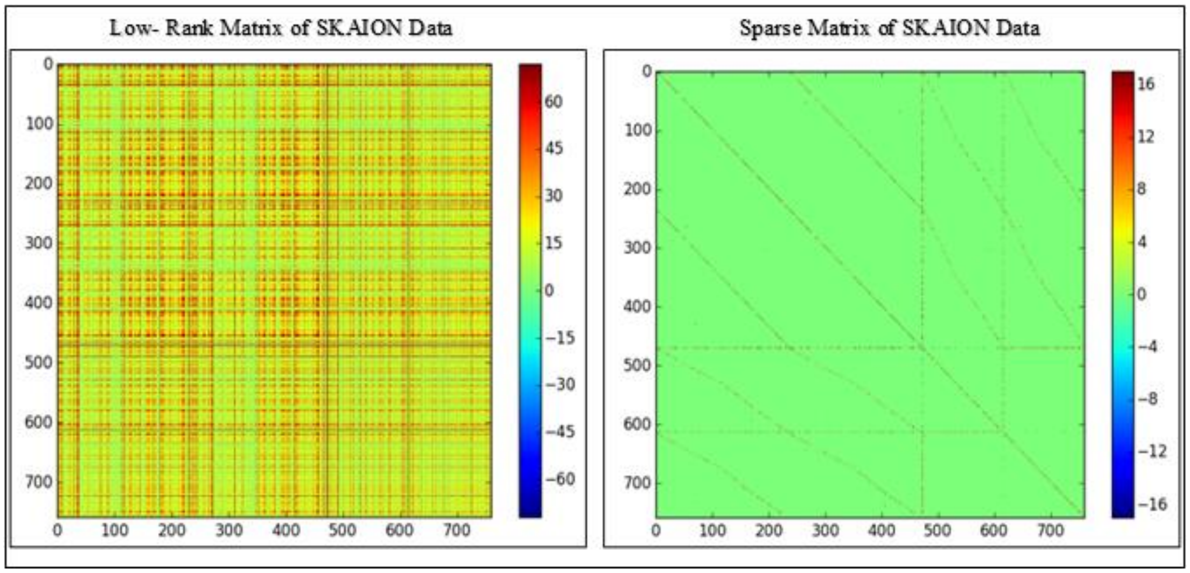


Figure 4.3: Low Rank Matrix and Sparse Matrix of SKAION Dataset has 759 sensor nodes, the color code bar represents the values in the matrix. The entries of L whose corresponding entries of S are zero come with $Type - LE$ errors in prediction, the entries of L whose corresponding entries of S are non-zero come with $Type - HE$ errors in prediction. This is a real example of the behavior we described in Figure 2.4.

In the typical settings of analyzing the second order matrix (M) using eRPCA, the number of zeroes and non-zeroes in the low rank matrix and the sparse matrix are given by the equations below. The number of zeroes in the sparse matrix is classically more than the number of zeroes than the low rank matrix.

$$S_z \gg L_z \text{ and } S_{nz} \ll L_{nz}, \qquad 24)$$

Where,

S_z: The number of zeroes in Sparse matrix

L_z: The number of zeroes in Low Rank matrix

S_{nz}: The number of non-zeroes in sparse matrix

L_{nz}: The number of non-zeroes in Low Rank matrix

The Figure 4.4 represents the singular values of the L produced by eRPCA, the y-axis represents the size of singular values, and the x-axis represents the index of the singular value order from largest to smallest. The singular values decreases and becomes almost zero and constant after 1 sensor nodes. The singular values imply the number of variables that is required to predict other variables. Having only, a single non-zero singular value implies we require just one variable to predict rest of the other 758 variables in the matrix of 759*759.

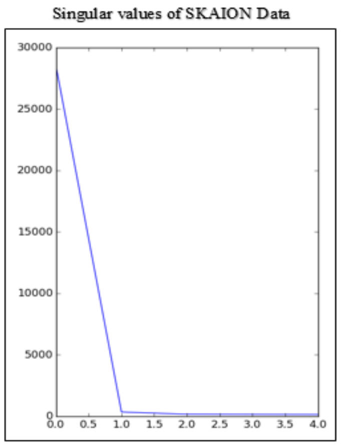


Figure 4.4: In this Figure the y-axis represents the size of singular values, the x-axis represents the index of the singular value order from largest to smallest, the singular values decreases and becomes almost zero and constant after 1 sensor node. A single non-zero singular value implies we require just one variable to predict rest of the other 758 variables.

The minimum, maximum, mean(μ) and standard deviation(σ) of the entries in the sparse matrix are -1987.34, 579.47, 0.16, and 0.37 respectively. Similarly, the minimum, maximum, mean and standard deviation of the entries in the low rank matrix are 0.98, 3186.08, 22.62, and 29.95 respectively. The above graph of low rank and sparse matrix is extracted from the range of $(\mu + 2\,\sigma\,,\ \mu - 2\,\sigma)$ the values beyond this range can be considered as outliers and anomalies. These anomalies just act as an alert, and there could be multiple reasons for this anomalous behavior. This information helps us to mitigate the risk of being attacked. The anomalies act as an alarm to the user to go back and validate the reason on sensor nodes transmission of such data.

4.4 Amazon Communities dataset

In this Section, we present the experiments conducted on singular values of second order matrices using the community definitions (CD) for the Amazon communities data mentioned in Chapter 3. In Section 4.4.1 and 4.4.2., we analyze the singular values using the SVD on the four different community definitions and choose the best CD. We compare the singular values using the best CD between SVD, RPCA and eRPCA techniques.

4.4.1 Singular values of Communities Strength

In Section 3.3.1, we have defined four metrics to construct the second order matrix from first order matrix of Amazon communities product dataset. However, theoretically we believe that CD-II was better metric to consider than other metrics, as CD-II includes not only the edges but also the number of nodes in each of the communities. However, we wanted to understand the practical significance of each of the three metrics, and we use the SVD technique to practically understand the difference among the metrics.

We analyze the second order matrix of Amazon communities data of size C_S = 1000. Figure 4.5 represents the singular values of the second order matrix using CD-I metric of size C_S = 1000. The x-axis represents the index of the singular values ordered from largest to smallest, and y-axis represents the size of the singular values of the matrix. We observe the exponential decrease of singular values, which is a good indication of a decreasing smooth trend, which captures each of the singular values correctly. The first non-zero singular value is 500, which implies that we require only 500 variables to predict rest of the other variables in the second order matrix of CD-I, but as the CD-I doesn't consider using nodes in the communities, so we would be skeptical of using the CD-I metric.

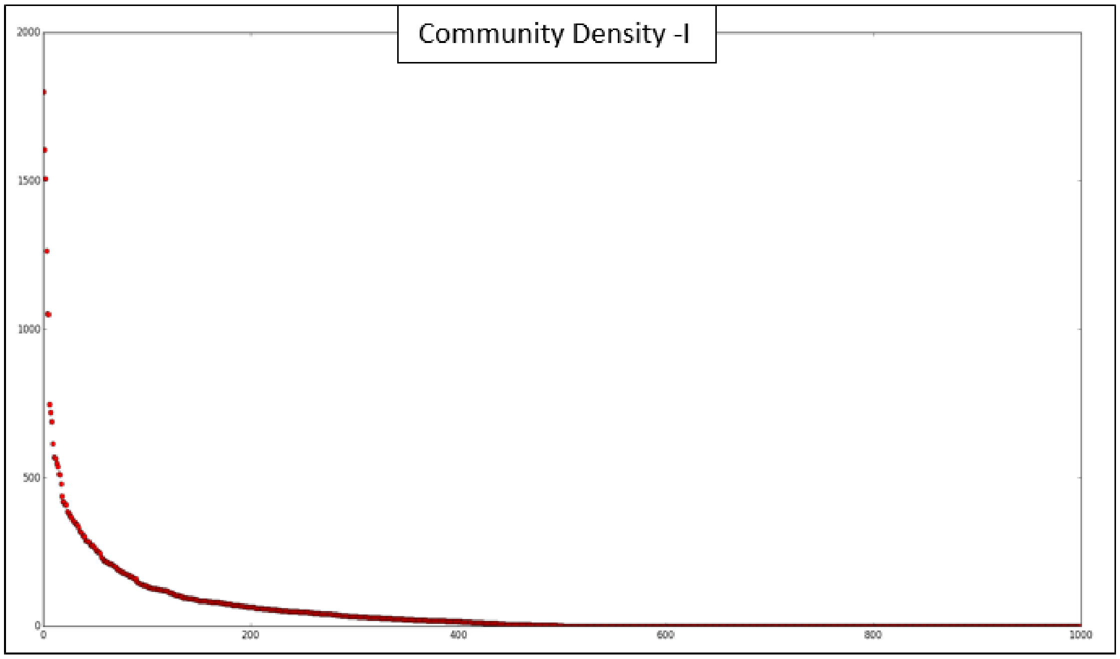


Figure 4.5: In this Figure, the y-axis represents the size of singular values of the matrix L, which is constructed by the CD-I, and the x-axis represents the index of the singular values ordered from largest to smallest. The singular values decrease and becomes almost zero and constant after 500 sensor nodes, but as the CD-I doesn't consider using nodes in the communities, so we would be skeptical of using the CD-I metric.

Figure 4.6 represents the singular values of the second order matrix using CD-II metric of size C_S = 1000. The x-axis represents the index of the singular values ordered from largest to smallest, and y-axis represents the size of the singular values of the matrix. We observe the exponential decrease of singular values, which is a good indication of a decreasing smooth trend, which captures each of the singular values correctly. The first non-zero singular value is 500, which implies that we require only 500 variables to predict rest of the other variables in the second order matrix of CD-II.

Figure 4.7 shows the singular values of the second order matrix that is formed by CD-III. The x-axis represents the index of the singular values ordered from largest to smallest, and y-axis represents the size of singular values of the matrix. We see that there is no consistent exponential decrease of singular values, and most importantly, the singular values remain constant until 500 and suddenly drops almost to zero after 500. This is anomalies graph for a graph of SVD, which is highly insensitive until singular value 500. In conclusion, among all these three metric, the CD-II has an exponential decrease in singular values and cuts off to almost zero at 500, which shows that we can perform dimensionality reduction techniques like SVD, RPCA and eRPCA better using CD-II.

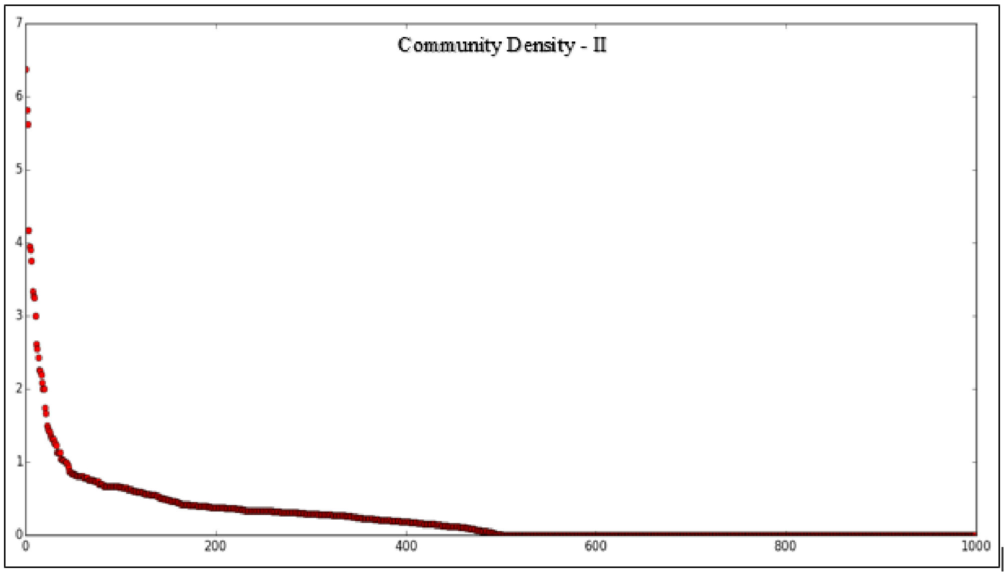


Figure 4.6: In this Figure, the y-axis represents the size of singular values of the matrix L whose corresponding M is constructed by the CD-II, and the x-axis represents the index of the singular values ordered from largest to smallest. The singular values decrease and becomes almost zero and constant after 500 sensor nodes, and we observe the exponential decrease of singular values, which is a good indication of a decreasing smooth trend, which captures each of the singular values correctly.

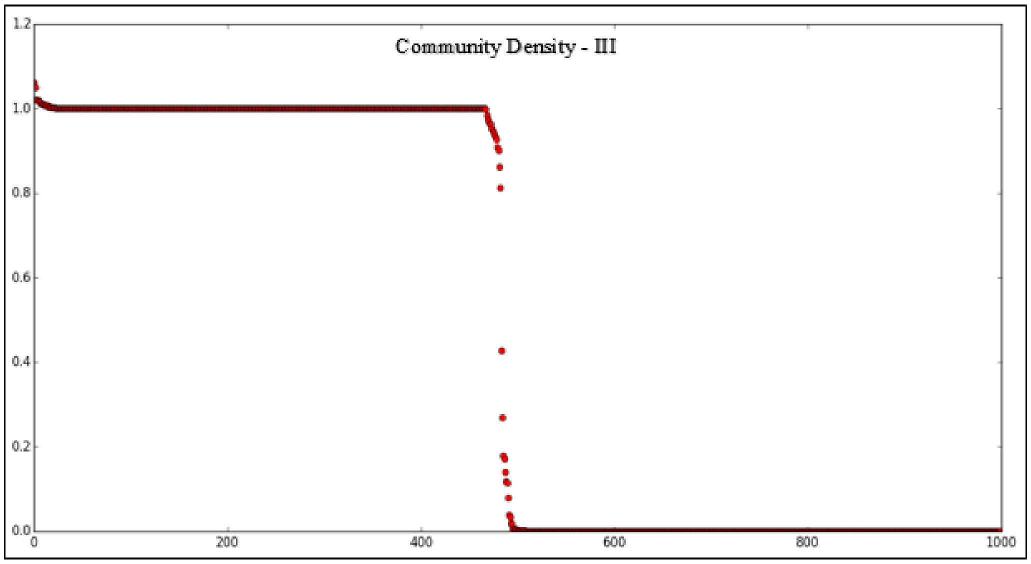


Figure 4.7: In this Figure, the y-axis represents the size of singular values of the matrix L whose corresponding M is constructed by the CD-III, and the x-axis represents the index of the singular values ordered from largest to smallest. The singular values decreases and becomes almost zero and constant after 500 sensor nodes, but there is no consistent exponential decrease of singular values, and most importantly, the singular values remain constant until 500 and suddenly drops almost to zero after 500. This is anomalies graph for a graph of SVD, which is highly insensitive until singular value 500.

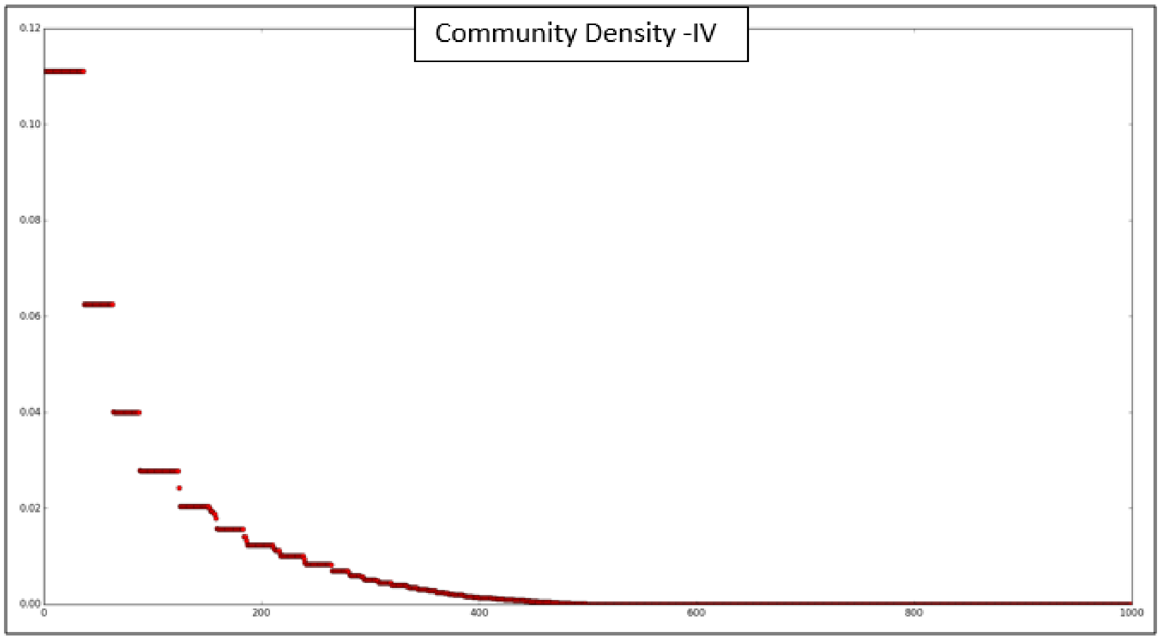


Figure 4.71 In this Figure, the y-axis represents the size of singular values of the matrix L, which is constructed by the CD-IV, and the x-axis represents the index of the singular values ordered from largest to smallest. The singular values decrease and becomes almost zero and constant after 500 sensor nodes, but the decrease of singular values is a discretely continuous trend, which shows the instability of decomposition of such a matrix, so we would be skeptical of using the CD-IV metric.

Figure 4.71 shows the singular values of the second order matrix that is formed by CD-IV. The x-axis represents the index of the singular value ordered from largest to smallest, and the y-axis represents the size of singular values of the matrix. We observe that the singular values decrease in discretely continuous trend, which shows the instability of decomposition of such a matrix, so we would be skeptical of using the CD-IV metric.

Figure 4.72 shows the second order matrix of Amazon communities data constructed from metric CD-2, x-axis and y-axis represents the communities, the values inside represent the strength of community connection of CD-2. As we can see from the graph, most of the values are zero and only a few of them are nonzero values.

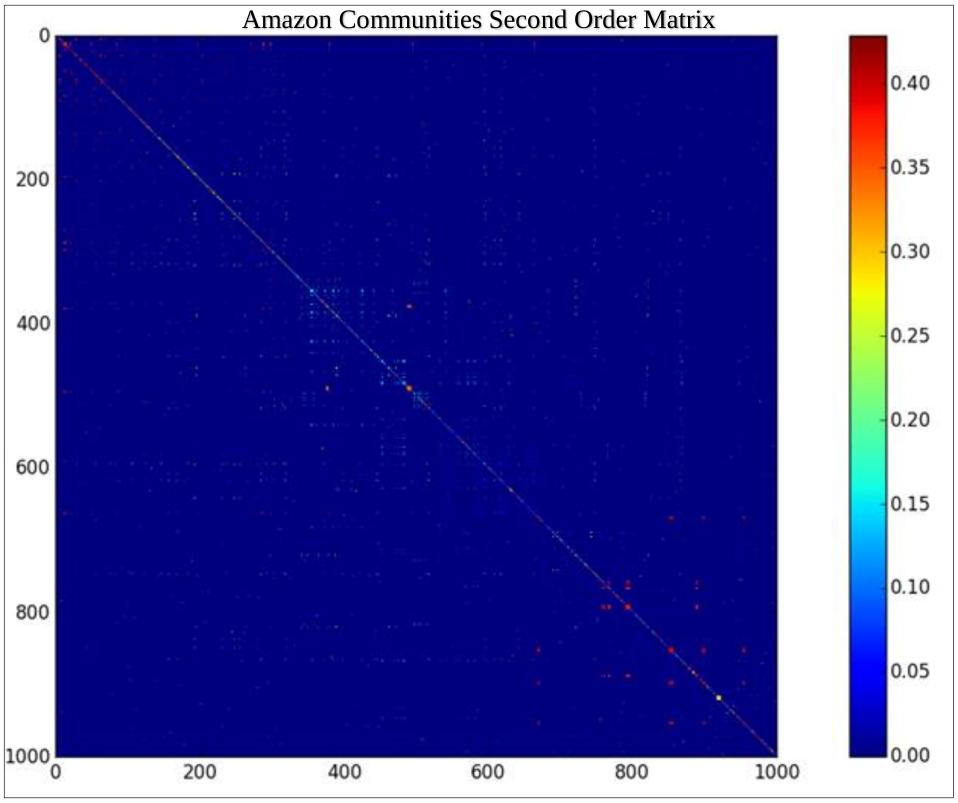


Figure 4.72: In this second order matrix, which is constructed by CD2 metric. The x-axis and y-axis represents the individual communities, the values inside that represent the strength of connection between two communities measured using CD2.

4.4.2 Singular Values Comparison of SVD, RPCA and eRPCA

In the Figure 4.8, we have used dimensionality reduction techniques like SVD, RPCA and our own eRPCA to compare the singular values of the second order matrix of the Amazon communities data [2], which is constructed using the CD-II metric. We observe that the SVD produces larger number of non-zero singular values compared to Robust and point wise error techniques like RPCA and eRPCA. The first non-zero singular value for the eRPCA is the eighth singular value and for the SVD is much larger. This implies that using the eRPCA techniques require a very few variables for prediction or dimensionality reduction as compared to the SVD. In practical settings of the Amazon communities data, the large number of non-zero singular values for the SVD implies that we require a large number of communities for the SVD to predict the strength of connection between the other communities. In contrast RPCA and eRPCA requires fewer communities to predict rest of the entries for other communities. Therefore, we use eRPCA technique for further analysis in our thesis with the CD-II metric definition of the second order matrix of the Amazon communities data.

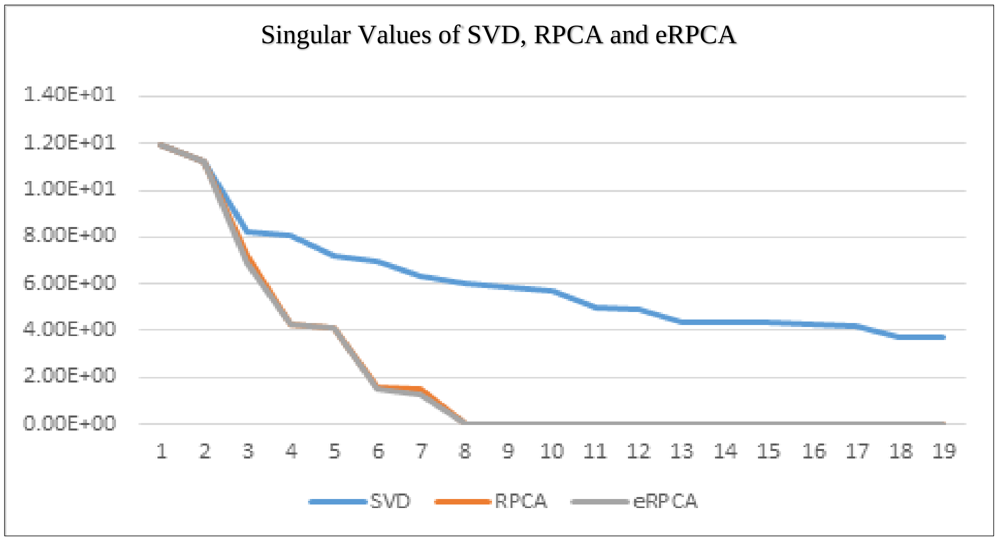


Figure 4.8: In this Figure, the y-axis represents the size of singular values, and the x-axis represents the index of the singular value order from smallest to largest. It shows that the singular values of three techniques: SVD, RPCA and eRPCA. We see that SVD produces larger number of non-zero singular values than RPCA and eRPCA. These singular values of RPCA and eRPCA are similar and that is why the lines are coincided.

In using the eRPCA algorithm for the second order matrix of the Amazon communities data with community definition metric of the CD-II, we mainly experimented on three scenarios. In the following sections, we detail those experiments on some of the eRPCA parameters and observe the corresponding matrices generated from eRPCA. In particular, first, we change λ values. Second, we change the error values to understand the relationship between the low rank and the sparse matrix. Third, we measure the prediction error for low rank matrix.

4.5 Effect of λ values on Low Rank and Sparse Matrices

The idea of experimenting with λ is based on the observation that the second order matrix of the Amazon communities data was highly low rank and sparse using the default value λ_0 [1]. From the principles of the eRPCA algorithm, a dense low rank matrix is easily predictable, however a sparse matrix is not easily predictable since all of the entries can be viewed as anomalies[ ]. Technically speaking, those entries of L whose corresponding sparse entries are zero $(S_{ij} = 0)$ are of $Type - LE$. However, if the entries of L whose corresponding sparse entries are not zero $(S_{ij} \neq 0)$ are of $Type - HE$. The purpose of using λ in the second order matrix of Amazon communities data is to adjust the density of non-zero values between the low rank and the sparse matrix. In other words, the control parameter λ helps us to control the rank of the low rank matrix and the prediction usability of L, and to control the sparsity in the sparse matrix.

In the case of applying the eRPCA algorithm to the Amazon communities data, the default value of $\lambda = 1 * \lambda_0$ does not yield any non-zero values in L . All of the non-zero entries of M appear in in S and all of the singular value of L is zero. A low rank matrix L with no non-zero singuar values or, in other words, no non-zero entries provides no information for prediction. Therefore, the experiment of increasing the λ value is intended to get more non-zero values in L, and hence increase the singular value of L, which would help us to predict the strength of connection between the communities. When eRPCA is run with $\lambda = 0.1 \times \lambda_0$ to decompose M into L and S, as shown in Figure 4.9, there are no non-zero entries in the low rank matrix and the largest singular value is zero. In addition, in the sparse matrix, all of the non-zero entries appear. This observation is quite opposite to what happens in the classical decomposition of typical second order matrix (where $S_z \gg L_z$ and $S_{nz} \ll L_{nz}$).

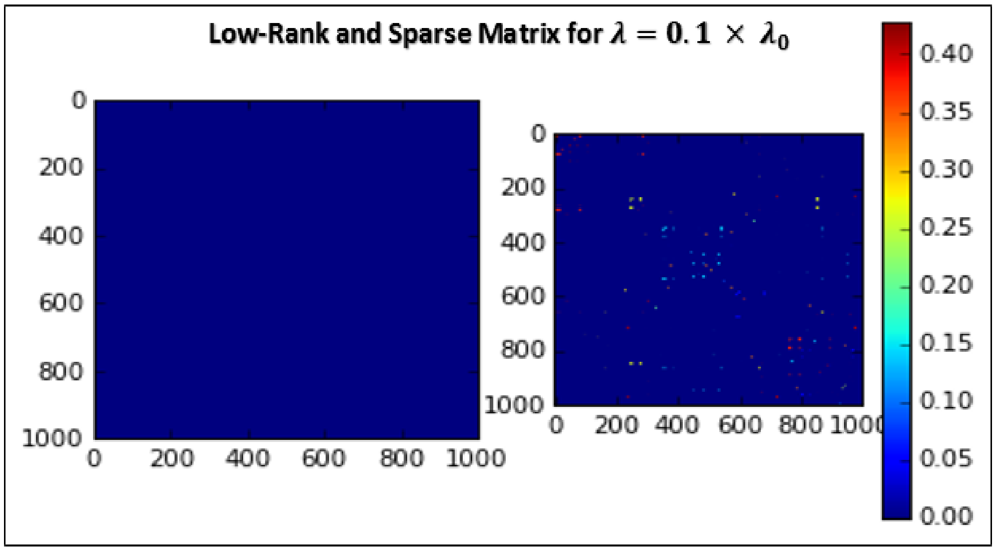


Figure 4.9: In this Figure, the left plot is the visualization of the low rank matrix and the right graph is the visualization of the sparse matrix. These plots are obtained from the decomposition of M using eRPCA with $\boldsymbol{\lambda} = \mathbf{0.1} \times \boldsymbol{\lambda_0}$. We can see from the above Figure, there are no non-zero values in the low rank matrix, and all of the non-zero values appear in sparse matrix.

The Figure 4.9 shows the visualization of the entry values in the low rank and the sparse matrix obtained by the decomposition of M using the eRPCA algorithm for $\lambda = 1 * \lambda_0$ shown on the left and right side of the graph respectively. We observe that the low rank matrix has no non-zero entries, whereas all of the non-zero values appear in sparse matrix. This can be observed from the color dots, where the sparse matrix has more dots than the low rank matrix. Comparing

Figure 4.3, which is a classic example of an L and S from the eRPCA algorithm, to Figure 4.9, one can easily that the L matrices are substantially different.

Figure 4.10 shows the visualization of the entries in the low rank and the sparse matrix, shown on the left and right side of the graph respectively, obtained by the decomposition of M using the eRPCA algorithm for $\lambda = 8 * \lambda_0$. We observe that the low rank matrix has a large number of non-zero entries, whereas the sparse matrix has a smaller number of non-zero entries as compared to L. This can be observed from the color dots, where the sparse matrix has lesser dots than the low rank matrix.

The sparse matrix is denser than the low rank matrix in the Figure 4.9 compared to the Figure 4.10. With the increased value of λ, S becomes more sparse (i.e., less dense, as observed from the color dots). This accounts for more entries in the low rank matrix (i.e., more dense), which is helpful for prediction but at the cost of a greater number of known community strengths. The various values of λ in the experiment are shown in the Table 4.1.

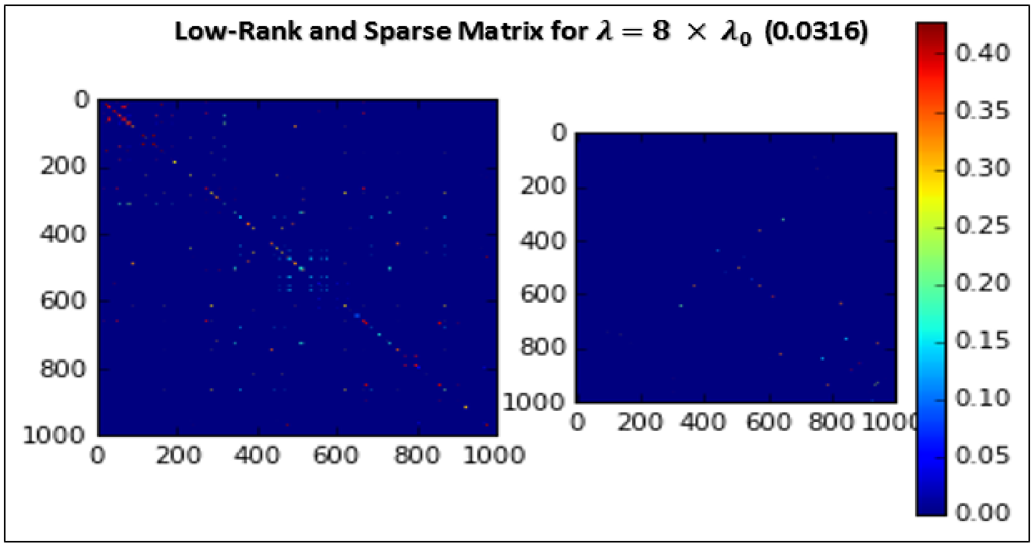


Figure 4.10: In this Figure, the left plot is the visualization of the low rank matrix and the right graph is the visualization of the sparse matrix. These plots are obtained from the decomposition of M using eRPCA with $\boldsymbol{\lambda} = \mathbf{8} \times \boldsymbol{\lambda_0}$. We can see from the above Figure, there is large number of non-zero entries in the low rank matrix, and there is lesser number of non-zero entries in the sparse matrix.

In the Figure 4.11, the x-axis represents the λ values and the y-axis represents the number of non-zero entries in the sparse matrix S. We observe that the number of non-zero entries in S for $\lambda = 0.1 * \lambda_0$ is high, and as we increase the value of λ the number of non-zero entries in S decreases. We observe that the number of non-zero entries in S is 4300 for $\lambda = 0.1 * \lambda_0$, and the number of non-zero entries in S is 800 for $\lambda = 8 * \lambda_0$.

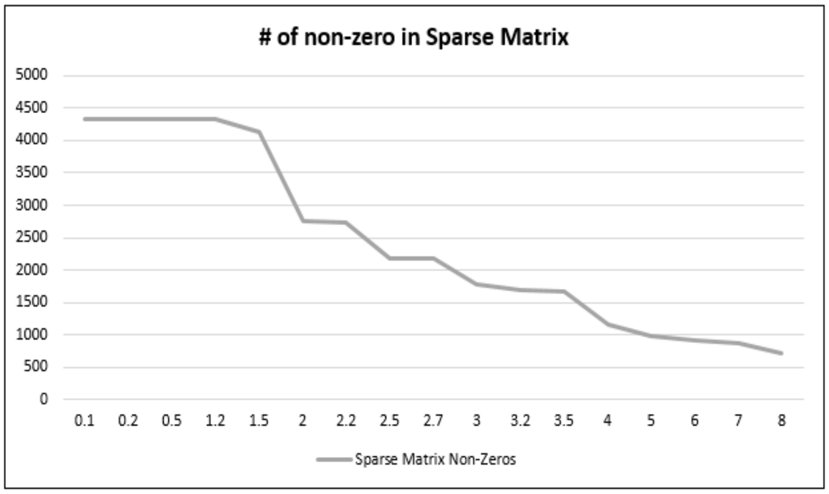


Figure 4.11: In this Figure, x-axis represents the λ values and y-axis represents the number of non-zero entries in sparse matrix. This plot shows the decrease in non-zero entries in sparse matrix with increasing λ.

In the Figure 4.12, the x-axis represents the λ values and y-axis represents the number of non-zero singular values. We observe that the all the singular values are zero for $\lambda = 0.1 * \lambda_0$, which means we cannot predict any of the entries in M. As we increase the value of λ, the number of non-zero singular values increases in the low rank matrix, which means we need to observe more of the strengths between the communities *but we are actually able to predict something*. We observe that the number of non-zero singular values is around 20 for $\lambda = 8 * \lambda_0$ which means we require 20 dimensions to predict rest of the 980 dimensions.

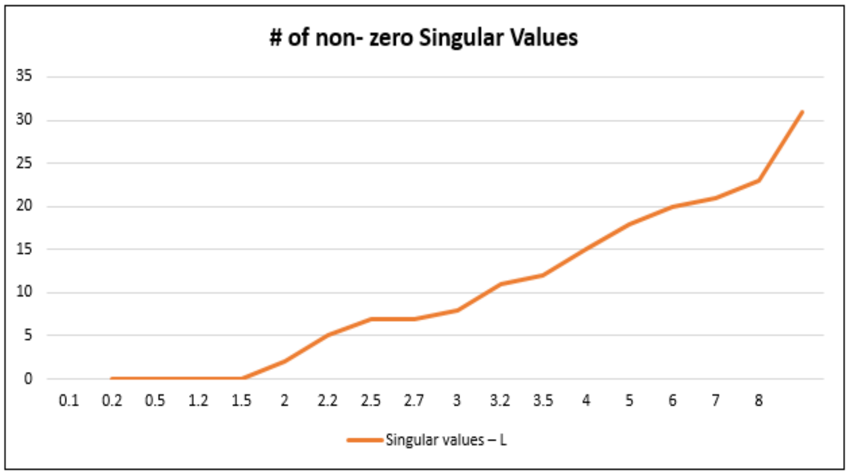


Figure 4.12: In this Figure, x-axis represents the λ values and y-axis represents the number of non-zero singular values of the low rank matrix. This plot shows the increase in the number of non-zero singular value of the low rank matrix with increasing λ.

$\boldsymbol{\lambda = l * \lambda_0}$ $\boldsymbol{l}$=	**Singular values**	**Sparse Matrix Non-Zeros**	**Sparse Matrix Zeros**
0.1	0	4324	995676
0.2	0	4324	995676
0.5	0	4324	995676
1.2	0	4324	995676
1.5	2	4126	995874
2.0	5	2763	997237
2.2	7	2742	997258
2.5	7	2188	997812
2.7	8	2183	997817
3.0	11	1786	998214
3.2	12	1696	998304
3.5	15	1676	998324
4.0	18	1164	998836
5.0	20	973	999027
6.0	21	920	999080
7.0	23	871	999129
8.0	31	722	999278

Table 4.1: This table shows the details of the λ values used in our experiment and it shows the corresponding singular values and number of non-zero and zero entries of the sparse matrix.

As seen can be seen from Figures 4.11 and 4.12 there is a balance between the large and small values of λ. In particular, small values of λ force all entries into S and accordingly L does not allow you to predict anything. Similarly, large values of λ cause too many non-zero singular values in L requiring many observations in M to make accurate predictions. So our goal is to choose a value of λ that balances these two competing interests.

4.6 Changing $\boldsymbol{vE}$ values on $\boldsymbol{M_p}$to understand the Error in prediction

This Section explains the analysis we conducted on various forms of error due to prediction of entries. These errors arise by changing the number of entries to be predicted (M_p), and by keeping the controlling parameter λ constant. We use vE_H in the eRPCA algorithm to designate the entries that needs to be predicted. Further, we have analyzed a few different ways to measure errors in this scenario.

When the eRPCA algorithm decomposes M by using vE_L on the off-diagonal entries and vE_H on the diagonal entries, we get L and S. We analyze both L and S by looking for entries where $S_{ij} \neq 0$. Since these entries are not predictable we set vE_H on those entries and recompute the decomposition to get L_p and S_p. We test whether L_p is a good approximation of L and we repeat the process of using the eRPCA algorithm by changing the number of entries to be predicted (M_p) in incremental value and measure the error for each of the M_p = {2,5,10,..100 ,..5000 ,...996000).

We measure various forms of error, to understand the effect of M_p. As we change M_p, the error of each of the entries changes as well, and we want to account for the error for each of the M_p values used in the error metric, so we normalize the error based upon the value of M_p. We define the first error metric, to account for M_p and call it the $lized\ Error\ (NE)$, it is the error between the low rank matrices divided by the number of entries taken for prediction. It is represented mathematically as

$$NE = \frac{\sum_{i,j}|L_p - L|}{2 \times M_p}. \qquad 25)$$

Where M_p, represents number of communities that needs to be predicted, and we multiply by 2 in the denominator because the second order matrix is symmetric across the diagonals, L is the low rank matrix arising from M with vE_L on all off-diagonal entries and L_p is the low rank matrix arising from M with vE_H on M_p entries.

In the above equation, our hypothesis is that as we increase the number of entries M_p to be predicted, the error should converge to some mean error. Surprisingly it seems not to happen as we hypothesize. In the Figure 4.13, which represents the error metric NE, x-axis represents the increasing number of M_p (from 2 to 99600, and refer to the appendix for detailed values of x-axis), and y-axis represents the error metric NE. As we observe, the errors (NE) do not appear to be converging to any mean (at least for the range of value of M_p we have studied). Figure 4.13 seems to raise many questions on the nature of fluctuating pattern of errors NE.

Accordingly, we delve into the error metric (NE) in order to understand the reasons for the fluctuations of NE. Mathematically, the NE trend line for specific M_p decreases because the value in denominator M_p has changed relatively more than the numerator in the equation ($i.e., \sum_{n=1}^{n=n}|L_p - L| < 2 \times M_p$), which means the errors of individual entries accounted by those additional entries are relatively small as compared to the previous entries. In addition, the NE trend line for specific M_p increases because of the denominator M_p has changed relatively more than the numerator in the equation ($i.e., \sum_{n=1}^{n=n}|L_p - L| > 2 \times M_p$) which means the error accounted by those additional entries are relatively small as compared to the previous entries.

Figure 4.13 demonstrates that the distributions of errors in approximating L by L_p are almost certainly not stationary as a function of M_p. Accordingly, we need to be very careful in analyzing the prediction errors.

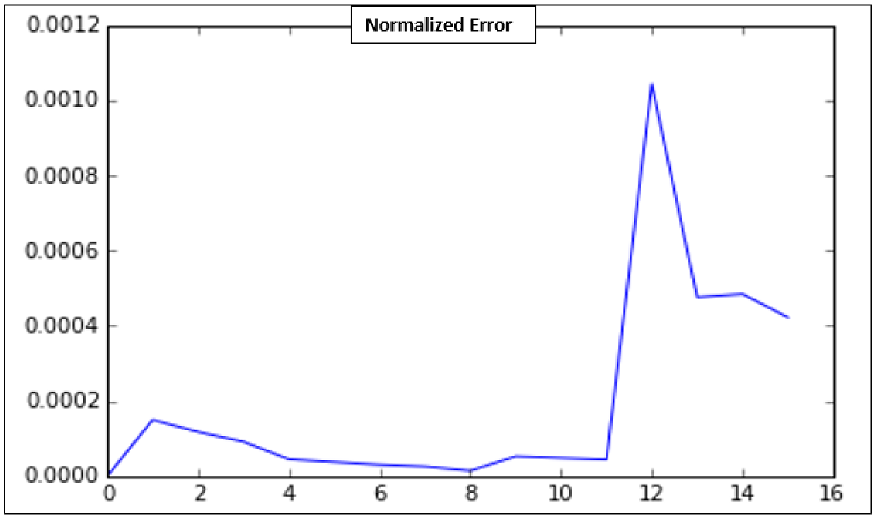


Figure 4.13: In this Figure, it represents the error metric NE, the x-axis represents the number of entries to be predicted (M_p), and the y-axis represents the absolute un-normalized error or in short, we call Normalized error(NE). The NE is not showing any trend and is fluctuating even for large values M_p.

In the previous error metric NE, we observe the errors are not stationary as a function of M_p , so we examine NE carefully to see if there is a better way to measure the errors in the entries. In order to do this, we define a new error metric called as *absolute value of un-normalized error* denoted by A_UNE. A_UNE is defined as the absolute difference between the default low rank matrix (L), and the predicted low rank matrix (L_p), mathematically it is represented in the equation below. As we observe in the error metric it does not include M_p, which would give us a better picture of the error variation of the entries between the low rank matrices. Therefore, we define

$$A\_UNE = \sum_{i,j} |L_p - L|. \quad 26)$$

In Figure 4.14, represents the error metric A_UNE. The x-axis represents the increasing number of M_p and the y-axis represents the error metric A_UNE. We observe from the graph that the error increases as a non-linear exponential function.

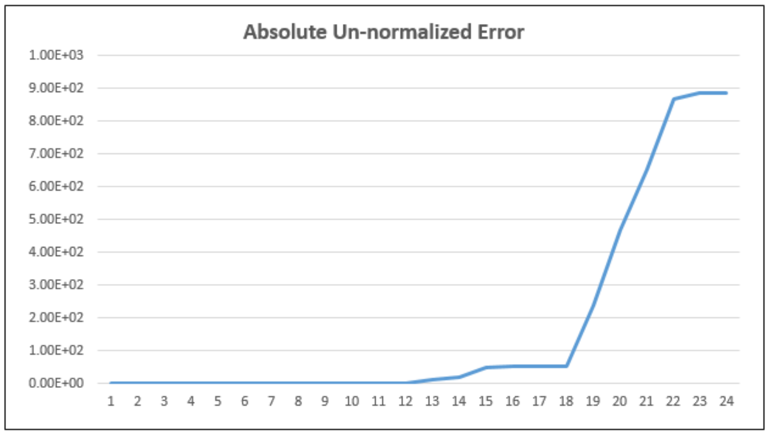


Figure 4.14: In this Figure, it represents the error metric A_UNE, the x-axis represents the number of entries used to predict (M_p), and the y-axis represents the absolute un-normalized error(A_UNE). It shows the trend that with the increase in number of entries to be predicted the error keeps increasing.

We observe the graph in the Figure 4.14, with the increase in number of entries to be predicted, the algorithm tries to find the predicted value of those entries, which would be slightly different from the actual values, and the error keeps increasing with increasing number of entries to be predicted. However, the rate of increase keeps changing. Therefore, this means the normalized error keeps fluctuating because of the large values of M_p with small change in numerator or relatively large change in numerator and small change in M_p.

4.7 Prediction of $\boldsymbol{M}$ using Low Rank matrix by changing $\boldsymbol{\lambda}$ and $\mathbf{P}_{\mathbf{max}}$

Attributes of Error Metrics: When we do prediction of entries of M, such predictions always comes with a cost of error. In order to know the effect of these errors on the problems of interest, we need to quantify the error metrics based on the specific problem situation. We present primarily four metrics to measure the error of prediction. First, we use **SSE (Sum of Squared Errors)** which is defined as the sum of the squares of the difference between predicted matrix (L or L_p) and the second order matrix M. Second, we use **Absolute Error** which is defined as the sum of the absolute values of the differences between the entries of predicted matrix (L or L_p) and M.

In practice, we evaluate the prediction accuracy by comparing the second order matrix M (which represents the real world data) and various low rank matrices obtained by changing the parameters of the eRPCA algorithm. In this Section, we understand the nature of the prediction error of M using L by varying the values of λ and $\mathrm{P}_{\max}$. We take three different values of $\mathrm{P}_{\max}$, one is the optimal singular value using SVD (i.e., $\mathrm{P}_{\max} = 300$) . This $\mathrm{P}_{\max}$ value was derived using an analysis similar to that in Chapter 3 using a 500×500 second order matrix. To study how the algorithm performs away from the optimal value of $\mathrm{P_{max}} = 300$ we also use the values of $\mathrm{P_{max}}$ = 100 and 400.

In this Section of the experiment, we consider $C_S = 500$ which implies that $N = 250{,}000$ and we do not use M_p (in other words we compare L to M instead of comparing L to L_p). We measure the error for a given λ in three different settings ($\mathrm{P_{max}}$ =100, 300 and 400). We start by taking the default value of $\lambda = \lambda_0$ and increment the value of λ in a range from 1 to 40, and measure the error between M and L using an error metric, similar to A_UNE, for each of the λ values defined as

$$A\_UNE = \sum_{i,j} |M - L|. \tag*{27)}$$

In the Figure 4.16, the x-axis represents the λ values, and the y-axis represents the A_UNE error. There are two lines, which are captured with $\mathrm{P_{max}}$ values of 300 and 400. Herein, we observe that after $\mathrm{P_{max}}$ is changed from the optimal value of $\mathrm{P_{max}}$ = 300 to 400, there is negligible difference in the measure of error with changing values of λ for both of the values of $\mathrm{P_{max}}$, hence we observe from Figure 4.16 that the gray line coincides with orange line. In addition, we observe that after λ = 25, the error becomes completely zero which shows that all the entries of L are perfectly predictable after λ = 25 (but we do require many singular values to make these predictions). In particular, we need to note that as we increase the value of λ, we are making the low rank matrix denser (i.e., we are moving all the values from sparse matrix to low rank matrix). This increase comes at the cost of the rank of the low rank matrix. By increasing the value of λ, though we are making the low rank matrix denser but the rank of low rank matrix may be much higher as compared to the default value of λ = λ_0.

In Figure 4.17, the x-axis represents the λ values, and the y-axis represents the A_UNE error. There are two lines, which are captured with the $\mathrm{P_{max}}$ values of 100 and 300. Herein, we observe that after the $\mathrm{P_{max}}$ is changed from the optimal value of $\mathrm{P_{max}}$ = 300 to 100, there is slight difference in the measure of error with changing values of λ when compared to Figure 4.16. There is a gap in the error metric at λ = 25 when measured for $\mathrm{P_{max}}$= 100 and 300. The error is zero for $\mathrm{P_{max}}$ = 300 at λ = 25 whereas the error is not zero for $\mathrm{P_{max}}$= 100. This observation allows us to draw an inference that by using $\mathrm{P_{max}}$ less than optimal value of singular value (< 300), the error may converge to nearly zero after some higher values of λ (> 25) than using the optimal value($\mathrm{P_{max}} \geq 300$). In other words, the error converging to almost zero is slightly faster for $\mathrm{P_{max}} \geq 300$ than for $\mathrm{P_{max}} < 300$. Though the trend of the error line is very similar to both $\mathrm{P_{max}}$ = 300 and 100. In addition, the error at different values of λ could have slightly more error for $\mathrm{P_{max}} < 300$ compared to $\mathrm{P_{max}} \geq 300$.

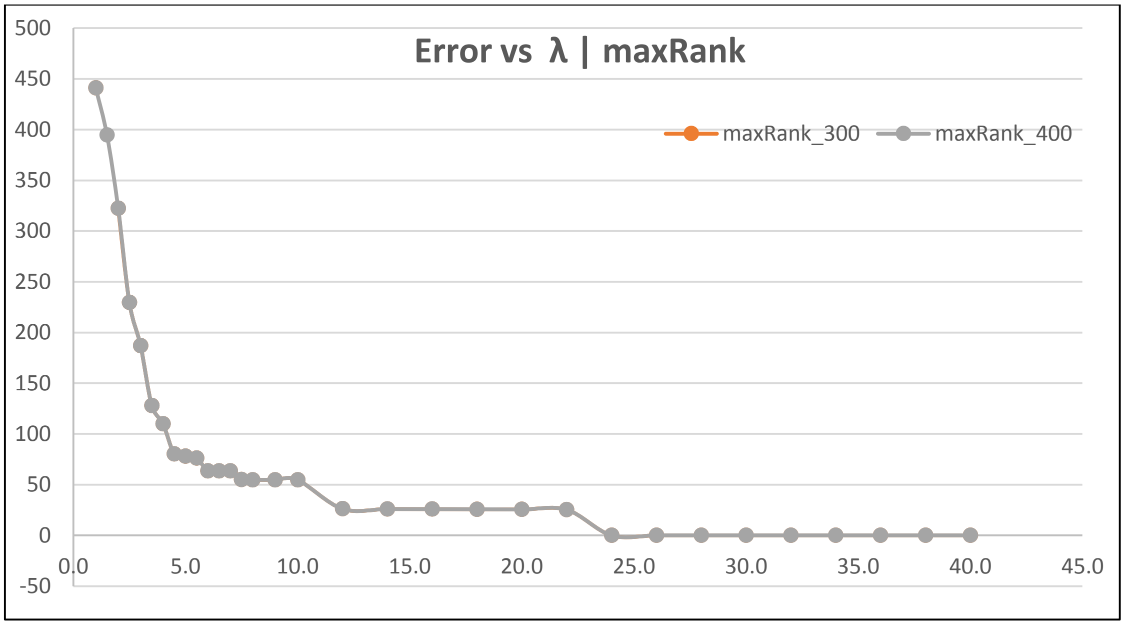


Figure 4.16: In this Figure, x-axis represents the λ values, and the y-axis represent the A_UNE error. This graph shows the variation of error with variation of λ values, with two different values of P_{max} = 300 and 400. The orange line represent the P_{max} of 300 and the gray line represents the P_{max} of 400, both the lines coincides as seen in the graph.

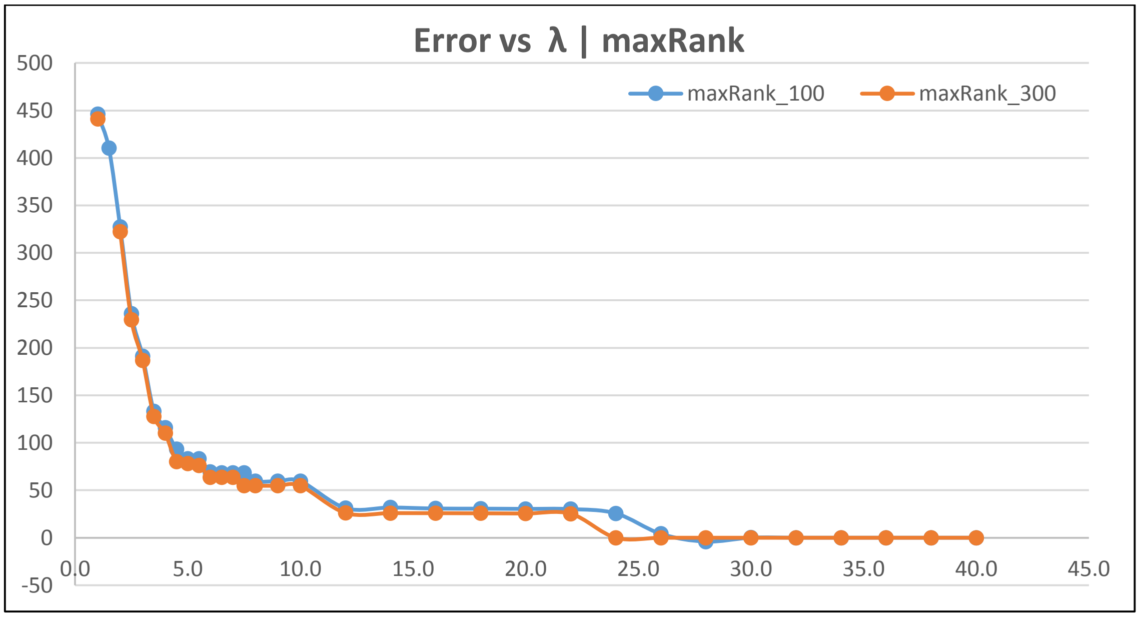


Figure 4.17: In this Figure, x-axis represents the λ values, and the y-axis represent the A_UNE error. This graph shows the variation of error with variation of λ values, with two different values of P_{max} = 100 and 300. The orange line represent the P_{max} of 300 and the blue line represents the P_{max} of 100, both the lines coincides as seen in the graph.

4.8 Prediction of $\boldsymbol{M}$ using the low Rank matrix $(\boldsymbol{L})$ by changing $\boldsymbol{\lambda}$ and $\boldsymbol{vE}$ on $\boldsymbol{M_p}$

In Section 4.7, we demonstrated that we should keep $\mathrm{P_{max}} \geq 300$ to optimize errors, so we proceed with that value in this Section. In addition, we measured errors with only one metric. However, in this Section, we will be measuring different kinds of error metrics in prediction between M and L. We sub-divide this section into 4.8.1 and 4.8.2, where the former one deals with $M - L_0$ and the latter one deals with $M - L_p$. In this Section, we vary two parameters of eRPCA i.e. λ and vE, and we keep M_p constant at 100,000 entries. We measure two metrics of errors i.e. SSE (Sum of Squared Errors) and Absolute errors.

When we run the eRPCA algorithm on the second order matrix, it is decomposed into a low rank and a sparse matrix. With different value of λ the low rank and sparse matrix are different. As we observed previously, as we increase the value of λ, the matrix becomes much denser. With this understanding of the principles of the eRPCA algorithm, we study the effect of λ on various kinds of errors. In addition, when we measure the error, we have four different views of observing the entries. 1) **Complete Entries without Diagonal entries,** herein, we consider only off diagonal entries to calculate the error. 2) **Complete Entries,** herein, we consider all the entries to calculate the error. 3) **GPE Entries:** In here, we consider only the GPE-Good predictable Entries ($S_{ij} \coloneqq 0\ or\ Type - LE$) to calculate the errors. **4) GPE Entries without Diagonal entries:** In here, we consider all the Good Predictable Entries ($S_{ij} \coloneqq 0\ or\ Type - LE$) without diagonal entries to calculate the errors.

We will first analyze the second order matrix using eRPCA using the default value of vE, i.e., using vE_H on the diagonals and vE_L on all off diagonal elements. We run the eRPCA algorithm to get a low rank and a sparse matrix, and we analyze each of the differences between the entries from the low rank matrix (L) with the second order matrix(M).

We use sum of squared errors to magnify the tiny errors of each entries, which is helpful for better analysis. In addition, we used absolute difference of the entries to understand the actual error difference of each entries. SSE is given by the below equation and here $L = L_0$ in 4.8.1 and $L = L_p$ in section 4.8.2

$$SSE = |M - L|^2, \qquad 28)$$

Absolute valued error is given by the below equation and here $L = L_0$

$$Abs = |M - L|, \qquad 29)$$

Where,

M Represents the second order matrix
L Represents the low rank matrix (L_0 or L_p)

4.8.1 Prediction by default $\boldsymbol{vE}$ $[\boldsymbol{M} - \boldsymbol{L_0}]$

Figure 4.18 shows the SSE error distribution of $M - L_0$, the x-axis represents the λ value and the y-axis represents the SSE. The four different lines represent four different views of measuring error. The L_0 represent the low rank matrix produced by the eRPCA algorithm using default value of vE, i.e., using vE_H on diagonals and vE_L on off diagonals.

In the Figure 4.18 and 4.19, in the yellow line "GPE- diagonal", we subtract the entries of second order matrix and low rank matrix of those off diagonal GPE entries. We observe that most of the errors of GPE are focused on diagonals (by comparing GPE and GPE-diagonal), so there are hardly any errors on off diagonal entries, which imply that we make very best prediction on GPE for off diagonal entries. In addition, the trend line for the error is flat which also shows that, its dependency on λ is almost negligible.

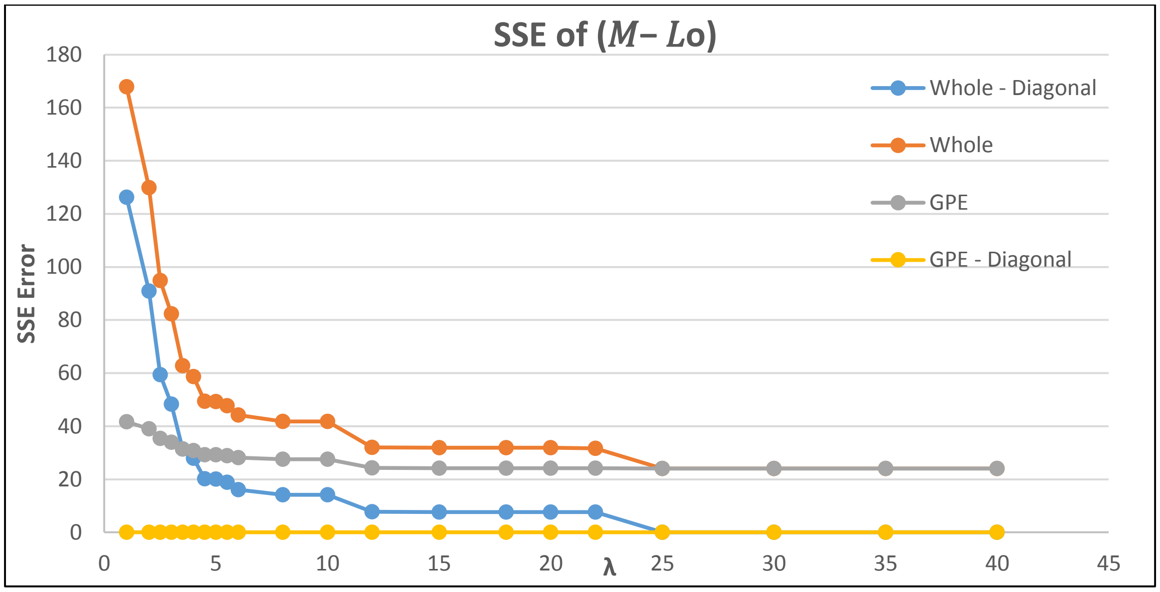


Figure 4.18: This Figure represents the SSE error distribution of $M - L_0$, the x-axis represents the λ value and the y-axis represents the SSE. The four different lines represent four different views of measuring error.The L_0 represents the low rank matrix by processing eRPCA with vE_L on non diagonal entries and vE_H on diagonal entries.

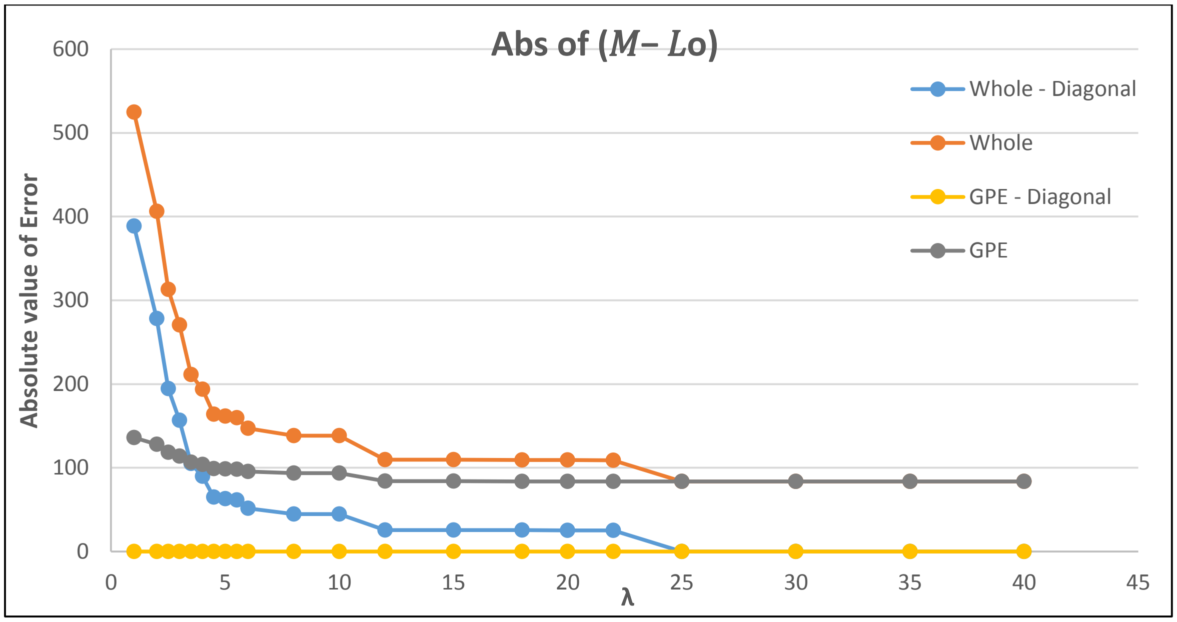


Figure 4.19: This Figure represents the Absolute error distribution of $M - L_0$, the x-axis represents the λ value and the y-axis represents the Absolute error. The four different lines represent four different views of measuring error.The L_0 represents the low rank matrix by processing eRPCA with vE_L on non diagonal entries and vE_H on diagonal entries.

For the gray line in Figure 4.18 and 4.19, we subtract the entries of second order matrix and low rank matrix of all the Good Predictable Entries. We observe that the error becomes completely constant after λ = 11, this shows that in order to predict GPE very well. We may need to use λ = 11 or more to find the best prediction, increasing the value of λ after 11 would cost more computation, but gives the same error results, so we suggest to use λ = 11 for this case. In addition, when we compare the yellow and gray line, we see that most of the errors are concentrated at the diagonal entries.

In the Figure 4.18 and Figure 4.19, we see from the blue line "Whole - Diagonal" that the error are high relative compared to other lines in the Figure, which is obvious because we are considering all the entries for error calculation. We can also make an inference that the errors becomes constant between λ = 11 and λ = 22, and after λ = 25. From λ = 1 to 4, we observe that the errors has a greater negative slope, which means the errors drastically change in this range of λ, and the error change after λ=4 is not so drastic. In the gray and orange line, we observe that from λ = 4 to 25 they become similar in their error profile and after λ = 25 we observe that they completely converge in error value, becoming constant after that. In addition, there is a similar trend between the blue and yellow lines after λ =25, the errors converge to zero and become constant. Now carefully observing these all four lines, we can infer that there are some *super anomalies*, whose error cannot be decrease further with increasing λ, and interestingly this fall under diagonal entries. Such, super anomalies are not predicted by the standard eRPCA theory [].

4.8.2 Prediction by varying $\boldsymbol{vE}$ $[\boldsymbol{M} - \boldsymbol{L_p}]$ (high $\boldsymbol{vE}$ for $\boldsymbol{S_{ij}} = \mathbf{0}$ entries)

Figure 4.18 shows the SSE error distribution of $M - L_p$, the x-axis represents the λ value and the y-axis represents the SSE. The four different lines represent four different views of measuring error. The $\mathrm{L_p}$ represent the low rank matrix produced by the eRPCA algorithm using vE_H on diagonals and on M_p = 100000 off diagonal entries.

In all the four lines except blue line “Whole-Diagonal”, there is a drastic increase in error from λ = 1 to 4, from 4 to 10, there is drastic decrease in the error, and later it becomes much constant. As we increase λ from 4 to 10, the low rank matrix becomes more dense and correspondingly increases the sparsity of sparse matrix, so the error decreases drastically. After λ =10, there are entries which are super anomalies which could not be predicted and hence the error remains almost constant after λ=10. There is negligible increase in error from λ=20 to 22 and then there is negligible decrease from 22 to 25 in all the 4 lines.

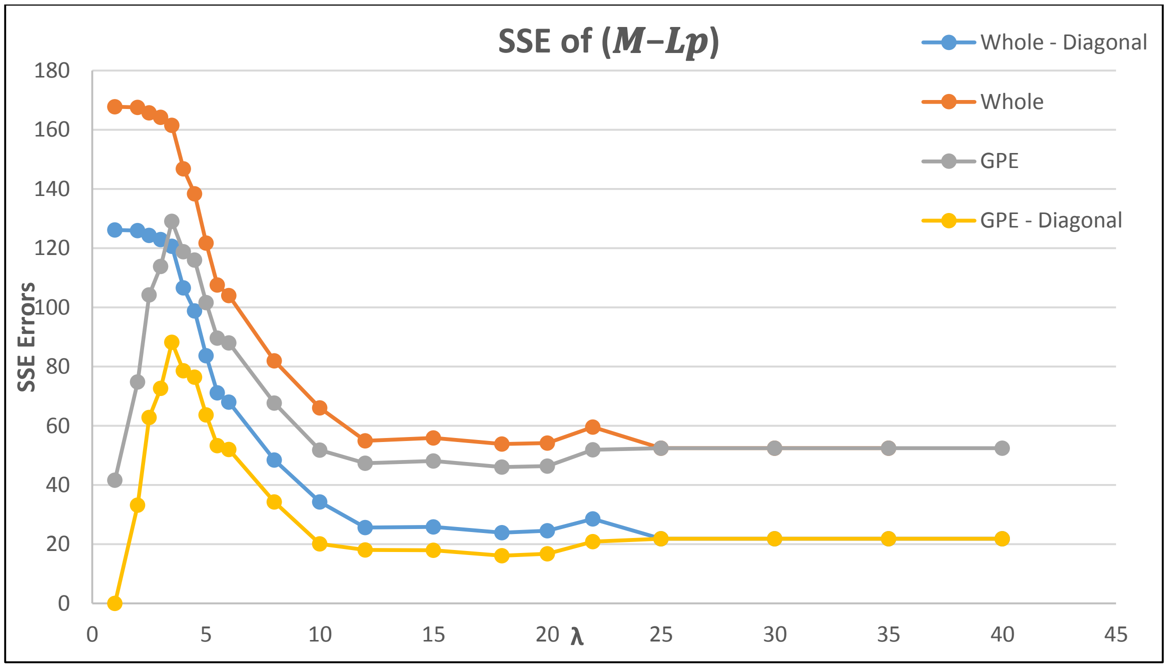


Figure 4.21: This Figure represents the SSE error distribution, the x-axis represents the increasing λ value and the y-axis represents the SSE, and all the four different lines represent various ways to observe the entries. L_p is the low rank matrix , which is decomposed by eRPCA using vE_H for M_p .

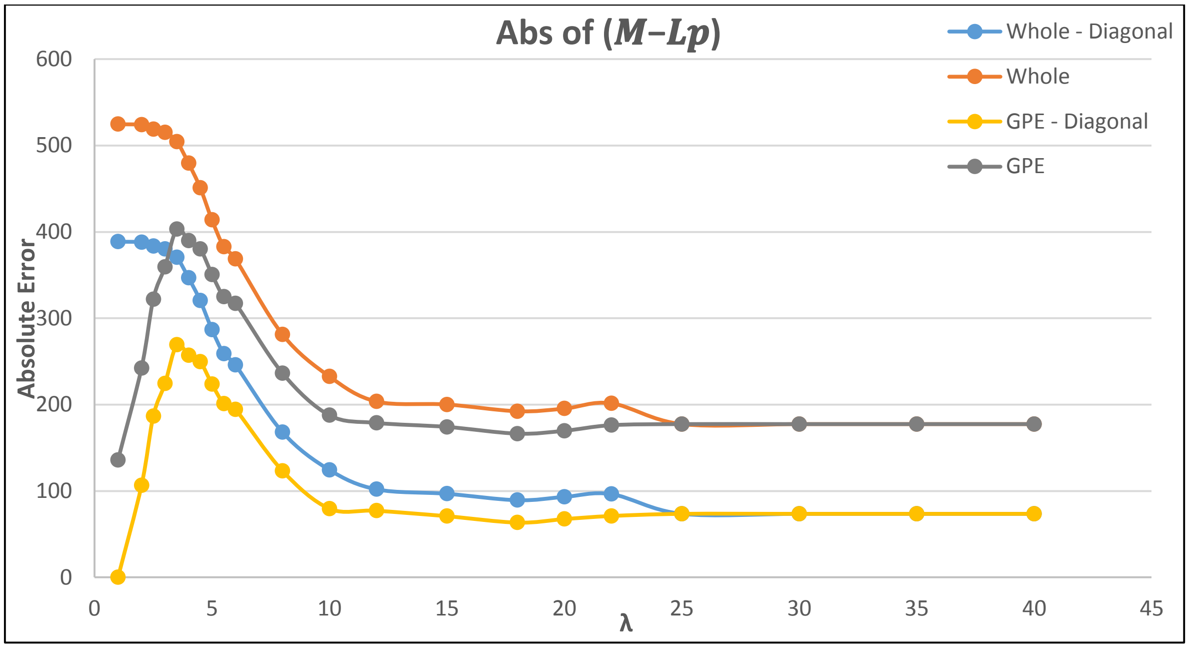


Figure 4.22: This Figure represents the Absolute error distribution, the x-axis represents the increasing λ value and the y-axis represents the SSE, and all the four different lines represent various ways to observe the entries. L_p is the low rank matrix , which is decomposed by eRPCA using vE_H for M_p .

For the yellow line "GPE- diagonal" in Figure 4.22, we subtract the entries of the second order matrix and the low rank matrix of those off diagonal GPEs. In here, we find that most of the errors of GPEs are focused on the diagonals, so there are small errors on off- diagonal entries. This observation implies that we make good predictions for GPEs for off diagonal entries. In addition, the trend line for the error is flat after λ =10, which show that, its dependency on, λ is almost negligible after certain value of λ.

For the gray line "GPE" in Figure 4.22, we subtract the entries of the second order matrix and low rank matrix of all the GPEs. In here, we find that the error becomes completely constant after λ = 10, this shows that in order to predict GPEs very well we may need to use λ = 11 or more to find the best prediction. Increasing the value of λ after 10 would cost more computation but with the same results, so we suggest to use λ = 10 for this case. In addition, when we compare case-3 and case-4, we see that most of the errors are concentrated at diagonal entries in GPE.

For the orange line "Whole" in Figure 4.22, we subtract all the entries of second order matrix and low rank matrix. We find that the error is high relative to other lines in the Figure, which is obvious because we are considering all the entries for error calculation. We can also make an inference that the errors becomes constant between λ = 11 and λ = 12, and after λ = 25 we see that errors becomes negligible. From λ = 1 to 4, we observe that the errors has greater negative slope, which means the errors drastically change in this range of λ, and the error change after λ = 4 is not that drastic.

From the blue and yellow line in Figure 4.22, we see that from λ = 4 to 25, they are similar in error profile and after λ = 25, we see that they completely converge in the error values and become constant. In addition, there is a similar trend between orange and gray line, after λ =25, these lines error converge to zero and become constant. Now carefully observing these two scenarios, we can infer there are some super anomalies whose error cannot be decreased further by changing λ, and interestingly these are diagonal entries.

4.9 Empirical relation between $\boldsymbol{\lambda}$ and number of non-zeroes in $\boldsymbol{S}$

In this Section, we analyze the relationship between the number of non-zeroes entries in the sparse matrix (S_0 *and* S_p) with changing λ. We conducted the experiment in five different settings of M_p and observe the relationship between λ and S_{nz}: 1) Running eRPCA and getting low rank matrix L and sparse matrix S 2) Running eRPCA with M_p as 20,000 entries 3) Running eRPCA with M_p as 50,000 entries 4) Running eRPCA with M_p as 100,000 entries 5) Running eRPCA with M_p as 200,000 entries. We vary M_p values on those entries, which gives $Type - LE$ errors. In each of the above settings, we vary the λ value and measure the number of non-zeroes in the sparse matrix for every change in λ value and Table 4.2 shows the results of the analysis.

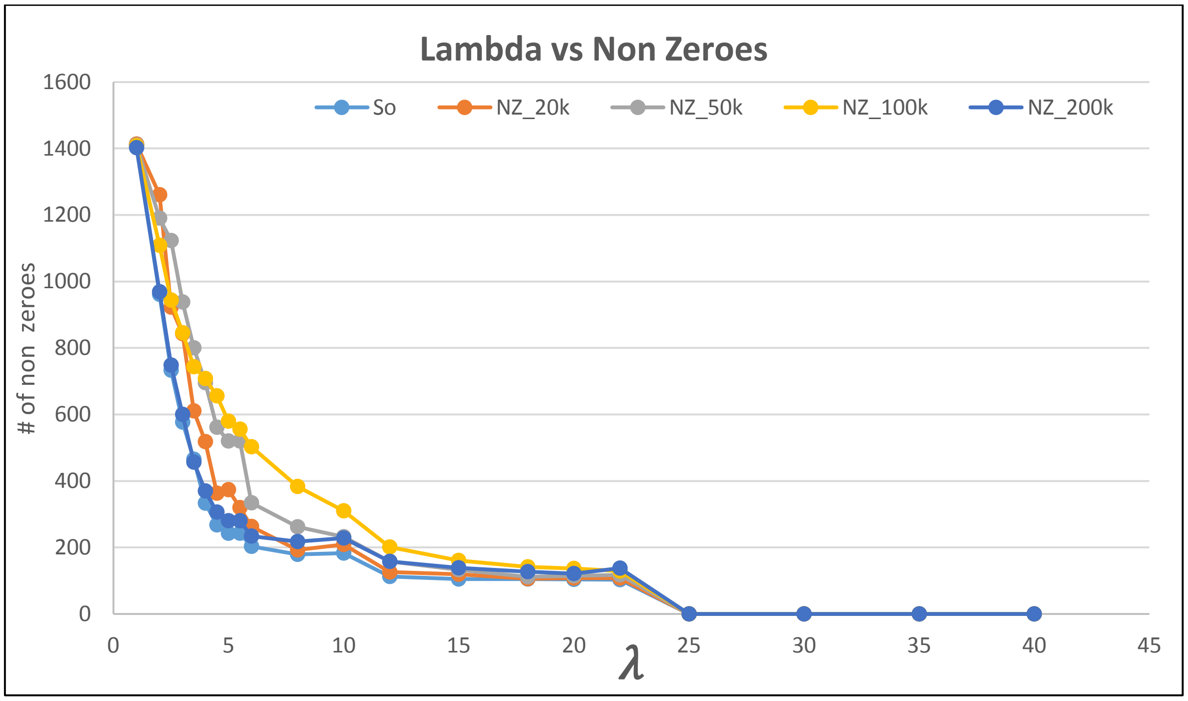


Figure 4.23 (a): The Figure shows the variation of S_{nz} and λ, x-axis represents the λ values and the y-axis represents S_{nz}. The different colored lines represent non-zero entries in S_0, S_p from M_p entries of 20k, 50k, 100k, 200k.

In the Figure 4.23(a), the x-axis represents the λ values and the y-axis represents S_{nz}. The different colored lines represent S_0, $\mathrm{S_p}$ from $\mathrm{M_p}$ entries of 20k, 50k, 100k, 200k. The graph shows that there is an exponential decay of number of non-zeroes in sparse matrix with increasing value of λ. Interestingly, complete entries in sparse matrix become zero after $\lambda = 25$ and this does not depend on the M_p entries. We observe that if the entries of M_p is higher, than the number of non-zeroes in sparse matrix is high compared to the lower entries of M_p for given value of λ.

We construct the relationship between the number of non-zero entries in sparse matrix and the λ values using the exponential curves from the Figure 4.23(a) and from the Table 4.2.In the various settings of M_p, we observe the relationship between S_{nz} and λ which appears to be exponential. Therefore, we fit the above values with the nonlinear equation using R's *nls* function and we get the below generalized equation from each of the different settings of M_p.

$$Log(S_{NZ}) = 7.2788 – 0.19704 * (\lambda) \qquad 30)$$

Where,

S_{NZ} represents the number of non-zeroes in sparse matrix

λ is the controlling parameter to control the sparsity of S_{NZ}

Therefore, the equation is with intercept value of 7.2788, and the coefficient value for λ is -0.1970.

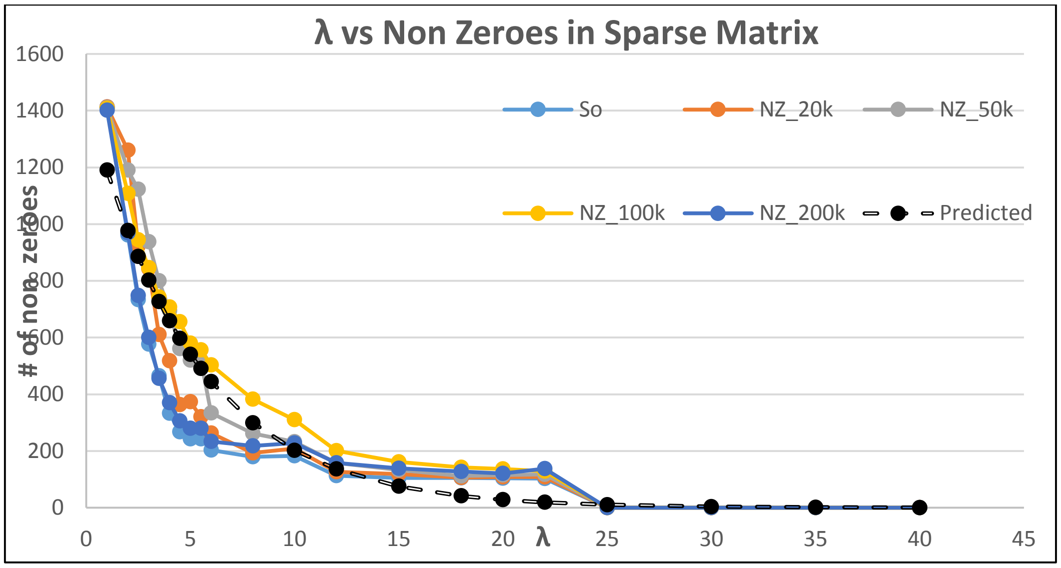


Figure 4.23: The Figure shows the variation of S_{nz} and λ, x-axis represents the λ values and the y-axis represents S_{nz}. The different colored lines represent non-zero entries in S_0, S_p from M_p entries of 20k, 50k, 100k, 200k, and the predicted value.

λ	S_{nz}	NZ_20k	NZ_50k	NZ_100k	NZ_200k	Predicted
1	1413	1413	1410	1408	1402	1190
2	961	1261	1190	1108	969	977
2.5	733	923	1123	944	748	886
3	577	843	938	846	600	802
3.5	465	611	800	743	456	727
4	333	518	695	708	370	659
4.5	268	363	561	656	306	597
5	243	374	520	580	280	541
5.5	243	320	520	556	280	490
6	203	263	334	503	234	444
8	179	193	262	383	218	300
10	183	209	232	310	228	202
12	113	126	158	201	158	136
15	105	119	133	161	139	75
18	105	106	112	142	127	42
20	104	108	115	137	121	28
22	103	108	117	129	138	19
25	0	0	0	0	0	11
30	0	0	0	0	0	4
35	0	0	0	0	0	1
40	0	0	0	0	0	1

Table 4.2: This table shows the experimental results between λ values and the number of non-zeroes corresponding to the λ value. The predicted value corresponds to the values we have predicted based on the experiments on other columns.

Table 4.2 has the λ values used in our experiment, S_{nz} is the number of non-zero values with no M_p, NZ_20k represents the number of non-zero values in sparse matrix obtained by using $M_p = 20000$ entries. NZ_50k represents the number of non-zero values in sparse matrix obtained by using $M_p = 50000$ entries. NZ_100k represents the number of non-zero values in sparse matrix obtained by using $M_p = 100000$ entries. NZ_200k represents the number of non-zero values in sparse matrix obtained by using $M_p = 200000$ entries. Predicted is the number of non-zeroes predicted from all the five settings.

In the Figure 4.23, the x-axis represents the λ values and the y-axis represents S_{nz}. The different colored lines represent S_0, $\mathrm{S_p}$ from $\mathrm{M_p}$ entries of 20k, 50k, 100k, 200k, and the predicted line. The graph shows that there is an exponential decay of number of non-zeroes in sparse matrix with increasing value of λ. Interestingly, complete entries in sparse matrix become zero after $\lambda = 25$ and this does not depend on the M_p entries. We observe that if the entries of M_p is higher, than the number of non-zeroes in sparse matrix is high compared to the lower entries of M_p for given value of λ. The predicted dot line shows the fit line from the above equation, which fits the exponential curves.

4.9.1 Making a few Non Zero Entries as High

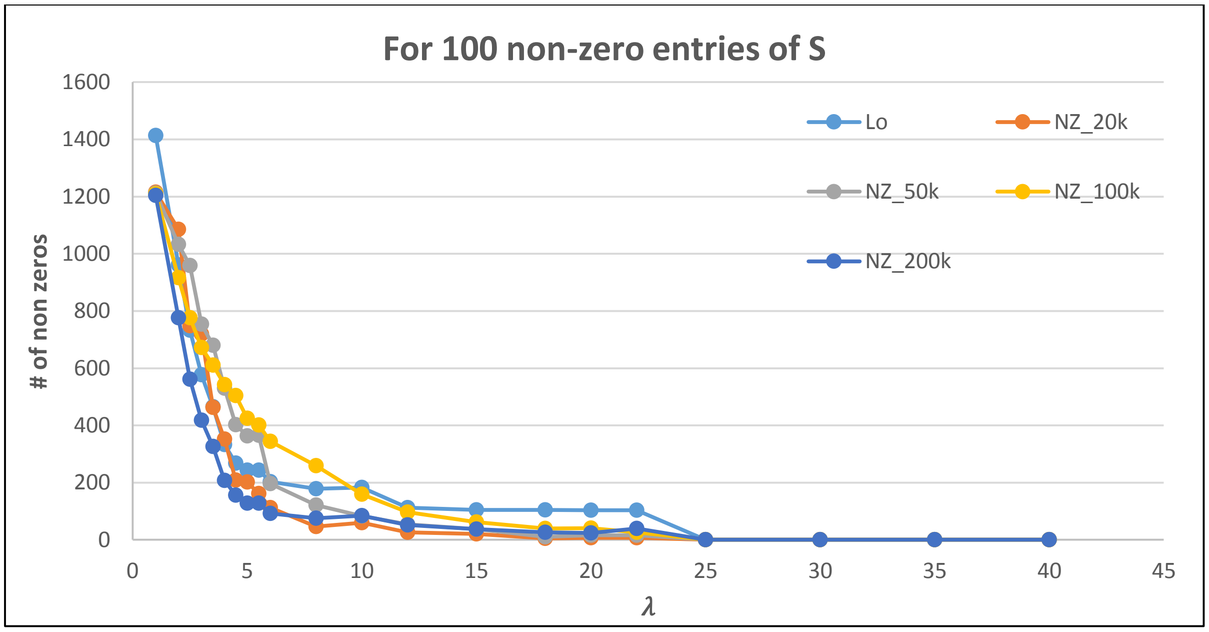


Figure 4.24: The Figure shows the variation of $\mathrm{S_{nz}}$ and λ, x-axis represents the λ values and the y-axis represents $\mathrm{S_{nz}}$. The different colored lines represent non-zero entries in $\mathrm{S_0}$, $\mathrm{S_p}$ from $\mathrm{M_p}$ entries of 20k, 50k, 100k, 200k. In those $\mathrm{M_p}$ entries that needs to be predicted, 100 entries of those $\mathrm{M_p}$ entries are used for prediction whose corresponding sparse entries are not zero.

In the previous experiments, we predict only those entries of $M \, or \, L$ whose corresponding sparse entries are zero. In this experiment, we use vE_H for 100 of those entries, whose corresponding sparse matrix are non-zero. There are 1000 non-zero entries in the sparse matrix and we use vE_H for 100 of those entries and run the eRPCA as mentioned in the Section 4.9. In the Figure 4.24, shows the variation of S_{nz} and λ, x-axis represents the λ values and the y-axis represents S_{nz}. The different colored lines represent non-zero entries in S_0, S_p from M_p entries of 20k, 50k, 100k, 200k. In those M_p entries that needs to be predicted, 100 entries of those M_p entries are used for prediction whose corresponding sparse entries are not zero. We observe that Figure 4.24 and 4.23(a) are very similar in the exponential curve pattern, which makes us to draw inference that, the relationship between λ and the number of non-zero entries in sparse matrix S_{nz}, doesn't depend on the entries which are either of $Type - LE$ or $Type - HE$.

4.10 Distribution of Error entries in prediction of $\boldsymbol{M} - \boldsymbol{L_p}$

The decomposition of L and S helps us to predict the entries of M using L. The interesting observation is that each of the entries has different errors, and most of the entries have very small errors and only a few of the entries have large errors. These large errors tend to dominate the overall error metric of the matrix prediction, so in this Section we analyze the distribution of the errors of each of the entries.

To explain the distribution of errors, we take the results obtained by the eRPCA algorithm using the parameters $\lambda = 10 \times \lambda_0$, $M_p = 100000$ and $\mathrm{P_{max}} = 300$ (based upon the analysis in previous Sections). The sparse matrix obtained from the decomposition of M using the eRPCA for these parameters, is highly sparse as compared to the low rank matrix L. Therefore, we expect the errors of most of the entries to be very low as we are only attempting to predicts GPEs.

Histogram of errors of all entries: In Figure 7.1, the y-axis represents the number of entries and the x-axis represents the absolute error of each of the entries, the histogram shows the error distribution of all the entries of the predicted matrix $(M - L_p)$. We observe from the graph that the maximum error of all the entries is less than 0.45 and minimum error of all the entries is zero. The mean of errors is 0.05, the standard deviation is 0.0005, and the median of the distribution is 0.04. 95% of the entries are below 0.0504 which demonstrates that the distribution is highly skewed towards lower values of errors. In addition, there are a very few entries, in fact less than 0.01 % of all entries, whose errors are in the range of 0.075 to 0.155 and 0.25 to 0.42.

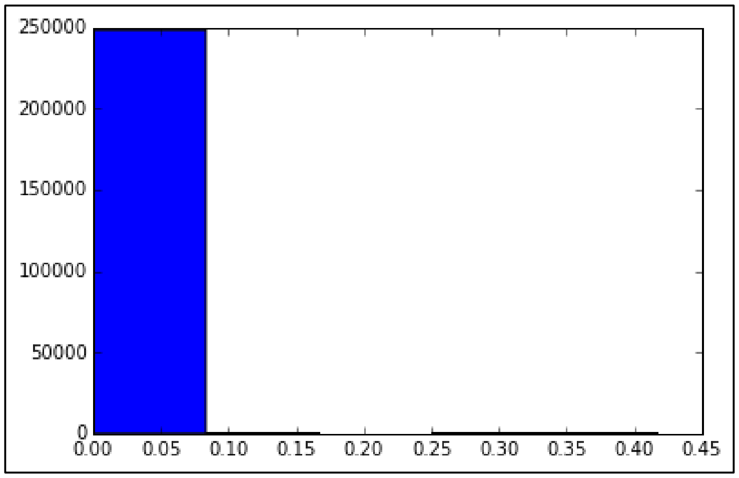


Figure 7.1: This Figure shows the plot of histogram of errors of all the entries. The x-axis represents the absolute error and the y-axis represents the number of entries.

Histogram of outlier entries (Only top ~ 2000 entries): In the Figure 7.2, the y-axis represents the number of entries and the x-axis represents the absolute error of each of the entries. It shows the error distribution of only the outlier entries (magnified view of histogram of error greater than 0.28) from Figure 7.1. As we see from the graph that all of the entries have relatively high absolute error in the range of 0.28 to 0.45. The minimum value is 0.285 and maximum value is 0.418. As we can see from the distribution shown in Figure 7.2, there are 350 highly concentrated entries in the error range from 0.365 to 0.38, and around 220 entries in the error range from 0.402 to 0.418. There are also other cohorts of 150 and 80 entries, which have different error range. We observe from Figure 7.1 and 7.2 the distribution of error entries is completely non-normal.

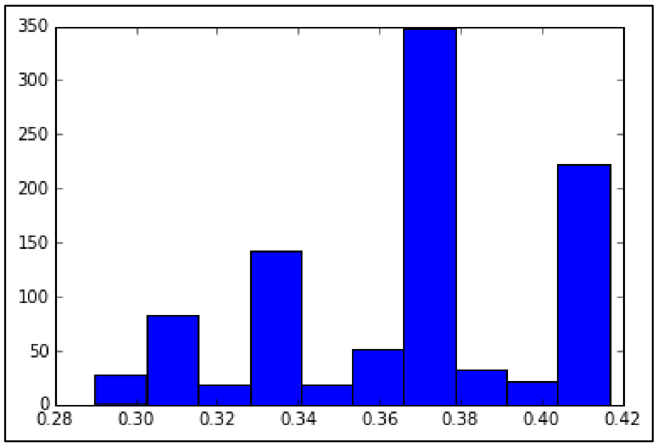


Figure 7.2: It shows the plot of histogram of top 2000 large error entries. The x-axis represents the absolute error and the y-axis represents the number of entries. These entries are difficult to predict and dominate the errors shown in many of the plots in this Section.

Histogram of non-outlier entries (excluding top ~ 2000 entries): In Figure 7.3, the y-axis represents the number of entries and the x-axis represents the absolute error of each of the entries. It shows the error distribution of non-outlier entries, we can see from the graph the errors of the entries are in the range from zero that is the minimum value and to 0.025 that is the maximum value. More than 99% entries (total ~ 250,000 entries) lie in the small error range (< 0.025).

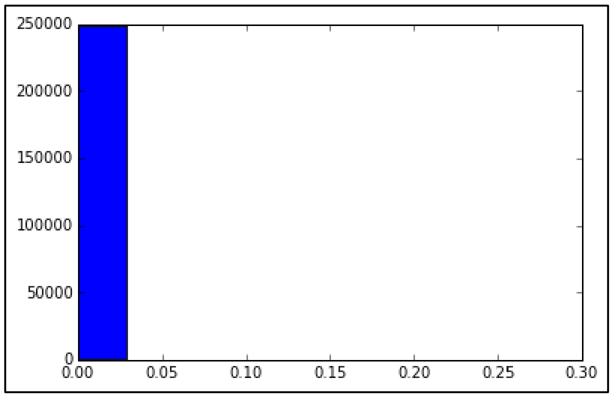


Figure 7.3: It shows the plot of histogram of error entries excluding top 2000 large error entries. The x-axis represents the absolute error and y-axis represents the number of entries.

The scenario distribution pattern of error entries that is explained above will be mostly same for any value of λ, M_p and $\mathrm{P}_{\max}$. In addition, we found that the high outlier errors are due to $Type - HE$, and most interestingly, the entries of $Type - HE$ error are mostly focused on the diagonals. In addition, there are large number of entries with small errors that is because of the $Type - LE$. As we increase the value of λ, we observe that the entries move from $Type - HE$ to $Type - LE$ which is explained in detail in Section 4.10.1.

4.10.1 Percentile to measure the entries of errors

In the previous Section, we observed that the distribution of errors is quite non-normal and has interesting structure. After analyzing the distribution of errors of all the entries, we find that each of the cohorts of entries is going to have different errors with changing λ, and we need to quantify error metric for each of such cohort of entries. In this scenario, the better way to quantify such errors is to look at the *percentiles* of the errors.

In this problem, we observe that below 99.25 percentile of the entries, most of their errors are zero and therefore the errors would start much above 99 percentile. Hence, we chose to start by looking at the 99.25 percentile of error entries and increasing the percentile gradually from there.

We look at 7 different percentile ranges, 99.25, 99.40, 99.50, 99.60, 99.75, 99.85 and 99.99 and compare the error percentiles across increasing λ values as shown in Figure 7.4 (keeping with our previous analysis we take $M_p = 100000$ and $\mathrm{P}_{\max} = 300$ as parameters used in eRPCA).

In Figure 7.4, the x-axis represents λ of ranging from 1 to 40 ($\times\, \lambda_0$) and the y-axis represents absolute error, and each of the lines represents the corresponding percentile values. The orange and gray line, which is percentile value of 99.40 and 99.5, shows that the error remains constant until for λ = 2 and suddenly we see a drastic decrease in error from λ = 2 to 2.5.

In the Figure 7.4, the blue line represents 99.25 percentile, we see that the error is very low and the error becomes completely zero after λ > 2.5. The orange represents 99.40 percentile, the error becomes zero after λ =3.5. The gray line represents 99.50 percentile, from λ = 2.5 to 5, there is a gradual decrease in error and then the error drops to zero for $\lambda \geq 5$. It tells us a story that some of these entries would move from $Type - HE$ to $Type - LE$ and be better predicted after λ=2.5 for 99.25 percentile , after λ=3.5 for 99.40 percentile, after λ=5 for 99.50 percentile.

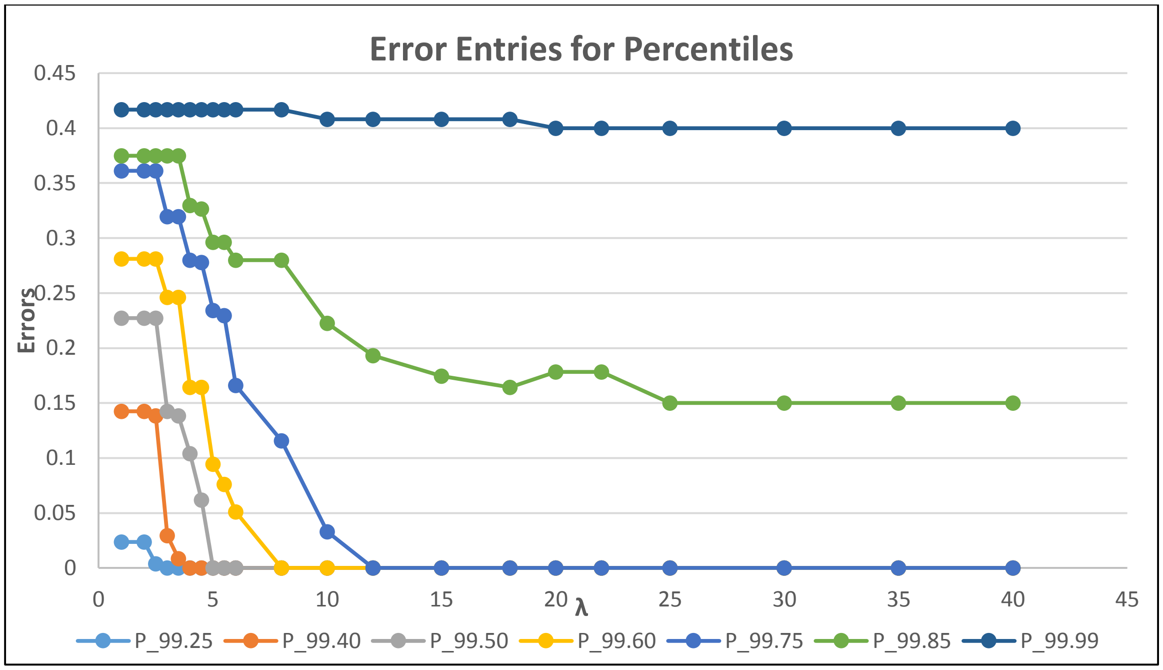


Figure 7.4: The percentile values of various error entries for λ as the x-axis (1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 8, 10, 12, 15, 18, 20, 22, 25, 30, 35, 40) and absolute errors as the y-axis. Each of the line represents the percentile value of the error entries starting from 99.25 percentile to 99.99 percentile.

The error entries for percentile 99.85 could be changed to an extent but after the value of $\lambda \geq 25$, the errors for these entries would remain completely constant. We observe that for the range of λ values from 18 to 25, the errors increases, remain constant and decrease, which is a peculiar behavior as seen in the plot.

The dark blue color represents 99.99 percentile, the errors would mostly remain constant irrespective of large change in λ value. This is because of *super anomalies* behavior of sparse values, which cannot be changed with any increasing in value of λ. All the percentile values except 99.99 and 99.85, the errors would come down to zero after λ = 11. Therefore, for the entries of this kind, the errors could not be changed with change in λ values.

This idea of super anomalies is not part of the current eRPCA or RPCA literature. A full explanation of Figure 7.4 and super anomalies would be an excellent avenue for future work. In particular, understanding such super anomalies requires a delicate analysis of the interplay of λ and vE_H. In effect, a large value of λ not only reduces the number of entries in S, but can also serve to move them into *different positions*. Understanding the precise mechanism of how such anomalies move is beyond the scope of the current work.

Chapter 5

Conclusion

5.1 Summary

The Amazon communities data in its raw form provides challenges to analyze the strength of communities connection. Firstly, we have analyzed several methods for measuring the strength of communities connection, including three definitions from the literature [12] and derived one novel definition of our own. We have used SVD to asses the better way of representing the strength of the communities connection for the eRPCA algorithm.

The second order matrix M is sparse (many of the observed entries are zero) and highly low rank which makes the current theory of the eRPCA algorithm ill suited for such data. Accordingly, we develop a method for analyzing a matrix M, which is incomplete, sparse and low rank (which the Amazon data forces on us), and this method is a novel contribution of this thesis.

We conducted experiments on balancing the entries of a second order matrix filling between the low rank and the sparse components. We understand the behavior of the low rank and the sparse matrix by changing the controlling parameter λ, where higher λ values will make putting non-zero entries into the sparse matrix more difficult. In other words, a low λ value will make L lower rank by putting more entries in S.

We have also found an empirical relationship between the number of non-zeroes in the sparse matrix and the controlling parameter λ, which would give us the ability to predict the non-zeroes in the sparse matrix for a given value of λ.

When dealing with such a second order data matrix, which is sparse, it becomes difficult to understand how to measure the errors in the various entries with usual way of measuring errors. Therefore, we layout a different way of looking into such problems, we have used the percentile values to measure the errors between the entries. Such an analysis gives a clear understanding of the errors of the entries than usual methods.

5.2 Future work

The Amazon communities data which has been analyzed using the eRPCA algorithm, has given us the direction of using various parameters from the eRPCA. However, there are still many more parameters of the eRPCA algorithm, which could be analyzed on this data. In practical settings of using Amazon communities data, we are still not aware of what those products or communities actual mean, that is we are currently dealing with real data of Amazon communities data but those products and communities are anonymous to us. The interesting problem to work is to measure the practical implication of these predictions by knowing the business context (real products and communities) and tweaking the algorithm based on the understanding of the business implications of these products and communities.

We also want to analyze the various parameters of the eRPCA algorithm on the second order matrix, which is not much sparse, as we have currently dealt with Amazon data, which is sparse and low rank. We also want to understand the super anomalies behavior in detail by varying the parameters λ and vE_H. As we have explored on the prediction of entries in this thesis, we want to build this algorithm in real time predictions of various other data sources and make it more robust.

Appendix

Changing vE values for M_p to understand the Error in prediction:

Observations	Error	Un-normalized Error
2	0.00E+00	0.00E+00
5	1.50E-04	1.50E-03
10	1.18E-04	2.35E-03
20	9.15E-05	3.66E-03
50	4.49E-05	4.49E-03
60	3.74E-05	4.48E-03
100	3.02E-05	6.04E-03
200	2.54E-05	1.02E-02
500	1.44E-05	1.44E-02
1000	5.23E-05	1.05E-01
1500	4.81E-05	1.44E-01
2000	4.41E-05	1.76E-01
5000	1.04E-03	1.04E+01
20000	4.76E-04	1.90E+01
50000	4.84E-04	4.84E+01
60000	4.22E-04	5.07E+01
70000	3.62E-04	5.07E+01
100000	2.61E-04	5.22E+01
300000	3.94E-04	2.36E+02
400000	5.83E-04	4.66E+02
500000	6.52E-04	6.52E+02
600000	7.22E-04	8.66E+02
800000	5.54E-04	8.87E+02
996000	4.45E-04	8.87E+02

Figure A.1: This figure mainly represents the x-axis values of the Figure 4.13 and the Figure 4.14, the observations in this table represents the x-axis values ordered in the same order as it appears of both the figures. The y-axis in the Figures 4.13 and Figure 4.14 represents the second and third column of this table.

Bibliography

[1] R. Paffenroth, P. d. Toit, R. Nong, L. Scharf and A. Jayasumana, "Space-time signal processing for distributed pattern detection in sensor networks," *IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING,* vol. 7, no. 1, FEBRUARY 2013.

[2] J. Leskovec and A. Krevl, *Amazon product co-purchasing network and ground-truth communities,* 2014.

[3] F. Li, *SVD and PCA,* 2008.

[4] S. Arora, "Singular Value Decomposition," in *Singular Value Decomposition and its properties*.

[5] E. J. Cand`es, X. Li, Y. Ma and J. Wright, "Robust Principal Component Analysis ?," 18 December 2009.

[6] "Internet World Stats," Miniwatts Marketing Group , January 2016. [Online]. Available: http://www.internetworldstats.com/stats.htm. [Accessed 20 January 2016].

[7] L. Danon, J. Duch, A. Diaz-Guilera and A. Arenas, "Comparing community structure and definition," *J. of Stat. Mech,* 2005.

[8] M. S and P. F, "A brief survey of machine learning methods for classification in networked data and an application to suspicion scoring," in *Statistical Network Analysis: Models, Issues, and New Directions*, vol. 4503.

[9] M. Walker, "Structured vs. Unstructured Data: The Rise of Data Anarchy," 19 December 2012. [Online]. Available: http://www.datasciencecentral.com/profiles/blogs/structured-vs-unstructured-data-the-rise-of-data-anarchy. [Accessed 20 January 2016].

[10] J. Yang, J. McAuley and J. Leskovec, "Community Detection in Networks with Node Attributes," in *IEEE International Conference On Data Mining (ICDM)*, 2013.

[11] J. Leskovec and A. Krevl, *Stanford Large Network Dataset Collection,* 2014.

[12] J. Yang and J. Leskovec, "Defining and Evaluating Network Communities based on Ground-truth," 6 November 2012. [Online]. Available: http://arxiv.org/pdf/1205.6233v3.pdf.

[13] L. E. Ghaoui, *Low-rank approximation of a matrix.*

[14] Y. Ma, *Low Rank Matrix Recovery and Completion via Convex Optimization (Introduction),* Perception and Decision Laboratory, 2012.

[15] Y. Ma, *Low Rank Matrix Recovery and Completion via Convex Optimization (Applications),* Perception and Decision Laboratory, 2012.

[16] C. Bishop, Pattern Recognition And Machine Learning, ser. Information Science and Statistics, Springer, 2006.

[17] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika,* 3 September 1936.

[18] E. J. Cand`es and B. Recht, "Exact Matrix completion via convex optimization," *Foundations of computational Mathematics,* vol. 9, no. 6, pp. 717-712, December 2009.

[19] Z. Zhou, X. Li, J. Wright, E. Cand`es and Y. Ma, "Stable Principal Component Pursuit," in *Proceedings of IEEE International Symposium on Information Technology*, 2010.

[20] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo and A. S. Willsky, "Sparse and Low-Rank Matrix Decompositions," in *IFAC Symposium on System Identification (SYSID)*, Saint-Malo, France, 2009.

[21] J. Wright, Y. Peng, Y. Ma, A. Ganesh and S. Rao, "Robust Principal Component Analysis: Exact Recovery of Corrupted Low-Rank Matrices by Convex Optimization," in *NIPS Proceedings*.

[22] E. J. Candes and Y. Plan, "Matrix completion with noise," *Proceedings of the IEEE,* vol. 98, no. 6, pp. 925-936, 2010.

[23] "Protected Repository for the Defense of Infrastructure Against Cyber Threats," [Online]. Available: https://predict.org. [Accessed December 2014].

[24] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen and Y. Ma, "Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix," *University of Illinois Urbana-Champaign, Tech. Rep. UILU ENG-09-2214,* August 2009.

[25] Y. Ren, R. Kraut and S. Kiesler, "Applying common identity and bond theory to design of online communities," in *Organization Studies*, 2007, pp. 377-408.