Low Rank Matrix Completion through Semi-definite Programming with Facial Reduction

by

Xinghang Ye

A report
presented to the University of Waterloo
in fulfilment of the
project requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Waterloo, Ontario, Canada,

I hereby declare that I am the sole author of this report. This is a true copy of the report, including any required final revisions, as accepted by my examiners.

I understand that my report may be made electronically available to the public.

Abstract

The matrix completion problem is a hot problem in data science. In recent years, many theories and practices on efficiently recovering the partially observed matrix have been introduced. A good example is the Netflix Challenge [15], where the provided data set is a partially observed matrix with each row represents a user and each column represents a film. The winning team of this competition applied matrix completion algorithm to this challenge, and improved the success rate of rating system by 10%. In this report, We will show how to convert the original low rank matrix completion problem which is non-convex to convex problem. Furthermore we will convert the convex optimization problem to a semi-definite programming problem. For the semi-definite programming problem, although Slater's condition holds, the facial reduction method can still be applied to obtain a proper face containing the optimal set and so can dramatically shrink the size of original problem while guaranteeing a low-rank solution. We include numerical tests for both the case without noise and the case with noise. In all cases the *target rank* is pre-defined.

Acknowledgements

I would like to thanks my supervisor Prof. Henry Wolkowicz and my second reader Prof. Wayne Oldford for their guidance on this report.

Dedication

This is dedicated to my parents, my brother, my sister, and my grand parents.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In this project, the **low-rank matrix completion problem** is considered, which can be expressed as follows:

Problem 1. *We are given a* ***partially observed matrix*** $Z \in \mathbb{R}^{m\times n}$. *Our goal is to find all the missing entries so that the completion has a low rank.*

This problem is strongly related to some real world applications, such as the well-known Netflix problem[15], sensor network localization [2], and system identification [14]. This problem can be relaxed using the nuclear norm, and further converted to a Semi-definite Programming(**SDP**)problem. Although for the **SDP** formulation of this problem, Slater's condition holds, we still find that its special structure at optimum can be utilized by **Facial Reduction Method**[3] through the **exposing vector approach**. Moreover, the result of **Facial Reduction** is a significant reduction in the numeber of the variables and a decrease in the rank of the solution. In the noiseless case, **Facial Reduction** always ends up with redundant constraints, which can be removed from the system through the QR approach, while if the observed entries of the matrix Z contain noise, **Facial Reduction** will end up with an overdetermined system. In order to handle the overdetermined system, we applied the **sketch matrix** method [16] to reduce the size of the overdetermined system. Further more, in order to simply the computation of this problem, we flip the problem by exchanging its objective function and constraint, using the Pareto approach as suggested by [1] [20].

1.2 Original problem

First, the following notation is introduced:

$\widehat{E} = \{\{i,j\} : Z_{i,j}\ is\ observed\}$ represents the set containing indices of observed entries of the matrix Z. $\mathcal{P}_{\widehat{E}}(\cdot)$ is the projection onto the corresponding entries in $\widehat{E}$; $b = \mathcal{P}_{\widehat{E}}(Z)$ is the vector containing all the observed entries from the matrix Z; and $\delta > 0$ measures the perturbation allowed in the observed entries of the recovered matrix. $\widehat{Z}$ is the recovered matrix.

In the noiseless case, the original low rank matrix completion problem (1) can be formulated as follows:

$$\begin{aligned} \min_{\widehat{Z}} \quad & \text{rank}(\widehat{Z}) \\ \text{s.t.} \quad & \mathcal{P}_{\widehat{E}}(\widehat{Z}) = b \end{aligned} \tag{1.1}$$

The constraints in the above problem require that $\widehat{Z}_{ij} = Z_{ij}, \forall ij\ observed\ in\ Z$

In contrast, in the noisy case, we need to introduce δ to set up the tolerance of perturbation in observed entries of the matrix $\widehat{Z}$ we recovered to get the following:

$$\begin{aligned} \min_{\widehat{Z}} \quad & \text{rank}(\widehat{Z}) \\ \text{s.t.} \quad & \|\mathcal{P}_{\widehat{E}}(\widehat{Z}) - b\| \leq \delta \end{aligned} \tag{1.2}$$

The constraint in (1.2) suggests that some perturbation is allowed but should be restricted by the pre-defined δ.

Although the above two optimization problems are easy to understand, they are very hard to compute, because the objective function for both problems is rank$(\cdot)$, which is neither convex nor continuous. Therefore, neither optimization problems (1.1) or (1.2) is a convex optimization problem. Thus, it is helpful to introduce a convex function to replace the function rank$(\cdot)$, so that the original problems can be converted to convex optimization problems. In next two sections, we will introduce how to convert the original problems to their convex surrogates and further to semi-definite programming problems which we are actually working on.

1.3 Convex Surrogates of Original Problems

In order to introduce the convex surrogates of the original problem, the following definitions and theorems are provided for convenience.

Definition 1. *Let C be a given convex set. The convex envelope of a (possibly non-convex) function $f : C \to \mathbb{R}$ is defined as the largest convex function $g(\cdot)$ such that $g(x) \leq f(x), \forall x \in C$.*

Thus, in the set C, among all the convex functions, $g(\cdot)$ is the best point-wise approximation of the function $f(\cdot)$. In particular, if $g(\cdot)$ can be conveniently described, it can serve as an approximation of $f(\cdot)$ that can be minimized efficiently.

Following the definition of convex envelop function, we provide the definition of nuclear norm and operator norm, which will be used to construct the convex surrogates for original problems.

Definition 2. *The nuclear norm of matrix X is $\|X\|_* = \sum_i \sigma_i(X)$, where $\sigma_i(X)$ represents the ith largest singular value of X.*

Definition 3. *The operator norm of matrix X is $\|X\| = \sigma_1(X)$, where $\sigma_1(X)$ represents the largest singular value of X.*

From the above definition of these two norms. It is easy to note the following inequality for any matrix X with $\text{rank}(X) = r$:

$$\|X\|_* \leq r\|X\| \tag{1.3}$$

Because for matrix X with rank r, there are r number of singular values. And because $\sigma_i \leq \sigma_1, \forall i \in \{1 \cdots r\}$. Therefore, $\|X\|_* = \sum_{i=1}^{r} \sigma_i \leq \sum_{i=1}^{r} \sigma_1 = r\|X\|$. Thus we could conclude that $\|X\|_* \leq r\|X\|$.

Finally, we present the theorem which gives the convex envelop function of function $\text{rank}(\cdot)$ in the set $\{X \in \mathbb{R}^{m\times n} : \|X\| \leq 1\}$. And we will use this theorem to show why the nuclear norm is the best convex surrogates to the function $\text{rank}(\cdot)$.

Theorem 1. *([4] Theorem 2.2) The convex envelope of $\text{rank}(X)$ on the set $\{X \in \mathbb{R}^{m\times n} : \|X\| \leq 1\}$ is the nuclear norm $\|X\|_*$.*

Theorem 1 provides the following interpretation for the rank minimization problem. Suppose $\mathcal{A}(\cdot) : \mathbb{R}^{m\times n} \to \mathbb{R}^k$ is a linear mapping, $z \in \mathbb{R}^k$, and $X_\$$ is the minimum rank

solution of $\mathcal{A}(X) = z$. The convex envelope of the function rank$(\cdot)$ on the set $C = \{X \in \mathbb{R}^{m\times n} : \|X\| \leq \|X_\$\|\}$ is $\|X\|_*/\|X_\$\|$. The reason is simple that $\forall X \in C$ we have $\|X\| \leq \|X_\$\|$. Let X_* be the minimum nuclear norm solution of $\mathcal{A}(X) = z$. Then we have:

$$\|X_*\|_*/\|X_\$\| \leq \text{rank}(X_\$) \leq \text{rank}(X_*)$$

providing an upper and lower bound on the optimal rank if the minimum rank solution $X_\$$ is known. Furthermore, the nuclear norm is the tightest lower bound among all convex lower bounds on the set C. This suggests that, in order to convert the original optimization problems to convex problems, we can approximate the objective function rank$(\cdot)$ with $\|\cdot\|_*$ [5] [6]

Proof. For the left inequality, recall the inequality provided in (1.3). Here for our problem, we have $\|X_\$\|_* \leq \text{rank}(X_\$) \times \|X_\$\|$, which is equivalent to $\|X_\$\|_*/\|X_\$\| \leq \text{rank}(X_\$)$. Because X_* is the minimum nuclear norm solution, meaning $\|X_*\|_* \leq \|X_\$\|_*$. Thus we have $\|X_*\|_*/\|X_\$\| \leq \text{rank}(X_\$)$, which is the left inequality. The right inequality is base on the fact that $X_\$$ is the minimum rank solution. Thus we have proved both the left and right inequalities. □

Therefore, by approximating rank$(\cdot)$ with $\|\cdot\|_*$, the noiseless case (1.1) can be expressed as:

$$\begin{aligned} \min_{\widehat{Z}} \quad & \|\widehat{Z}\|_* \\ \text{s.t.} \quad & \mathcal{P}_{\widehat{E}}(\widehat{Z}) = b \end{aligned} \tag{1.4}$$

Similarly, the noisy case (1.2) can be expressed as follows:

$$\begin{aligned} \min_{\widehat{Z}} \quad & \|\widehat{Z}\|_* \\ \text{s.t.} \quad & \|\mathcal{P}_{\widehat{E}}(\widehat{Z}) - b\| \leq \delta \end{aligned} \tag{1.5}$$

1.4 Semi-definite formulation

In this section, we consider how to convert the nuclear norm minimization problems in previous section to semi-definite programming problems. First we introduce some definitions and theorems that can be used in constructing equivalent semi-definite programming problems.

Theorem 2. *([8] Lemma 1) For any $X \in \mathbb{R}^{m\times n}$ and $t \in \mathbb{R}$, $\|X\|_* \leq t \iff$ there exists $A \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{n\times n}$such that*

$$Y = \begin{bmatrix} A & X \\ X^T & B \end{bmatrix} \succeq 0$$

$$\textit{and}$$
$$\operatorname{trace}(A) + \operatorname{trace}(B) \leq 2t$$

Theorem 2 suggests that introducing the matrix $Y = \begin{bmatrix} A & X \\ X^T & B \end{bmatrix}$ and minimizing the $\operatorname{trace}(\cdot)$ of Y is equivalent to minimizing the nuclear norm $\|\cdot\|_*$ of matrix X, which is embedded in the top-right block matrix of Y.

In the noiseless situation, in order to complete this matrix $\widehat{Z}$, we can apply theorem 2 to formulate the following SDP problem [8] [18].

$$\begin{aligned} \min_{\widehat{Z}} \quad & \operatorname{trace}(A) + \operatorname{trace}(B) = \operatorname{trace}(Y) \\ \text{s.t.} \quad & \\ & Y = \begin{bmatrix} A & \widehat{Z} \\ \widehat{Z}^T & B \end{bmatrix} \succeq 0 \\ & \mathcal{P}_{\widehat{E}}(\widehat{Z}) = b \end{aligned} \tag{1.6}$$

Similarly, in the noisy case, the problem can be formulated as follows:

$$\begin{aligned} \min_{\widehat{Z}} \quad & \operatorname{trace}(A) + \operatorname{trace}(B) = \operatorname{trace}(Y) \\ \text{s.t.} \quad & \\ & Y = \begin{bmatrix} A & \widehat{Z} \\ \widehat{Z}^T & B \end{bmatrix} \succeq 0 \\ & \|\mathcal{P}_{\widehat{E}}(\widehat{Z}) - b\| \leq \delta \end{aligned} \tag{1.7}$$

These two optimization problems are surrogates for the nuclear norm minimization problems where the semi-definite constraint $Y = \begin{bmatrix} A & \widehat{Z} \\ \widehat{Z}^T & B \end{bmatrix} \succeq 0$ is introduced. The semi-definite matrix Y has a special structure at optimum. Once we apply the facial reduction method through exposing vectors (will be discussed later), both optimization problems above can be simplified further.

Chapter 2

Facial Reduction and Exposing Vectors

2.1 Basic elements of convex geometry

In our semi-definite programming formulation of original problem (1.6) or (1.7), because the optimal Y is a positive semi-definite matrix, therefore it resides on a positive semi-definite cone.

For the facial reduction method, what it really do is to locate a specific face of the original convex cone of optimization problem, and shrink the size of problem from the whole convex cone to that specific face. We provide followings to define a face of a convex cone.

Definition 4. *The **relative interior** of a set contains all points which are not on the edge of the set, relative to the smallest subspace in which this set lies.*

Definition 5. *For a convex cone C, the convex subset $F \subseteq C$ is a **face** of C, denoted as $F \trianglelefteq C$, if F contains all convex combinations in C whose relative interior intersects F.*

Definition 6. *For a convex cone C, the **minimum face** containing a set $S \subseteq C$, denoted as $face(S, C)$, is the intersection of all the faces of C containing S.*

For our case, we need to apply facial reduction to locate a face by finding the vectors which expose this face. Therefore, here we first provide the definition of non-negative polar cone where exposing vectors reside and then provide the definition of exposed face and exposing vector.

Definition 7. *For the cone C of a vector space V, the* ***non-negative polar cone*** *associated with cone C is the set $\{y \in V : \langle y, x\rangle \geq 0,\ \forall x \in C\}$, which we denote as C^*.*

Definition 8. *A face F of C is an* ***exposed face*** *when there exists a vector $v \in C^*$ satisfying $F = C \cap v^\perp$. In this case we say that v exposes F, and v is called the* ***exposing vector*** *of F. The cone C is* ***facially exposed*** *when all faces of C are exposed.*

Note that for a convex cone, all of its faces are exposed.

2.2 Facial Reduction on Semi-definite matrices

The facial reduction method was first introduced by J.Borwein and H.Wolkowicz [3]. For our problem, this method works to find exposing vectors that expose the face containing the optimal Y in our semi-definite programming formulation in (1.6) or (1.7), and through the exposing vectors to locate that face. In this section, we will discuss how to apply facial reduction method in positive semi-definite matrices.

Here we denote the set of n-by-n real symmetric matrices as S^n. The inner product between two n-by-n real symmetric matrices M_1 and M_2 is defined as: $\langle M_1, M_2\rangle = \text{trace}(M_1 M_2)$. The symbols S^n_+ and S^n_{++} stand for sets of the positive semi-definite and positive definite matrices in S^n, respectively.

For a positive semi-definite matrix $M \in S^n_+$ with rank r, its eigenvalue decomposition is:

$$M = [U, V] \begin{pmatrix} \Lambda & 0 \\ 0 & 0 \end{pmatrix} [U, V]^T$$

The columns of $U \in \mathbb{R}^{n\times r}$, $V \in \mathbb{R}^{n\times(n-r)}$ are eigenvectors. Thus the minimal face containing matrix M, $F = face(M, S^n_+)$ could be expressed as:

$$F_{U,V} = \{[U, V] \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} [U, V]^T : A \in S^r_{++}\} \tag{2.1}$$

From the above definition of the exposed face, we can easily find the exposing vectors associated with that face F which are in the following set:

$$E_F = \{[U, V] \begin{pmatrix} 0 & 0 \\ 0 & B \end{pmatrix} [U, V]^T : B \in S^{(n-r)}_{++}\} \tag{2.2}$$

Through the above expression of the exposing vectors to the face $F_{U,V}$, we can easily find an exposing vector associated with the face F, which is the following:

$$F^{\Delta} = [U, V] \begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix} [U, V]^T \tag{2.3}$$

Where I_{n-r} is an identity matrix.
For $\forall M_A \in F_{U,V}$, we can get:

$$M_A = [U, V] \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} [U, V]^T \ where \ A \in S^r_{++} \tag{2.4}$$

Note that the inner product between M_A and F^{Δ} is:

$$\begin{aligned} &\langle M_A, F^{\Delta} \rangle = \text{trace}(M_A F^{\Delta}) \\ &= \text{trace}([U, V] \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} [U, V]^T [U, V] \begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix} [U, V]^T) \\ &= \text{trace}([U, V] \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix} [U, V]^T) \\ &= \text{trace}([U, V] \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} [U, V]^T) \\ &= \text{trace}(\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}) \\ &= 0 \end{aligned} \tag{2.5}$$

Thus, we have shown that F^{Δ} is indeed an exposing vector to the face $F_{U,V}$.

2.3 Facial Reduction on the face containing optimal Y

Recall that for the semi-definite programming formulation of the original problems, the optimal Y is a positive semi-definite matrix. If we want Y to be minimum rank, we could find an E such that $\langle E, Y \rangle = 0$ where by definition E is an exposing vector to the face containing Y. The eigenvalue decomposition of E is:

$$E = [\widehat{U}, \widehat{V}] \begin{pmatrix} E_d & 0 \\ 0 & 0 \end{pmatrix} [\widehat{U}, \widehat{V}]^T \tag{2.6}$$

Here, E_d is a diagonal matrix with positive eigenvalues of E on the main diagonal of it, and $\widehat{U}$ and $\widehat{V}$ are eigenvectors.

Then through the definition of exposing vector, we know that the optimal Y should now be in the face expressed as: $\widehat{V}R\widehat{V}^T$ where $R \in S_{++}^{m+n-r_E}$. Consequently, the original problem can be converted to an equivalent problem with smaller dimensions (from $m+n$ to $m+n-r_E$). From the above explanation, it is easy to notice that the larger the rank of exposing vector E, the smaller the dimension of R.

Therefore, in order to sufficiently simplify the original problem, we need to construct the exposing vector E with a high rank. In later chapters, details about this special structure and how to construct such exposing vector E will be discussed.

Chapter 3

Facial Reduction for Low-rank Matrix Completion

3.1 Special Structure at Optimum

Consider problems (1.6) or (1.7). Because there is no constraint on the main diagonal of matrix Y, we can make the main diagonal of Y to be arbitrarily large. In this situation, we have $Y \succ 0$, which means we could find positive definite matrices in the feasible region of the problems (1.6) and (1.7). Thus, Slater's condition holds, which means that there is no duality gap between the primal problem and the dual problem. This suggests that we now can only work on the primal problem without considering the dual problem, because the optimal value of primal problem equals to the optimal value of the dual problem.

Theorem 3. *([11] Corollary 3.1) Suppose $\widehat{Z}^*$ is the optimal solution corresponding to $\widehat{Z}$ for the primal problem (1.6) (1.7) with $\operatorname{rank}(\widehat{Z}^*) = r_Z$. Through the spectral decomposition of the corresponding Y, we have:*

$$0 \preceq Y = \begin{bmatrix} A & \widehat{Z}^* \\ \widehat{Z}^{*T} & B \end{bmatrix} = \begin{bmatrix} U \\ V \end{bmatrix} D \begin{bmatrix} U \\ V \end{bmatrix}^T, \ D \in S^{r_Y}_{++}, \ \operatorname{rank}(Y) = r_Y = r_Z$$

Further more, we get:

$$A = UDU^T, \ B = VDV^T, \ \widehat{Z}^* = UDV^T, \ \|\widehat{Z}^*\|_* = \frac{1}{2}\operatorname{trace}(Y) = \frac{1}{2}\operatorname{trace}(D) \tag{3.1}$$

Proof. Suppose the optimal solution we get from (1.6) or (1.7) is $\widehat{Z}^*$. Applying a singular value decomposition on $\widehat{Z}^*$, gives $\widehat{Z}^* = U_{\widehat{Z}^*}\Sigma_{\widehat{Z}^*}V_{\widehat{Z}^*}$. Set

$$D = 2\Sigma_{\widehat{Z}^*}$$
$$U = \frac{1}{\sqrt{2}}U_{\widehat{Z}^*}$$
$$V = \frac{1}{\sqrt{2}}V_{\widehat{Z}^*}$$

Consequently, we can get

$$Y = \begin{bmatrix} U \\ V \end{bmatrix} D \begin{bmatrix} U \\ V \end{bmatrix}^T$$

Suppose $[U_{\widehat{Z}^*}]_i$, $[V_{\widehat{Z}^*}]_i$ and $\begin{bmatrix} U_i \\ V_i \end{bmatrix}$ represent the ith column of $U_{\widehat{Z}^*}$, $V_{\widehat{Z}^*}$ and $\begin{bmatrix} U \\ V \end{bmatrix}$,respectively. Since each column of $U_{\widehat{Z}^*}$ and $V_{\widehat{Z}^*}$ is orthonormal, each column of $\begin{bmatrix} U \\ V \end{bmatrix}$ is orthonormal. Thus we have $\text{trace}(Y) = 2 \times \text{trace}(\Sigma_{\widehat{Z}^*}) = 2\times \parallel \widehat{Z^*} \parallel_*$ and $r_Y = r_Z = \text{rank}(D)$. Since we get optimal Y from the primal problem, and Slater's condition holds, there must exist an optimal z for the dual problem. □

3.2 Graph Representation of the Problem

In this section, we will introduce how to view the original optimization problem as a graph. This new formulation will help us to fully utilize the partially observed information of the matrix Z which could be used to construct exposing vectors.

Here, we can associate the partially observed matrix $Z \in R^{m\times n}$ with a **weighted undirected graph** $G = (V, E, W)$, having **nodes set** $V = \{1, ..., m, m+1, ..., m+n\}$ and **edge set** E, defined as:

$$E = \{ij \in V\times V : i < j \leq m\}\cup\{ij \in V\times V : m+1 \leq i < j \leq n\}\cup\{ij : Z_{i,j-m} \in \widehat{E}\} \quad (3.2)$$

and **weights** for all $ij \in E$

$$W_{ij} = \begin{cases} Z_{i(j-m)}, & \forall ij \in \bar{E} \\ 0, & \forall ij \in E\backslash\bar{E}. \end{cases}$$

where $\bar{E}$ is defined to be:

$$\bar{E} = E \setminus (\{ij \in V \times V : i \leq j \leq m\} \cup \{ij \in V \times V : m+1 \leq i \leq j \leq m+n\})$$

Note that every $ij \in \bar{E}$ corresponds to $i(j-m) \in \widehat{E}$, representing the position of an observed entry in the large matrix Y and in the partially observed matrix Z.

We can now construct the *adjacency matrix* $Ad = [Ad_{ij}]$ for the graph G:

$$Ad_{ij} = \begin{cases} 1, & if\ ij \in E \\ 0, & if\ ij \notin E. \end{cases}$$

Based on the adjacency matrix, we can apply the clique-finding algorithm in [13] to find cliques.

For this graph, the cliques $C = \{i_1, ..., i_k\} \subset \{1, ..., m\}$ and $C = \{j_1, ..., j_k\} \subset \{m+1, ..., m+n\}$ are not of interest. But what we are interested in is the cliques (possibly after row and column permutations) that correspond to a full (specified) sub-matrix X in Z.

$$|C \cap \{1, ..., m\}| = \ p > \ 0 \ , \ |C \cap \{m+1, ..., m+n\}| = \ q \ > 0$$

Consequently we can find a specified sub-matrix X in Z:

$$X \ = \{Z_{i,(j-m)} : ij \in \bar{E}, X \in \mathbb{R}^{p \times q}\}$$

Example 1. *Suppose the partially observed matrix Z is:*

$$Z = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} ? & ? & 1 & 2 & 3 & ? \\ ? & ? & ? & ? & ? & ? \\ ? & ? & 4 & 5 & 6 & ? \\ ? & ? & ? & ? & ? & ? \\ ? & ? & 7 & 8 & 9 & ? \end{pmatrix} \end{array}$$

The entries with numbers represent the observed entries in Z, while the entries with question marks represent the unobserved entries. For this case, number of rows $m = 5$ and number of columns $n = 6$. Based on this we can construct the $(m+n) \times (m+n)$ adjacency

matrix Ad associated with Z as follows :

$$Ad = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ \hline 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array} \left(\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ \hline 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right)$$

This adjacency matrix has the same dimension as matrix Y, and its top left block is where the information of Z is embedded. Consequently, we can find one node set {1,3,5,8,9,10} where all the nodes in this set are pairwise connected. Furthermore, $|\{1,3,5,8,9,10\} \cap \{1,\ldots,5\}| = |\{1,3,5\}| = 3 > 0$ *and* $|\{1,3,5,8,9,10\} \cap \{5+1,\ldots,5+6\}| = |\{8,9,10\}| = 3 > 0$*, meaning this clique is a clique that we are interested in. Based on this clique, we can find a 3-by-3 specified sub-matrix of Z, which resides on the intersections of rows 1,3,5 and columns 3,4,5 of matrix Z which is:* $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$.

3.3 Finding Exposing Vectors

Through the graph representation G introduced in previous section, every time we find a clique on the graph G, we consequently find a *specified sub-matrix* in the matrix Z. Now suppose a specified sub-matrix X is found, $X \in \mathbb{R}^{p \times q}$ with rank r_X. Although the entries of *specified sub-matrix* X might scatter around the matrix Z, we can apply row and column permutations to move to the bottom-left of Z:

$$Z = \begin{bmatrix} Z_{11} & Z_{12} \\ X & Z_{22} \end{bmatrix}$$

. The following lemma shows that we can find the X with rank $r_X = r_Z = r$.

Lemma 1. *([11] Lemma 3.2) Suppose $Z \in \mathbb{R}^{m\times n}$ is a random matrix with entries generated from a continuous random distribution, and* $\text{rank}(Z) = r$ *and we can always find specified sub-matrices $X \in \mathbb{R}^{p\times q}$ in Z with* $\min\{p, q\} \geq r$ *then* $\text{rank}(X) = r$ *with probability 1 (generically).*

Proof. Note that, because X is a specified sub-matrix of matrix Z, with $\text{rank}(Z) = r$, therefore, it is intuitive that the rank of matrix X which is $\text{rank}(X)$ is less than or equal to r. Otherwise we can find contradictions on how many linearly independent rows or columns are there in the matrix Z. Thus we know $\text{rank}(X) \leq r$. And from above definition, we have $\min\{p, q\} \geq r$, thus we could select r columns from X which is $[x^1, x^2, \cdots, x^r]$ with $x^i \in \mathbb{R}^{p\times 1}$, $i \in \{1, \cdots, r\}$. If $\text{rank}(X) < r$, it means that we can find linearly dependent relations on this r columns. This means that there exists $a_i, i \in \{1, \cdots, r\}$, such that $y = \sum_{i=1}^{r} a_i x^i = 0$. Since $y = 0$, we know that the first element of vector y which is $y_1 = 0 = \sum_{i=1}^{r} a_i x_1^i$. Because all the x_1^i are generated from continuous random distribution, then so is y_1. Thus we know that the possibility that $\text{rank}(X) < r$ is equivalent to the possibility that $y1 = 0$. The equivalence is generated from the fact that if $\text{rank}(X) < r$, then we could find linear combination to make $y = 0$. And y_1 is a linear combination of $x_1^i, i \in \{1, \cdots, r\}$, so y_1 also follows a continuous random distribution. Because the possibility that the random variable generated from a continuous random distribution equals to a certain value equals to zero. Thus we can conclude that $\mathbb{P}(rank(X) < r) = \mathbb{P}(y_1 = 0) = 0$. Thus from previous two steps, we conclude that $\text{rank}(X) = r$ with probability 1. □

Now we can rewrite the structure of the optimal Y in Theorem 3 as:

$$0 \preceq Y = \begin{bmatrix} U_1 \\ P \\ Q \\ V_1 \end{bmatrix} D \begin{bmatrix} U_1 \\ P \\ Q \\ V_1 \end{bmatrix}^T = \left[\begin{array}{c|ccc|c} U_1DU_1^T & U_1DP^T & U_1DQ^T & U_1DV_1^T \\ \hline PDU_1^T & PDP^T & PDQ^T & PDV_1^T \\ QDU_1^T & QDP^T & QDQ^T & QDV_1^T \\ \hline V_1DU_1^T & V_1DP^T & V_1DQ^T & V_1DV_1^T \end{array}\right]. \tag{3.3}$$

It is easy to see that the corresponding $Z = \begin{bmatrix} U_1DQ^T & U_1DV^T \\ PDQ^T & PDV_1^T \end{bmatrix}$, $X = PDQ^T$ and $X^T = QDP^T$. Let $\bar{P} = PD^{\frac{1}{2}}$ and $\bar{Q} = QD^{\frac{1}{2}}$. Note that $\bar{P}$ and $\bar{Q}$ have the same range with X and X^T, respectively.

$$\mathcal{R}(X) = \mathcal{R}(P) = \mathcal{R}(\bar{P}) \text{ and } \mathcal{R}(X^T) = \mathcal{R}(Q) = \mathcal{R}(\bar{Q}) \tag{3.4}$$

Note that (3.4) is of significant importance to our problem, because we can construct the exposing vectors generated from the specified matrix X for our problem by using this fact.

Through exploiting the structure in (3.3), we have $PDP^T = \bar{P}\bar{P}^T \in S_+^p$ and $QDQ^T = \bar{Q}\bar{Q}^T \in S_+^q$ with $\text{rank}(\bar{P}\bar{P}^T) = r$ and $\text{rank}(\bar{Q}\bar{Q}^T) = r$. To check whether exposing vectors associated with $\bar{P}\bar{P}^T$ and $\bar{Q}\bar{Q}^T$ exist, we simply need to check the size of X. If the number of rows of X is greater than the target rank ($p > r$), we know that exposing vectors associated with the p-by-p matrix $\bar{P}\bar{P}^T$ exist. And if the number of columns of X is greater than the target rank ($q > r$), we know that exposing vectors associated with the q-by-q matrix $\bar{Q}\bar{Q}^T$ exist.

For the specified sub-matrix $\bar{M}_P = \bar{P}\bar{P}^T$, which is a positive semi-definite matrix, with $p > r$, it is easy to see that after eigenvalue decomposition as in (2.1), we can get:

$$\bar{M}_P = [U_p, V_P] \begin{pmatrix} A_P & 0 \\ 0 & 0 \end{pmatrix} [U_P, V_P]^T : A_P \in S_{++}^r \tag{3.5}$$

Where each column of U_P and V_P is an eigenvector obtained from eigenvalue decomposition. From (2.3) we can easily find a specified exposing vector $\bar{M}_P^\Delta = V_P V_P^T$ associated with $\bar{M}_P$.

For the specified sub-matrix $\bar{M}_Q = \bar{Q}\bar{Q}^T$, which is also a positive semi-definite matrix, with $q > r$, it is easy to see that after eigenvalue decomposition as in (2.1), we can get

$$\bar{M}_Q = [U_Q, V_Q] \begin{pmatrix} A_Q & 0 \\ 0 & 0 \end{pmatrix} [U_Q, V_Q]^T : A_Q \in S_{++}^r \tag{3.6}$$

Where each column of U_Q and V_Q is an eigenvector obtained from eigenvalue decomposition. From (2.3) we can easily find a specified exposing vector $\bar{M}_Q^\Delta = V_Q V_Q^T$ associated with $\bar{M}_Q$.

We now can construct M_P^Δ and M_Q^Δ, which are exposing vectors to the face containing the optimal Y in (1.6) (1.7) by putting $\bar{M}_P^\Delta$ and $\bar{M}_Q^\Delta$ on the corresponding positions in M_P^Δ and M_Q^Δ and filling out other entries with 0's to make M_P^Δ and M_Q^Δ of the same size of Y. For example, if the clique we found from the graph representation is $\{i_1, \cdots, i_k, j_1, \cdots, j_l\}$ where $1 \le i_1 < \cdots < i_k \le m$ and $m+1 \le j_1 < \cdots < j_l \le m+n$. Then $\bar{M}_P^\Delta$ will be placed on the intersections of rows $i_1, \cdots, i_k$ and columns $i_1, \cdots, i_k$ of M_P^Δ. At the same time, $\bar{M}_Q^\Delta$ will be placed on the intersections of rows $j_1, \cdots, j_l$ and columns $j_1, \cdots, j_l$ of M_Q^Δ. Other entries of M_P^Δ and M_Q^Δ will be set to 0.

To better illustrate how to construct the exposing vectors M_P^Δ and M_Q^Δ, we present following example.

Example 2. *Suppose the partially observed matrix Z with* $\text{rank} = 2$ *is:*

$$Z = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} ? & ? & 1 & 2 & 3 & ? \\ ? & ? & ? & ? & ? & ? \\ ? & ? & 1 & 2 & 3 & ? \\ ? & ? & ? & ? & ? & ? \\ ? & ? & 4 & 5 & 6 & ? \end{pmatrix} \end{array}$$

The adjacency matrix we construct from Z is:

$$Ad = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{ccccc|cccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{array} \\ \left(\begin{array}{ccccc|cccccc} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right) \end{array}$$

Following the procedure in the 1st example, we can get a clique $\{1, 3, 5, 8, 9, 10\}$*, and further get a specified matrix* $X = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ *which lies on the intersections of rows 1,3,5 and columns 3,4,5 of matrix Z. After applying the singular value decomposition on this specified sub-matrix X, we get the following:*

$$X = \begin{bmatrix} -0.3619 & -0.6075 & -0.7071 \\ -0.3619 & -0.6075 & 0.7071 \\ -0.8591 & 0.5118 & 0.0000 \end{bmatrix} \begin{bmatrix} 10.1961 & 0 & 0 \\ 0 & 1.0192 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -0.4080 & 0.8166 & -0.4082 \\ -0.5633 & 0.1268 & 0.8165 \\ -0.7185 & -0.5631 & -0.4082 \end{bmatrix} \tag{3.7}$$

Then we get:

$$\bar{P} = \begin{bmatrix} -0.3619 & -0.6075 & -0.7071 \\ -0.3619 & -0.6075 & 0.7071 \\ -0.8591 & 0.5118 & 0.0000 \end{bmatrix} \begin{bmatrix} 10.1961 & 0 & 0 \\ 0 & 1.0192 & 0 \\ 0 & 0 & 0 \end{bmatrix}^{\frac{1}{2}} = \begin{bmatrix} -1.1556 & -0.6133 & 0 \\ -1.1556 & -0.6133 & 0 \\ -2.7432 & 0.5167 & 0 \end{bmatrix} \tag{3.8}$$

$$\bar{Q} = \begin{bmatrix} -0.4080 & 0.8166 & -0.4082 \\ -0.5633 & 0.1268 & 0.8165 \\ -0.7185 & -0.5631 & -0.4082 \end{bmatrix} \begin{bmatrix} 10.1961 & 0 & 0 \\ 0 & 1.0192 & 0 \\ 0 & 0 & 0 \end{bmatrix}^{\frac{1}{2}} = \begin{bmatrix} -1.3029 & 0.8244 & 0 \\ -1.7986 & 0.1280 & 0 \\ -2.2943 & -0.5685 & 0 \end{bmatrix} \tag{3.9}$$

Through simple matrix multiplication, we get:

$$\bar{M}_P = \bar{P}\bar{P}^T = \begin{bmatrix} -1.1556 & -0.6133 & 0 \\ -1.1556 & -0.6133 & 0 \\ -2.7432 & 0.5167 & 0 \end{bmatrix} \begin{bmatrix} -1.1556 & -0.6133 & 0 \\ -1.1556 & -0.6133 & 0 \\ -2.7432 & 0.5167 & 0 \end{bmatrix}^T = \begin{bmatrix} 1.7116 & 1.7116 & 2.8532 \\ 1.7116 & 1.7116 & 2.8532 \\ 2.8532 & 2.8532 & 7.7922 \end{bmatrix} \tag{3.10}$$

$$\bar{M}_Q = \bar{Q}\bar{Q}^T = \begin{bmatrix} -1.3029 & 0.8244 & 0 \\ -1.7986 & 0.1280 & 0 \\ -2.2943 & -0.5685 & 0 \end{bmatrix} \begin{bmatrix} -1.3029 & 0.8244 & 0 \\ -1.7986 & 0.1280 & 0 \\ -2.2943 & -0.5685 & 0 \end{bmatrix}^T = \begin{bmatrix} 2.3771 & 2.4488 & 2.5205 \\ 2.4488 & 3.2513 & 4.0537 \\ 2.5205 & 4.0537 & 5.5870 \end{bmatrix} \tag{3.11}$$

After applying an eigenvalue decomposition, we get:

$$\bar{M}_P = \begin{bmatrix} 0.7071 & 0.6075 & 0.3619 \\ -0.7071 & 0.6075 & 0.3619 \\ 0 & -0.5118 & 0.8591 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1.0192 & 0 \\ 0 & 0 & 10.1961 \end{bmatrix} \begin{bmatrix} 0.7071 & 0.6075 & 0.3619 \\ -0.7071 & 0.6075 & 0.3619 \\ 0 & -0.5118 & 0.8591 \end{bmatrix}^T \tag{3.12}$$

$$\bar{M}_Q = \begin{bmatrix} -0.4082 & -0.8166 & 0.4080 \\ 0.8165 & -0.1268 & 0.5633 \\ -0.4082 & 0.5631 & 0.7185 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1.0192 & 0 \\ 0 & 0 & 10.1961 \end{bmatrix} \begin{bmatrix} -0.4082 & -0.8166 & 0.4080 \\ 0.8165 & -0.1268 & 0.5633 \\ -0.4082 & 0.5631 & 0.7185 \end{bmatrix}^T \tag{3.13}$$

Then we get the specified exposing vectors $\bar{M}_P^{\Delta}$ and $\bar{M}_Q^{\Delta}$ associated with $\bar{M}_P$ and $\bar{M}_Q$, respectively:

$$\bar{M}_P^{\Delta} = \begin{bmatrix} 0.7071 \\ -0.7071 \\ 0 \end{bmatrix} \begin{bmatrix} 0.7071 \\ -0.7071 \\ 0 \end{bmatrix}^T = \begin{bmatrix} 0.5 & -0.5 & 0 \\ -0.5 & 0.5 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.14}$$

$$\bar{M}_Q^{\Delta} = \begin{bmatrix} -0.4082 \\ 0.8165 \\ -0.4082 \end{bmatrix} \begin{bmatrix} -0.4082 \\ 0.8165 \\ -0.4082 \end{bmatrix}^T = \begin{bmatrix} 0.1667 & -0.3333 & 0.1667 \\ -0.3333 & 0.6667 & -0.3333 \\ 0.1667 & -0.3333 & 0.1667 \end{bmatrix} \tag{3.15}$$

To construct the exposing vectors generated from the clique we found:

Step 1: Initialize two zero matrices M_P^{Δ} and M_Q^{Δ} having the same dimension of matrix Y.

Step 2: Put $\bar{M}_P^{\Delta}$ on the intersections of rows 1,3,5 and columns 1,3,5 of matrix M_P^{Δ}.

$$M_P^{\Delta} = \begin{array}{c|ccccc|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ \hline 1 & 0.5 & 0 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & -0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Step 3: Put $\bar{M}_Q^{\Delta}$ on the intersections of rows 8,9,10 and columns 8,9,10 of matrix M_Q^{Δ}

$$M_Q^{\Delta} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{ccccc|cc|ccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{array} \\ \left(\begin{array}{ccccc|cc|ccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1667 & -0.3333 & 0.1667 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.3333 & 0.6667 & -0.3333 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1667 & -0.3333 & 0.1667 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}$$

After these steps, we have successfully found two exposing vectors, namely M_P^{Δ} and M_Q^{Δ}. And M_P^{Δ} and M_Q^{Δ} are exposing vectors with the same size of Y that are associated with the two specified sub-matrix $\bar{M}_P$ and $\bar{M}_Q$ of the optimal Y.

From the definition of exposing vectors on positive semi-definite matrices, we know that the sum of exposing vectors is still an exposing vector. Therefore, we can find enough exposing vectors that are associated with specified sub-matrices of the optimal Y, and then we just need to add them up to construct a large exposing vector to efficiently expose the face containing the optimal solution. To do so, we can collect all the exposing vectors we found through the approach discussed above, and then add them up. Details of summing up all the exposing vectors are given in next section

3.4 Weights of Exposing Vectors

After finding the exposing vectors from the previous step, weights must be assigned to each exposing vector to construct the final exposing vector to expose the face containing the optimal Y. At this stage, if noise is included in the original matrix, we cannot weight each exposing vector equally, because we know that the noise is not uniformly distributed, and it may not be possible to reconstruct each sub-matrix with the same accuracy.[7]

Suppose that all the cliques we found are in the set Θ. Now, we consider a specified matrix $X_\alpha \in \mathbb{R}^{p_\alpha \times q_\alpha}$ generated from a clique α in the set Θ. Through what has been discussed, using singular value decomposition, we can get $X_\alpha = \bar{P}_\alpha \bar{Q}_\alpha^T$. Then it is easy to get $\bar{M}_\alpha^P = \bar{P}_\alpha \bar{P}_\alpha^T \in \mathbb{R}^{p_\alpha \times p_\alpha}$ and $\bar{M}_\alpha^Q = \bar{Q}_\alpha \bar{Q}_\alpha^T \in \mathbb{R}^{q_\alpha \times q_\alpha}$.

In the noiseless case, if the *target rank* of the partially observed matrix Z is r, and $p_\alpha \geq r$ or $q_\alpha \geq r$, we will always get $\text{rank}(\bar{M}_\alpha^P) = r$ or $\text{rank}(\bar{M}_\alpha^Q) = r$, respectively.

But in noisy case, because we know that for sub-matrices of a large matrix, the rank of the sub-matrices must be bounded up by the rank of the large matrix otherwise we can find contradiction on how many linear independent rows/columns are there in the large matrix. For our case, because we know the target rank is r, thus if $\text{rank}(\bar{M}_\alpha^P) > r$ or $\text{rank}(\bar{M}_\alpha^Q) > r$, it is intuitive that this is the effect of noise.

In order to measure how noisy a corresponding exposing vector is, we introduce the **Eckart-Young distance** of matrix M which defines the distance between matrix M and the closest matrix with rank r and the same dimension on the semi-definite cone. If there is no noise, then (3.16) and (3.17) would result in 0. Through (3.18) and (3.19) we know that all the exposing vectors will have the same weight 1. However, if there is noise in $\bar{M}_\alpha^P$ and $\bar{M}_\alpha^Q$, the larger the noise, the larger the results of (3.16) and (3.17), therefore the weights associated of the corresponding exposing vectors (3.18) (3.19) will be smaller.

For $\bar{M}_\alpha^P$ the clique weight $v(\bar{M}_\alpha^P)$ is calculated as follows:

$$v(\bar{M}_\alpha^P) = \frac{\sum_{j=1}^{p_\alpha - r} \lambda_j^2(M\alpha^P) + \sum_{j=p_\alpha - r+1}^{p_\alpha} (\min(0, \lambda_j(\bar{M}_\alpha^P)))^2}{0.5 p_\alpha (p_\alpha - 1)} \tag{3.16}$$

For $\bar{M}_\alpha^P$ the clique weight $v(\bar{M}_\alpha^Q)$ is determined by:

$$v(\bar{M}_\alpha^Q) = \frac{\sum_{j=1}^{q_\alpha - r} \lambda_j^2(\bar{M}_\alpha^Q) + \sum_{j=q_\alpha - r+1}^{q_\alpha} (\min(0, \lambda_j(\bar{M}_\alpha^Q)))^2}{0.5 q_\alpha (q_\alpha - 1)} \tag{3.17}$$

Here, $\lambda_j(\bar{M}_\alpha^P)$ and $\lambda_j(\bar{M}_\alpha^Q)$ refer to the j'th smallest eigenvalue of the matrices $\bar{M}_\alpha^P$ and $\bar{M}_\alpha^Q$, respectively. p_α and q_α represent the dimension of $\bar{M}_\alpha^P$ and $\bar{M}_\alpha^Q$, respectively. The value of $v(\bar{M}_\alpha^P)$ and $v(\bar{M}_\alpha^Q)$ measure how noisy $\bar{M}_\alpha^P$ and $\bar{M}_\alpha^Q$ are. In the case without noise, we will always have $v(\bar{M}_\alpha^P) = v(\bar{M}_\alpha^Q) = 0$.

For the exposing vector $(\bar{M}_\alpha^P)^\Delta$, the weight associated ω is determined by:

$$\omega((\bar{M}_\alpha^P)^\Delta) = 1 - \frac{v(\bar{M}_\alpha^P)}{\textit{sum of all existing clique weights}} \tag{3.18}$$

For the exposing vector $(\bar{M}_\alpha^Q)^\Delta$, the weight associated ω is determined by:

$$\omega((\bar{M}_\alpha^Q)^\Delta) = 1 - \frac{v(\bar{M}_\alpha^Q)}{\textit{sum of all existing clique weights}} \tag{3.19}$$

3.5 Final Exposing Matrix

Now, we have collected all the exposing vectors and the weights associated with them. Then, simply adding up all the weighted exposing vectors, gives the final exposing vector E_{final}:

$$E_{final} = \sum_{\alpha \in \Theta} \omega((\bar{M}_\alpha^P)^\Delta)(M_\alpha^P)^\Delta + \omega((\bar{M}_\alpha^Q)^\Delta)(M_\alpha^Q)^\Delta \tag{3.20}$$

Recall that $(M_\alpha^P)^\Delta$ and $(M_\alpha^Q)^\Delta$ are the exposing vectors generated from $(\bar{M}_\alpha^P)^\Delta$ and $(\bar{M}_\alpha^Q)^\Delta$ by filling out 0's in non-corresponding entries of $(\bar{M}_\alpha^P)^\Delta$ and $(\bar{M}_\alpha^Q)^\Delta$.

3.6 Algorithm for Finding large exposing vector

In this section, we conclude how the actual algorithm is applied to construct the final exposing vector E_{final}.

Data: a partially observed matrix $Z \in \mathbb{R}^{m\times n}$, target rank: r_{target}, max clique size:$size$

Result: A final exposing vector E_{final} that exposes a face containing the optimal solution of $Y \in S_+^{m+n}$

1. form the corresponding adjacency matrix A
2. find a set Θ from A contains all the cliques of proper size in the range $\{r_{target} \times (r_{target}+1),\ size\ \}$

for *each clique* $\alpha \in \Theta$ **do**
- Get the corresponding specified sub-matrix X_α;
- $[p_\alpha, q_\alpha] \leftarrow \text{size}(X_\alpha)$;
- $[\bar{P}, \bar{Q}] \leftarrow \text{fullrankdecomposition}(X_\alpha)$;
- **if** $p_\alpha > r_{target}$ **then**
  - $\bar{M}_\alpha^P = \bar{P}\bar{P}^T$
  - calculate $v(\bar{M}_\alpha^P)$ through (3.16)
- **end**
- **if** $q_\alpha > r_{target}$ **then**
  - $\bar{M}_\alpha^Q = \bar{Q}\bar{Q}^T$
  - calculate $v(\bar{M}_\alpha^P)$ through (3.17)
- **end**

end

calculate all the exposing vectors $(M_\alpha^P)^\Delta$ and $(M_\alpha^Q)^\Delta$.

calculate all the exposing vector weights $\omega((\bar{M}_\alpha^P)^\Delta)$ and $\omega((\bar{M}_\alpha^Q)^\Delta)$ for every $\alpha \in \Theta$ as described in (3.18) and (3.19)

sum over all existing weights using exposing vectors $(M_\alpha^P)^\Delta$ and $(M_\alpha^Q)^\Delta$associated with $\alpha \in \Theta$

$E_{final} \leftarrow \sum_{\alpha\in\Theta} \omega((\bar{M}_\alpha^P)^\Delta)(M_\alpha^P)^\Delta + \omega((\bar{M}_\alpha^Q)^\Delta)(M_\alpha^Q)^\Delta$

return E_{final};

Algorithm 1: Finding final exposing vector E_{final}

Chapter 4

Dimension Reduction and Local Refinement

4.1 Dimension Reduction

After constructing the final exposing vector E_{final}, we can apply eigenvalue decomposition on it to get:

$$E_{final} = \begin{bmatrix} \widehat{U} & \widehat{V} \end{bmatrix} \begin{bmatrix} R_0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \widehat{U} & \widehat{V} \end{bmatrix} \tag{4.1}$$

Where R_0 is a diagonal matrix with the positive eigenvalues of E_{final} on the main diagonal of it. $\widehat{U}$ and $\widehat{V}$ are the eigenvectors. Based on the number of non-zero eigenvalues found, through (2.2) we can easily find the optimal Y_{final} for our semi-definite programming formulation in (1.6) or (1.7) resides on the null space of E_{final} which can be expressed as $Y_{final} = \widehat{V} R \widehat{V}^T$, where $r_{R_0} + r_R = m + n$. Because $r_{R_0} > 0$, therefore R is a positive definite matrix with smaller dimensions compared to the matrix Y_{final}. Thus, we have successfully reduced the original problem to a smaller face of the original semi-definite cone. Also note that, since the column vectors in $\widehat{V}$ are orthonormal, we have $\text{trace}(Y_{final}) = \text{trace}(R)$.

4.2 Noiseless Case

Based on smaller dimension matrix R, the optimization problem in the noiseless case can be expressed as:

$$\begin{array}{ll} \min_R & \text{trace}(R) \\ \text{s.t.} & \mathcal{P}_{\bar{E}}(\widehat{V}R\widehat{V}^T) = b \\ & R \succeq 0, \end{array} \tag{4.2}$$

The constraints in (4.2) are exactly low rank constraints of the following type:

$$e_i^T \widehat{V} R \widehat{V}^T e_j = b_{ij}, \ \forall ij \in \bar{E} \tag{4.3}$$

where $\bar{E} \subset E$ is an index set containing the linearly independent constraints. These constraints can be further written as rank two constraints though the cyclic permutation property of trace$(\cdot)$:

$$\begin{aligned} b_{ij} &= e_i^T \widehat{V} R \widehat{V}^T e_j \\ &= \text{trace}(e_i^T \widehat{V} R \widehat{V}^T e_j) \\ &= \tfrac{1}{2} \text{trace}((\widehat{V}_{j,:}^T \widehat{V}_{i,:} + \widehat{V}_{i,:}^T \widehat{V}_{j,:})R) \\ &= \tfrac{1}{2} \langle (\widehat{V}_{j,:}^T \widehat{V}_{i,:} + \widehat{V}_{i,:}^T \widehat{V}_{j,:}), R \rangle \end{aligned} \tag{4.4}$$

Definition 9. *(e.g see [17]) A matrix $M \in S^n$ can be identified linear isometrically as a vector in $\mathbb{R}^{\frac{n\times(n+1)}{2}}$ through the following vectorization operator:*

$$svec(M) = [M_{11}, \sqrt{2}M_{12}, M_{22}, \cdots, \sqrt{2}M_{1n}, \cdots, \sqrt{2}M_{n-1,n}, M_{nn}]^T$$

Example 3. *Suppose we have two symmetric matrices. The first one is* $L_1 = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}$. *The second one is* $L_2 = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$. *Thus through above definition,* $svec(L_1) = \begin{bmatrix} 1 \\ 2\sqrt{2} \\ 1 \\ 2\sqrt{2} \\ 2\sqrt{2} \\ 1 \end{bmatrix}$ *and* $svec(L_2) = \begin{bmatrix} 2 \\ \sqrt{2} \\ 2 \\ \sqrt{2} \\ \sqrt{2} \\ 2 \end{bmatrix}$. *Thus it is easy to see that* $svec(L_1)^T svec(L_2) = \langle L_1, L_2 \rangle = 18$.

Note that, if M_s and N_s are both symmetric matrices, then through the definition above, it is easy to see that $\langle M_s, N_s \rangle = svec(M_s)^T svec(N_s)$. For our case, we know $T_{ij} = \frac{1}{2}(\widehat{V}_{j,:}^T \widehat{V}_{i,:} + \widehat{V}_{i,:}^T \widehat{V}_{j,:})$ and R are both symmetric matrices; therefore, based on the property of the *inner product* and equation (4.4), we can convert both $T_{ij} = \frac{1}{2}(\widehat{V}_{j,:}^T \widehat{V}_{i,:} + \widehat{V}_{i,:}^T \widehat{V}_{j,:})$ and R to long vectors $svec(T_{ij})$ and $svec(R)$, respectively. The positive semi-definite constraint in (4.4) may now be rewritten as:

$$svec(T_{ij})^T svec(R) = b_{ij} \tag{4.5}$$

By constructing a large matrix $\widehat{M}$ whose rows are $sevc(T_{ij}), \forall ij \in \bar{E}$ we have converted the original problem with its semi-definite constraints into a new problem with linear constraints:

$$\begin{array}{ll} \min_R & \text{trace}(R) \\ s.t. & \widehat{M} svec(R) = b \\ & R \succeq 0 \end{array} \tag{4.6}$$

Note that the number of columns of $\widehat{M}$ equals the length of vector $svec(R)$. The reason is that, each row of matrix $\widehat{M}$ is actually a $svec(T_{ij}) \forall (i,j) \in \bar{E}$. The dimension of T_{ij} equals to the dimension of R, thus the length of $svec(T_{ij})$ equals to the length of $svec(R)$.

It is worth mentioning that the matrix $\widehat{M}$ will have linearly dependent constraints. To obtain a set of linearly independent constraints, we need to apply QR factorization to drop those dependent constraints. This approach takes longer time than the noisy case. After dropping out redundant rows of $\widehat{M}$ and b, we will get $\tilde{M}$ and $\tilde{b}$. Thus for the noiseless case, the final optimization problem we are going to solve is:

$$\begin{array}{ll} \min_R & \text{trace}(R) \\ s.t. & \tilde{M} svec(R) = \tilde{b} \\ & R \succeq 0 \end{array} \tag{4.7}$$

For the final formulation of noiseless problem (4.7), we have successfully reduced the number of constraints of the original problem. And with fewer constraints, we can directly solve (4.7) using standard semi-definite programming packages like CVX[10].

4.3 Noisy Case

For the noisy case, we investigate the performance of the Pareto Search Strategy as suggested by [7]. This strategy was originated in portfolio optimization, where the constraints are far more complicated than the objective function[1] [20]. The idea is simple: we only need to exchange the objective function and the difficult constraint, and then use this easier flipped problem to solve the origin.

First, we present the resulting optimization problem associated with the noisy case after dimension reduction:

$$\begin{array}{ll} \min_R & \text{trace}(R) \\ s.t. & ||\mathcal{P}_{\bar{E}}(\widehat{V}R\widehat{V}^T) - b|| \leq \delta \\ & R \succeq 0 \end{array} \tag{4.8}$$

The objective function trace(R) is easy to compute by summing up all the main diagonal elements. However, computing the constraint $||\mathcal{P}_{\bar{E}}(\widehat{V}R\widehat{V}^T) - b|| \leq \delta$ is much harder. Therefore, it is intuitive to apply Pareto search strategy to flip the problem and get the following problem with easier constraint:

$$\begin{array}{lll} \varphi(\tau) := \min_R & \|\mathcal{P}_{\bar{E}}\left(\hat{V}R\hat{V}^T\right) - b\| \\ \text{s.t.} & \text{trace}(R) = \tau \\ & R \succeq 0. \end{array} \tag{4.9}$$

Now the objective function is convex and the feasible region is very simple. We simply need the smallest τ to make the objective function $\varphi(\tau) \leq \delta$.

For our problem, we can even make the flipped problem simpler. Firstly, we solve the following problem:

$$\begin{array}{lll} \varphi(\tau_0) := \min_R & \|\mathcal{P}_{\bar{E}}\left(\hat{V}R\hat{V}^T\right) - b\| \\ \text{s.t.} & R \succeq 0. \end{array} \tag{4.10}$$

Using the solution of (4.10), we get τ_0 and δ_0. where δ_0 is the minimum residual of the optimization, and τ_0 is the trace associated with it. Therefore, we know that it is impossible to get a smaller residual than δ_0 in the optimization problem (4.9), because we cannot get a solution with constraint on the trace better than the problem without such a constraint.

We can then solve (4.9) iteratively, that is, in each iteration of the algorithm we keep reducing the τ_0 value we get from the first step, until we get a $\widehat{Z}$ with the target rank or meet the exit criteria. We then exit from the loop, and can use the $\widehat{Z}$ from last iteration as the recovered matrix.

In order to illustrate the idea, we present the Figure (4.1). On this image, point P represents the solution from the problem (4.10). Note that the x-axis represents the trace value, therefore if we keep reducing the τ_0, the solution from each iteration will keep moving left, until we get to the point Q that we is the solution with a pre-defined target rank or we meet the exit criteria.

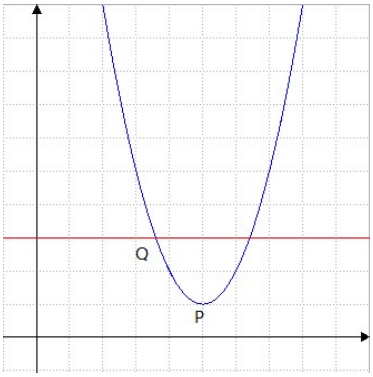

Figure 4.1: Pareto illustration

Both optimization problems presented in this chapter can be easily solved by CVX [10][9] with the embedded SDPT3 package [12][17].

For the $\mathcal{P}_{\bar{E}}(\widehat{V}R\widehat{V}^T)$ part in the objective function, we can construct a large matrix $\widehat{M}$ following what is done in the noiseless case (4.6). Consequently, all the semi-definite constraints can be formed as rows of a large matrix $\widehat{M}$, and the matrix R is converted into the long vector $svec(R)$. In addition, we add a regularization term, similar to a ridge regression type problem e.g. see [19]. The reason for applying regularization is simply that there appear to be some correlated columns in the big matrix $\widehat{M}$. If we do not apply the regularization to impose constraints on the size of the parameters, the resulting $svec(R)$ of the optimization problem will have high variance and be poorly determined. A large positive coefficient can be cancelled by a large negative coefficient. And with regularization, the problem is alleviated. The optimization problems we work on with regularizations are the following:

$$\begin{aligned} \varphi(\tau_0) := \min_R \quad & \|\widehat{M} svec(R) - b\| + \gamma \parallel R \parallel_F \\ \text{s.t.} \quad & R \succeq 0. \end{aligned} \tag{4.11}$$

and

$$\begin{aligned} \varphi(\tau) := \min_R \quad & \|\widehat{M} svec(R) - b\| + \gamma \parallel R \parallel_F \\ \text{s.t.} \quad & \text{trace}(R) = \tau \\ & R \succeq 0. \end{aligned} \tag{4.12}$$

where $\gamma > 0$ and very small

Moreover, the number of rows of $\widehat{M}$ equals the number of the observed entries in Z ($|\widehat{E}|$), and the number of columns equals the length of vector $svec(R)$. For a matrix $N \in S^{s\times s}$, the length of $svec(N)$ equals $t(s) = \frac{s(s+1)}{2}$, which is the number of upper triangular entries and diagonal entries of matrix N. For our case, We know that, after finding enough cliques and applying dimension reduction, generally we can get $R \in S^s_{++}$ with s very close or even equal to r, which is the rank of matrix Z. Note that r is a low rank, meaning it is much smaller than $|\widehat{E}|$. Consequently, the system presented above (4.12) is highly over determined. In order to alleviate this situation, we introduce the sketch matrix method as suggested in [16]. A sketch matrix is a random matrix with proper size. For our case, we set a sketch matrix $\bar{M} \in \mathbb{R}^{2t(s)\times|\widehat{E}|}$. we then multiply on the left of both $\widehat{M}$ and b, and get: $\bar{M}\widehat{M} = \mathbb{R}^{2t(s)\times t(s)}$ and $\bar{M}b \in \mathbb{R}^{2t(s)\times 1}$. Thus the number of constraints of the above system can be shrunk significantly from $|\widehat{E}|$ to $2t(s)$.

Thus, the two optimization problems that we actually applied in the noisy case are as follows:

$$\begin{aligned} \varphi(\tau_0) := \min_R \quad & \|\bar{M}\widehat{M} svec(R) - \bar{M}b\| + \gamma \parallel R \parallel_F \\ \text{s.t.} \quad & R \succeq 0. \end{aligned} \tag{4.13}$$

$$\begin{aligned} \varphi(\tau) := \min_R \quad & \|\bar{M}\widehat{M} svec(R) - \bar{M}b\| + \gamma \parallel R \parallel_F \\ \text{s.t.} \quad & \text{trace}(R) = \tau \\ & R \succeq 0. \end{aligned} \tag{4.14}$$

Now we present the actual algorithm we used to solve the flipped problem

Data: null space basis vectors: $\widehat{V}$, target rank: r_{target}, limit of iterations: I
Result: $\text{rank}(\widehat{Z}) = r_{target}$
solve the problem (4.14) through CVX,and get the initial value of τ_0 and δ_0
set $\tau = \tau_0$ and $\delta = \delta_0$
while $\text{rank}(\widehat{Z}) > r_{target}$ *and* $k \leq I$ **do**
    reduce τ by 5 percent:

$$\tau \leftarrow 0.95 * \tau$$

    Then solve the problem:(4.14) through CVX, and get δ_1 and R_1

$$\delta \leftarrow \delta_1$$
$$\bar{R} \leftarrow R_1$$

    Increment the iterate:

$$k \leftarrow k + 1$$

end
return the $\widehat{Z}$ part in $\widehat{V}\bar{R}\widehat{V}^T$

Algorithm 2: Local Refinement Process

Chapter 5

Numerical Results

5.1 Data Generation

The Table 5.1 shows the results of matrix recovery with target rank equals 2 without noise. While Table 5.2 represents the results of the noisy matrix recovery with target rank equals to 2. The Table 5.3 shows the results of the matrix recovery with target rank equals to 3 without noise. While Table 5.4 shows the results of the matrix recovery with target rank equals to 3 with noise.

We generate the instance $Z = Z_L Z_R^T$, where $Z_L \in \mathbb{R}^{m\times r}$ and $Z_R \in \mathbb{R}^{n\times r}$, r is the target rank for the recovery algorithm. Both Z_L and Z_R are generated from a standard normal distribution N(0,1). For the noisy case, we generate noise on the matrix through the approach below:

$$Z_{ij} \leftarrow Z_{ij} + \sigma\epsilon_t \parallel Z \parallel_\infty, \forall ij \in \bar{E},$$

where ϵ_t is also generated from a standard normal distribution. We also noted that $\|Z\|_\infty$ is about 5.

The test result is generated from the *Relative Residual*, which is defined as:

$$RResidual = \frac{\parallel \widehat{Z} - Z \parallel_F}{\parallel Z \parallel_F},$$

where $\widehat{Z}$ is the matrix that recovered from out algorithm.

For each row of the following tables, we have run 5 tests and take their average value to present.

5.2 Test Results

Here, we use the Matlab code to generate the numerical results. The main algorithm is embedded in the file **completZ.m**. The codes to generate test cases are in the files with names starting with **table** and the file **run_tests.m** will generate test results in table format. In each of the files with names start with **table**, **mm** and **nn** represents the number of rows and columns of the matrix to be recovered, respectively. **NF** represents the noise level, and **PP** represents the density of the observed entries in the matrix to be recovered.

All of the tests were run on MATLAB 2015a(32-bit version) running on a DELL Inspiron 14 5000 series with Windows10, 16G RAM and Intel(R) Core(TM) i7-4510U CPU @ 2.60GHz.

For the following tables, r represents the target rank, m represents the number of rows of the recovery matrix, and n represents the number of columns of the recovery matrix. The proportion of entries of matrix Z that are observed are represented by p. For the noisy case, the σ represents the noise level.

For the noiseless case, we present the results of recovering matrices with rank equal 2 and 3, respectively. It can be seen from the results that we almost perfectly recovered the matrix Z. However, the running time of this case is much longer because of the QR approach which is applied to drop off the redundant constraints in original problem.

We also present the results of rank 2 and rank 3 matrices recovery with noise. It can be seen that even there exists noise, the result is still promising. The **initial time** refers to the time spend before local refinement, and **refine time** stands for the time spent in the whole process, including the local refinement step. The **initial rank** stands for the rank we get before the local refinement, and the **refine rank** stands for the rank we get after the local refinement. It can be seen from these tables that in most of the time, we can get an acceptable result without any local refinement.

It can be seen from the following 4 tables that even though sometimes facial reduction method might fail to get the target rank value we set beforehand, it can still produce promising results. As can be seen, for all the test cases, facial reduction method always gives low rank solutions with very small relative residuals. This suggests that facial reduction method can be applied to low-rank matrix completion problem and can generate good results.

Table 5.1: no noise: $r = 2$; $m \times n$ size; density p.

Specifications			Time (s)	Rank	RResidual ($\%Z$)
m	n	p			
700	1000	0.40	34.13	2.00	4.079e-14
1000	1400	0.40	55.27	2.00	3.0711e-13
1400	2000	0.40	70.70	2.00	8.0114e-14

Table 5.2: noisy: $r = 2$; $m \times n$ size $\uparrow$; density $p \downarrow$; noise $\uparrow$.

Specifications				Time (s)		Rank		RResidual ($\%Z$)	
m	n	σ	p	initial	total	initial	refine	initial	refine
700	1000	0.0000	0.40	18.38	18.38	2.00	2.00	2.788e-14	2.788e-14
700	1000	0.0010	0.40	18.53	35.49	2.20	2.00	6.127e-01	6.238e-01
700	1000	0.0015	0.40	16.59	24.35	3.20	3.00	7.627e-01	7.737e-01
700	1000	0.0030	0.40	17.47	31.18	3.00	3.00	5.643e-01	5.913e-01
700	1000	0.0040	0.40	22.06	29.99	3.20	3.00	5.881e-01	5.881e-01

Table 5.3: no noise: $r = 3$; $m \times n$ size; density p.

Specifications			Time (s)	Rank	RResidual ($\%Z$)
m	n	p			
700	1000	0.40	36.93	3.00	4.079e-14
1000	1400	0.40	67.72	3.00	3.3711e-13
1400	2000	0.40	76.70	3.00	7.7184e-14

Table 5.4: noisy: $r = 3$; $m \times n$ size $\uparrow$; density $p \downarrow$; noise $\uparrow$.

Specifications				Time (s)		Rank		RResidual ($\%Z$)	
m	n	σ	p	initial	total	initial	refine	initial	refine
400	700	0.0000	0.40	8.92	8.92	3.00	3.00	1.010e-14	1.010e-14
700	1000	0.0001	0.40	20.99	20.99	1.00	1.00	9.889e-01	9.889e-01
700	1000	0.0010	0.40	31.29	72.54	4.00	4.00	8.827e-01	9.238e-01
700	1000	0.0015	0.40	37.31	37.31	2.00	2.00	9.709e-01	9.709e-01
700	1000	0.0020	0.40	16.60	16.60	3.00	3.00	9.206e-01	9.206e-01
700	1000	0.0000	0.40	12.90	12.90	3.00	3.00	1.436e-14	1.436e-14
700	1000	0.0001	0.45	37.45	37.45	3.00	3.00	9.373e-01	9.373e-01
700	1000	0.0015	0.50	16.76	48.41	4.00	4.00	9.087e-01	9.316e-01
700	1000	0.0030	0.55	38.04	38.04	2.00	2.00	9.351e-01	9.351e-01
700	1000	0.0045	0.60	45.44	113.57	4.00	4.00	9.108e-01	9.371e-01

Chapter 6

Conclusion

In this report, we present how facial reduction with the exposing vectors approach can be applied to pre-process a low-rank matrix completion problem together with a nuclear norm heuristic and positive semi-definite programming.

Instead of directly solving large scale problems, such as the matrix with more than a million entries, we can fully utilize the high degenerate structure of the optimal solution. By finding enough exposing vectors associated with the "cliques" and summing them up to get a large exposing vector that can locate the minimum face containing the optimal solution, we successively convert the original problem into an equivalent problem with much smaller dimensions.

Although it is not guaranteed that we can always find the minimal face containing the optimal solution, even in this situation, we can still reduce the dimension of the original problem significantly such that the final optimal solution is partially localized. This suggests that by applying the exposing vector approach of facial reduction, large dimension problems can be efficiently converted to equivalent problems with much smaller dimensions.

Solving large scale low-rank minimization problems consequently becomes much more efficient.

Index

References

[1] Van Den Berg.E and M.P. Friedlander. Spgl1: A solver for large-scale sparse reconstruction. *http://www.cs.ubc.ca/labs/scl/spgl1*, 2007.

[2] P. Biswas, T. Liang, K. Toh, K. Wang, and Y. Ye. Semidefinite programming approaches for sensor network localization with noisy distance measurement. *IEEE Tran. Autom. Sci. Eng*, 3:360–371, 2006.

[3] J.M. Borwein and H. Wolkowicz. Facial reduction for a cone-convex programming problem. *J. Austral. Math. Soc. Ser. A*, 30(3):369–380, 1980/81.

[4] B.Recht, M.Fazel, and P.Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Rev.*, 52(3):471–501, 2010.

[5] E.J Candes and B. Rechet. Exact matrix completion via convex optimization. *Found. Comput. Math*, 9:717–772, 2008.

[6] E.J. Candes and Tao.T. The power of convex relaxation: Near-optimal matrix completion. *IEEE Tran. Inform. Theory*, 56:2053–2080, 2010.

[7] D.Drusvyatskiy, N.Krislock, Y.-L.Voronin, and H.Wolkowicz. Noisy euclidean distance realization:robust facial reduction and the pareto frontier. *American Mathematics Society*, 2015.

[8] Maryam Fazel, Haitham Hindi, and Stephen P.Boyd. A rank minimization heuristic with application to minimum order system approximation. *American Contral Conference*, 6, 2001.

[9] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. http://stanford.edu/~boyd/graph_dcp.html.

[10] M. Grant, S. Boyd, and Y. Ye. Disciplined convex programming. In *Global optimization*, volume 84 of *Nonconvex Optim. Appl.*, pages 155–210. Springer, New York, 2006.

[11] Shimeng Huang and Henry Wolkowicz. Low-rank matrix completion using nuclear norm with facial reduction. *http://www.math.uwaterloo.ca/ hwolkowi//henry/reports/facredpsdcompl.pdf*, 2016.

[12] M.J. Todd K.C. Toh and R.H. Tutuncu. Sdpt3 — a matlab software package for semidefinite programming, 1999.

[13] Nathan Krislock and Henry Wolkowicz. Explicit sensor network localization using semidefinite representations and facial reduction. *SIAM J.OPTIM*, 20 No.5:2679–2708.

[14] Z. Liu and L. Vandenberghe. Interior point method for nuclear norm approximation with application to system identification. *SIAM J. Matrix Anal, A*, 31:1235–1256, 2009.

[15] Netflix. Netflix problem. *http://www.netflixprize.com*, 2006.

[16] M. Pilanci and M.J. Wainwright. Randomized sketches of convex programs with sharp guarantees. *IEEE Trans. Info. Theory*, 61(9):5096–5115, 2015.

[17] K.C. Toh R.H Tutuncu and M.J. Todd. Solving semidefinite-quadratic-linear programs using sdpt3. *Mathematical Programming Ser. B*, 95:189–217, 2003.

[18] N. Srebro and R.R. Salakhutdinov. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. *Advances in Neural Information Processing Systems*, 2010.

[19] R Tibshirani T Hastie and J Friedman. *The Elements of Statistical Learning Learning*. Springer series in Statistics, 2003.

[20] E. Van Den Berg and M.P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM J.sci.Comput*, 31, 2008.