Systems Research Institute
Polish Academy of Sciences

Efficient matrix completion for data recovery in data-driven IT applications

Antonina Krajewska

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

Supervised by
Professor Ewa Niewiadomska-Szynkiewicz

Warsaw, 2023

Contents

List of Figures

List of Tables

Symbols

$\mathbf{x}$ vector $\mathbf{x}$

$\mathbf{X}$ matrix $\mathbf{X}$

$\|\mathbf{x}\|_0$ ℓ_0 pseudo-norm of $\mathbf{x}$
$\|\mathbf{x}\|_1$ ℓ_1 norm$\mathbf{x}$
$\|\mathbf{x}\|_2$ ℓ_2 norm (Euclidean norm) $\mathbf{x}$

$\|\mathbf{X}\|_2$ spectral norm of matrix $\mathbf{X}$

$\|\mathbf{X}\|_*$ nuclear norm of matrix $\mathbf{X}$

$\|\mathbf{X}\|_F$ Frobenius norm of matrix $\mathbf{X}$

$\mathbf{X}^T$ transpose of matrix $\mathbf{X}$

$\mathbf{X}^\dagger$ Moore-Penrose pseudoinverse of $\mathbf{X}$

$\sigma(\mathbf{X})$ vector of singular values of matrix $\mathbf{X}$
$\sigma_i(\mathbf{X})$ i-th largest singular value of matrix $\mathbf{X}$

$\lambda_{\max}(\mathbf{X})$ Maximum eigenvalue of matrix $\mathbf{X}$
$\lambda_{\min}(\mathbf{X})$ Minimum eigenvalue of matrix $\mathbf{X}$

$\mathbf{X} \otimes \mathbf{Y}$ Kronecker product of $\mathbf{X}$ and $\mathbf{Y}$

$D_\lambda(\mathbf{X})$ Singular Value Thresholding operator of $\mathbf{X}$

T linear space T

$T^\perp$ linear space perpendicular to T

$T_1 \oplus T_2$ direct sum of linear spaces T_1 and T_2

$\mu(U)$ coherence of subspace U

$\mathbf{M}$ matrix to be recovered

Ω set of observed indices

$\mathcal{R}_\Omega$ sampling operator
$\mathbf{C}$ column submatrix of $\mathbf{M}$

Abbreviations

AUC Area Under the Curve

CSMC Column Selected Matrix Completion
CSNN Column Selected Nuclear Norm
CSPGD Column Selected Proximal Gradient Descent
CSS Column Subset Selection

MC Matrix Completion

NMAE Normalized Mean Absolute Error
NN Nuclear Norm

PGD Proximal Gradient Descent

SCS Splitting Conic Solver
SDP Semidefinite Programming
SNR Signal to Noise Ration
SVD Singular Value Decomposition
SVT Singular Value Thresholding

Abstract

In today's data-driven era, methods for filling data gaps are becoming more critical. This dissertation proposes a novel two-step method for the matrix recovery problem. Our approach combines the theoretical foundations of the Column Subset Selection and Low-rank Matrix Completion problems. The proposed method, in each step, solves a convex optimization task. In the first step, a subset of columns is drawn at random. The resulting matrix is completed using a selected algorithm that solves the matrix completion task. The second step solves a least squares problem using the known elements and the completed columns.

We present three algorithms that implement our Column Selected Matrix Completion (CSMC) method, each dedicated to a different size problem. We performed a formal analysis of the presented method, in which we formulated the necessary assumptions and the probability of finding a correct solution.

In the second part of the paper, we present the results of the experimental work. Numerous numerical experiments verified the correctness and performance of the algorithms. To study the influence of the size of the matrix, its rank, and the proportion of missing elements on the quality of the solution and the computation time, experiments were performed on synthetic data. The presented method was applied to three problems: prediction of movie rates in a recommendation system, image completion and prediction of connections in a graph.

Our thorough analysis shows that CSMC provides solutions of comparable quality to matrix completion algorithms, which are based on convex optimization. However, CSMC offers notable savings in terms of runtime.

Acknowledgments

I am deeply thankful to Professor Ewa Niewiadomska-Szynkiewicz, my supervisor, for her constant support, enthusiastic guidance, and mentorship throughout my research journey.

I am incredibly grateful to Dr. hab. Michał Karpowicz for not just igniting my interest in linear algebra but also for his constant encouragement, sincere interest in my thoughts, and willingness to offer assistance whenever needed.

Most importantly, I would like to thank my Dad for teaching me the importance of finding joy in my work.

Chapter 1

Introduction

Matrix-based datasets are common in various domains. Images, time series, graphs, and recommendation systems, are just a few examples with natural representation as real-valued matrices. However, these matrices often miss entries due to data collection issues, sensor errors, or privacy considerations. To fully exploit the potential of matrix-based datasets, we need to address the missing data problem.

This dissertation proposes a novel, two-staged matrix completion method deriving benefits from integrating two fields of modern linear algebra. The first one is the theory of low-rank matrix completion. Many efficient algorithms exploit the low-rank structure of the data and principles of convex optimization to recover the missing entries. The second one is the theory of the column subset selection problem. This line of research focuses on finding the column of the matrix representing the original matrix approximately, i.e., preserving specific desirable properties.

Our method, Column Selected Matrix Completion (CSMC), addresses the problem of filling matrices with many columns. In the first stage, CSMC selects a subset of columns and recovers them with a chosen matrix completion algorithm. In the second stage, the relevant least squares problem is solved. We introduce three algorithms implementing CSMC. Those algorithms employ various numerical solvers depending on the size of the completed matrix.

The integration of the two approaches allows CSMC to decrease the computational cost of the convex matrix completion algorithms while maintaining their theoretical guarantees. This thesis provides a formal analysis of the CSMC, including required assumptions and the probability of a successful recovery.

Several numerical experiments on synthetic and real-world datasets were conducted to verify theoretical results. The performance of the presented algorithms was compared with the commonly used matrix completion algorithms described in the literature. The CSMC method was applied to three real-life scenarios: movie recommendation, image inpainting, and graph link prediction.

1.1 Research hypothesis

This section presents the research hypotheses that guide this dissertation study.

Hypothesis 1

The quality of the solution provided by the proposed new CSMC method is comparable to that of the matrix completion methods described in the literature.

This hypothesis proposes that there is no significant difference in the solution quality between the CSMC and the matrix completion methods, including nuclear norm minimization, proximal gradient descent, algorithm based on the factorization and iterative SVD. The first aim of the research is to provide theoretical guarantees of the CSMC. The study's second aim is to empirically test this hypothesis by comparing the performance, accuracy, or effectiveness of the methods using appropriate evaluation metrics or criteria.

Hypothesis 2

The new algorithms implementing the CSMC methods are efficient and scalable techniques for matrix recovery and competitive with other algorithms using nuclear norm minimization

This hypothesis suggests that algorithms implementing the CSMC method outperform algorithms based on the nuclear norm minimization in terms of computational efficiency, resulting in significantly shorter execution times. The research would aim to test this hypothesis by measuring the execution times of the new algorithms under various conditions or datasets and comparing their efficiency with other techniques described in the literature.

1.2 Structure of the dissertation

The structure of this thesis is outlined as follows. Chapter 2 introduces the topic, including preliminary concepts and notation. In Chapter 3, we delve into the motivation behind the theory of low-rank matrix completion and trace its development over time. We introduce CSMC and articulate the reasons and benefits behind its application. The chapter ends with a discussion of the various application of matrix completion. Chapter 4 systematically reviews the contemporary methods of matrix completion. The survey offers a comprehensive examination of the strengths and weaknesses inherent in the methods. Two methods based on convex optimization are discussed in Chapter 5. Those approaches differ in scalability and noise robustness. The matrix's properties determining the completion's success are concisely explained. Chapter 6 considers the column subset selection problem in the context of matrix completion. The CSMC is formally introduced in Chapter 7. We discuss three algorithms implementing the CSMC method dedicated to the various matrix completion problem sizes. An in-depth formal analysis

of our method is carried out in the last section of the chapter. The theoretical results are verified in Chapter 8, which describes numerical experiments on synthetic data sets. This chapter aims to compare the performance of the CSMC algorithms in various settings and benchmark them with the matrix completion algorithms described in the literature. To assess the practical effectiveness of the CSMC methods, they were employed in three real-world settings in Chapter 9. These applications include recommendation systems, image inpainting and link predictions in graphs. We conclude the dissertation in Chapter 10.

Chapter 2

Preliminaries and notation

This chapter provides basic definitions and notation used in the thesis. We closely follow the conventions established in the literature and build upon the foundations of the low-rank matrix completion theory.

The matrices and vectors are denoted in bold ($\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{x} \in \mathbb{R}^n$). The x_{ij} is the (i, j) entry of $\mathbf{X}$. We will denote as $\mathbf{x}_j \in \mathbb{R}^{n_1}$ j-th column vector of $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$. The $\mathbf{x}_{i:} \in \mathbb{R}^{n_2}$ will refer to the vector defined by the i-th row of $\mathbf{X}$. For the set of d indices, $I \in \{1, \ldots, n_2\}$, $\mathbf{X}_{:I} \in \mathbb{R}^{n_1 \times d}$ will denote the column submatrix of $\mathbf{X}$ build of columns with indices in I. $\mathbf{e}_i$ will denote a i-th standard basis vector of $\mathbb{R}^n$, equal to 1 in component i and 0 everywhere else.

We will now introduce some notation regarding the matrix completion The following definitions and notation will be used throughout the thesis.

We will denote as $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ the matrix we would like to recover in the matrix completion problem,

$$\mathbf{M} = \begin{pmatrix} -2 & 4 & 16 \\ -3 & 6 & 24 \end{pmatrix}. \tag{2.1}$$

For the matrix completion problem, we will use the following notation. We denote the set of known indices as Ω. We will also consider a sampling operator $\mathcal{R}_\Omega : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n_1 \times n_2}$ which sets entries from Ω to 0, e.g.

Definition 2.0.1. *We will denote as $\Omega \subseteq \{1, \ldots, n_1\} \times \{1, \ldots, n_2\}$ the set of the indices of the observed entries of $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$.*

Suppose, we know only two entries $\Omega = \{(1, 1), (2, 2)\}$, then we consider the following matrix completion problem,

$$\begin{pmatrix} -2 & ? & ? \\ ? & 6 & ? \end{pmatrix}, \tag{2.2}$$

Definition 2.0.2. *We will denote as $\Omega^\perp$ the complement of a set $\Omega \subseteq \{1, \ldots, n_1\} \times \{1, \ldots, n_2\}$, i.e. $\{1, \ldots, n_1\} \times \{1, \ldots, n_2\} \setminus \Omega$.*

For the example mentioned above, $\Omega^{\perp} = \{(1,2),(1,3),(2,1),(2,3)\}$.

Definition 2.0.3. *The operator defined be the Ω set, $\mathcal{R}_\Omega : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n_1 \times n_2}$, sets all matrix elements to 0, except those in the Ω set. Formally,*

$$\mathcal{R}_\Omega(\mathbf{X}) = \sum_{(i,j)\in\Omega} x_{ij}\mathbf{e}_i\mathbf{e}_j^T, \tag{2.3}$$

where x_{ij} is a (i,j)-th entry of $\mathbf{X}$, $\mathbf{e}_i$ for $i = 1,\ldots,n_1$ are standard basis vectors in $\mathbb{R}^{n_1}$, and $\mathbf{e}_j$ for $i = 1,\ldots,n_2$ are standard basis vectors in $\mathbb{R}^{n_2}$.

For $\Omega = \{(0,0),(1,1)\}$ and $\mathbf{M}$ defined in eq. (2.1),

$$\mathcal{R}_\Omega(\mathbf{M}) = \begin{pmatrix} -2 & 0 & 0 \\ 0 & 6 & 0 \end{pmatrix}. \tag{2.4}$$

Definition 2.0.4. *The rank of a matrix $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ is the dimension of the vector space spanned by its columns.*

We will consider Singular Value Decomposition (SVD) of $\mathbf{M}$,

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{2.5}$$

Where $\mathbf{U} \in \mathbb{R}^{n_1 \times n_1}$ and $\mathbf{V} \in \mathbb{R}^{n_2 \times n_2}$ are orthogonal matrices, and $\mathbf{\Sigma} \in \mathbb{R}^{n_1 \times n_2}$ is a rectangular diagonal matrix with non-negative numbers on its diagonal $\sigma_i = \mathbf{\Sigma}_{ii}$ called singular values. We will follow the convention $\sigma_1 \geq \sigma_2 \ldots \sigma_r > 0$, where r is the rank of $\mathbf{M}$. Eq. 2.5 can be rewritten as

$$\mathbf{M} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \tag{2.6}$$

where $\mathbf{u}_i$ denotes i-th column of $\mathbf{U}$ and $\mathbf{v}_i$ denotes i-th column of $\mathbf{V}$. The SVD is often referred to as the full SVD, while the compact SVD is a factorization,

$$\mathbf{M} = \bar{\mathbf{U}}\bar{\mathbf{\Sigma}}\bar{\mathbf{V}}^T, \tag{2.7}$$

where $\bar{\mathbf{U}} \in \mathbb{R}^{n_1 \times r}$, $\bar{\mathbf{V}} \in \mathbb{R}^{n_2 \times r}$, and $\bar{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$. A. The compact SVD retains only the r columns of $\mathbf{U}$, $\mathbf{V}$, associated with the non-zero singular values. We will denote as $\bar{U}$ a r-dimensional subspace of $\mathbf{R}^{n_1}$ spanned by the first r left singular vectors. We will denote as $P_{\bar{U}}$ the orthogonal projection onto the $\bar{U}$,

$$P_{\bar{U}}\mathbf{X} = \bar{\mathbf{U}}\bar{\mathbf{U}}^T\mathbf{X}, \tag{2.8}$$

since column vectors of $\mathbf{U}$ are orthonormal.

Definition 2.0.5. *The ℓ_0-(pseudo)norm of vector $\mathbf{x} \in \mathbb{R}^n$ is the number of its non-zero entries.*

Definition 2.0.6. *The ℓ_1-norm of vector $\mathbf{x} \in \mathbb{R}^n$ is defined as*

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|, \tag{2.9}$$

Three different norms on matrices will be discussed.

Definition 2.0.7. *The Frobenius norm of a matrix $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ is given by*

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} |x_{ij}|^2}, \tag{2.10}$$

equivalently,

$$\|\mathbf{X}\|_F = \sqrt{Tr(\mathbf{X^T X})}, \tag{2.11}$$

where $Tr(\mathbf{X^T X})$ denotes trace of the matrix $\mathbf{X^T X} \in \mathbb{R}^{n_2 \times n_2}$.

From (2.11) the Frobenius norm can be calculated according to the following remark.

Remark 2.0.1.

$$\|\mathbf{X}\|_F^2 = \sum_{i=1}^{r} \sigma_i^2, \tag{2.12}$$

where r denotes the rank of matrix $\mathbf{X}$.

Definition 2.0.8. *The spectral norm of a matrix $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ is the largest singular value of $\mathbf{X}$*

$$\|\mathbf{X}\|_2 = \max_{1 \le i \le r} \sigma_i, \tag{2.13}$$

where σ_i is a non-zero singular value of $\mathbf{X}$ for $i = 1, \cdots, r$ and r is a rank of $\mathbf{X}$.

Remark 2.0.2. *Let $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ be a matrix with rank r,*

$$\frac{1}{r}\|\mathbf{X}\|_F^2 \le \|\mathbf{X}\|_2^2 \le \|\mathbf{X}\|_F^2. \tag{2.14}$$

Proof. It follows from the following inequality,

$$\frac{1}{r}\sum_{i=1}^{r} \sigma_i^2 \le \max_{1 \le i \le r} \sigma_i^2 \le \sum_{i=1}^{r} \sigma_i^2. \tag{2.15}$$

□

Definition 2.0.9. *The nuclear norm of a matrix* $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ *is given by the sum of singular values of* $\mathbf{X}$*.*

$$\|\mathbf{X}\|_* = \sum_{1 \leq i \leq r} \sigma_i, \tag{2.16}$$

where σ_i *are the non-zero singular values of* $\mathbf{X}$ *for* $i = 1, \cdots, r$ *and* r *is a rank of* $\mathbf{X}$*.*

The nuclear norm is also known as the matrix Schatten-1 or trace norm.

Remark 2.0.3. *Let* $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ *be a matrix with rank* r*, and let* $\sigma \in \mathbb{R}^{\min\{n_1, n_2\}}$ *be the vector of its singular values. Since the rank is the number of the non-zero singular values, then*

$$r = \|\sigma\|_0. \tag{2.17}$$

From the definition of the nuclear norm and the fact that $\sigma_i \geq 0$ *for* $i = 1, \ldots, \min\{n_1, n_2\}$*,*

$$\|\mathbf{X}\|_* = \|\sigma\|_1. \tag{2.18}$$

Definition 2.0.10. *Let* $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n_1 \times n_2}$*, the Frobenius inner product is defined as,*

$$\langle \mathbf{X}, \mathbf{Y} \rangle = tr(\mathbf{X}^T \mathbf{Y}). \tag{2.19}$$

Methods introduced in this thesis are based on the Column Subset Selection and least-squares problems. The Moore-Penrose inverse of a matrix, also known as the pseudoinverse, is a generalization of the matrix inverse for non-square matrices or matrices that are not full rank [1, 2]. The pseudoinverse inverse allows us to solve linear systems of equations even when the matrix is singular or not of full rank.

Definition 2.0.11. *Let* $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$*, the pseudoinverse of* $\mathbf{X}$*, defined as a matrix* $\mathbf{X}^\dagger \in \mathbb{R}^{n_2 \times n_1}$*, satisfying the following the Moore-Penrose conditions:*

1. $\mathbf{X}\mathbf{X}^\dagger\mathbf{X} = \mathbf{X}$,
2. $\mathbf{X}^\dagger\mathbf{X}\mathbf{X}^\dagger = \mathbf{X}^\dagger$,
3. $(\mathbf{X}\mathbf{X}^\dagger)^T = \mathbf{X}\mathbf{X}^\dagger$,
4. $(\mathbf{X}^\dagger\mathbf{X})^T = \mathbf{X}^\dagger\mathbf{X}$.

The pseudoinverse can be calculated using the singular value decomposition. If SVD of $\mathbf{X}$ is given by

$$\mathbf{X} = \mathbf{U}_x \mathbf{\Sigma}_x {\mathbf{V}_\mathbf{x}}^T, \tag{2.20}$$

where $\mathbf{U} \in \mathbb{R}^{n_1 \times n_1}$, $\mathbf{V} \in \mathbb{R}^{n_2 \times n_2}$, and $\mathbf{\Sigma} \in \mathbb{R}^{n_1 \times n_2}$, then

$$\mathbf{X}^\dagger = \mathbf{V}_x \mathbf{\Sigma}_x^\dagger {\mathbf{U}_\mathbf{x}}^T, \tag{2.21}$$

The pseudoinverse of $\mathbf{\Sigma}$ is obtained by taking the reciprocal of the non-zero singular values and taking the transpose of the resulting matrix.

Chapter 3

Motivation and problem definition

Matrix Completion (MC) gained significant attention and popularity with the Netflix Prize competition, which aimed to improve movie recommendations. The competition was launched by the popular streaming platform in 2006. The goal of the contest was to improve the accuracy of Netflix's movie recommendation algorithm by 10%. The company offered a prize of $1 million to the team that could achieve this improvement. The company provided a large dataset containing anonymized movie ratings from their users. The dataset comprised over 100 million ratings from hundreds of thousands of subscribers. The participants were tasked with developing a recommendation algorithm that could predict how a user would rate a movie based only on the previous ratings in the system. The contest lasted for several years, and during that time, many teams from around the world participated and submitted their algorithms. The Netflix Prize brought matrix completion techniques into the spotlight, leading to field advancements. Since then, matrix completion has been extensively studied and applied in various domains, including recommendation systems, image inpainting, sensor networks, and more [3].

Matrix completion algorithms can employ different optimization frameworks, such as convex optimization, non-convex optimization, iterative methods, or probabilistic approaches [4–6]. This thesis focuses on convex optimization methods. The elegance of convex optimization lies in its solid theoretical foundations, which allow for the development of efficient algorithms with guaranteed convergence and optimality properties.

3.1 Collaborative filtering

In collaborative filtering, the goal is to predict the ratings or preferences of users for items they have not yet rated or interacted with [7–10]. This problem can be formulated as a matrix completion task, where the rows represent users, the columns represent items, and the matrix entries correspond to the user-item ratings. The basing idea behind collaborative filtering is that people with similar

preferences or behaviours will likely have similar preferences in the future. The Singular Value Decomposition (SVD) offers valuable insight into collaborative filtering [11, 12]. Let $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ be a user-movie rating matrix. The SVD is a matrix decomposition such that

where $\mathbf{U} \in \mathbb{R}^{n_1 \times n_1}$ and $\mathbf{V} \in \mathbb{R}^{n_2 \times n_2}$ are orthonormal matrices, and $\mathbf{\Sigma} \in \mathbb{R}^{n_1 \times n_2}$ is a rectangular diagonal matrix with non-negative numbers on its diagonal $\sigma_i = \mathbf{\Sigma}_{ii}$ called singular values. We will follow the convention $\sigma_1 \geq \sigma_2 \ldots \sigma_r > 0$, where r is the rank of $\mathbf{M}$. Eq. 2.5 can be rewritten as

$$\mathbf{M} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \tag{3.1}$$

where $\mathbf{u}_i$ denotes i-th column of $\mathbf{U}$ and $\mathbf{v}_i$ denotes i-th column of $\mathbf{V}$. In the context of collaborative filtering, the singular vectors obtained from SVD are often associated with latent factors. Latent factors represent underlying dimensions or features contributing to a recommendation system's user-item interactions. Each of the left singular vectors represents the importance of the given factor to a user, while the right singular vectors represent the relationships between items and latent factors. Singular values express the significance of each latent factor. The larger the singular value, the more significant the corresponding latent factor in explaining the variances or patterns in the rating matrix. The latent factors are abstract features that influence user preferences and movie characteristics. For example, latent factors might represent dimensions such as *romantic comedy*, *documentary*, or *horror*. The presence or absence of these latent factors in the user-movie interactions determines the recommendations made by the system. Collaborative filtering algorithms can identify similar users or items based on their latent factor representations by mapping users and movies to the latent factor space. Users with similar preferences will have similar weights for the same latent factors, and items with similar characteristics will also have similar weights. This similarity in the latent factor space allows for accurate predictions. Matrix completion techniques can fill in the missing entries in the rating matrix and provide personalized recommendations [3, 13, 14].

3.2 Low-rank matrix completion

A convenient strategy for solving matrix completion would be to seek the matrix which matches the observed entries while having the lowest rank [3, 15–17]. This could be interpreted as looking for the simplest explanation of the observed ratings, where simplicity is expressed as the number of latent factors. From a socio-psychological perspective, this can be interpreted as an attempt to identify collective preferences among individuals. By reducing the rank, matrix completion algorithms aim to uncover underlying dimensions of shared tastes among users, reflecting social norms or trends.

Suppose that one of the singular vectors in $\mathbf{U}$ is a standard basis vector. It implies the existence of the latent factor, which has some meaning only to one user. Therefore, we can not infer anything from other users' ratings. If we do not know all ratings made by this user, we can not successfully predict missing ratings. This phenomenon motivates the definition of the matrix *coherence*, which measures how much singular vectors (either left or right) are colinear with standard basis vectors. Coherence is essential in matrix completion and affects the number of observed entries required for accurate recovery. On the other hand, incoherent matrices may require fewer observed entries for successful completion compared to coherent matrices. The low coherence property enables fewer measurements or samples to recover the original matrix accurately. This can be advantageous when dealing with limited or sparse data [3, 13, 15–18].

3.3 From compressed sensing to convex matrix completion

Compressed sensing, also known as compressive sensing, emerged as a field of research in the early 2000s. It was introduced by Emmanuel Candès, Terence Tao, and David Donoho, among others, who laid the foundations and contributed to its development [19–24]. The key idea of compressed sensing was to exploit the compressibility of signals to acquire and process them more efficiently. Compressed sensing acquires a small number of measurements, which capture the essential information of the signal. The breakthrough in compressed sensing was the realization that sparse signals could be accurately reconstructed from these limited measurements using convex optimization. It came with the realization that ℓ_0 pseudo-norm, which measures the sparsity of a vector by counting the number of its non-zero elements, can be relaxed to ℓ_1 norm. The ℓ_1 norm is the sum of the absolute values of the elements of a vector and allows for the formulation of convex optimization problems.

The success of compressed sensing encouraged researchers to apply analogous relaxation to the constrained rank minimization problem, which aims to recover matrix $\mathbf{M}$. The rank may be expressed as the number of non-zero singular values: ℓ_0 norm of the singular values vector σ. The ℓ_1 norm of σ is referred to as the nuclear norm of $\mathbf{M}$ and has found applications in various areas, including signal processing and robust optimization [3, 15, 17, 18, 25].

In their seminal work, Candes and Recht proved that, under certain assumptions, solving nuclear norm minimization leads to successful recovery [26]. The theoretical guarantees are based on the properties such as matrix rank and coherence, number and distribution of observed entries. There has been a significant amount of research extending these results to noisy settings, improved bounds on the sample complexity and analysing the robustness of nuclear norm minimization algorithms to noise and outliers. Recht [15] greatly simplified the analysis and provided a foundation for understanding the potential and limitations of nuclear

norm minimization in practical matrix completion scenarios. The research on theoretical guarantees for nuclear norm based matrix completion has significantly advanced our understanding of the conditions under which accurate recovery is possible.

3.4 Column Selected Matrix Completion

The nuclear norm minimization can be solved in polynomial time with the Semidefinite Programming (SDP) [3,17,25]. As such it can be handled with one of the shelf SDP solvers including those using interior point methods or first-order solvers such as Splitting Conic Solver (SCS). However, SDP solvers can consume substantial memory resources, particularly for large problem sizes. This can limit the scalability of SDP solvers when dealing with large matrices [17]. To limit computational burden, an algorithm dedicated to nuclear norm minimization, including Proximal Gradient Descent (PGD) has been developed [27–29]. The PGD for nuclear norm minimization leverages the closed formula for the proximal operator, which incorporates Singular Value Thresholding (SVT) operation. While SVT offers an ease of implementation, there is a computational cost associated with this approach. The straightforward implementation of SVT involves computing the SVD of the updated matrix in each iteration. Consequently, SVT can become computationally expensive for large-scale problems and inherits its difficulty in parallelizing efficiently. Thus, it is desired to develop methods limiting the problem size while providing theoretical guarantees. In this thesis, we introduce the Column Selected Matrix Completion (CSMC), a two-staged method for low-rank matrix recovery. In the first stage, CSMC randomly samples columns of $\mathbf{M}$ and recovers the column submatrix using the preferred matrix completion algorithm. In the second stage, the relevant least squares problem is solved to reconstruct $\mathbf{M}$. Least squares regression is a fundamental technique in machine learning. It is employed in linear regression models for prediction and estimation tasks and can be solved efficiently using various numerical methods and optimization techniques [30–32].

The CSMC offers computational savings when one dimension of $\mathbf{M}$ is significantly bigger. This scenario is valid in the Netflix Prize setup, where the number of movies was fairly bigger than the number of movies.

The CSMC benefits from the recent research in the area of Column Subset Selection (CSS) problem and CUR factorization [33–37]. Those linear algebra techniques aim to reduce the dimensionality of a matrix while preserving essential information. Their popularity stems from the ability to provide interpretable low-rank approximations, preserve structural information, and offer computational efficiency. CSS refers to selecting a subset of columns from a given matrix. At the same time, CUR is a low-rank matrix approximation technique that decomposes a given matrix into the column submatrix, row submatrix, and an intersection of the selected rows and columns. Recent work has shown that for incoherent matrices, uniform sampling achieves satisfying results [37]. Uniform sampling

does not require any complex algorithms or information about entries in $\mathbf{M}$.

3.5 Applications of low-rank matrix completion

The motivation behind low-rank matrix completion is to exploit the inherent low-rank structure of the matrices. By assuming that a low-rank matrix can well approximate the underlying matrix, the missing entries can be effectively imputed. We have already discussed its applicability in recommendation systems. While low-rank matrix completion is indeed widely used in recommendation systems, it has found applications in various other domains as well [17, 38–42].

In many real-world datasets, including images, text, and biological data, there are often redundant elements or patterns. This redundancy leads to a low-rank structure, as the data can be well approximated by a smaller number of underlying factors or components.

The low-rank matrix completion finds applications in image and video processing tasks such as image inpainting and video denoising [13, 43–47]. In image inpainting, missing or corrupted parts of an image are reconstructed by completing the observed pixel values. Matrix completion can also be used for video denoising by exploiting temporal correlations between video frames to recover missing or noisy frames [43, 46].

In bioinformatics, low-rank matrix completion techniques have been employed for tasks such as gene expression analysis, protein structure prediction, and DNA sequencing. By leveraging the low-rank property of biological data, missing values in high-dimensional datasets can be imputed, leading to more accurate analysis and predictions [39, 48–50].

In Magnetic Resonance Imaging (MRI), low-rank matrix completion techniques have been used for accelerated imaging and reconstruction. By exploiting the low-rank structure of MR images, fewer measurements can be acquired, leading to faster imaging and reduced scan times [51–53].

A matrix with a rank equal to the number of latent variables can be used to approximate the inverse marginal covariance matrix of the observed variables in a sparse graphical model with latent variables [54, 55].

Hankel matrix completion techniques leverage the structure and properties of Hankel matrices to estimate missing or incomplete entries and enable various analysis tasks in time series analysis, system identification, and signal processing. These techniques aim to capture the underlying patterns, dynamics, and relationships in the data, ultimately improving prediction, estimation, and analysis in time-dependent or sequential scenarios [40, 56, 57].

Low-rank matrix completion techniques can be applied to network traffic analysis and anomaly detection. By reconstructing the traffic matrix from incomplete or sampled measurements, anomalies and patterns in network behaviour can be identified. It is also useful in IoT networks, where data collection from sensor networks or IoT devices is incomplete or prone to missing readings. By applying

matrix completion techniques, missing sensor measurements can be estimated, enabling accurate data analysis, anomaly detection, and system monitoring [58–60].

Low-rank matrix completion has also been used in environmental monitoring applications, such as air quality monitoring and pollution estimation. By recovering missing or sparse measurements, low-rank matrix completion enables accurate estimation and prediction of environmental conditions [61, 62].

Low-rank matrix completion techniques have been employed in natural languages processing tasks, such as text completion, sentiment analysis, and document recommendation. By exploiting the low-rank structure of text corpora, missing words or documents can be predicted, and latent features can be extracted for text analysis [41, 63].

Low-rank matrix completion can be applied to resource allocation tasks [64–66], i.e., to predict and assess how well an incoming application will run on different hardware platforms available. By treating the problem as a matrix completion task, where the matrix represents the relationship between applications and hardware platforms, we may predict the performance of an application on different hardware platforms. [67, 68].

Matrix completion methods can be utilized in social network analysis to estimate unobserved social connections or interactions between individuals. By completing the partially observed social network matrix, researchers can infer missing connections, predict links, and understand the structure and dynamics of social networks [69, 70].

These are just a few examples of the broad range of applications for matrix completion. The technique is versatile and can be adapted to various domains where incomplete data is encountered, enabling the recovery of missing information and facilitating data analysis and decision-making processes [42].

Chapter 4

Matrix completion methods

This chapter discusses various formulations of the low-rank matrix completion problem. Low-rank matrix completion algorithms aim to estimate the missing entries of a partially observed matrix by exploiting the low-rank assumption. As discussed in the previous chapter, the low-rank assumption is often made based on the intuition that many real-world matrices exhibit underlying structures that low-rank representations can effectively capture. These algorithms typically leverage convex or non-convex optimisation techniques to recover the low-rank structure and complete the matrix. Using the low-rank assumption, these algorithms can often achieve accurate and efficient matrix completion even with a limited number of observed entries.

4.1 Convex method for low-rank matrix completion

The straightforward approach to the low-rank matrix completion is to seek the matrix with the lowest rank that matches the observations,

$$\begin{vmatrix} \text{minimize} & \text{rank}(\mathbf{X}) & \\ \text{s.t.} & \mathbf{X}_{ij} & = \mathbf{M}_{ij} \text{ for } (i,j) \in \Omega, \end{vmatrix} \tag{P1}$$

where by definition 2.0.1, Ω denotes the set indices of known data of $\mathbf{M}$. A plausible interpretation to put on P1 is that solution would be the simplest explanation of the observed data. In collaborative filtering, problem P1 minimizes the number of latent factors behind users' preferences. However, solving rank minimization is NP-hard [71, 72]. To overcome this drawback, problem P1 may be relaxed to the convex optimization problem. Because matrix rank is equal to the number of non-zero singular values of $\mathbf{X}$, the objective function can be approximated by their sum, i.e. its nuclear norm of the matrix introduced in Definition 2.0.9. The nuclear norm can be used as a convex surrogate for the

rank of a matrix, enabling the formulation of certain optimization problems with a nuclear norm regularization term.

Exact matrix completion algorithms As discussed in Section 3.3, the success of compressed sensing in sparse signal recovery [73–76] inspired researchers to explore similar ideas for matrix completion and motivated the following optimization problem

$$\begin{vmatrix} \text{minimize} & \|\mathbf{X}\|_* & \\ \text{s.t.} & \mathbf{X}_{ij} & = \mathbf{M}_{ij} \text{ for } (i,j) \in \Omega. \end{vmatrix} \tag{P2}$$

The problem P2 can be solved in polynomial time by Semidefinite Programming (SDP) [71]. SDP is an optimization problem where the objective function and constraints involve linear matrix inequalities or semidefinite constraints. SDPs generalize linear and quadratic programming problems where the variables are positive semidefinite matrices. According to the Definition 2.0.3, the constraints of the problem P2 can be expressed as $\|\mathcal{R}_\Omega(\mathbf{X}) - \mathcal{R}_\Omega(\mathbf{M})\|_F = 0$. To reduce computational costs of the off-the-shelf SDP solvers, Cai et al. introduced the regularized constrained optimization,

$$\begin{vmatrix} \text{minimize} & \frac{1}{2}\|\mathcal{R}_\Omega(\mathbf{X}) - \mathcal{R}_\Omega(\mathbf{M})\|_F^2 + \lambda\|\mathbf{X}\|_* & \\ \text{s.t.} & \mathbf{X}_{ij} = \mathbf{M}_{ij} & \text{for } (i,j) \in \Omega, \end{vmatrix} \tag{P3}$$

where $\lambda > 0$ is a regularization parameter [19]. The authors have shown that the solution converges to the solution of the P2 problem, as $\lambda \to \infty$ (Theorem 3.1 in [19]). The algorithm uses Singular Value Thresholding (SVT) operator, which we discuss in the next chapter.

Inexact matrix completion To handle noisy data and avoid model overfitting, Mazumder et al. [27] employed Proximal Gradient Descent (PGD) algorithm for the following optimization problem,

$$\underset{\mathbf{X}}{\text{minimize}}\ \frac{1}{2}\|\mathcal{R}_\Omega(\mathbf{X}) - \mathcal{R}_\Omega(\mathbf{M})\|_F^2 + \lambda\|\mathbf{X}\|_*. \tag{P4}$$

The loss function penalizes the difference between the observed and estimated entries while promoting low-rank solutions. The alternative formulation of the inexact matrix completion problem is to minimize the mismatch between the observed entries with the constraint on its nuclear norm,

$$\begin{vmatrix} \text{minimize} & \frac{1}{2}\|\mathcal{R}_\Omega(\mathbf{X}) - \mathcal{R}_\Omega(\mathbf{M})\|_F^2 \\ \text{s.t.} & \|\mathbf{X}\|_* \leq \lambda, \end{vmatrix} \tag{P5}$$

where $\lambda > 0$ is a tuning parameter. It can be memory-efficiently solved by the Frank-Wolfe (conditional-gradient descent) schema [77,78]. Jaggi popularized these methods in his PhD Thesis [77]. Those methods have much lower iteration space complexity since they require only the rank of one SVD. The classical Frank-Wolfe schema converges very slowly. Recently, much work has been done to achieve speedups of this method [79–81], including SketchyCG, which uses random projections [82].

4.2 Matrix factorization based matrix completion

The more recent methods use the non-convex optimization problem and aim to minimize least squares error on the observed entries while constraining the rank of the solution,

$$\begin{vmatrix} \text{minimize} & \|\mathcal{R}_\Omega(\mathbf{X}) - \mathcal{R}_\Omega(\mathbf{M})\|_F^2 \\ \text{s.t.} & \text{rank}(\mathbf{X}) \leq k, \end{vmatrix} \tag{P6}$$

where $k > 0$ denotes the maximum possible rank. Non-convex rank constraint can be imposed by the parametrizing matrix $\mathbf{X}$ as the product of two matrices $\mathbf{L} \in \mathbb{R}^{n_1 \times k}, \mathbf{R} \in \mathbb{R}^{n_2 \times k}$, The problem P6 is equivalent to

$$\min_{\mathbf{L} \in \mathbb{R}^{n_1 \times k}, \mathbf{R} \in \mathbb{R}^{n_2 \times k}} \|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\mathbf{L}\mathbf{R}^T)\|_F. \tag{P7}$$

The problem is often referred to as Burer-Monteiro factorization [17,83]. This parametrization is non-unique. If $\mathbf{X} = \mathbf{L}\mathbf{R}^T$, then for any orthonormal matrix $\mathbf{Q} \in \mathbb{R}^{k \times k}$, $\mathbf{X} = \mathbf{L}\mathbf{Q}\mathbf{Q}^\mathbf{T}\mathbf{R}^T$, since $\mathbf{Q}\mathbf{Q}^\mathbf{T} = \mathbf{I}$. On the other hand, this formulation can limit the computational burden. The rank r is often much smaller than $\min\{n_1, n_2\}$, thus the size of the variables $(\mathbf{L}, \mathbf{R})$ is roughly linear in $(n_1 + n_2)$ rather than quadratic. This opens up an opportunity to create linear-time algorithms [17]. Chen et al. [17] distinguish three classes of methods solving P7.

Alternating Minimization The Alternating Minimization (AM) optimizes the loss function alternatively over one of the factors while fixing the other [84,85]. The subproblem solved in each iteration is convex. In particular, each iteration takes the form,

$$\mathbf{L}_{t+1} = \arg\min_{\mathbf{L} \in \mathbb{R}^{n_1 \times k}} \mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\mathbf{L}\mathbf{R}_t^T), \tag{4.1}$$

$$\mathbf{R}_{t+1} = \arg\min_{\mathbf{R} \in \mathbb{R}^{n_2 \times k}} \mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\mathbf{L}_{t+1}\mathbf{R}^T), \tag{4.2}$$

where t denotes the iteration number. A practical extension of AM is Generalized Low-Rank Models (GLRM) [86]. GLRM extends the concept of low-rank

matrix completion to handle more general data types and provides a flexible approach for data analysis and machine learning tasks.

Gradient descent methods The second class of non-convex algorithms use gradient descent methods to solve problem P6 with respect to $\mathbf{L}$ and $\mathbf{R}$ [83,87,88]. The most straightforward routine is given by

$$\mathbf{L}_{t+1} = \mathbf{L}_t - \tau \nabla_{\mathbf{L}} f(\mathbf{L}_t, \mathbf{R}_t), \tag{4.3}$$
$$\mathbf{R}_{t+1} = \mathbf{R}_t - \tau \nabla_{\mathbf{R}} f(\mathbf{L}_t, \mathbf{R}_t), \tag{4.4}$$

where τ is the stepsize parameter and $f(\mathbf{L}_t, \mathbf{R}_t) = \frac{1}{2}\|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\mathbf{L}_t\mathbf{R}_t^T)\|_F^2$. However, these methods often incorporate different variants of gradient descent, including projected gradient descent and stochastic gradient descent [88].

Singular value projection The last algorithm group incorporates singular value projection technique [89, 90]. They are based on the observation that the singular values of a low-rank matrix decrease rapidly, while the singular values of a matrix with random entries do not. The non-convex methods have worse theoretical guarantees for matrix completion. However, they have better running time and space complexity and are the most popular approach in practice [17].

4.3 Other methods for low-rank matrix completion

Classical optimisation methods are not the only approach to solving low-rank matrix completion problems. While these methods are commonly used and have proven to be effective, a variety of methods are described in the literature.

Iterative SVD Troyanskaya et al. [91] proposed an expectation maximization, iterative algorithm, which at iteration t replaces missing values from the truncated SVD of the matrix from previous iteration $t-1$. It is noted that the number of the largest singular values used in the reconstruction needs to be determined empirically.

Riemannian optimization methods Riemannian optimization techniques provide a robust framework for matrix completion problems and can be used when the rank of $\mathbf{M}$ is known. Those consider the underlying manifold structure and leverage techniques from differential geometry to improve optimization efficiency and convergence properties [92–95]. Much research has been dedicated to the MC algorithms which perform optimization over the Grassmannian manifold [96, 97].

Active sampling methods In machine learning, adaptivity refers to the ability of a model or algorithm to adjust its behaviour based on the observed data. It involves dynamically updating the model or algorithm as new information becomes available, allowing it to improve its performance or adapt to changes in the underlying data distribution. Incorporating adaptivity into low-rank matrix completion enables the model to handle dynamic data, adapt to changes, and improve the accuracy of the completion process over time. It allows the system to continuously learn and update its predictions, making it more robust and applicable in real-world scenarios where the data distribution is not static. Active sampling methods for matrix completion aim to select the most informative entries to observe in a partially observed matrix, seeking to maximise the completion process's accuracy. Instead of passively relying on randomly observed entries, active sampling methods actively choose which entries to observe based on specific criteria or strategies [98–100]. Active sampling methods can significantly reduce the observations required for matrix completion, making them particularly useful when observations are costly or time-consuming.

Bayesian methods Bayesian methods model the uncertainty in the missing entries and use Bayesian inference techniques to estimate the low-rank matrix [101–103]. Markov Chain Monte Carlo (MCMC) algorithms are often employed for sampling the posterior distribution [104, 105]. Bayesian methods offer a principled framework for low-rank matrix completion, allowing for the incorporation of prior knowledge, modelling uncertainty, and quantifying uncertainty. However, implementing Bayesian methods can be computationally demanding, especially for large-scale problems.

Inductive matrix completion Inductive matrix completion refers to the task of completing a partially observed matrix by leveraging additional information or side information that is available during both the training and testing phases [48,106,107]. It extends the traditional matrix completion problem, which focuses on recovering missing entries based solely on observed entries in a matrix. The side information can take various forms depending on the specific task and domain. It can include features associated with the rows or columns of the matrix, metadata about the entities represented in the matrix, temporal or spatial information, or any other relevant information that can provide clues for completing the matrix.

Low-rank matrix completion refers to estimating or recovering a low-rank matrix from a subset of its entries. All methods discussed in this chapter can be successfully employed in the matrices that can be well-approximated by a matrix with a low rank. In cases where the matrix does not exhibit a clear low-rank structure, a high-rank matrix completion should be considered. However, those methods are more challenging, as they require estimating more parameters without the benefit of low-rank constraints.

The formulation of the matrix completion problem can vary depending on the noise level in the data. For the noiseless scenario, exact matrix completion should be applied, while inexact matrix completion will handle noisy data.

Chapter 5

Nuclear norm based matrix completion

Nuclear norm minimization is a well-known and widely studied technique for low-rank matrix completion. It has gained popularity due to its convex formulation, theoretical guarantees, and good empirical performance. Besides matrix completion, the nuclear norm has several applications in various fields, including data compression, denoising, and dimensionality reduction. Robust Principal Component Analysis (RPCA) is a technique that separates a given matrix into a low-rank component and a sparse component. The nuclear norm plays a crucial role in RPCA by promoting low-rank while the ℓ_1-norm encourages sparsity in the sparse component [108,109]. RPCA has applications in video surveillance, image processing, and anomaly detection. Nuclear norm can also be applied in system identification problems, where the goal is to estimate the underlying dynamics of a system from observed input-output data. By promoting low-rank solutions, nuclear norm regularization helps identify systems with structured dynamics [110, 111]. Recent applications in signal processing include phase retrieval [112–114], which aims to recover the phase information of a signal or an image from only its magnitude measurements.

In this chapter, we discuss two nuclear norm based algorithms for the matrix completion problem - exact nuclear norm minimization as Semidefinite Programming (SDP) and Proximal Gradient Descent for inexact matrix completion. In the last section, we discuss theoretical guarantees for nuclear norm based matrix completion. If the matrix satisfies certain incoherence, and the observed entries are sampled uniformly at random, nuclear norm minimization can recover the actual low-rank matrix with high probability. These theoretical guarantees provide confidence in the effectiveness of the approach.

5.1 Nuclear norm minimization as Semidefinite Programming

Fazel [25, 115] introduced the concept of using the nuclear norm as a convex relaxation for the rank function in matrix completion. The formulation of the nuclear norm minimization as a Semidefinite Programming (SDP) problem involves introducing two auxiliary positive semidefinite matrix variables $\mathbf{W}_1 \in \mathbb{R}^{n_1 \times n_1}$, $\mathbf{W}_2 \in \mathbb{R}^{n_2 \times n_2}$. By incorporating these variables, the problem can be cast as an optimization problem that can be solved using SDP techniques. Fazel used the following formula for the nuclear norm in her seminal work,

$$\|\mathbf{X}\|_* = \min_{\mathbf{W_1}, \mathbf{W_2}} \left\{ \frac{1}{2}\mathrm{Tr}(\mathbf{W_1}) + \frac{1}{2}\mathrm{Tr}(\mathbf{W_2}) \middle| \begin{pmatrix} \mathbf{W_1} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{W_2} \end{pmatrix} \succeq 0 \right\} \tag{5.1}$$

Hence, the problem P2 is equivalent to

$$\left| \begin{array}{ll} \text{minimize} & \|\mathbf{X}\|_* \\ \text{s.t.} & \mathbf{X}_{ij} = \mathbf{M}_{ij} \quad \text{for } (i,j) \in \Omega, \\ & \begin{pmatrix} \mathbf{W_1} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{W_2} \end{pmatrix} \succeq 0, \end{array} \right. \tag{P8}$$

considered in [25]. The problem P8 can be solved with the off-the-shelf semidefinite programming solvers. Those solvers employ interior point methods (SeDuMi [116], SDPT3 [117], and MOSEK [118], or first-order methods (SCS [119]). However, using off-the-shelf SDP solvers for matrix completion problems can face several challenges and limitations. SDP solvers typically have high computational complexity, making them less efficient for large-scale matrix completion problems. As the problem size increases, SDP solvers' computation time and memory requirements can become prohibitive.

5.2 Proximal Gradient Descent for inexact matrix completion (Soft-Impute)

To overcome discussed challenges, researchers have developed specialized algorithms and techniques that exploit the specific characteristics of matrix completion problems. One such method is Proximal Gradient Descent (PGD). PGD is an optimization algorithm for solving convex optimization problems, particularly in cases where the objective function consists of a smooth and a nonsmooth term [120]. It is an extension of the standard gradient descent method incorporating a proximal operator.

Definition 5.2.1. *The Proximal Operator (or Proximal Mapping) of the convex function h from Hilbert space to $\mathbb{R}$ is defined as,*

$$prox_h(x) = \arg\min_{z \in \mathcal{X}} \left[h(z) + \frac{1}{2}\|z - x\|_2^2 \right]. \tag{5.2}$$

In the context of the iterative optimization methods, it is often evaluated on function $\tilde{h} := \tau h$, where $\tau > 0$ is a parameter representing the stepsize, and formulated as,

$$\text{prox}_{\tau h}(x) = \arg\min_{z \in \mathcal{X}} \left[h(z) + \frac{1}{2\tau}\|z - x\|_2^2 \right]. \tag{5.3}$$

The general form of a convex optimization problem that can be solved using PGD is,

$$f(x) = g(x) + h(x), \tag{5.4}$$

where g is convex and differentiable, and h is convex and possibly non-differentiable, but with the easy to compute proximal operator [121].

The Proximal Gradient Descent (PGD) method et each iteration t, sets

$$x^{(t)} = \text{prox}_{\tau_t h}(x^{(t-1)} - \tau_t \nabla g(x^{(t-1)})), \tag{5.5}$$

where $\tau_t > 0$ is a stepsize at the iteration $t = 1, \cdots, T$, where T is the maximum iterations number.

The PGD uses a quadratic approximation of the smooth part g to define a step towards the minimum value, with the update rule

$$x^{(t)} = \arg\min_{z} \left[h(z) + \frac{1}{2\tau_t}\|z - (x^{(t-1)} - \tau_t \nabla g(x^{(t-1)}))\|_2^2 \right]. \tag{5.6}$$

The first term is minimized when the value of h is as small as possible, and the second term is minimized z is close to the gradient update of the smooth part g.

Proximal gradient descent has proven to be particularly useful in problems with sparsity-inducing regularization terms. It has been successfully applied in various fields, including compressed sensing, machine learning, and signal processing.

To apply PGD to the inexact matrix completion defined optimization problem P4 in Section 4.1, we need to decompose the objective function $f : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}$ into

$$f(\mathbf{X}) = \underbrace{\frac{1}{2}\|\mathcal{R}_\Omega(\mathbf{X}) - \mathcal{R}_\Omega(\mathbf{M})\|_F^2}_{g(\mathbf{X})} + \underbrace{\lambda\|\mathbf{X}\|_*}_{h(\mathbf{X})}. \tag{P9}$$

Function $g(\mathbf{X}) : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}$ is smooth, with gradient equal to

$$\nabla g(\mathbf{X}) = -(\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\mathbf{X})). \tag{5.7}$$

The analytical form of the proximal operator is given by the Singular Value Thresholding (SVT) operator [18, 27]. The operator bases on the SVD of the matrix, but controls the rank of the output matrix by nullifying singular values smaller than given threshold value λ. For $\mathbf{X}$ with the SVD given by

$$\mathbf{X} = \mathbf{U}_x \mathbf{\Sigma}_x \mathbf{V}_x^T, \tag{5.8}$$

the SVT is defined as follow,

$$D_\lambda(\mathbf{X}) = \sum_{i=1}^{r} \mathbf{u}'_i \mathbf{v}_i'^T (\sigma'_i - \lambda)_+, \tag{5.9}$$

where $\mathbf{v}'_i$ are left singular vectors of $\mathbf{X}$ associated with non-zero singular values σ'_i for $i = 1, \ldots, r$, $\mathbf{u}'_i$.

Following lemma provides the explicit formula for the PGD for the problem P9 [27, 122].

Lemma 5.2.1. *Proximal operator for the function $h : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}$ defined as*

$$h(\mathbf{X}) = \lambda \|\mathbf{X}\|_*, \tag{5.10}$$

$$\text{prox}_{\tau h}(\mathbf{X} - \tau \nabla g(\mathbf{X})) = D_\lambda(\mathbf{X} + \tau(\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\mathbf{X}))) \tag{5.11}$$

Since $\nabla g(\mathbf{X})$ is Lipshitz continuous with the Lipshitz constant $L = 1$, we can choose fixed step size $\tau_t = 1$ for every iteration $t = 1, \cdots, T$. Since

$$\mathbf{X} - \mathcal{R}_\Omega(\mathbf{X}) = \mathcal{R}_{\Omega^\perp}(\mathbf{X}) \tag{5.12}$$

where $\Omega^\perp$ follows the Definition 2.0.2.

The update step of the PGD algorithm has the form,

$$\mathbf{X}^{\tau+1} = D_\lambda(\mathcal{R}_\Omega(\mathbf{M}) + \mathcal{R}_\Omega^\perp(\mathbf{X}^\tau))) \tag{5.13}$$

The PGD for the regularized matrix completion problem was firstly used by Mazumder et al. [27] with the iterative procedure called Soft-Impute.

5.3 Guarantees for the Nuclear Norm Minimization based Matrix Completion

Convex matrix completion methods, which use nuclear norm minimization, offer strong theoretical guarantees. Under certain conditions, convex methods can provably recover the exact low-rank matrix with a high probability. Candes and Recht [26] provided guarantees for the matrix completion problem which relied on the three assumptions:

1. $\mathbf{M}$ has a low rank,
2. entries of $\mathbf{M}$ are observed uniformly at random,
3. singular vectors of $\mathbf{M}$ are uncorrelated with the standard basis vectors ($\mathbf{M}$ is incoherent).

In further work, Candes and Tao [123] improved the assumptions about the required number of the observed entries. Matrix completion theory was then greatly simplified by Recht [15] and Gross [16] by changing the assumptions about the sampling schema. Instead of sampling the whole set of known entries, each entry was sampled uniformly with replacement.

In this chapter, we discuss theoretical guarantees provided by Recht [15]. We begin with the definition of the matrix coherence.

5.3.1 Matrix coherence

The coherence parameter reflects the extent to which the singular vectors of the matrix are spread out over many rows or columns [3]. The matrix with the low value of the coherence parameter is said to be incoherent. Coherence is a useful measure, which provides insights into the difficulty of matrix completion problems. Incoherent matrices are more amenable to recovery using techniques such as nuclear norm minimization or low-rank matrix factorization [18, 85]. Conversely, coherent matrices may pose challenges and require more sophisticated algorithms or additional assumptions for successful completion.

Let us now revisit the Netflix Prize example discussed in Section 2.2. Let

$$\mathbf{M} = \sum_{i}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T. \tag{5.14}$$

Suppose the $\mathbf{u}_j$ tells how much j-th latent factor is important to the users, and $\mathbf{v}_j$ contains information on how this factor characterizes each of the movies. The matrix $\sigma_j \mathbf{u}_j \mathbf{v}_j^T$ represents the contribution of the j-th factor to movie rates of all users. If vector $\mathbf{u}_j$ equals to the standard basis vector, it means that j-th factor affects only one of the users and the contribution matrix can not be recovered

unless we sample all entries from the first row. This motivates the definition of the *coherence* which measures how much the singular vectors are correlated with the standard basis [26, 123].

Definition 5.3.1. *Let U be a subspace of $\mathbb{R}^n$ of dimension r_U and P_U be the orthogonal projection onto U. The coherence of U is defined as*

$$\mu(U) = \frac{n}{r_U} \max_{1 \leq i \leq n} ||P_U \mathbf{e}_i||_2^2, \tag{5.15}$$

where $\mathbf{e}_i$ for $i = 1, \cdots n$ are the standard basis vectors of $\mathbb{R}^n$.

The coherence parameter of the rank-$\tilde{r}$ matrix$\mathbf{X}$ is given by $\mu_0(\mathbf{X}) = \max\{\mu(U), \mu(V)\}$ where U and V are the linear spaces spanned by its $\tilde{r}$ left and right singular vectors respectively.

The coherence of the r_U-dimensional subspace is bounded according to the following lemma [3, 17, 18].

Lemma 5.3.1. *Let U be the r_U-dimensional subspace of $\mathbb{R}^n$, then $1 \leq \mu(U) \leq \frac{n}{r_U}$.*

Proof. To show the upper bound,

$$\mu(U) = \frac{n}{r_U} \max_{1 \leq i \leq n} ||P_U \mathbf{e}_i||_2^2 \leq \frac{n}{r_U} ||\mathbf{e}_i||_2^2 = \frac{n}{r_U}. \tag{5.16}$$

The upper bound is achieved when $\mathbf{e}_j \in U$ for some $j \in \{1, \cdots, n\}$.

To show the lower bound,

$$\sum_i^n ||P_U \mathbf{e}_i||_2^2 = r_U. \tag{5.17}$$

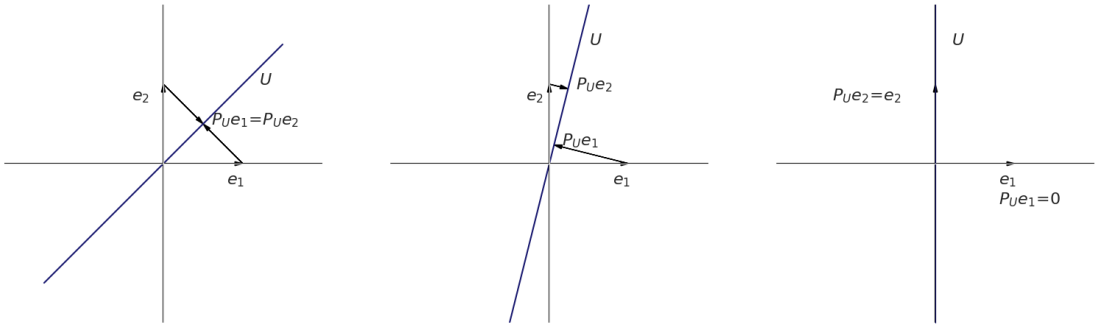
Thus, $\mu(U) \geq 1$. The lower bound is achieved if U is spanned by r_U vectors $(\frac{1}{\sqrt{n}}, \cdots, \frac{1}{\sqrt{n}})$.

□

To illustrate the role of the incoherence in the matrix completion, let us consider an orthogonal decomposition of the linear space of matrices $\mathbf{R}^{n_1 \times n_2}$ into two subspaces T and $T^\perp$, determined by the matrix $\mathbf{M}$,

$$\mathbb{R}^{n_1 \times n_2} = T \oplus T^\perp. \tag{5.18}$$

The projector operators onto T and $T^\perp$ were introduced by Candes and Recht [26] and play a crucial role in understanding the theoretical guarantees of the matrix completion problem [15, 16, 26].



(a) $\mu(U) = 1$. (b) $1 < \mu(U) < 2$ (c) $\mu(U) = 2$

Figure 5.1: Coherence of the one-dimensional subspace $U \subset \mathbb{R}^2$. 5.1a depicts U spanned by $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ with the smallest coherence $\mu(U) = 1$, while 5.1c presents U spanned by one of the standard basis vectors $\mathbf{e}_2$ which achieves maximal coherence $\mu(U) = 2$

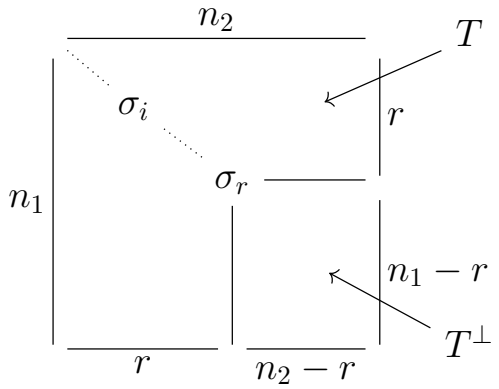


Figure 5.2: The decomposition of $\mathbf{R}^{n_1 \times n_2}$ determined by the SVD of $\mathbf{M}$. Any matrix in $T^\perp$ has rows perpendicular to the left singular vectors and columns perpendicular to the right singular vectors of $\mathbf{M}$ ($\dim(T^\perp) = (n_1 - r)(n_2 - r)$.

Definition 5.3.2. *Let T be the subspace of $\mathbb{R}^{n_1 \times n_2}$,*

$$T = span\{\mathbf{u}_i \mathbf{v}_j^T \mid 1 \leq i \leq r \text{ or } 1 \leq j \leq r \text{ or both}\} \tag{5.19}$$

Then

$$T^\perp = span\{\mathbf{u}_i \mathbf{v}_j^T \mid r+1 \leq i \leq n_1 \text{ and } r+1 \leq j \leq n_2\}. \tag{5.20}$$

The dimension of $\dim(T^\perp)$ is equal to $(n_1 - r)(n_2 - r)$ and $\dim(T) = r(n_1 + n_2 - r)$. The definition implies that any $\mathbf{X} \in T^\perp$ has rows perpendicular to the left singular vectors of $\mathbf{M}$ and columns perpendicular to the right singular vectors. The representation of the discussed decomposition is shown in Fig. 5.2) and was adopted from work of Gross [16].

To build some intuition behind discussed concepts, let us consider completion of a matrix

$$\mathbf{M} = \begin{pmatrix} 2 & 2 \\ 0 & 0 \end{pmatrix}, \tag{5.21}$$

with SVD,

$$\mathbf{M} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{\mathbf{U}} \underbrace{\begin{pmatrix} 2\sqrt{2} & 0 \\ 0 & 0 \end{pmatrix}}_{\mathbf{\Sigma}} \underbrace{\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}}_{\mathbf{V}^T}. \tag{5.22}$$

In accordance with the definition 5.3.1 and the lemma 5.3.1, $\mathbf{M}$ achieves the maximal coherence equal to 2, since the columns of $\mathbf{U}$ are the standard basis vectors $\mathbf{e}_1 = (1, 0)$ and $\mathbf{e}_2 = (0, 1)$.

The rank-one matrices $\mathbf{u}_1\mathbf{v}_1^T, \mathbf{u}_1\mathbf{v}_2^T, \mathbf{u}_2\mathbf{v}_1^T, \mathbf{u}_2\mathbf{v}_2^T$ constitute the orthonormal basis of the $\mathbf{R}^{2\times 2}$. The space T is spanned by the following matrices

$$T = \text{span}\left\{ \underbrace{\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 \end{pmatrix}}_{\mathbf{u}_1\mathbf{v}_1^T}, \underbrace{\begin{pmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 \end{pmatrix}}_{\mathbf{u}_1\mathbf{v}_2^T}, \underbrace{\begin{pmatrix} 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}}_{\mathbf{u}_2\mathbf{v}_1^T} \right\}, \tag{5.23}$$

and

$$T^{\perp} = \text{span}\left\{ \underbrace{\begin{pmatrix} 0 & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}}_{\mathbf{u}_2\mathbf{v}_2^T} \right\}. \tag{5.24}$$

Suppose, we know all of the entries of $\mathbf{M}$ except the one in the upper right corner,

$$\begin{pmatrix} 2 & ? \\ 0 & 0 \end{pmatrix}, \tag{5.25}$$

thus $\Omega = \{(1, 1), (1, 2), (2, 2)\}$. We would like to recover $\mathbf{M}$ with the nuclear norm minimization. Any feasible $\mathbf{X}$ consistent with the observation set, can be parameterized as

$$\mathbf{X} = \begin{pmatrix} 2 & x \\ 0 & 0 \end{pmatrix}. \tag{5.26}$$

In particular, the second row of $\mathbf{X}$ is a vector $(0, 0)$. Let $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ be the coefficients of the matrix $\mathbf{X}$ with respect to the basis $\mathbf{u}_1\mathbf{v}_1^T, \mathbf{u}_1\mathbf{v}_2^T, \mathbf{u}_2\mathbf{v}_1^T, \mathbf{u}_2\mathbf{v}_2^T$.

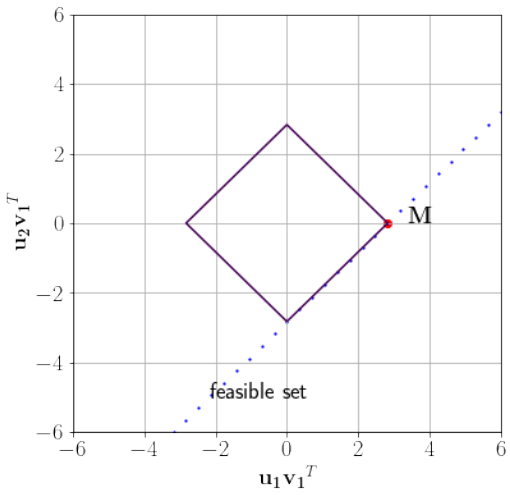


Figure 5.3: Completion of the matrix $\mathbf{M} = \left(\begin{smallmatrix} 2 & 2 \\ 0 & 0 \end{smallmatrix}\right)$, when three entries are known $M = \left(\begin{smallmatrix} 2 & ? \\ 0 & 0 \end{smallmatrix}\right)$. Any feasible $\mathbf{X}$ belongs to the linear space spanned by $\mathbf{u}_1\mathbf{v}_1^T, \mathbf{u}_1\mathbf{v}_2^T$. The dotted lines denote a feasible set for the optimization problem and the violet line denotes the set of the matrices with the same nuclear norm as $\mathbf{M}$ belonging to the two-dimensional subspace span$\{\mathbf{u}_1\mathbf{v}_1^T, \mathbf{u}_1\mathbf{v}_2^T\}$.

Thus, $\alpha_3 = \alpha_4 = 0$. As a consequence, $\mathbf{X}$ belongs to the linear space spanned by $\mathbf{u}_1\mathbf{v}_1^T, \mathbf{u}_1\mathbf{v}_2^T$. Fig. 5.3 presents the projection of the matrix $\mathbf{M}$ onto subspace spanned by $\mathbf{u}_1\mathbf{v}_1^T$ and $\mathbf{u}_1\mathbf{v}_2^T$. The dotted lines denote a feasible set for the optimization problem and the violet line denotes the set of the matrices with the same nuclear norm as $\mathbf{M}$ belonging to the two-dimensional subspace span$\{\mathbf{u}_1\mathbf{v}_1^T, \mathbf{u}_1\mathbf{v}_2^T\}$. We can see that $\mathbf{M}$ is not the only, feasible solution to the optimization problem P2.

5.3.2 Theoretical guarantees for matrix completion

A strong theoretical foundation makes nuclear norm minimization the most developed and well-understood method. It has been shown that under certain conditions, matrix completion problems can be efficiently solved by minimizing the nuclear norm of the matrix. The following theorem is derived from the work of Recht [15] and defines minimal assumptions about the low-rank matrix $\mathbf{M}$.

Theorem 5.3.1 (Theorem 1.1 in [15])**.** *Let $\mathbf{M}$ be a $n_1 \times n_2$ matrix of rank r with the singular value decomposition $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. Without a loss of generality, impose the conventions $n_1 \leq n_2$, $\mathbf{\Sigma}$ is $r \times r$, $\mathbf{U}$ is $n_1 \times r$ and $\mathbf{V}$ is $n_2 \times r$. Assume that*

1. *$\mu_0(\mathbf{M})$ is the coherence of $\mathbf{M}$.*
2. *The matrix $\mathbf{U}\mathbf{V}^T$ has maximum entry bounded by $\mu_1\sqrt{\frac{r}{n_1 n2}}$ in absolute value for some $\mu_1 > 0$.*

Suppose that m entries of $\mathbf{M}$ are observed with locations sampled uniformly with replacement at random. Then if

$$|\Omega| \geq 32 \max[\mu_1^2, \mu_0(\mathbf{M})] r(n_1 + n_2)\beta \log^2(2n_2) \tag{5.27}$$

for some $\beta \geq 1$, the minimizer to the problem

$$\begin{vmatrix} \text{minimize} & \|\mathbf{X}\|_* & \\ \text{s.t.} & \mathbf{X}_{ij} & = \mathbf{M}_{ij} \text{ for } (i,j) \in \Omega \end{vmatrix} \tag{5.28}$$

is unique and equal to $\mathbf{M}$ with probability at least $1 - 6\ln(n_2)(n_1+n_2)^{2-2\beta} - n_2^{2-2\beta^{1/2}}$

The convex optimization, which is based on nuclear norm minimization is the most developed and understood. There are several factors behind the success of the matrix recovery. The key properties include the number and distribution of observed entries, matrix rank and incoherence parameter. When these conditions are satisfied, nuclear norm minimization can recover the original low-rank matrix accurately. For the non-convex alternating minimization, Jain et al. [84] and Hardt [85] gave provable guarantees. However, these guarantees require low coherence, rank and condition number of $\mathbf{M}$, i.e. the coherence must be lower. The nuclear norm minimization can be formulated as SDP and solved with off-the-shelf solvers. However, those solvers are often not accessible for large-scale problems. Thus many first-order methods including Proximal Gradient Descent (PGD) have been developed.

Chapter 6

Column Subset Selection

Low-rank models play a significant role in various fields, including machine learning, data analysis, and signal processing. These models base on approximating data by a low-rank matrix, thereby capturing the most important information or structure while reducing dimensionality. We have already discussed their application to the matrix completion task and collaborative filtering. More broadly, these models enable effective data visualization, feature extraction, and compression by approximating high-dimensional data with a low-rank representation. They help in reducing noise, redundancy, and computational complexity. However, such models may lack interpretability. The derived latent factors or components might not have clear and intuitive interpretations, making it challenging to interpret and extract meaningful insights from the model. For this reason, much research has been directed into creating the low-rank representation using original data points. In this chapter, we discuss two such models. Column Subset Selection (CSS) provides such representation using the column submatrix of the data matrix $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$. On the other hand, CUR decomposition explicitly selects a subset of columns and rows from the original matrix to construct the factorization. In both problems, the crucial step is the selection of a column (or row). Fortunately, for incoherent matrices, uniform sampling is an effective strategy [13, 37]. This considerably increases computational efficiency. Moreover, it is handy in the setup with missing entries in the original matrix. This chapter discusses the role of CSS and CUR in the matrix completion task.

6.1 Column Subset Selection problem

Column Subset Selection (CSS) problem aims to identify the most relevant and informative subset of columns from a given matrix $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ [124]. In machine learning, CSS can serve as an unsupervised feature selection algorithm. In many real-world applications, datasets often contain numerous features, but not all contribute significantly to the predictive task. Moreover, having many irrelevant or redundant features can introduce noise, increase computational complexity, or

lead to overfitting. CSS addresses this task by selecting the d columns with capture as much of $\mathbf{M}$ as possible in terms of spectral or Frobenius norm [35, 125]. Formally, given a matrix $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ and positive integer d, CSS seeks for the column submatrix $\mathbf{C} \in \mathbb{R}^{n_1 \times d}$ such that

$$\|\mathbf{M} - P_C(\mathbf{M})\|_\xi \tag{6.1}$$

is minimized over all possible $\binom{n_2}{d}$ choices for the matrix $\mathbf{C}$. Parameter $\xi \in \{2, F\}$ $\xi = 2$ denotes spectral or $\xi = F$ is the Frobenius norm. Here,

$$P_C = \mathbf{C}\mathbf{C}^\dagger, \tag{6.2}$$

is the projection onto the d-dimensional space spanned by the columns $\mathbf{C}$. The $\mathbf{C}^\dagger$ denotes Moore-Penrose pseudoinverse from the Definition 2.0.11. The Moore-Penrose pseudoinverse, named after E. H. Moore and Roger Penrose, generalises the matrix inverse for not invertible matrices.

Considerable attention has been given to researching column subset selection methods [126–130]. It has been demonstrated that Rank Revealing QR (RRQR) is nearly optimal for addressing the Column Subset Selection problem and is the basis for many CSS algorithms [126, 127]. On the other hand, sampling-based approaches [128–130] attempt to choose columns by randomly selecting from a set of distributions across all columns of an input matrix. If the sample distribution is appropriately chosen, these algorithms are substantially faster than RRQR and achieve comparable performance [129, 130]. The sampling methods include uniform sampling, l_2-norm sampling and leverage scores based sampling.

6.2 CUR factorization

CUR factorization is closely related to the column subset selection problem. CUR factorization is a matrix factorization technique that approximates a given matrix into the product of three matrices: $\mathbf{C}$, $\mathbf{U}$, and $\mathbf{R}$. There are several formulation of CUR decomposition [33, 131, 132] including following,

$$\mathbf{X} \approx \mathbf{C}\mathbf{U}^\dagger\mathbf{R}, \tag{6.3}$$

where $\mathbf{U}^\dagger$ denotes Moore-Penrose pseudoinverse of $\mathbf{U}$ $\mathbf{C}$ and $\mathbf{R}$ are column and row submatrices of $\mathbf{X}$, and $\mathbf{U}$ is the intersection submatrix of $\mathbf{C}$ and $\mathbf{R}$. Hamm proves that once the rank of $\mathbf{U}$ is equal to the rank of $\mathbf{X}$, CUR is an exact factorization [33].

CUR factorization provides a low-rank approximation of the original matrix while retaining important structural properties. The advantage of CUR factorization lies in its ability to offer a low-rank approximation of the original matrix

while using a subset of its columns and rows [33, 36, 133, 134]. This can be beneficial for reducing the dimensionality of the data, compressing information, or extracting essential features. Factorization can be used in various applications such as dimensionality reduction, collaborative filtering, and system identification [109, 135, 136]. CUR factorization provides a more interpretable representation of the data compared to Singular Value Decomposition (SVD), Principal Component Analysis (PCA) or Non-negative Matrix Factorization (NNMF). The $\mathbf{C}$ matrix in CUR consists of selected columns from the original matrix, which directly corresponds to the meaningful features or attributes in the data. This makes it easier to understand and interpret the results in terms of the original variables. In CUR factorization, the $\mathbf{C}$ matrix represents the selected subset of columns, which directly relates to the column subset selection problem. The goal is to choose informative, relevant, or important columns accurately representing the original matrix.

6.3 CSS, CUR, and low-rank matrix completion

In the context of the matrix completion, two fundamental questions may be posed about CSS and CUR. Firstly, how to select meaningful columns (or/and rows) when only a subset of its entries is observed. Regarding the fully-observed input matrix, both problems have been researched extensively [33,36,133,134]. However, observing all matrix entries is frequently difficult or even impossible. For the genetic variation identification challenge, for instance, obtaining the complete DNA sequences of an entire population could be costly and time-consuming [137]. The second question concerns the application of CSS and CUR in matrix completion problems.

For the incoherent matrices, uniform sampling is an effective strategy [13, 37]. Wang and Singh [137] discuss the limitation of the existing sampling schemes for the coherent matrix design and use active sampling schemes as a remedy. This strategy dynamically adjusts the selection of columns based on the information gained during the sampling process. It uses the estimated importance scores to iteratively select additional columns likely to contribute the most valuable information. By adopting the column subset selection process based on the knowledge gained during the sampling and evaluation stages, an adaptive sampling strategy can lead to a more efficient and effective selection of columns. It allows for a targeted exploration of the column space, focusing on the most informative columns while avoiding redundancy or irrelevant features. Such strategies can be beneficial in scenarios where the data distribution or characteristics change dynamically, as they can adapt to the evolving requirements of the problem.

Adaptive column sampling helps to tackle the completion of matrices which do not meet assumptions of the Theorem 5.3.1. Krishnamurthy and Sin designed adaptive sampling algorithms to complete and find a low-rank approximation of highly coherent matrices [98, 100]. This approach also diminishes the sample

complexity of the problem, i.e. the cardinality of the Ω set. The algorithm actively selects columns to observe fully, achieving better sampling complexity than passive sampling frameworks. In [138], authors propose a two-phased matrix completion algorithm based on the leverage scores. In the first phase, the algorithm estimates leverage scores based on uniform sampling. Each matrix entry is sampled in the second phase according to the estimated row and column leverage score. After the resampling matrix is completed using the nuclear norm minimization approach.

In [37], authors present the CUR++, the algorithm for partially observed matrices. Low-rank approximation of the matrix is computed based on the randomly selected columns and rows and a subset of observed entries. The findings and conclusions of this paper played a pivotal role in our research for two reasons. Firstly, the authors show that under assumptions about matrix incoherence, a high-quality column subset can be obtained by sampling each column uniformly at random. Secondly, the authors formulate the least squares optimization problem for the matrix recovery based on the fully observed columns and rows.

In [13], authors demonstrate the potential of the CUR factorization in low-rank matrix completion. They introduce a new sampling strategy and non-convex Iterative CUR completion algorithm. The so-called Cross-Concentrated Sampling bridges uniform sampling and CUR sampling and offers additional flexibility that may reduce sampling costs potentially. The authors provide provable guarantees for matrix completion with Cross-Concentrated Sampling.

The Column Subset Selection and CUR matrix decomposition provide a low-rank approximation of the original matrix while explicitly selecting a subset of columns (or columns and rows). Interpretability and computational efficiency have contributed to their popularity and adoption in various domains. They have data compression, feature selection, system identification, and machine learning applications. The decomposition quality is determined by the columns (and row) subset selection. The vast amount of conducted research refers to the fully-observed matrices and may be impractical in real-data applications. Incorporating matrix completion algorithms into the column subset selection process can enhance the selection of informative columns by considering the missing entries and utilizing the information provided by the observed entries.

There is a shortage of studies integrating column subset selection with matrix completion. Several researchers, ourselves included, have recognized the need to address the research gap between column subset selection and matrix completion. The connection between CUR/CSS and matrix completion lies in the fact that both techniques can be used for data approximation. Matrix completion can be seen as a way to fill in missing entries of a matrix, which approximates the complete matrix. CUR decomposition, on the other hand, provides a low-rank approximation of a given matrix by selecting a subset of columns and rows. Sometimes, matrix completion algorithms can leverage the CUR decomposition or CSS problem. In the application discussed in the literature, modification of

the sampling pattern allowed to completion of highly coherent matrices or reduced sampled complexity of the problem. Depending on the problem characteristic passive or active sampling schemes can be employed. This combination of CUR/CSS decomposition and matrix completion can improve results in scenarios with missing data and low-rank structure.

The CUR and CSS may reduce the problem size in matrix completion methods by only considering a smaller set of columns for the matrix completion task [13, 109]. The next chapter introduces Column Selected Matrix Completion (CSMC). CSMC selects a representative subset of columns to reduce the computational complexity and memory requirements of the problem.

Chapter 7

Matrix Completion using Column Subset Selection

This chapter introduces a novel method for the low-rank matrix completion problem, which we call Column Selected Matrix Completion (CSMC). The CSMC draws inspiration from the broad techniques in the literature discussed in chapters 5 and 6. In particular, it employs the Column Subset Selection (CSS) algorithms to reduce the size of the matrix completion problem. Recovering the most important columns considerably limits the computational costs of the matrix completion task.

In the CSMC, the completion of the matrix $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ is divided into two stages. In the first stage, CSMC selects the subset of d columns and fills it with an established matrix completion algorithm. In the second stage, the entries recovered in the first stage and previously known entries are used to recover $\mathbf{M}$. This is done by solving the standard least squares problem.

The motivation behind the second stage is that if $\mathbf{C}$ is a solution to the Column Subset Selection problem, then

$$\mathbf{M} \approx \mathbf{C}\mathbf{C}^{\dagger}\mathbf{M}, \tag{7.1}$$

with respect to the Frobenius or spectral norm, where $\mathbf{C}^{\dagger}$ denotes Moore-Penrose pseudo-inverse (Definition 2.0.11).

The CSMC tackles the task of filling thick ($n_1 << n_2$) or thin ($n_1 >> n_2$) matrices. Our analysis is focused on the case when $\mathbf{M}$ is thick. The recovery of the thin matrix can be reduced to the recovery of a thick matrix using the transpose operator. Thick matrices are encountered in various tasks, including image recognition, natural language processing, and speech recognition. The Netflix Prize, discussed in Chapter 3, is an example of a thick matrix completion problem. The number of users is significantly bigger than the number of movies.

By selecting the informative columns, CSMC reduces the problem's dimensionality and improves the completion process's efficiency. Thus, this strategy allows

the application of Semidefinite Programming (SDP) solvers for larger matrices. It addresses computer memory limitations.

The least squares problem is a widely used technique in machine learning, particularly for regression problems. Leading researchers has extensively worked on it and developed many efficient algorithms to solve it. Randomized linear algebra methods involve sampling techniques or matrix sketching, making them suitable for large-scale problems [30, 139–142]. On the other hand, the least squares problem can be solved with iterative methods, including stochastic gradient descent (SGD). At each iteration, SGD uses only a subset of the data (one sample or a mini-batch). Thus it is suitable for large-scale problems where calculation of the gradient over the entire dataset may be computationally expensive [142–148].

The success of our CSMC depends on several factors. The column submatrix matrix must meet the assumptions of the matrix completion (MC) algorithm used in the first stage, including rank condition, sampling pattern and number of observations. Here, we consider nuclear norm based algorithms, whose theoretical guarantees additionally depend on the coherence parameter. When the column submatrix $\mathbf{C} \in \mathbb{R}^{n_1 \times d}$ selected according to the uniform distribution, the desired properties are preserved [109]. Secondly, a subset of columns from the matrix $\mathbf{M}$ must capture the essential information needed to complete the matrix. Fortunately, when a matrix is incoherent, uniform sampling is an effective strategy [37].

Our approach takes inspiration from the work of Xu et al. [37] and their CUR++ algorithm for partially observed matrices discussed in Section 6.3. To obtain a low-rank approximation, the CUR++ method involves randomly selecting rows and columns and a sample of observed matrix entries, which are then used to solve a linear regression problem. In the case of the low-rank matrices, the algorithm can serve as the matrix recovery method. However, CSMC differs from CUR++ in three essential aspects. Firstly, we do not observe whole columns. Instead, we fill them with a selected matrix completion algorithm. Secondly, recovery of $\mathbf{M}$ is based only on selected columns, not columns and rows. Finally, CUR++ requires finding singular vectors of column and row submatrices. CSMC does not and thus limits computational burden.

Our work is also closely related to the other method discussed in Section 6.3, i.e. Matrix Completion with Cross-Concentrated Sampling (MCCCS). The MCCCS bridges uniform sampling and CUR sampling in matrix completion problem [13]. The authors of [13] also propose a non-convex iterative scheme (ICURC). The CSMC leverages their result of how incoherence is transformed into column (and row) submatrices [13, 109]. However, CSMC does not require more observations on selected columns and rows.

In the following sections, we discuss our CSMC in detail. We introduce three algorithms implementing the CSMC intending to recover the matrix of the various dimensions. We also discuss the key aspects that contribute to the algorithm's success. Notably, we formulate assumptions underlying the theoretical guarantees of the CSMC. The notation and definitions are provided in Chapter 2.

7.1 Column Selected Matrix Completion method

The CSMC is a versatile method for solving matrix completion problems. This section introduces the algorithms implementing the CSMC method:

1. **Column Selected Nuclear Norm (CSNN)** in which submatrix is filled with the exact nuclear norm minimization using SDP solver.
2. **Column Selected Proximal Gradient Descent (CSPGD)** in which inexact nuclear norm minimization is solved by Proximal Gradient Descent (PGD).
3. **CSPGD-adam** a variant of CSPGD in which the least squares problem is solved using Adam optimization solver [143].

Each of the presented CSMC methods adopts the following procedure (Fig. 7.1).

Stage I: Sample and fill

In the stage I, the CSMC selects d columns of $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ according to the uniform distribution. Let $I \subseteq \{1, \cdots, n_2\}$ denote the set of the indices of the selected columns. The goal of this stage is to recover the submatrix formed by selected columns, $\mathbf{C} := \mathbf{M}_{:I}$, $\mathbf{C} \in \mathbb{R}^{n_1 \times d}$ using the chosen matrix completion algorithm.

Stage II: Solve least squares

Let $\hat{\mathbf{C}} \in \mathbb{R}^{n_1 \times d}$ be the output of the Stage I. Now, the CSMC solves the following convex optimization problem,

$$\min_{\mathbf{Z} \in \mathbb{R}^{d \times n_2}} \frac{1}{2} \|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\hat{\mathbf{C}}\mathbf{Z})\|_F^2, \tag{P10}$$

where $\mathcal{R}_\Omega : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n_1 \times n_2}$ is the operator following Definition 2.0.3. Let $\hat{\mathbf{Z}} \in \mathbb{R}^{d \times n_2}$ be a solution to the problem P10. The CSMC output with the matrix $\hat{\mathbf{M}} = \hat{\mathbf{C}}\hat{\mathbf{Z}}$.

The problem P10 is a variant of the least squares problem, which minimize the sum of the squares over the set of observed entries Ω. It can be efficiently solved with convex solvers or as a variant of the least squares problem. In Section 7.2, we conduct an in-depth analysis of the theoretical guarantees for the CSMC.

We will refer as CSMC-α to the CSMC in which in Stage I, the number of selected columns is given by $d = \lfloor \alpha \cdot n_2 \rfloor$, i.e. α denotes the ratio of sampled columns in the first stage. Algorithm 1 contains the CSMC-α scheme.

Below, we present three algorithms implementing the CSMC method. Those algorithms employ various optimization solvers depending on the size of the matrix completion problem.

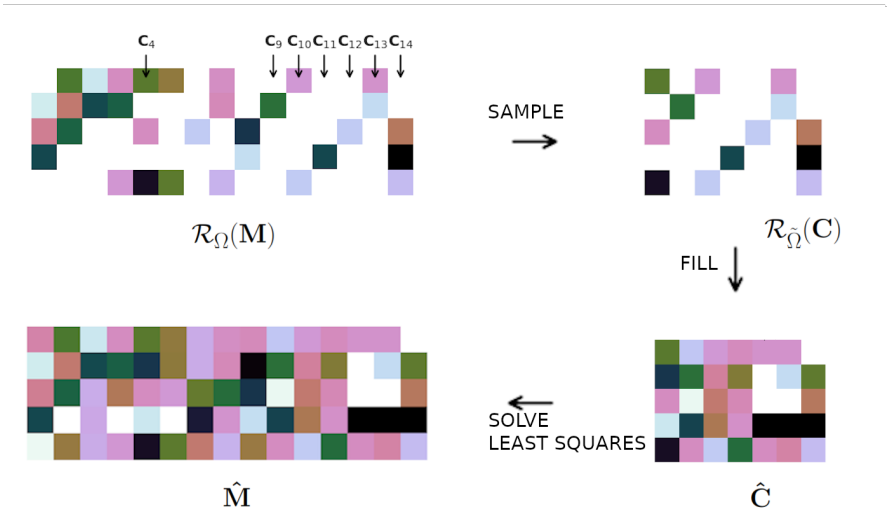


Figure 7.1: The CSMC method.

Algorithm 1 CSMC-α

Require: $\mathcal{R}_\Omega(\mathbf{M})$: $n_1 \times n_2$-matrix
Require: α: Ratio of the selected columns
$I \leftarrow$ Uniformly sample $\alpha \cdot n_2$ indices
$\mathbf{C}_{\text{missing}} \leftarrow \mathcal{R}_\Omega(\mathbf{M})_{:I}$
$\hat{\mathbf{C}} \leftarrow \text{MC}(\mathbf{C}_{\text{missing}})$ ▷ Complete submatrix $\mathbf{C} \in \mathbb{R}^{n_1 \times d}$.
$\hat{\mathbf{Z}} \leftarrow \arg\min_{\mathbf{Z} \in \mathbb{R}^{d \times n_2}} \frac{1}{2}\|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\hat{\mathbf{C}}\mathbf{Z})\|_F^2$
$\hat{\mathbf{M}} \leftarrow \hat{\mathbf{C}}\hat{\mathbf{Z}}$
return $\hat{\mathbf{M}}$

7.1.1 Column Selected Nuclear Norm (CSNN)

The CSNN is dedicated to recovering the small and medium size $\mathbf{M}$. It inherits the theoretical guarantees of the SDP solvers.

Stage I: Sample and fill

After selecting the column submatrix $\mathbf{C} \in \mathbb{R}^{d \times n_2}$, CSNN solves the following optimization problem,

$$\begin{vmatrix} \text{minimize} & \|\mathbf{Y}\|_* & \\ \text{s.t.} & \mathbf{Y}_{ji} & = \mathbf{C}_{ji} \text{ for } (j,i) \in \tilde{\Omega}, \end{vmatrix} \tag{P11}$$

where $\tilde{\Omega}$ denote the set of indices of the known entries in $\mathbf{C}$. We found the first-order Splitting Conic Solver (SCS) as an efficient way to solve the SDP [149, 150]. SCS employs a first-order algorithm based on operator-splitting techniques. It solves the dual formulation of the SDP problem, using a combination of primal and dual variables and applies a fast iterative solver based on the alternating direction method of multipliers (ADMM) [151]. This approach allows for efficient and scalable solutions to SDP problems.

Stage II: Solve least squares

To find the optimum of the regression problem P10, we directly solve the least squares for the observed entries in each column. This approach takes advantage of the explicit form of the cost function as the Frobenius norm. Solving large-scale linear systems is the core of many scientific and machine-learning problems. One of the most remarkable recent development in numerical linear algebra is new randomized algorithms that are fast, scalable, robust, and reliable [30, 32, 152–156]. This approach is simple and allows to create fairly efficient distributed implementations. We will refer to this subroutine as Direct Least Squares (DLS) (Algorithm 2). Let $\hat{\mathbf{Z}}$ be the output of the DLS. The CSNN results with $\hat{\mathbf{M}} = \hat{\mathbf{C}}\hat{\mathbf{Z}}$.

Algorithm 2 Direct Least Squares (DLS) for P10

DECLARE $\hat{\mathbf{Z}}$: ARRAY$[d, n_2]$
for $i = 1, \cdots, n_2$ **do**
  $\hat{\mathbf{z}}_i \leftarrow \arg\min_{z \in \mathbb{R}^{d \times n_2}} \frac{1}{2}\|\mathcal{R}_\Omega(\mathbf{M})_{:i} - \mathcal{R}_\Omega(\hat{\mathbf{C}}\mathbf{z})_{:i}\|_2^2$
  $\hat{\mathbf{Z}}_{:i} \leftarrow \hat{\mathbf{z}}_i$
end for
return $\hat{\mathbf{Z}}$

7.1.2 Column Selected Proximal Gradient Descent (CSPGD)

For large-scale problems, we solve the matrix completion problem with the Proximal Gradient Descent method. The CSPGD benefits from sampling a subset of columns due to the per-iteration cost of the PGD. In the most straightforward implementation for nuclear norm regularization, the proximal operator involves performing a singular value thresholding operation, which can be relatively expensive, especially for large matrices.

Stage I: Sample and fill

Complete the selected column submatrix $\mathbf{C}$ by solving following optimization problem,

$$\underset{\mathbf{Y}\in\mathbb{R}^{n_1\times d}}{\text{minimize}}\ \frac{1}{2}\|\mathcal{R}_{\tilde{\Omega}}(\mathbf{Y})-\mathcal{R}_{\tilde{\Omega}}(\mathbf{C})\|_F^2+\lambda\|\mathbf{Y}\|_*. \tag{P12}$$

The problem is solved by the PGD routine described in Algorithm 3, where λ is a regularization parameter.

Algorithm 3 Proximal Gradient Descent (Soft-Impute)

Require: ϵ: Error tolerance
Initialize $\mathbf{Y}^{(0)}=0$
for $t=1,2,\ldots,T$ **do**
  $\mathbf{Y}^{(t)}\leftarrow D_\lambda(\mathcal{R}_{\tilde{\Omega}}(\mathbf{C})+\mathcal{R}_{\tilde{\Omega}^\perp}(\hat{\mathbf{Y}}^{(\mathbf{t-1})}))$
  if $\frac{\|(\hat{\mathbf{C}}^{(\mathbf{t})})-\mathbf{Y}^{(\mathbf{t-1})})\|_F^2}{\|\hat{\mathbf{Y}}^{(\mathbf{t-1})}\|_F^2}\le\epsilon$ **then**
    break
  end if
end for
$\hat{\mathbf{C}}\leftarrow\hat{\mathbf{Y}}^{(t)}$

Stage II: Solve least squares

As the CSNN, the CSPGD solves the least squares problem using DLS 2. Let $\hat{\mathbf{Z}}$ be the obtained minimizer of the problem P10. The CSPGD outputs with the matrix $\hat{\mathbf{M}}=\hat{\mathbf{C}}\hat{\mathbf{Z}}$.

7.1.3 Column Selected Proximal Gradient Descent - Adam (CSPGD-adam))

For large-scale problems, solving the least squares problem with the fast convex solver may be suitable. The CSPGD-adam algorithm solves the problem P10 using the Adam solver. Adam is an algorithm for the first-order gradient-based optimization of stochastic objective functions based on adaptive estimates

of lower-order moments [143]. Adam (short for Adaptive Moment Estimation) is widely used in machine learning and extends the Stochastic Gradient Descent (SGD). The primary difference between Adam and SGD is that Adam uses adaptive learning rates (stepsize), whereas SGD uses a fixed learning rate (stepsize). Thus, SGD may lead to slow convergence or overshooting the minimum of the function. In contrast, Adam dynamically adapts the learning rate for each decision variable. Additionally, while SGD only uses the current gradient to update the parameters, Adam uses a moving average of the past gradients to prevent oscillations. The adaptive learning rates and using past gradients in Adam make it more robust and efficient than SGD. However, there are cases where SGD can be more effective.

Stage I: Sample and fill

This stage is the same as in the CSPGD algorithm.

Stage II: Solve least squares

The function $f : \mathbb{R}^{d\times n_2} \to \mathbb{R}$,

$$f(\mathbf{Z}) = \frac{1}{2}\|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\hat{\mathbf{C}}\mathbf{Z})\|_F^2, \tag{7.2}$$

is minimized using the Adam procedure (Algorithm 4).

Algorithm 4 Adam

Require: τ: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: ϵ: Small number to avoid zero division error
Require: $f(\mathbf{Z})$: Objective function
Require: $\mathbf{Z}^{(0)}$: Initial value of $\mathbf{Z}$
for $t = 1, \cdots, T$ **do**
  $g_t \leftarrow \nabla f^{(t-1)}(\mathbf{Z}^{(t-1)})$
  $\eta_t \leftarrow \beta_1 \cdot \eta_{t-1} + (1 - \beta_1) \cdot g_t$ ▷ Update biased first moment estimate
  $\omega_t \leftarrow \beta_2 \cdot \omega_{t-1} + (1 - \beta_2) \cdot g_t^2$ ▷ Update biased second raw moment estimate
  $\hat{\eta}_t \leftarrow \frac{\eta_t}{(1-\beta_1^t)}$ ▷ Compute bias-corrected first moment estimate
  $\hat{\omega}_t \leftarrow \frac{\omega_t}{(1-\beta_2^t)}$ ▷ Compute bias-corrected first moment estimate
  $\mathbf{Z}^{(t)} \leftarrow \mathbf{Z}^{(t-1)} - \frac{\tau\cdot\hat{\eta}_t}{(\sqrt{\hat{\omega}_t}+\epsilon)}$ ▷ Update parameters
end for
return $\mathbf{Z}^{(t)}$

7.2 Formal analysis of CSMC

Theoretical guarantees allow the performance evaluation and bring confidence, reliability, and problem understanding. In this section, we provide the theoretical guarantees for the CSMC. We perform a formal analysis for each stage of

the proposed convex CSMC. The goal of this section is to answer the following questions.

1. Can the uniformly sampled $\mathbf{C}$ be correctly recovered by the nuclear norm minimization algorithm?

2. Is the solution of the least squares problem P10 a good approximation of the matrix $\mathbf{M}$?

We assume that $\mathbf{C} \in \mathbb{R}^{n_1 \times d}$ is a column submatrix of $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$. $\mathbf{C}$ is obtained by sampling uniformly without replacement d columns of $\mathbf{M}$. Let r be the rank of $\mathbf{M}$ and let $\tilde{r}$ be the rank of $\mathbf{C}$. We assume that the compact SVD of $\mathbf{C}$ is given by,

$$\mathbf{C} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^{\mathbf{T}}, \tag{7.3}$$

where $\tilde{\mathbf{U}} \in \mathbb{R}^{n_1 \times \tilde{r}}$, $\tilde{\mathbf{V}} \in \mathbb{R}^{n_2 \times \tilde{r}}$, and $\tilde{\mathbf{\Sigma}} \in \mathbb{R}^{\tilde{r} \times \tilde{r}}$.

We denote as $\mu_0(\mathbf{M})$ and $\mu_0(\mathbf{C})$ the coherence parameters of $\mathbf{M}$ and $\mathbf{C}$. We also refer to the condition number of $\mathbf{M}$,

$$\kappa(\mathbf{M}) := \|\mathbf{M}\|_2 \|\mathbf{M}^\dagger\|_2 = \frac{\sigma_{\max}(\mathbf{M})}{\sigma_{\min}(\mathbf{M})}, \tag{7.4}$$

where $\sigma_{\max}(\mathbf{M})$ and $\sigma_{\min}(\mathbf{M})$ denote maximum and minimum non-zero singular value of $\mathbf{M}$ respectively, and $\mathbf{M}^\dagger \mathbb{R}^{n_2 \times n_1}$ is a pseudoinverse of $\mathbf{M}$ (Definition 2.0.11).

7.2.1 Formal analysis of the first stage of CSMC

To address the first question, it is necessary to understand how the characteristics of the problem described in Theorem 5.3.1 are transferred to the task of filling in the submatrix $\mathbf{C}$. These properties include:

1. **Matrix rank**: $\mathbf{C}$ consists of randomly selected d columns of $\mathbf{M}$, the rank of $\mathbf{C}$, rank of $\mathbf{C}$, denoted as $\tilde{r}$), is bounded by,

   $$\tilde{r} \leq \min\{r, d\}, \tag{7.5}$$

   where r is the rank of $\mathbf{M}$.

2. **Number and distribution of observed entries:** The entries of $\mathbf{C}$ also follow uniform distribution, and the expected number of observed entries in $\mathbf{C}$ is equal to

   $$\tilde{m} = |\Omega| \frac{d}{n_2}. \tag{7.6}$$

3. **Matrix coherence** Following theorem shows that under certain assumptions, the coherence of $\mathbf{C}$ bounds depends on the coherence and condition number of $\mathbf{M}$.

   Theorem 7.2.1 (Corollary 3.6 in Cai et al. [109]). *Suppose that $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ has rank r and coherence bounded by $\mu_0(\mathbf{M})$. Suppose that $I \subseteq \{1, \dots n_2\}$ is chosen by sampling uniformly without replacement to yield $\mathbf{C} = \mathbf{M}_{:I}$. Let d be the number of sampled columns such that $d \geq 1.06\mu_0(\mathbf{M}) r \ln(rn_2)$. Then*

$$\mu_0(\mathbf{C}) \leq 100\kappa^2(\mathbf{M})\mu_0(\mathbf{M}), \tag{7.7}$$

   with a probability at least $1 - \frac{1}{n_2}$, where $\kappa(\mathbf{M})$ (7.4)denotes condition number of $\mathbf{M}$.

The following findings enable an examination of the efficiency of Stage I of CSMC. They demonstrate the transfer of three key features from a matrix completion task to the task of completing a column submatrix obtained by sampling uniformly columns of $\mathbf{M}$.

7.2.2 Formal analysis of the second stage of CSMC

In the following analysis of the second stage of CSMC, we assume that submatrix $\mathbf{C}$ is fully observed or perfectly recovered. We focus on the recovery ability of the least-squares problem in the second stage of the CSMC. In particular, we provide assumptions implying that solving

$$\min_{\mathbf{Z} \in \mathbb{R}^{d \times n_2}} \frac{1}{2} \|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\mathbf{CZ})\|_F^2 \tag{P13}$$

will output with $\hat{\mathbf{Z}} \in \mathbb{R}^{d \times n_2}$, such that $\mathbf{M} = \mathbf{C}\hat{\mathbf{Z}}$ holds with high probability.

The following analysis is guided by the analysis of the CUR+ in the work of Xu et al. [37]. However, in contrast to CUR+, solving P13 does not require computing singular vectors of $\mathbf{C}$. The main result of this section is given by the Theorem 7.2.2. We believe that the following result broadens the result of Xu et al. formulated as Theorem 2 in [37].

Theorem 7.2.2. *Let r be the rank of $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ and let $\mathbf{C} \in \mathbb{R}^{n_1 \times d}$ be a column submatrix of $\mathbf{M}$ formed by uniformly sampled without replacement d columns. Let $\tilde{r}$ denote rank of $\mathbf{C}$. Assume that for the parameter $\gamma > 0$,*

1. $d \geq 7\mu_0(\mathbf{M}) r(\gamma + \ln r)$,
2. $|\Omega| \geq \tilde{r} n_2 \mu_0(\mathbf{C}) \left(\gamma + \ln\left(\frac{n_2 \tilde{r}}{2}\right)\right)$.

Let $\hat{\mathbf{Z}} \in \mathbb{R}^{d \times n_2}$ *be the minimizer of the problem P13. Then, with a probability at least* $(1 - 3e^{-\gamma})$*, we have* $\mathbf{M} = \mathbf{C}\hat{\mathbf{Z}}$*.*

We denote as $\tilde{U}$ the subspace spanned by the first $\tilde{r}$ singular vectors of $\mathbf{C}$. The orthogonal projection onto $\tilde{U}$ is given by

$$P_{\tilde{U}} = \tilde{\mathbf{U}}\tilde{\mathbf{U}}^T. \tag{7.8}$$

The following lemma can be found in [37] and shows that under the incoherence assumption, uniform sampling outputs a high-quality solution to the CSS problem, with high quality.

Theorem 7.2.3 (Theorem 9 in Xu et al. [37])**.** *Let* $\tilde{U}_\psi$ *denote subspace spanned by the* ψ *first singular vectors of* $\mathbf{C}$ *and let* $P_{\tilde{U}_\psi}$ *denote orthogonal projection on* $\tilde{U}_\psi$*. Then for parameter* $\gamma > 0$*, with probability* $1 - 2e^{-\gamma}$ *we have*

$$\|\mathbf{M} - P_{\tilde{U}_\psi}\mathbf{M}\|_2^2 \leq \sigma_{\psi+1}^2 \left(1 + \frac{2n_2}{d}\right), \tag{7.9}$$

if $d \geq 7\mu(U_\psi)\psi(\gamma + \ln \psi)$*, where* d *denotes number of sampled columns in* $\mathbf{C}$*,* $\sigma_{\psi+1}$ *is the* $(\psi+1)$*-th largest singular value of* $\mathbf{M}$ *and* $\mu(U_\psi)$ *is the coherence of the subspace spanned by* ψ *first left singular vectors of* $\mathbf{M}$*,*

The following remark is the immediate consequence of the Theorem 7.2.3.

Remark 7.2.1. *Let* $\psi \geq r$*, where* r *is the rank of* $\mathbf{M}$*. Then* $\sigma_{\psi+1} = \sigma_{r+1} = 0$ *and with probability* $1 - 2e^{-\gamma}$*,*

$$\|\mathbf{M} - P_{\tilde{U}}\mathbf{M}\|_2^2 = 0, \tag{7.10}$$

provided that $d \geq 7\mu_0(\mathbf{M})r(\gamma + \ln r)$*.*

We will now bound the distance measured in the spectral norm between $\mathbf{M}$ and $\hat{\mathbf{M}} = \mathbf{C}\hat{\mathbf{Z}}$. To do that, we will assume that the objective function $g : \mathbb{R}^{\tilde{r} \times n_2} \to \mathbb{R}$,

$$g(\mathbf{Y}) = \frac{1}{2}\|\mathcal{R}_\Omega(\mathbf{M}) - \tilde{\mathbf{U}}\mathbf{Y}\|_F^2 \tag{7.11}$$

is strongly convex [121], where the strong convexity is defined as follows

Definition 7.2.1. *Function* $g : \mathbb{R}^{\tilde{r} \times n_2} \to \mathbb{R}$ *is strongly convex with parameter* β *if* $h(\mathbf{X}) = g(\mathbf{X}) - \frac{\beta}{2}\|\mathbf{X}\|_F^2$ *is convex.*

The following remarks provide a helpful characterization of strongly convex functions.

Remark 7.2.2. *A function $g : \mathbb{R}^{\tilde{r}\times n_2} \to \mathbb{R}$ is strongly convex with parameter β if g is everywhere differentiable and*

$$g(\mathbf{Y}) \geq f(\mathbf{X}) + \langle \nabla g(\mathbf{X}), \mathbf{Y} - \mathbf{X}\rangle + \frac{\beta}{2}\|\mathbf{Y} - \mathbf{X}\|_F^2, \tag{7.12}$$

for any $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{\tilde{r}\times n_2}$, where inner product follows Definition 2.0.10.

Remark 7.2.3. *If g is twice differentiable, then $g(\mathbf{X})$ is strongly convex with parameter β if $\nabla^2 g(\mathbf{X}) \succeq \beta\mathbf{I}$ for any $\mathbf{X} \in \mathbb{R}^{\tilde{r}\times n_2}$.*

We now provide the formula for the first-order and the second-order partial derivatives of g. Let y_{sw} denote the (s, w) entry of the matrix $\mathbf{Y}$, and m_{lw} denote (l, w) entry of $\mathbf{M}$ then the partial derivative of g with respect to y_{sw} is given by following formula,

$$\frac{\partial g}{\partial y_{sw}} = -\sum_{l:(l,w)\in\Omega}(m_{lw} - \sum_{i=1}^{\tilde{r}}\tilde{u}_{li}y_{iw})\tilde{u}_{ls} = \sum_{l:(l,w)\in\Omega}(\sum_{i=1}^{\tilde{r}}\tilde{u}_{li}y_{iw} - m_{lw})\tilde{u}_{ls}. \tag{7.13}$$

The second-order partial derivative with respect to y_{pq} and y_{sw} is equal to

$$\frac{\partial^2 g}{\partial y_{sw}\partial y_{pq}} = \begin{cases} \sum\limits_{l:(l,w)\in\Omega} \tilde{u}_{ls}\tilde{u}_{lp} & w = q, \\ 0 & w \neq q. \end{cases} \tag{7.14}$$

The following theorem shows that under certain assumptions, $\hat{\mathbf{M}} = \mathbf{C}\hat{\mathbf{Z}}$ is close to the matrix $\mathbf{M}$ in the spectral norm. Those assumptions include that i) $P_{\tilde{U}}\mathbf{M}$ is close to $\mathbf{M}$, ii) g is strongly convex with β. The following proofs take inspiration from the work of Xu et al. [37].

Theorem 7.2.4. *Suppose that $\|\mathbf{M} - P_{\tilde{U}}\mathbf{M}\|_2^2 \leq \Delta$ for the parameter $\Delta > 0$, and let $\tilde{U}$ be the subspace of $\mathbb{R}^{n_1}$ spanned by the first $\tilde{r}$ singular vectors of $\mathbf{C}$, $P_{\tilde{U}} = \tilde{\mathbf{U}}\tilde{\mathbf{U}}^{\mathbf{T}}$. Assume that g is strongly convex with a parameter β, then*

$$\|\mathbf{M} - \hat{\mathbf{M}}\|_2^2 \leq \Delta + \frac{2n_1\tilde{r}\Delta}{\beta}. \tag{7.15}$$

Proof. Set $\mathbf{Y}_1 = \tilde{\mathbf{U}}^{\mathbf{T}}\mathbf{M}$, then from the assumptions

$$\|\mathbf{M} - \tilde{\mathbf{U}}\mathbf{Y}_1\|_2^2 \leq \Delta. \tag{7.16}$$

By remark 2.0.2

$$\frac{1}{r'}\|\mathbf{M} - \tilde{\mathbf{U}}\mathbf{Y}_1\|_F^2 \leq \|\mathbf{M} - \tilde{\mathbf{U}}\mathbf{Y}_1\|_2^2 \leq \Delta, \tag{7.17}$$

where r' denotes the rank of the matrix $(\mathbf{M} - \tilde{\mathbf{U}}\mathbf{Y}_1) \in \mathbb{R}^{n_1 \times n_2}$ and

$$\|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\tilde{\mathbf{U}}\mathbf{Y}_1)\|_F^2 \leq \|\mathbf{M} - \tilde{\mathbf{U}}\mathbf{Y}_1\|_F^2 \leq r'\Delta. \tag{7.18}$$

Since any of the matrix dimensions bounds rank, $r' \leq n_1$ (for thick matrices $n_1 < n_2$)

$$\|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\tilde{\mathbf{U}}\mathbf{Y}_1)\|_F^2 \leq n_1\Delta. \tag{7.19}$$

Let $\hat{\mathbf{Z}}$ be a solution to the problem P13, then

$$f(\hat{\mathbf{Z}}) = \|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\mathbf{C}\hat{\mathbf{Z}}\|_F^2 = \|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T\hat{\mathbf{Z}}\|_F^2 \tag{7.20}$$

and $\hat{\mathbf{Y}} := \tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T\hat{\mathbf{Z}}$ must be a minimum of g. Indeed, assume that $g(\mathbf{Y}_2) < g(\hat{\mathbf{Y}})$ for some $\mathbf{Y}_2 \neq \hat{\mathbf{Y}}$, i.e.

$$\begin{aligned} \|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\mathbf{C}\hat{\mathbf{Z}})\|_F^2 &= \|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\tilde{\mathbf{U}}\hat{\mathbf{Y}})\|_F^2 \\ &> \|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\tilde{\mathbf{U}}\mathbf{Y_2})\|_F^2 \\ &= \|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\mathbf{C}\underbrace{\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\mathbf{Y_2}}_{\mathbf{X}_2})\|_F^2 \end{aligned} \tag{7.21}$$

and $f(\mathbf{X}_2) < f(\hat{\mathbf{Z}})$ where f is the objective function in problem P13.

We will bound distance between $\mathbf{Y}_1$ and $\hat{\mathbf{Y}}$ using strong convexity of g. Since $\hat{\mathbf{Y}}$ is minimum of $g(\mathbf{Y})$ (7.11), then $\nabla g(\hat{\mathbf{Y}}) = \mathbf{0}$, and

$$\langle \nabla g(\hat{\mathbf{Y}}), \mathbf{Y}_1 - \hat{\mathbf{Y}} \rangle = 0. \tag{7.22}$$

Thus, by the Remark 7.2.2,

$$g(\mathbf{Y}_1) \geq g(\hat{\mathbf{Y}}) + \frac{\beta}{2}\|\mathbf{Y}_1 - \hat{\mathbf{Y}}|_F^2. \tag{7.23}$$

Using the definition of g (7.11),

$$\begin{aligned} \frac{\beta}{2}\|\mathbf{Y}_1 - \hat{\mathbf{Y}})\|_F^2 &\leq \|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\tilde{\mathbf{U}}\mathbf{Y}_1)\|_F^2 - \|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\tilde{\mathbf{U}}\hat{\mathbf{Y}})|_F^2 \\ &\leq \|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\tilde{\mathbf{U}}\mathbf{Y}_1)\|_F^2. \end{aligned} \tag{7.24}$$

Combining (7.24) and (7.19),

$$\|\mathbf{Y}_1 - \hat{\mathbf{Y}}\|_F^2 \leq \frac{2}{\beta}\|\mathcal{R}_\Omega(\mathbf{M}) - \mathcal{R}_\Omega(\tilde{\mathbf{U}}\mathbf{Y}_1)\|_F^2 \leq \frac{2n_1}{\beta}\Delta. \tag{7.25}$$

Using the triangle inequality and the fact that Frobenius norm of a matrix is always greater or equal to its spectral norm (Remark 2.0.2),

$$\begin{aligned}\|\mathbf{M}-\hat{\mathbf{M}}\|_2^2 &= \|\mathbf{M}-\mathbf{C}\hat{\mathbf{Z}}\|_2^2 \\ &\leq \|\mathbf{M}-\tilde{\mathbf{U}}\tilde{\mathbf{U}}^{\mathbf{T}}\mathbf{M}\|_2^2 + \|\tilde{\mathbf{U}}\tilde{\mathbf{U}}^{\mathbf{T}}\mathbf{M}-\mathbf{C}\hat{\mathbf{Z}}\|_2^2 \\ &\leq \|\mathbf{M}-P_{\tilde{U}}\mathbf{M}\|_2^2 + \|\tilde{\mathbf{U}}\tilde{\mathbf{U}}^{\mathbf{T}}\mathbf{M}-\mathbf{C}\hat{\mathbf{Z}}\|_F^2.\end{aligned} \tag{7.26}$$

The first component on the right side is bounded by the assumptions,

$$\|\mathbf{M}-P_{\tilde{U}}\mathbf{M}\|_2^2 \leq \Delta. \tag{7.27}$$

To bound the second component, we use the definition of the $\mathbf{Y}_1$ and $\hat{\mathbf{Y}}$, and the fact that the product of the Frobenius norms bounds Frobenius norm of the two matrices.

$$\begin{aligned}\|\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\mathbf{M}-\tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T\hat{\mathbf{Z}}\|_F^2 &= \|\tilde{\mathbf{U}}\mathbf{Y}_1-\tilde{\mathbf{U}}\hat{\mathbf{Y}}\|_F^2 \\ &\leq \|\tilde{\mathbf{U}}\|_F^2\|\mathbf{Y}_1-\hat{\mathbf{Y}}\|_F^2\end{aligned} \tag{7.28}$$

Using eq. 7.25 and the fact that $\|\tilde{\mathbf{U}}\|_F = \sqrt{\hat{r}}$, since columns of $\tilde{\mathbf{U}} \in \mathbb{R}^{n_1\times\tilde{r}}$ are orthonormal and have length equal to 1.

$$\|\tilde{\mathbf{U}}\|_F^2\|\mathbf{Y}_1-\hat{\mathbf{Y}}\|_F^2 \leq \frac{2n_1\tilde{r}\Delta}{\beta}, \tag{7.29}$$

implying

$$\|\mathbf{M}-\hat{\mathbf{M}}\|_F^2 \leq \Delta + \frac{2n_1\tilde{r}\Delta}{\beta}. \tag{7.30}$$

□

To bound strong convexity, we will use Remark 7.2.3 and bound the smallest eigenvalue of the Hessian of g. Following [37], to do that, we will use the following result of Tropp [125].

Theorem 7.2.5 (Theorem 5 in Xu et al. [37] derived from Theorem 2.2 in Tropp [125])**.** *Let $\mathcal{X}$ be a finite set of the positive-semidefinite (PSD) matrices with dimension $k \times k$, and suppose that*

$$\max_{\mathbf{X}\in\mathcal{X}} \lambda_{\max}(\mathbf{X}) \leq B \tag{7.31}$$

for some parameter $B > 0$, where $\lambda_{\max}(\mathbf{X})$ is the maximum eigenvalue of $\mathbf{X}$). Sample $\{\mathbf{X}_1, \ldots, \mathbf{X}_\Psi\}$ uniformly from $\mathcal{X}$ without replacement. Compute:

$$\mu_{\min} := \Psi\lambda_{\min}(\mathbb{E}\mathbf{X}_1), \tag{7.32}$$

and

$$\mu_{\max} := \Psi\lambda_{\max}(\mathbb{E}\mathbf{X}_1), \tag{7.33}$$

where $\mathbb{E}\mathbf{X}_1$ *is the expected value of a random variable* $\mathbf{X}_1$, $\lambda_{\max}(\mathbb{E}\mathbf{X}_1)$ *and* $\lambda_{\min}(\mathbb{E}\mathbf{X}_1)$ *denote its maximum and minimum eigenvalue.*

$$P\big(\lambda_{\min}(\sum_{j=1}^{\Psi}\mathbf{X}_j) \leq (1-\rho)\mu_{\min}\big) \leq k\exp\frac{-\mu_{\min}}{B}[(1-\rho)\ln(1-\rho)+\rho], \tag{7.34}$$

for parameter $\rho \geq 0$.

Theorem 7.2.6. *Let* $\gamma > 0$ *be a parameter. With probability* $1 - e^{-\gamma}$ *we have that the objective function* $g : \mathbb{R}^{\tilde{r}\times n_2} \to \mathbb{R}$ *defined in (7.11) is* β*-strongly convex, provided that*

$$|\Omega| \geq \tilde{r}n_2\mu(\tilde{U})\left(\gamma + \ln(\frac{n_2\tilde{r}}{2})\right). \tag{7.35}$$

Proof. By remark 7.2.3 to bound strong convexity, we can instead bound the smallest eigenvalue of the Hessian matrix $\mathbf{H} = \nabla^2 g$.

The Hessian matrix of a function g is a $\tilde{r}n_2 \times \tilde{r}n_2$ matrix. Let assume that second-order derivative with respect to the y_{sw} and y_{pq} entries of matrix $\mathbf{Y}$ is the $\tilde{r}(s-1)+w$, $\tilde{r}(p-1)+q$ entry of the Hessian matrix. Then using eq. 7.14 the Hessian matrix $\mathbf{H}$ can be written as

$$\begin{pmatrix} \mathbf{H}^{1,1} & \mathbf{H}^{1,2} & \dots & \mathbf{H}^{1,\tilde{r}} \\ \mathbf{H}^{2,1} & \ddots & \dots & \mathbf{H}^{2,\tilde{r}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}^{\tilde{r},1} & \mathbf{H}^{\tilde{r},2} & \dots & \mathbf{H}^{\tilde{r},\tilde{r}} \end{pmatrix} \tag{7.36}$$

where $\mathbf{H}^{s,q}$ is a diagonal $n_2 \times n_2$ matrix containing partial derivatives $\frac{\partial^2 g}{\partial y_{sw}\partial y_{pq}}$ for $w, q \in \{1, \dots, n_2\}$.

$$\mathbf{H}^{s,q} = \begin{pmatrix} \sum\limits_{l:(l,1)\in\Omega} \tilde{u}_{ls}\tilde{u}_{lp} & 0 & \dots & 0 \\ 0 & \ddots & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sum\limits_{l:(l,n_2)\in\Omega} \tilde{u}_{ls}\tilde{u}_{lp} \end{pmatrix} \tag{7.37}$$

Then the Hessian of g is a sum of the random matrices,

$$\mathbf{H} = \sum_{(i,j)\in\Omega} \tilde{\mathbf{u}}_{i:}\tilde{\mathbf{u}}_{i:}^T \otimes \mathbf{e}_j\mathbf{e}_j^T, \tag{7.38}$$

where $\mathbf{e}_j \in \mathbb{R}^{n_2}$ is a standard basis vector and $\tilde{\mathbf{u}}_{i:} \in \mathbb{R}^r$ is a vector defined by the i-th row of matrix $\tilde{\mathbf{U}}$.

Thus the Hessian of g is a sum of the $|\Omega|$ random matrices of the form

$$\mathbf{H}^{i,j} := \tilde{\mathbf{u}}_{i:}\tilde{\mathbf{u}}_{i:}^T \otimes \mathbf{e}_j\mathbf{e}_j^T, \tag{7.39}$$

where $\tilde{\mathbf{u}}_{i:}\tilde{\mathbf{u}}_{i:}^T \in \mathbb{R}^{\tilde{r}\times\tilde{r}}$ is PSD and $\mathbf{e}_j\mathbf{e}_j^T \in \mathbb{R}^{n_2\times n_2}$ is also PSD. Thus,

$$\mathbf{H} = \sum_{(i,j)\in\Omega} \mathbf{H}^{i,j}. \tag{7.40}$$

Each $\mathbf{H}^{i,j}$ is PSD as the Kronecker product of the two PSD matrices. Moreover,

$$\begin{aligned} \lambda_{\max}(\mathbf{H}^{i,j}) &= \lambda_{\max}(\tilde{\mathbf{u}}_{i:}\tilde{\mathbf{u}}_{i:}^T) \\ &\leq \max_{1\leq j\leq r} |u_{ij}|^2 \\ &\leq \frac{\tilde{r}\mu(\tilde{U})}{n_1}, \end{aligned} \tag{7.41}$$

for each $i = 1, \cdots n_1$, $j = 1, \cdots n_2$. Thus,

$$\max_{ij} \lambda_{\max}(\mathbf{H}^{i,j}) \leq \frac{\tilde{r}\mu(\tilde{U})}{n_1}. \tag{7.42}$$

Let (i_1, j_1) be the first pair of indices in Ω, i.e., the indices of the first observed entry in $\mathbf{M}$. The expected value of the random matrix $\mathbf{H}^{i_1,j_1}$ is given by

$$\begin{aligned} \mathbb{E}(\mathbf{H}^{i_1,j_1}) &= \frac{1}{n_1 n_2}\sum_{l=1}^{n_1}\sum_{q=1}^{n_2} \mathbf{H}^{l,q} \\ &= \frac{1}{n_1 n_2}\tilde{\mathbf{U}}^T\tilde{\mathbf{U}} \otimes \mathbf{I}_{n_1\times n_1} \\ &= \frac{1}{n_1 n_2}\mathbf{I}_{\tilde{r}\times\tilde{r}} \otimes \mathbf{I}_{n_1\times n_1} \\ &= \frac{1}{n_1 n_2}\mathbf{I}_{\tilde{r}n_1\times\tilde{r}n_1}. \end{aligned} \tag{7.43}$$

Following the notation of the theorem 7.2.5,

$$\mu_{\min} = |\Omega|\lambda_{\min}(\mathbb{E}(\mathbf{H}^{i_1,j_1})) = \frac{|\Omega|}{n_1 n_2}, \tag{7.44}$$

and

$$\max_{ij} \lambda_{\max}(\mathbf{H}^{i,j}) \leq \frac{\tilde{r}\mu(\tilde{U})}{n_1} = B. \tag{7.45}$$

Combining theorem 7.2.5 with $\rho = \frac{1}{2}$ and eq. 7.40

$$\Pr(\lambda_{\min}(\mathbf{H}) \leq \frac{1}{2}\mu_{\min}) \leq \tilde{r}n_2 \exp\left(\frac{-\mu_{\min}}{B}\right)\left(\frac{1-\ln 2}{2}\right) \leq \frac{rn_2}{2}\exp\left(\frac{-\mu_{\min}}{B}\right), \tag{7.46}$$

implying,

$$\Pr\left(\lambda_{\min}(\mathbf{H}) \leq \frac{|\Omega|}{2n_1n_2}\right) \leq \frac{\tilde{r}n_2}{2}\exp\left(-\frac{|\Omega|}{rn_2\mu(\tilde{U})}\right), \tag{7.47}$$

Hence, with probability at least $1 - e^{-\gamma}$,

$$\lambda_{\min}(\mathbf{H}) \geq \frac{|\Omega|}{2n_1n_2}, \tag{7.48}$$

provided that

$$|\Omega| \geq \tilde{r}n_2\mu(\tilde{U})\left(\gamma + \ln(\frac{n_2\tilde{r}}{2})\right). \tag{7.49}$$

□

Theorem 7.2.2 can be proved by combining the results of 7.2.4 and 7.2.6.

Proof of Theorem 7.2.2. Since $d \geq 7\mu_0(\mathbf{M})r(\gamma + \ln r)$, from the Remark 7.2.1, we have

$$\|\mathbf{M} - P_{\tilde{U}}\mathbf{M}\|_2^2 = 0, \tag{7.50}$$

with the probability at least $1 - 2e^{-\gamma}$.

From Theorem 7.2.6, the fact that $\mu_0(\mathbf{C}) > \mu(\tilde{U})$
and the fact $|\Omega| \geq \tilde{r}n_2\mu_0(\mathbf{C})\left(\gamma + \ln(\frac{n_2\tilde{r}}{2})\right)$, function g is β-strongly convex with probability at least $1 - e^{-\gamma}$.

The probability the fact, that eq. (7.50) holds and g is β-strongly convex is grater or equal than the product $(1 - 2e^{-\gamma})(1 - e^{-\gamma})$.

Thus we can apply 7.2.4 with $\Delta = 0$ and show that

$$\|\mathbf{M} - \hat{\mathbf{M}}\|_2^2 \leq 0, \tag{7.51}$$

implying $\mathbf{M} = \hat{\mathbf{M}}$ with probability at least $(1 - 3e^{-\gamma})$.

□

This chapter presented the Column Selected Matrix Completion, a two-staged method for the matrix completion problem. In the first stage, CSMC randomly selects the column submatrix of $\mathbf{M}$ and completes it with the selected MC algorithm. In the second stage, CSMC recovers $\mathbf{M}$ by minimizing the least squares error on the previously observed and filled-in the first stage entries.

This chapter gives a systematic, formal analysis of the CSMC algorithm. We discuss how the crucial properties of the $\mathbf{M}$ are inherited by the uniformly sampled column submatrix. Those properties include matrix rank, coherence, and the distribution of the observed entries Ω. This offers valuable insight into the first stage of the CSMC.

We have also investigated the theoretical guarantees of the second stage of the CSMC. Here, we assume that we know all of the entries of $\mathbf{C}$. We either observed them, or we perfectly recovered them in the first stage. The results of this analysis are provided in Theorem 5.3.1. The underlying assumptions refer to the rank and the coherence of $\mathbf{M}$, the rank and the coherence of $\mathbf{C}$, the number of columns in $\mathbf{C}$ and the size of Ω. Those results deepen the understanding of the performance of the CSMC. In particular, the rank and the coherence of $\mathbf{C}$ can be further bounded by the rank and the coherence of $\mathbf{M}$.

Chapter 8

Numerical evaluation

To evaluate the performance of the proposed Column Selected Matrix Completion (CSMC) methods, i.e. Column Selected Nuclear Norm (CSNN), Column Selected Proximal Gradient Descent (CSPGD), and CSPGD-adam, we conducted numerous experiments on the synthetic data set in a controlled setting. The primary goal of this chapter is to compare the performance of the CSMC algorithms with the matrix completion methods based on convex optimization, particularly nuclear norm minimization with Semidefinite Programming (NN algorithm) and Proximal Gradient Descent (PGD algorithm). To assess the sample complexity of the presented methods, i.e. the number of measurements required for the successful completion, we vary the cardinality of the Ω set. We also explore the recovery ability of CSMC methods depending on the number of sampled columns. Moreover, we benchmark the CSMC and other low-rank matrix completion methods - Alternating Minimization (AM) and Iterative SVD (ISVD) methods. Finally, we address the scalability demands and explore the performance of PGD-based methods for matrices with numerous columns. Since direct least squares might be slower in this case, we compare CSPGD, PGD, CSPGD-adam, and PGD-adam.

8.1 Implementation overview

The author has developed and released open-source code with CSNN minimization algorithm and CSPGD methods. Both algorithms support Numpy arrays [157] and Pytorch tensors [158]. The latter can benefit from GPU acceleration. The semidefinite programming is solved with the Splitting Conic Solver (SCS) [119]. The code is written in Python 3.10 and provided in `https://github.com/akrajewska/css-matrix-completion`.

All testing examples were implemented in Python 3.10. The related codes are available at `https://github.com/akrajewska/css-matrix-completion/examples`. The Alternating Minimization and Iterative SVD algorithms are taken from the *fancyimpute* library and used for benchmarking ([159]).

8.2 Experimental setup

8.2.1 Data set

The goal of each experiment was to recover random $n_1 \times n_2$ matrix $\mathbf{M}$ with the ratio of missing entries ρ. To control its rank r, the test matrix was generated as a product of the $n_1 \times r$ matrix $\mathbf{A}$ and $r \times n_2$ matrix $\mathbf{B}$. Matrices $\mathbf{A}$ and $\mathbf{B}$ were generated in two steps. In the first one, matrix entries were sampled from the normal distribution $\mathcal{N}(0, 1)$. In the second step, noise matrices with the ratio 0.3 of non-zero entries were added to each matrix. The $\mathbf{M}$ matrix was generated according to formula 8.1,

$$\underset{(n_1 \times n_2)}{\mathbf{M}} = \underset{(n_1 \times r)}{\mathbf{A}} \underset{(r \times n_2)}{\mathbf{B}}. \tag{8.1}$$

Let Ω_ρ be the set of known indices from Definition 2.0.1 of a the cardinality $\rho n_1 n_2$, and $\mathcal{R}_{\Omega_\rho}$ follow Definition 2.0.3. The input matrix $\mathcal{R}_{\Omega_\rho}(\mathbf{M})$ was generated randomly. The entries of $\mathbf{M}$ with indices outside the set Ω_ρ were set as null values. Experiments were conducted on the three types of data sets with a dimension (n_1, n_2) equal to (300×1000), (2000×3000) and (600×10000) and maximal rank $r \in \{5, 10, 15, 25, 50\}$. Each trial of the experiments was run on a different test matrix.

8.2.2 Experimental procedures

We conducted four series of experiments to investigate the performance of matrix completion methods on both smaller and larger problem sizes. These experiments aimed to compare the effectiveness and efficiency of matrix completion techniques under different problem-size scenarios. S I and S II were conducted on the data set of smaller matrices. In S III and S IV, we focused on larger datasets to investigate the scalability and performance of the presented method on more substantial problem sizes.

Experiment S I The goal of the first experiment was to compare the CSNN-α for $\alpha \in \{0.1, \cdots, 0.9\}$ algorithm and the exact Nuclear Norm minimization (NN) algorithm. Each experiment run was conducted on the randomly generated $\mathbf{M} \in \mathbb{R}^{300 \times 1000}$ under one of the rank settings $r \in \{5, 10\}$. To assess the sample complexity of the algorithms, we varied the ratio of known entries $\rho \in \{0.4, 0.3, 0.2, 0.1\}$. Every setup was executed over $N_{\text{trial}} = 20$ independent trial runs. The performance of the algorithms NN and CSNN-α, $\alpha = \{0.1, \cdots, 0.9\}$ was evaluated over all trials.

Experiment S II The goal of the second experiment was to benchmark CSNN-α for $\alpha \in \{0.1, \cdots, 0.5\}$ algorithm, Matrix Factorization [83, 87, 88], Iterative

SVD [91], and exact Nuclear Norm minimization (NN). As in Experiment S I, each experiment run was conducted on the randomly generated $\mathbf{M} \in \mathbb{R}^{300\times1000}$ under one of the rank settings $r \in \{5, 10\}$. The ratio of known entries ρ was taken from the set $\{0.2, 0.1\}$. Every setup was executed over $N_{\text{trial}} = 20$ independent trial runs.

Experiment S III In the third scenario, we evaluated the sample complexity and performance of the CSPGD and equated it with the PGD algorithm. The data set consisted of randomly generated 2000×3000 matrices of rank $r \in \{5, 10, 15, 25, 50\}$. The ratio of known entries was equal $\rho = \{0.5, 0.3, 0.1\}$. We examined different value of the parameter $\lambda = \{0.1, \ldots, \sigma_{max}\}$, where λ is a regularization parameter for the problems P12 and P4 for CSPGD and PGD, respectively, and $\sigma_{\max}$ denote the largest singular value of the matrix $\mathcal{R}_{\Omega_\rho}(\mathbf{M})$. Every setup was executed over $N_{\text{trial}} = 20$ independent trial runs.

Experiment S IV The last experiment was designed to compare CSPGD and PGD algorithms with the CSPGD-adam algorithm, in which the least squares problem was solved using the Adam optimization scheme [143]. Again, we tested different regularization parameter values $\lambda = \{0.1, \ldots, \sigma_{max}\}$, where $\sigma_{\max}$ denote the largest singular value of the matrix $\mathcal{R}_{\Omega_\rho}(\mathbf{M})$. The initial point matrix for the Adam optimizer was sampled from the distribution $\mathcal{N}(0, 1)$, the learning rate (stepsize) was set to $1e^{-04}$ and the maximum number of iterations was set to 1000. The exponential decay parameters $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 1e^{-08}$ remained as default values as suggested in the original paper [143]. The experiment was conducted on the set of 600×10000 matrices of rank $r \in \{5, 10, 15\}$ with the rate of known entries $\rho = \{0.2, 0.1\}$. Every setup was executed over $N_{\text{trial}} = 20$ independent trial runs. Each of the algorithm PGD, CSPGD-α and CSPGD-adam-α, for $\alpha = \{0.3, 0.4, 0.5\}$ was evaluated over all trials.

Performance measures

Following [13, 27, 160], we the quality of the solution $\hat{\mathbf{M}}$ will be expressed as a relative error,

$$\epsilon := \frac{\|\mathbf{M} - \hat{\mathbf{M}}\|_F}{\|\mathbf{M}\|_F} \tag{8.2}$$

After Cai et al. [13], the run of the experiment will be considered to be successfully solved if

$$\epsilon \leq 10^{-2}. \tag{8.3}$$

To inquire how many columns should be sampled to achieve a satisfying algorithm error, we calculate an empirical cumulative distribution function (ECDF)

to display the proportion of trials achieved given approximation error [161, 162]. ECDF is defined as $\hat{F}_S : \mathbb{R}_+ \to [0, 1]$,

$$\hat{F}_S(a) = \frac{|\{s|\epsilon_s \leq a\}|}{|S|}, \tag{8.4}$$

Where each trial is represented as s, ϵ_s denotes the relative error reached by the solution of s, and S denotes the set of all trials for the given parameters setup.

Moreover, we compare two other quality measures widely used in real data matrix completion tasks. Normalized Mean Absolute Error (NMAE) calculated as

$$\text{NMAE} = \frac{1}{|\Omega_{\text{test}}|(m_{\max} - m_{\min})} \sum_{(i,j)\in\Omega_{\text{test}}} |\hat{\mathbf{M}} - \mathbf{M}|, \tag{8.5}$$

where $m_{\max}$ and $m_{\min}$ denote the maximum and minimum rating, respectively, and Ω_{test} denote the set of indices in test set. This metric can be found in [13,163,164] and will be a reference for collaborative filtering tasks. We also consider ECDF for NMAE values.

$$\hat{G}_S(a) = \frac{|\{s|\text{NMAE}_s \leq a\}|}{|S|}, \tag{8.6}$$

where each trial is represented as s, NMAE_s denotes the approximation error reached by the solution of s, and S denotes the set of all of the trials for the given parameters setup.

The image reconstruction quality is often measured by signal-to-noise ratio (SNR) [13] and is defined as

$$\text{SNR}(\hat{\mathbf{M}}) = 20 \log_{10} \left(\frac{\|\mathbf{M}\|_F}{\|\hat{\mathbf{M}} - \mathbf{M}\|_F} \right). \tag{8.7}$$

SNR quantifies the level of a desired signal relative to the level of unwanted noise or interference in a system. A higher SNR indicates a stronger and more distinguishable signal relative to the noise, which is desirable in most applications. Conversely, a lower SNR means that the noise level is relatively high compared to the signal, making it more challenging to discern the signal accurately.

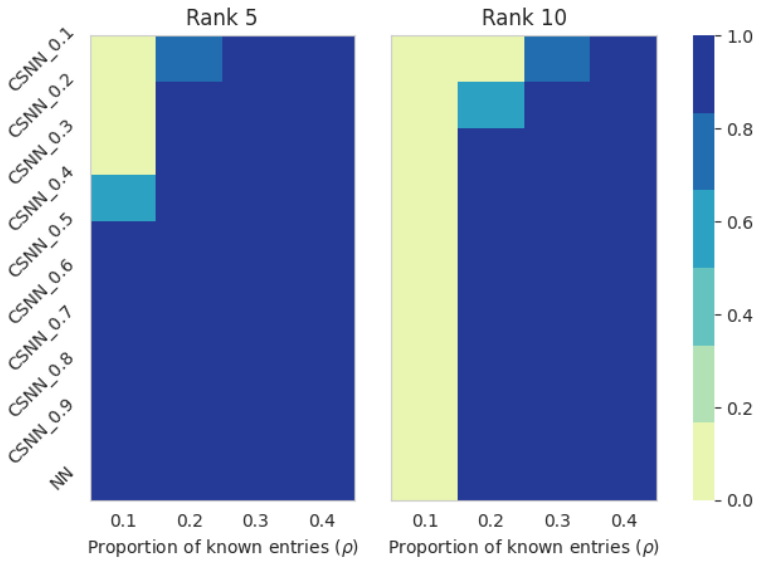
Finally, we compare the analyzed algorithms' runtime (in seconds). Since CSMC methods consist of two stages, while other algorithms are one staged, we do not compare iteration numbers of algorithms to achieve solutions of prescribed quality.

8.3 Results of experiments

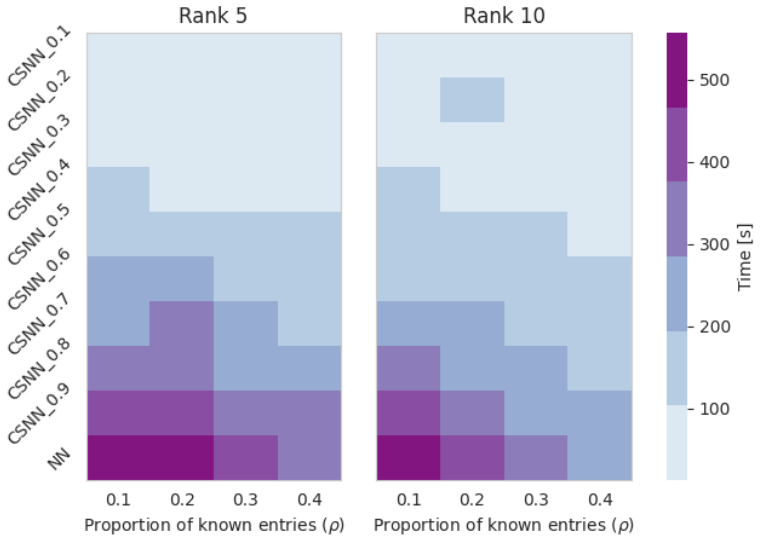
Experiments S I and S II were executed on the Linux workstation equipped with 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz (8 cores) and 32GB RAM. Experiments S III and S IV were run on a Linux station equipped with one NVIDIA T4 Tensor Core GPU 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz (4 cores) and 16 GiB RAM.

Experiment S I Firstly, we discuss results for the NN and CSNN-α, for $\alpha = \{0.1, \ldots, 0.9\}$, algorithms in completing the 300×1000 matrix. We expect a trade-off between the probability of a successful recovery and runtime. The lower column sampling rates should result in shorter runtimes. However, the quality of the obtained solutions can suffer. As presented in Fig. 8.1, completing the rank-5 matrices was successful in every trial, for every tested α, if the known part ρ was greater or equal to 0.3. In the case of the rank-10 matrices, $\rho = 0.4$ guaranteed successful recovery for every α. For 80% missing entries, CSNN-0.2 and CSNN-0.3 output with good solutions for rank 5 and 10, respectively. In the case of the matrices of rank 5, CSNN-0.5 succeeded with empirical probability 0.8 when $\rho = 0.1$. To inquire how many columns should be sampled to achieve a satisfying error of the algorithm, we compare ECDF (8.4) plots for the relative error depending on the matrix rank and sample frequency (Fig. 8.2). To inquire how many columns should be sampled to achieve satisfying runtime depending on ρ, we compare the runtime distribution over the matrix rank and sample frequency (Fig. 8.3). It can be seen that CSNN-0.1 was over ten times faster than NN (Fig. 8.3). For 10% known entries, CSNN-0.5 succeeded in 137 seconds, while NN required 474 seconds on average (see Table 8.1).

Fig. 8.4 displays NMAE (8.5) and Fig. 8.5 presents SNR (8.7) for all of the experimental setups. All CSNN-α with $\alpha \geq 3$ achieved comparable SNR and NMAE values as NN. Table 8.1 presents mean relative error and runtime values for NN and CSNN-α algorithms for the rank-5 matrix, while Table 8.2 presents results for the matrix with rank 10.



(a) Empirical probability of the successful recovery for matrix.



(b) Runtime distribution measured in seconds.

Figure 8.1: Probability of successful recovery of $\mathbf{M} \in \mathbb{R}^{300\times1000}$ and runtime distribution measured in seconds; NN and CSNN-α, $\alpha \in \{0.1, \cdots, 0.9\}$.

Known entries ratio (ρ) / Algorithm	Relative error (ϵ) 0.4	0.3	0.2	0.1	Time (s) 0.4	0.3	0.2	0.1
CSNN_0.1	**4.122e-06**	**3.708e-06**	**6.613e-03**	2.106e+00	**1.076e+01**	**1.374e+01**	**4.538e+01**	9.614e+00
CSNN_0.2	3.287e-06	5.717e-06	4.341e-06	2.393e-01	2.287e+01	2.397e+01	3.217e+01	2.598e+01
CSNN_0.3	2.567e-06	3.045e-06	3.839e-06	2.507e-02	4.638e+01	5.396e+01	6.519e+01	6.213e+01
CSNN_0.4	2.839e-06	1.823e-03	4.393e-06	1.100e-02	6.452e+01	7.624e+01	8.809e+01	9.253e+01
CSNN_0.5	2.547e-06	2.627e-06	3.085e-06	**3.141e-03**	9.119e+01	1.087e+02	1.280e+02	**1.369e+02**
CSNN_0.6	2.297e-06	2.712e-06	1.880e-06	1.853e-03	1.221e+02	1.424e+02	1.791e+02	1.794e+02
CSNN_0.7	2.552e-06	2.423e-06	1.799e-06	1.892e-03	1.559e+02	1.888e+02	2.483e+02	2.370e+02
CSNN_0.8	1.509e-06	2.346e-06	1.841e-06	1.485e-03	2.056e+02	2.364e+02	3.141e+02	3.016e+02
CSNN_0.9	2.361e-06	2.207e-06	2.161e-06	1.506e-03	2.518e+02	3.028e+02	3.821e+02	3.686e+02
NN	**3.031e-06**	**2.738e-06**	**2.724e-06**	**7.016e-04**	**3.149e+02**	**3.545e+02**	**4.842e+02**	**4.739e+02**

Table 8.1: Results S I: NN and CSNN-α, $\alpha \in (0.1, 0.9)$, $\mathbf{M} \in \mathbb{R}^{300\times1000}$, rank $r = 5$.

Known entries ratio (ρ) / Algorithm	Relative error (ϵ) 0.4	0.3	0.2	0.1	Time [s] 0.4	0.3	0.2	0.1
CSNN_0.1	**4.698e-06**	**4.154e-03**	7.119e-01	7.220e+00	**1.300e+01**	**4.799e+01**	1.359e+01	1.657e+01
CSNN_0.2	4.545e-06	4.117e-06	1.062e-02	1.036e+00	2.447e+01	3.040e+01	1.834e+02	4.383e+01
CSNN_0.3	2.424e-06	5.201e-06	**7.934e-05**	5.563e-01	5.222e+01	5.304e+01	**8.009e+01**	9.077e+01
CSNN_0.4	2.284e-06	4.148e-06	1.220e-03	3.652e-01	7.068e+01	7.665e+01	9.899e+01	1.127e+02
CSNN_0.5	2.183e-06	4.053e-06	6.413e-04	2.757e-01	8.944e+01	1.097e+02	1.425e+02	1.410e+02
CSNN_0.6	1.816e-06	2.459e-06	3.005e-06	2.159e-01	1.154e+02	1.514e+02	1.794e+02	1.796e+02
CSNN_0.7	1.462e-03	2.800e-06	3.946e-06	1.800e-01	1.403e+02	1.804e+02	2.247e+02	2.260e+02
CSNN_0.8	1.858e-06	3.253e-06	4.070e-06	1.503e-01	1.720e+02	2.236e+02	2.709e+02	3.093e+02
CSNN_0.9	4.096e-04	2.659e-06	3.750e-04	1.258e-01	2.132e+02	2.841e+02	3.300e+02	4.168e+02
NN	**1.635e-06**	**2.228e-06**	**4.961e-06**	**1.050e-01**	**2.443e+02**	**3.391e+02**	**3.860e+02**	**5.568e+02**

Table 8.2: Results S I; NN and CSNN-α, $\alpha \in (0.1, 0.9)$, $\mathbf{M} \in \mathbb{R}^{300\times1000}$, rank $r = 10$.

Experiment S II Experiment S II benchmarks performance of CSNN-α, $\alpha \in \{0.1, \ldots, 0.5\}$ algorithms with Matrix Factorization (MF) [84, 85], Iterative SVD [91] and NN algorithms. The known entries rate ρ was set to 0.1, and the rank of the matrix was either 5 or 10. Presented results were obtained for the best maximal rank parameter MF and Iterative SVD used. Fig. 8.6 displays ECDF plots for the relative error ϵ. Iterative SVD did not succeed for both rank settings. CSNN-0.1 failed to recover the original rank-10 matrices (Fig. 8.6). MF was faster than CSNN-α algorithms (Fig. 8.7). The magnitude of the relative error for CSNN-0.3 and CSNN-0.5 was significantly smaller for rank-5 and rank-10 cases, respectively. In that case, both algorithms returned a solution of superior quality.

Experiment S III We discuss results for the completion of the 2000×3000 matrix with algorithms PGD and CSPGD-α for $\alpha \in \{0.3, 0.4, 0.5\}$, known entries rate $\rho \in \{0.5, 0.3, 0.1\}$ and matrix rank $r \in \{5, 10, 15, 25, 50\}$. As shown in Fig. 8.10) ECDF values were similar for all of the algorithms. In almost all test scenarios, algorithms succeeded in matrix completion tasks. Only for $\rho = 0.1$ and $r = 50$ the relative error was greater than 10^{-2}. Algorithms CSPGD-α and PGD failed in this test scenario. In all successful trials, CSPGD-0.3 offered significant runtime savings (Fig. 8.11). For $\rho = 0.1$, CSPGD-0.3 completed the task in 123 seconds, while PGD required 375 seconds (see Table 8.4). We want to emphasize

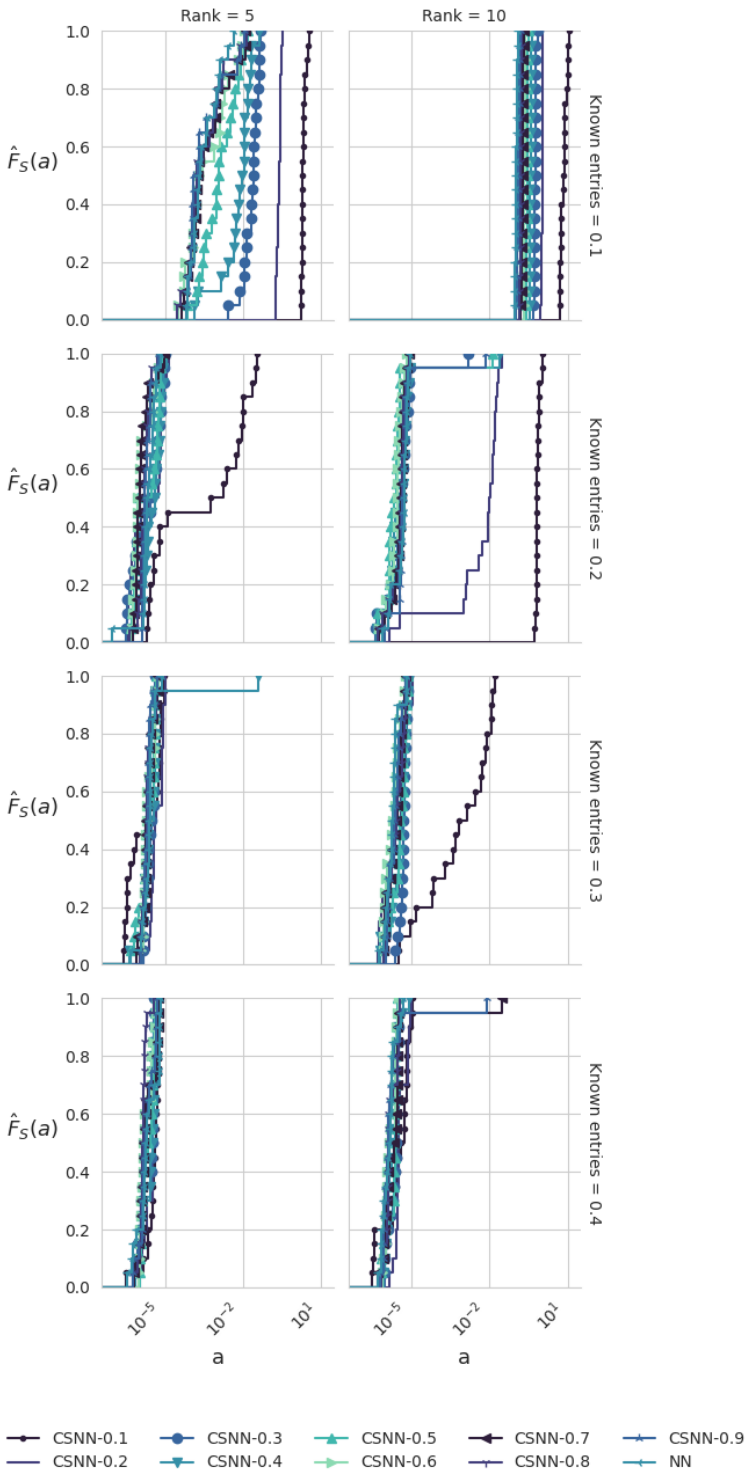


Figure 8.2: Results S I: ECDF (8.4) curves depending on the matrix rank and rate of the known entries for NN and CSNN-α, $\alpha \in \{0.1, \cdots, 0.9\}$, $\mathbf{M} \in \mathbb{R}^{300\times1000}$.

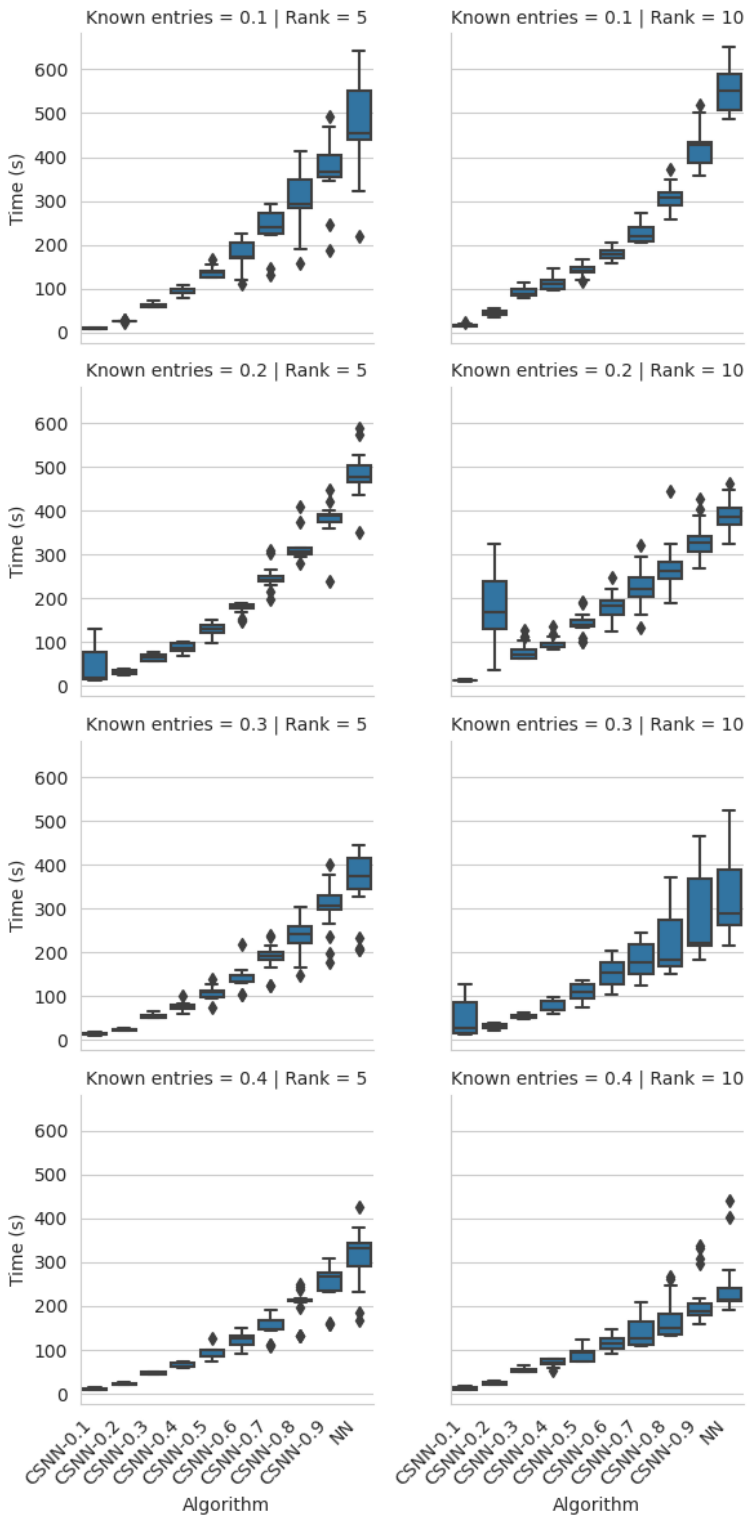


Figure 8.3: Results S I: Runtime distribution depending on the matrix rank and rate of the known entries for NN and CSNN-α, $\alpha \in \{0.1, \cdots, 0.9\}$, $\mathbf{M} \in \mathbb{R}^{300\times1000}$.

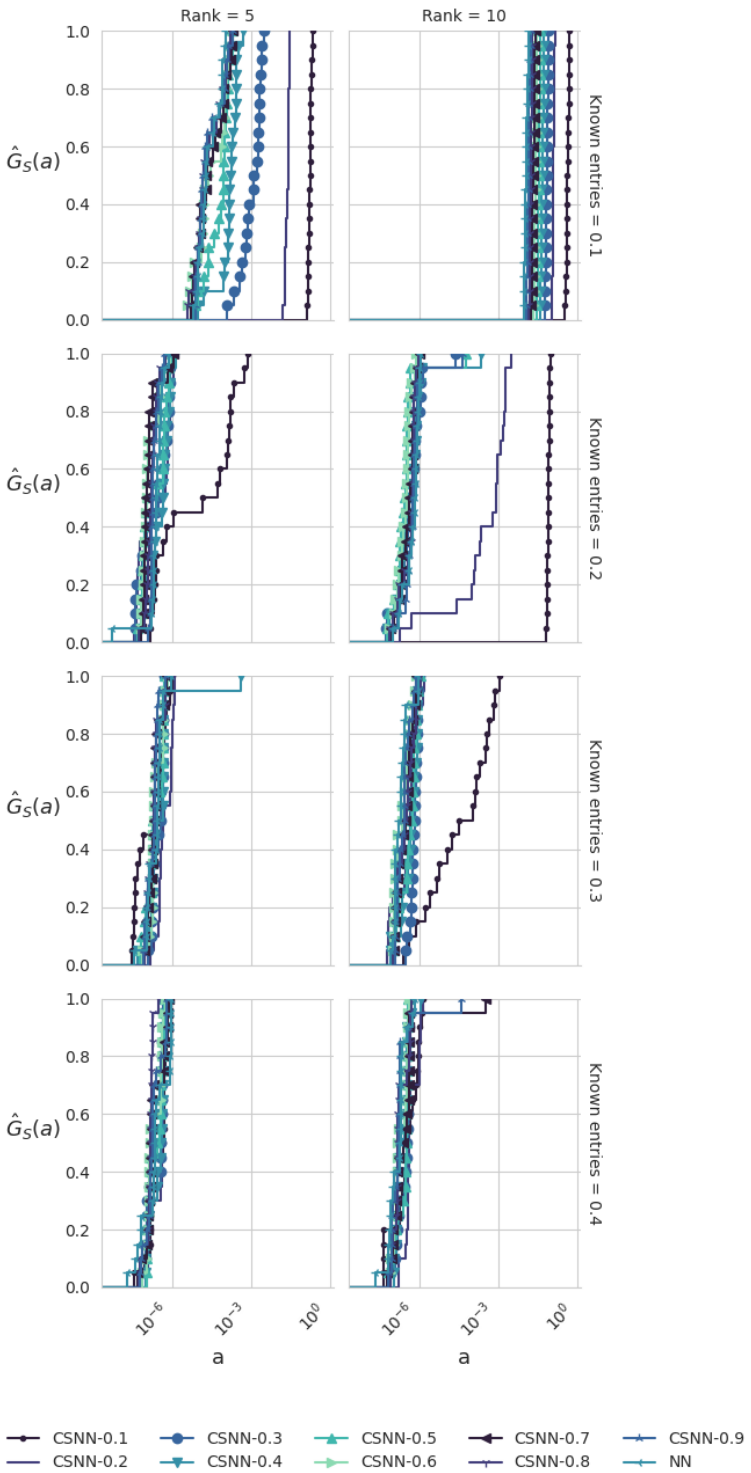


Figure 8.4: Results S I: ECDF curves (8.6) depending on the matrix rank and rate of the known entries for NN and CSNN-α, $\alpha \in \{0.1, \cdots, 0.9\}$, $\mathbf{M} \in \mathbb{R}^{300\times1000}$.

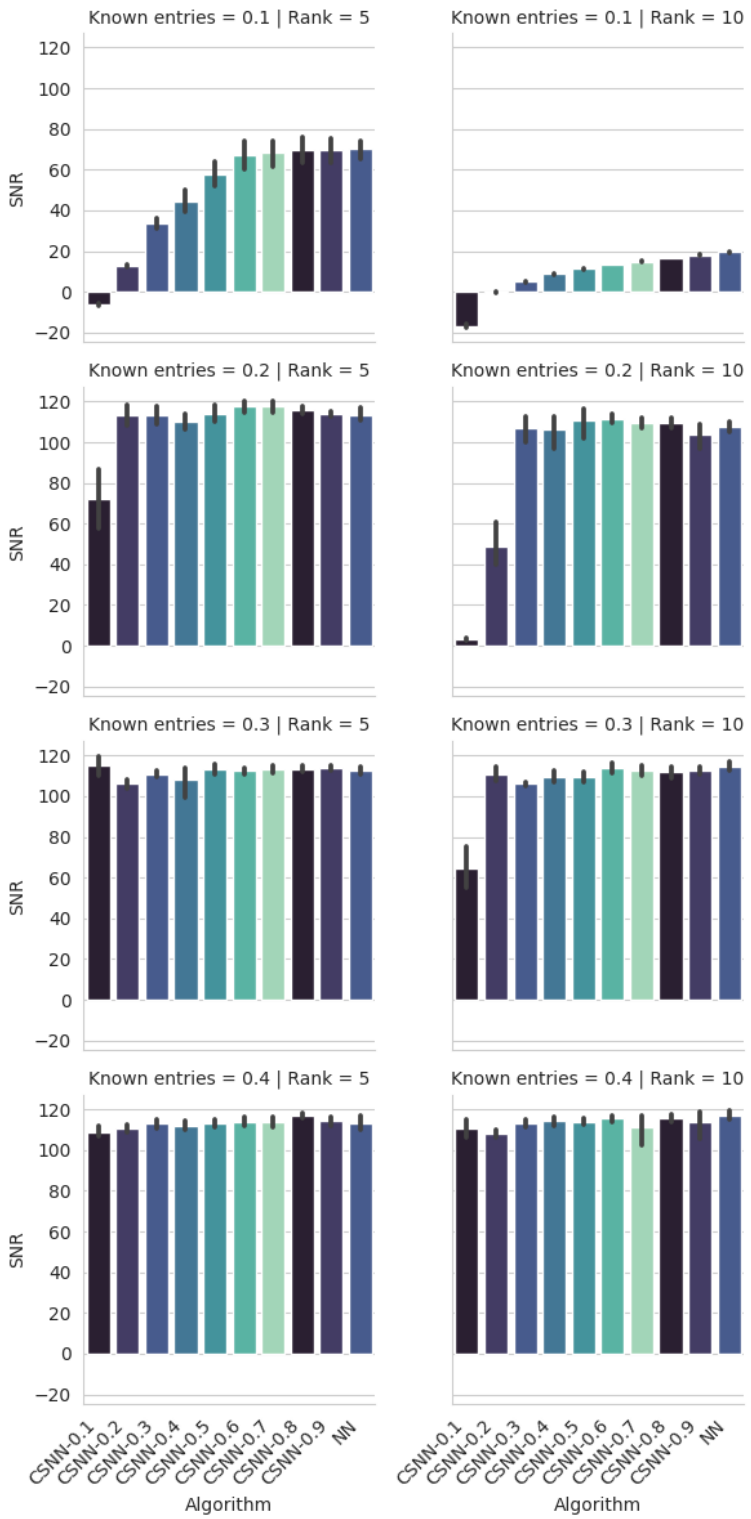


Figure 8.5: Results S I: SNR (8.7) depending on the matrix rank and rate of the known entries for NN and CSNN-α, $\alpha \in \{0.1, \cdots, 0.9\}$, $\mathbf{M} \in \mathbb{R}^{300\times 1000}$.

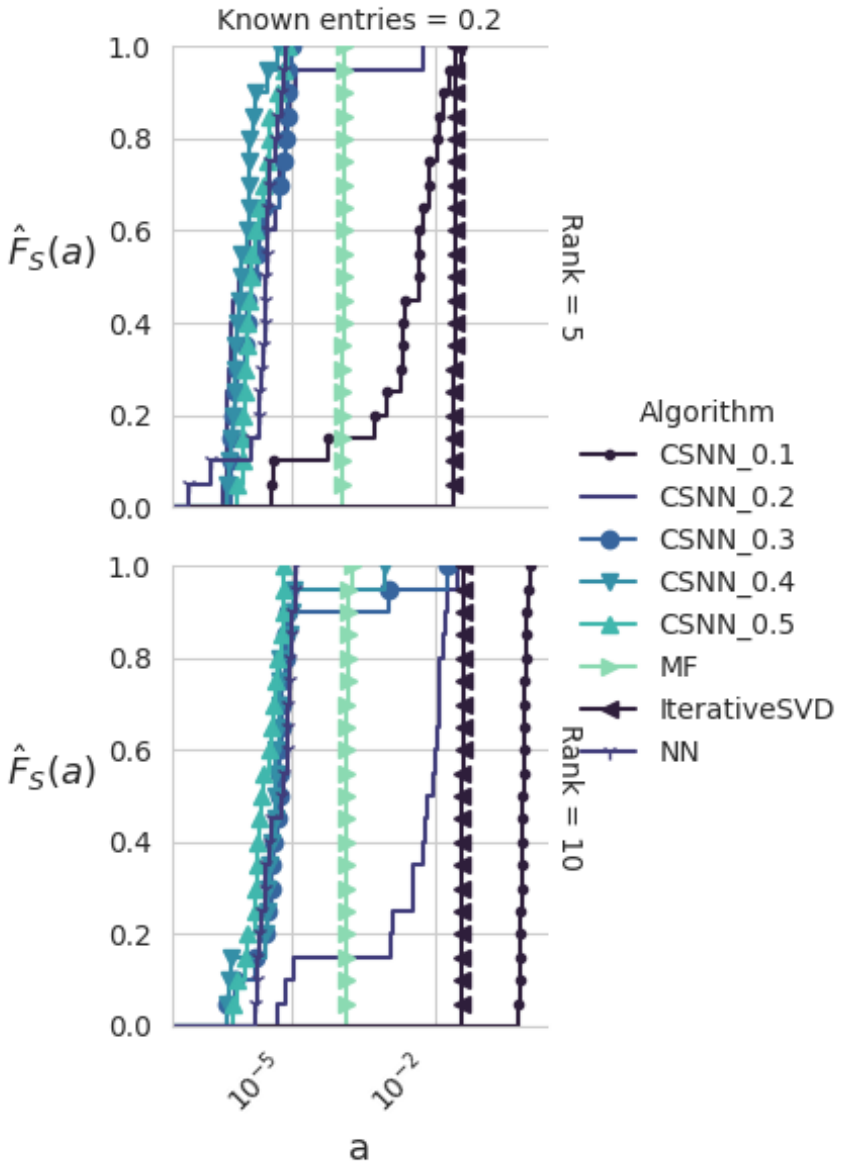


Figure 8.6: Results S II: ECDF curves (8.4) depending on the matrix rank for MF, Iterative SVD, NN and CSNN-α, $\alpha \in \{0.1, \cdots, 0.5\}$, $\mathbf{M} \in \mathbb{R}^{300\times1000}$, ratio of known entries $\rho = 0.2$.

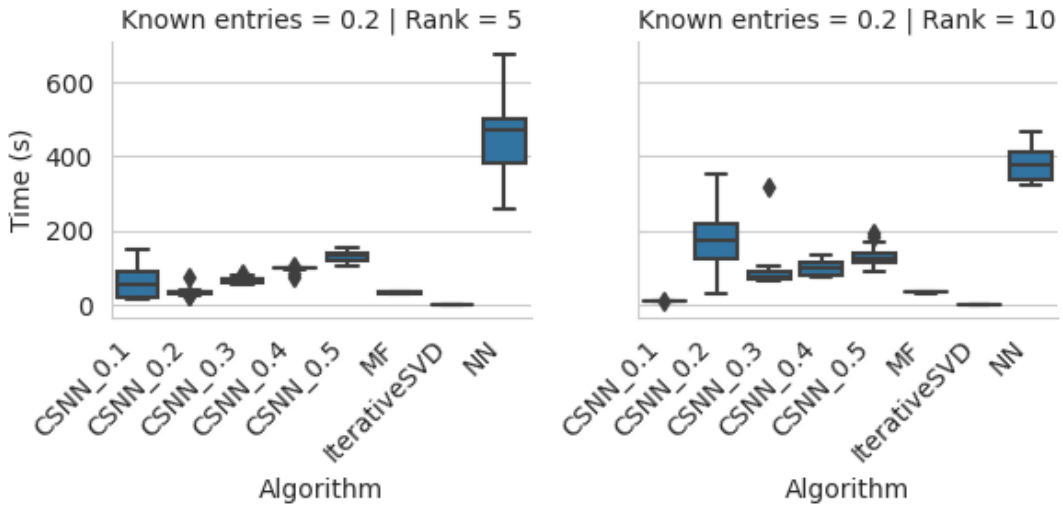


Figure 8.7: Runtime distribution depending on the matrix rank for MF, Iterative SVD, NN and CSNN-α, $\alpha \in \{0.1, \cdots, 0.5\}$, $\mathbf{M} \in \mathbb{R}^{300\times1000}$, ratio of known entries $\rho = 0.2$.

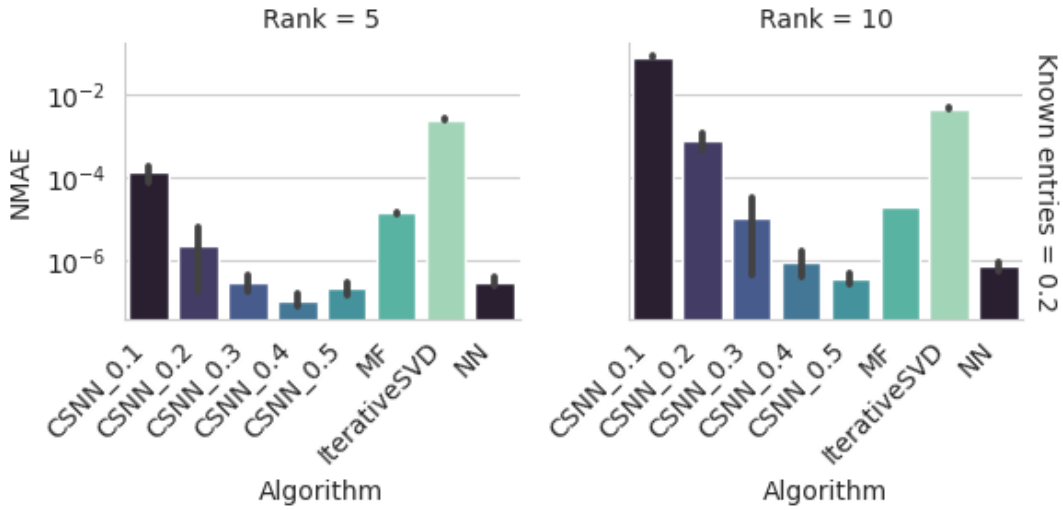


Figure 8.8: Results S II: NMAE depending on the matrix rank for MF, Iterative SVD, NN and CSNN-α, $\alpha \in \{0.1, \cdots, 0.5\}$, $\mathbf{M} \in \mathbb{R}^{300\times1000}$, ratio of known entries $\rho = 0.2$.

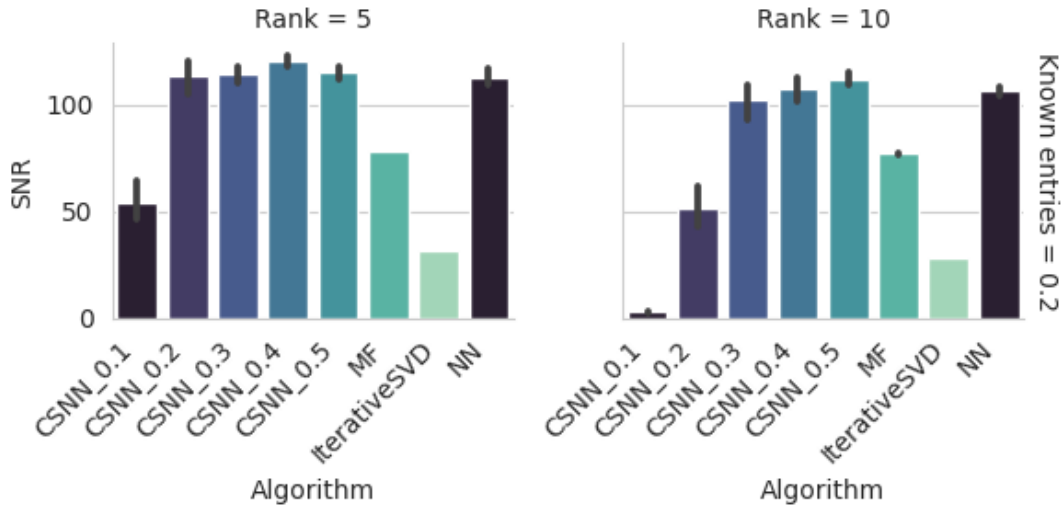


Figure 8.9: Results S II: SNR depending on the matrix rank for MF, Iterative SVD, NN and CSNN-α, $\alpha \in \{0.1, \cdots, 0.5\}$, $\mathbf{M} \in \mathbb{R}^{300\times1000}$, ratio of known entries $\rho = 0.2$.

		Relative error (ϵ)		Time [s]	
	Rank				
Known entries ratio (ρ)	Algorithm	5	10	5	10
0.2	MF	1.162e-04	1.401e-04	3.375e+01	3.380e+01
	CSNN_0.1	6.719e-03	6.954e-01	6.259e+01	1.321e+01
	CSNN_0.2	2.644e-04	8.406e-03	3.539e+01	1.738e+02
	CSNN_0.3	**3.290e-06**	9.235e-04	**6.801e+01**	9.180e+01
	CSNN_0.4	1.198e-06	4.539e-05	9.832e+01	9.942e+01
	CSNN_0.5	2.308e-06	**3.103e-06**	1.299e+02	**1.302e+02**
	IterativeSVD	2.518e-02	3.675e-02	5.527e-01	7.890e-01
	NN	**3.090e-06**	**5.858e-06**	**4.505e+02**	**3.790e+02**

Table 8.3: Results S II; MF, Iterative SVD, NN and CSNN-α, $\alpha \in (0.1, 0.5)$, for $\mathbf{M} \in \mathbb{R}^{300\times1000}$ and ratio of known entries $\rho = 0.2$.

Rank	Known part (ρ) / Algorithm	Relative error (ϵ) 0.5	0.3	0.1	Time [s] 0.5	0.3	0.1
5	CSPGD-0.3	**6.336e-04**	**1.260e-03**	**5.076e-03**	**5.868e+01**	**5.548e+01**	**5.103e+01**
	CSPGD-0.4	6.016e-04	1.209e-03	4.679e-03	7.554e+01	7.148e+01	6.608e+01
	CSPGD-0.5	5.906e-04	1.191e-03	4.491e-03	9.583e+01	9.136e+01	8.423e+01
	PGD	**5.906e-04**	**1.191e-03**	**4.491e-03**	1.396e+02	**1.408e+02**	**1.316e+02**
10	CSPGD-0.3	**6.882e-04**	**1.367e-03**	**6.241e-03**	**6.815e+01**	**6.669e+01**	**6.673e+01**
	CSPGD-0.4	6.252e-04	1.268e-03	5.417e-03	9.079e+01	8.858e+01	8.732e+01
	CSPGD-0.5	6.052e-04	1.235e-03	5.040e-03	1.184e+02	1.158e+02	1.121e+02
	PGD	**6.052e-04**	**1.235e-03**	**5.040e-03**	**2.036e+02**	**2.079e+02**	**1.995e+02**
15	CSPGD-0.3	7.408e-04	1.474e-03	7.737e-03	7.676e+01	7.714e+01	8.447e+01
	CSPGD-0.4	6.473e-04	1.325e-03	6.299e-03	1.034e+02	1.034e+02	1.091e+02
	CSPGD-0.5	6.174e-04	1.275e-03	5.687e-03	1.363e+02	1.362e+02	1.385e+02
	PGD	**6.174e-04**	**1.275e-03**	**5.687e-03**	**2.526e+02**	**2.615e+02**	**2.587e+02**
25	CSPGD-0.3	**8.454e-04**	**1.712e-03**	**1.251e-02**	**9.280e+01**	**9.693e+01**	**1.523e+02**
	CSPGD-0.4	6.900e-04	1.453e-03	8.794e-03	1.262e+02	1.309e+02	1.669e+02
	CSPGD-0.5	6.398e-04	1.365e-03	7.397e-03	1.679e+02	1.724e+02	2.009e+02
	PGD	**6.398e-04**	**1.365e-03**	**7.397e-03**	**3.350e+02**	**3.535e+02**	**3.775e+02**
50	CSPGD-0.3	**1.132e-03**	**2.483e-03**	1.212e+00	**1.345e+02**	**1.522e+02**	2.573e+02
	CSPGD-0.4	8.153e-04	1.849e-03	6.699e-01	1.820e+02	2.014e+02	3.353e+02
	CSPGD-0.5	7.073e-04	1.635e-03	4.372e-01	2.413e+02	2.620e+02	4.117e+02
	PGD	**7.073e-04**	**1.635e-03**	4.372e-01	**5.156e+02**	**5.598e+02**	8.183e+02

Table 8.4: Results S III; PGD and CSPGD-α, $\alpha \in \{0.3, 0.4, 0.5\}$, $\mathbf{M} \in \mathbb{R}^{2000\times3000}$.

that λ value for all algorithms was selected based on the relative error. Thus larger λ values could result in smaller runtimes and comparable relative errors. The difference between PGD and CSPGD could be less sharp in that case. Fig. 8.12 and Fig. 8.13 present NMAE and SNR and confirm that results obtained with CSPGD and PGD algorithms maintain equivalent quality.

Experiment S IV Finally, we would like to discuss results for PGD, CSPGD-α and CSPGD-α-adam, $\alpha \in \{0.3, 0.4, 0.5\}$ for matrices with ten thousand columns for rank $r \in \{5, 10, 15\}$ and known entries rates $\rho \in \{0.2, 0.1\}$. As shown in Fig. 8.14, all algorithms achieved the same magnitude of the relative error ($\epsilon \sim 10^{-3}$). Sampling $\alpha = 0.3$ columns of the matrix allowed to save up to 160 seconds in case of rank 15 matrices with 0.1 known entries (see Table 8.5). The CSPGD-adam algorithm was faster than the CSPGD algorithm. However, the runtime of both types of algorithms had the same magnitude (Fig. 8.15). All experiment trials took reasonably less time than trials of the S III experiment (although the number of elements in the matrices was the same). All algorithms maintained a good quality of performance in terms of NMAE (Fig. 8.16) and SNR values (Fig. 8.17).

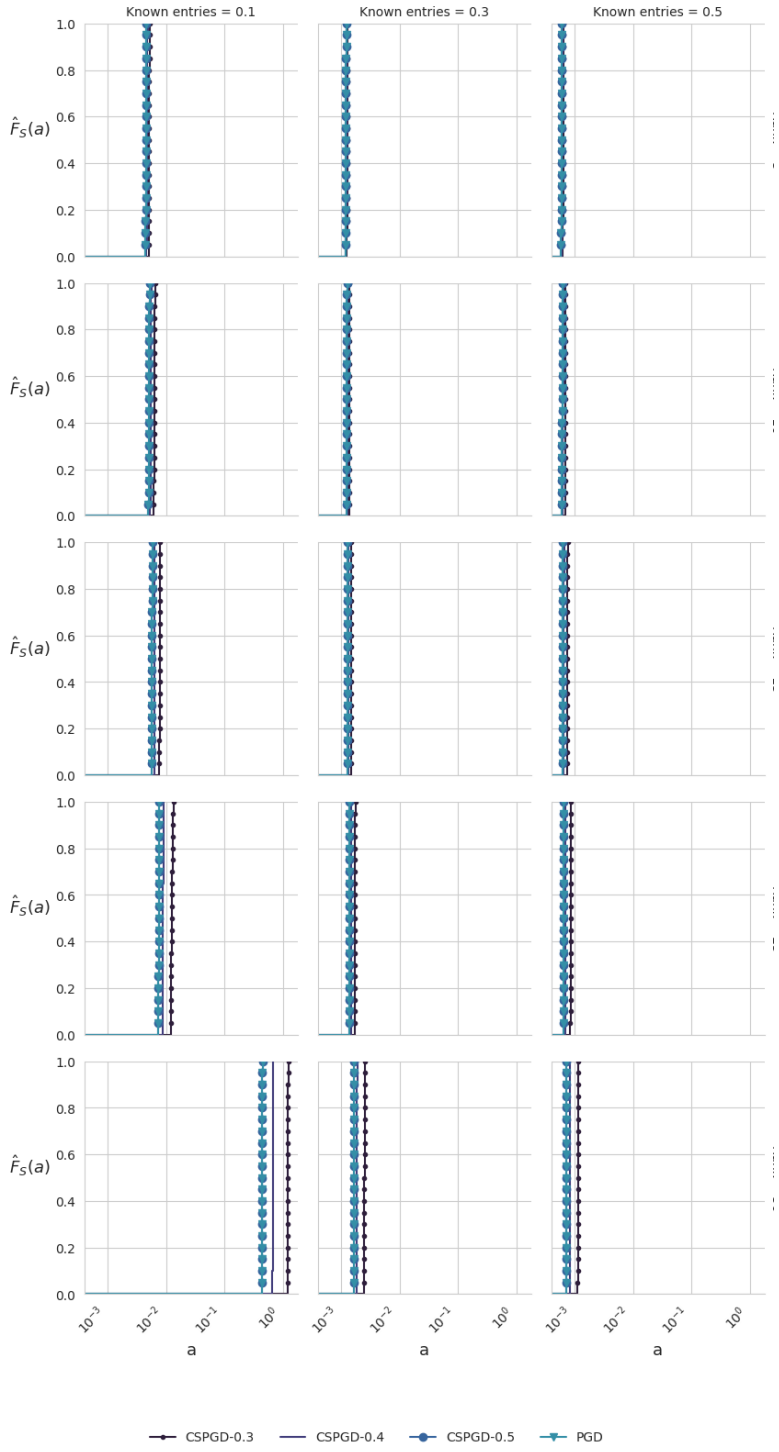


Figure 8.10: Results S III: ECDF curves (8.4) depending on the matrix rank and rate of the known entries for PGD and CSPGD-α, $\alpha \in \{0.3, 0.4, 0.5\}$, $\mathbf{M} \in \mathbb{R}^{2000\times 3000}$.

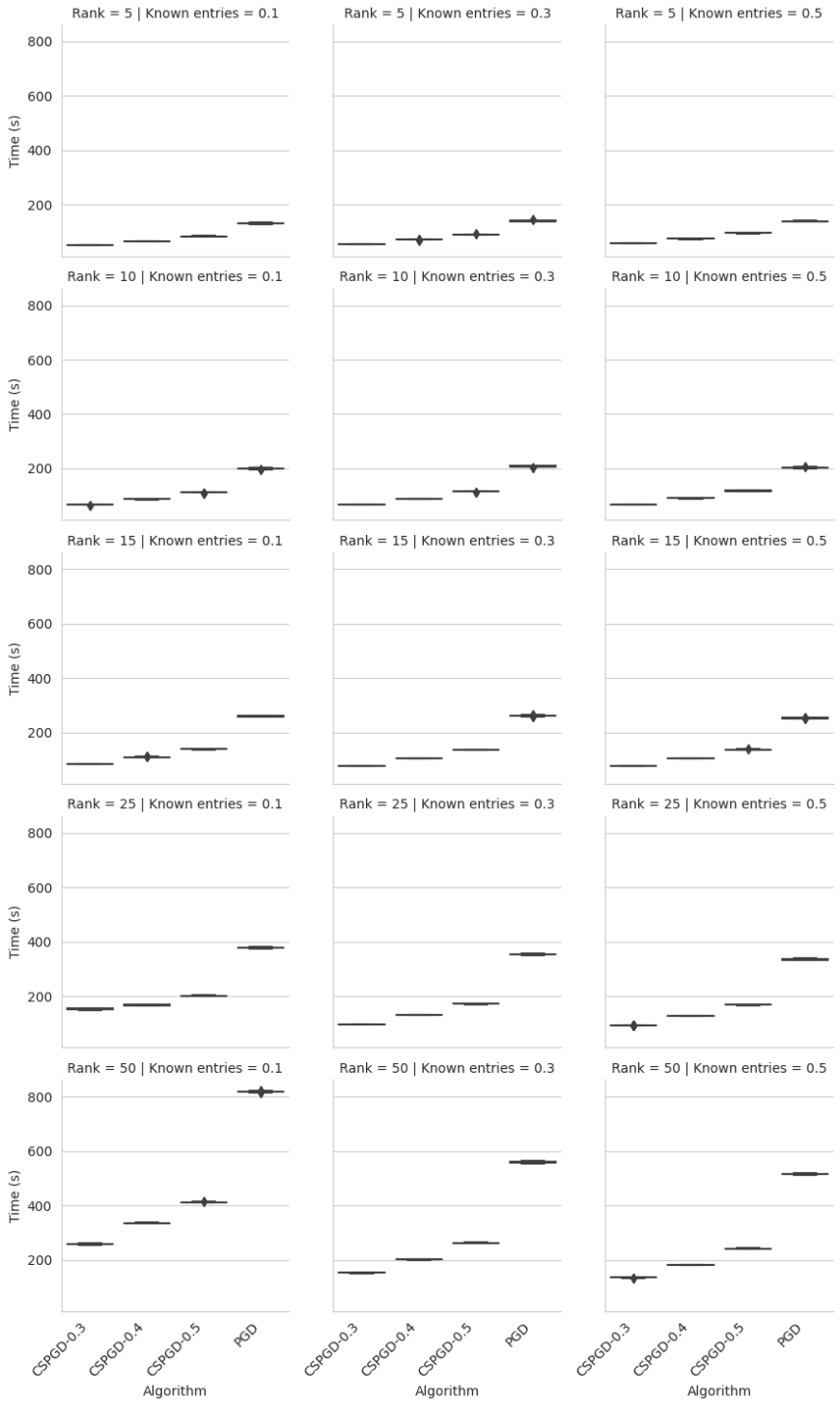


Figure 8.11: Results S III: Runtime distribution depending on the matrix rank and rate of the known entries for PGD and CSPGD-α, $\alpha \in \{0.3, 0.4, 0.5\}$, $\mathbf{M} \in \mathbb{R}^{2000\times3000}$.

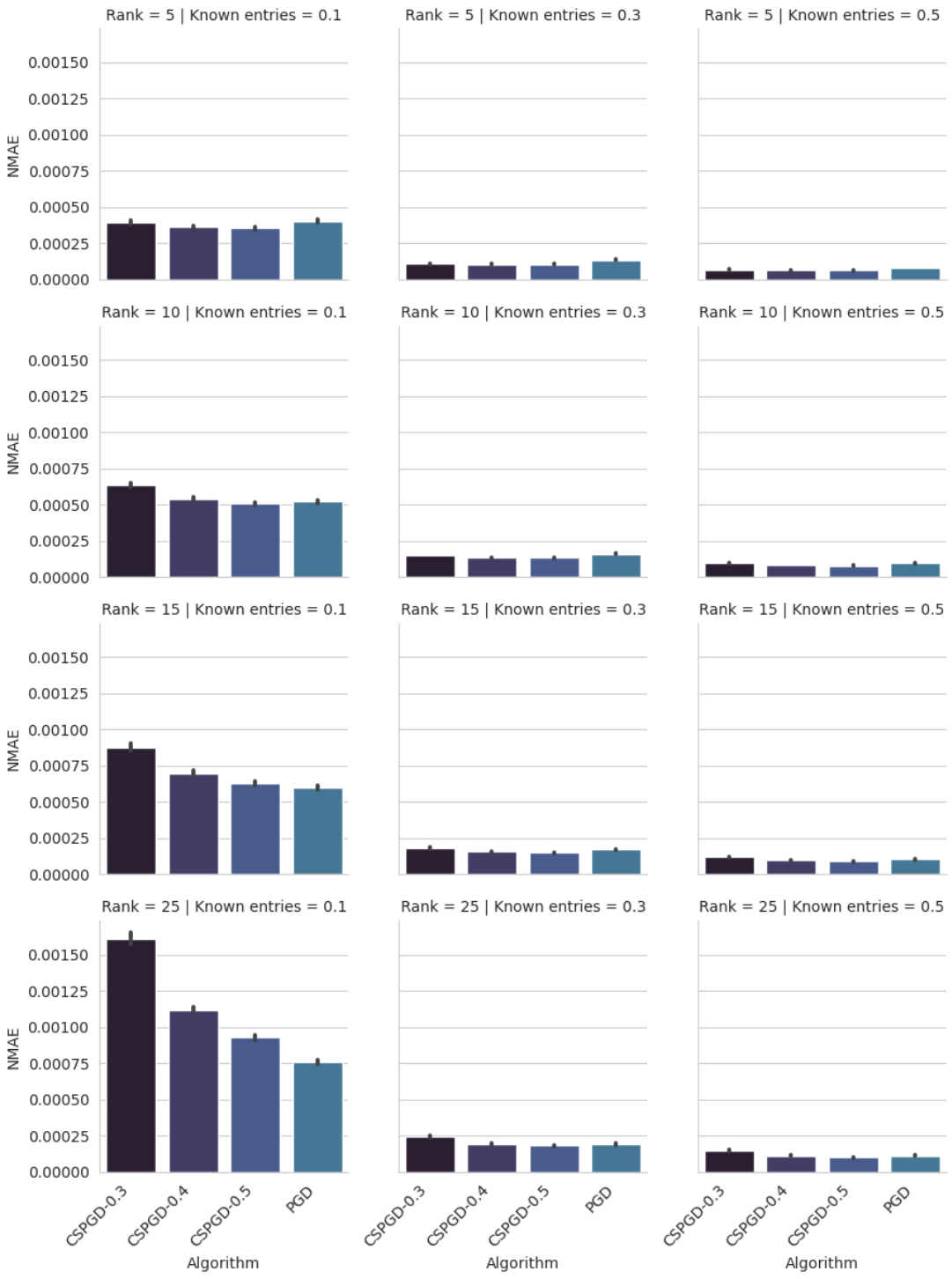


Figure 8.12: Results S III: NMAE depending on the matrix rank and rate of the known entries for PGD and CSPGD-α, $\alpha \in \{0.3, 0.4, 0.5\}$, $\mathbf{M} \in \mathbb{R}^{2000\times3000}$.

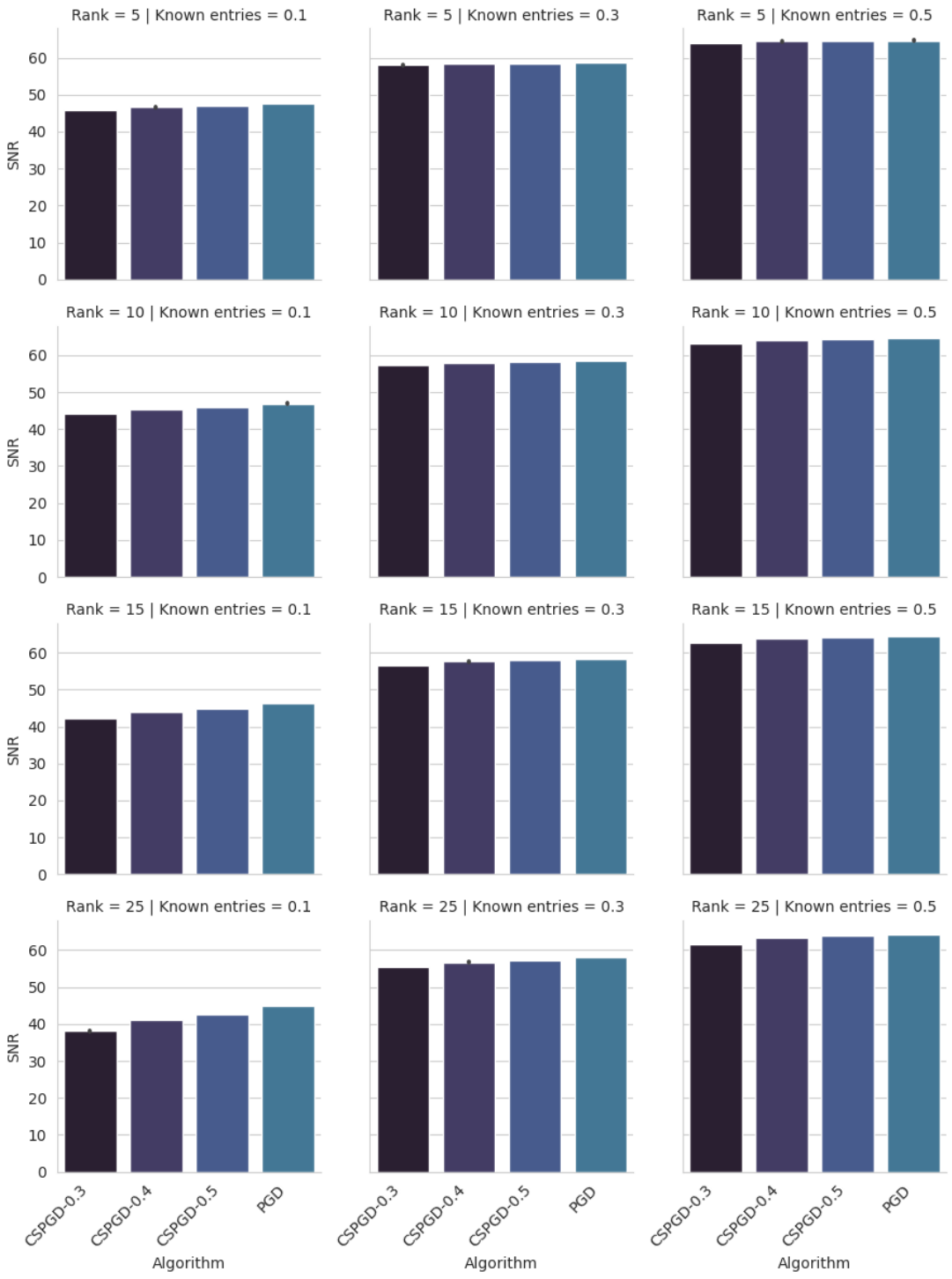


Figure 8.13: Results S III: SNR depending on the matrix rank and rate of the known entries for PGD and CSPGD-α, $\alpha \in \{0.3, 0.4, 0.5\}$, $\mathbf{M} \in \mathbb{R}^{2000\times3000}$.

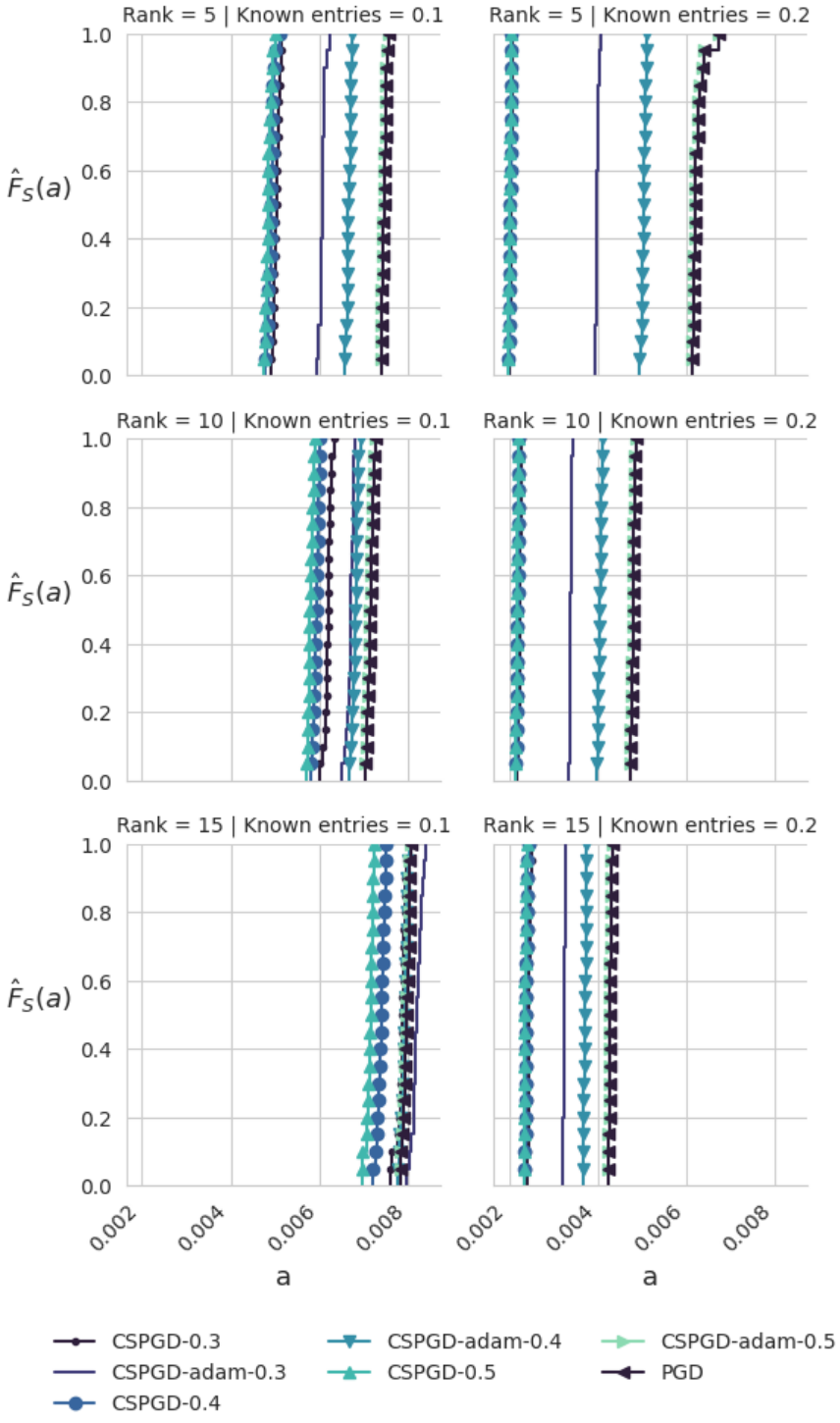


Figure 8.14: Results S IV: ECDF curves (8.4) depending on the matrix rank and rate of the known entries for PGD, CSPGD-α and CSPGD-adam-α, $\alpha \in \{0.3, 0.4, 0.5\}$, $\mathbf{M} \in \mathbb{R}^{600\times 10000}$.

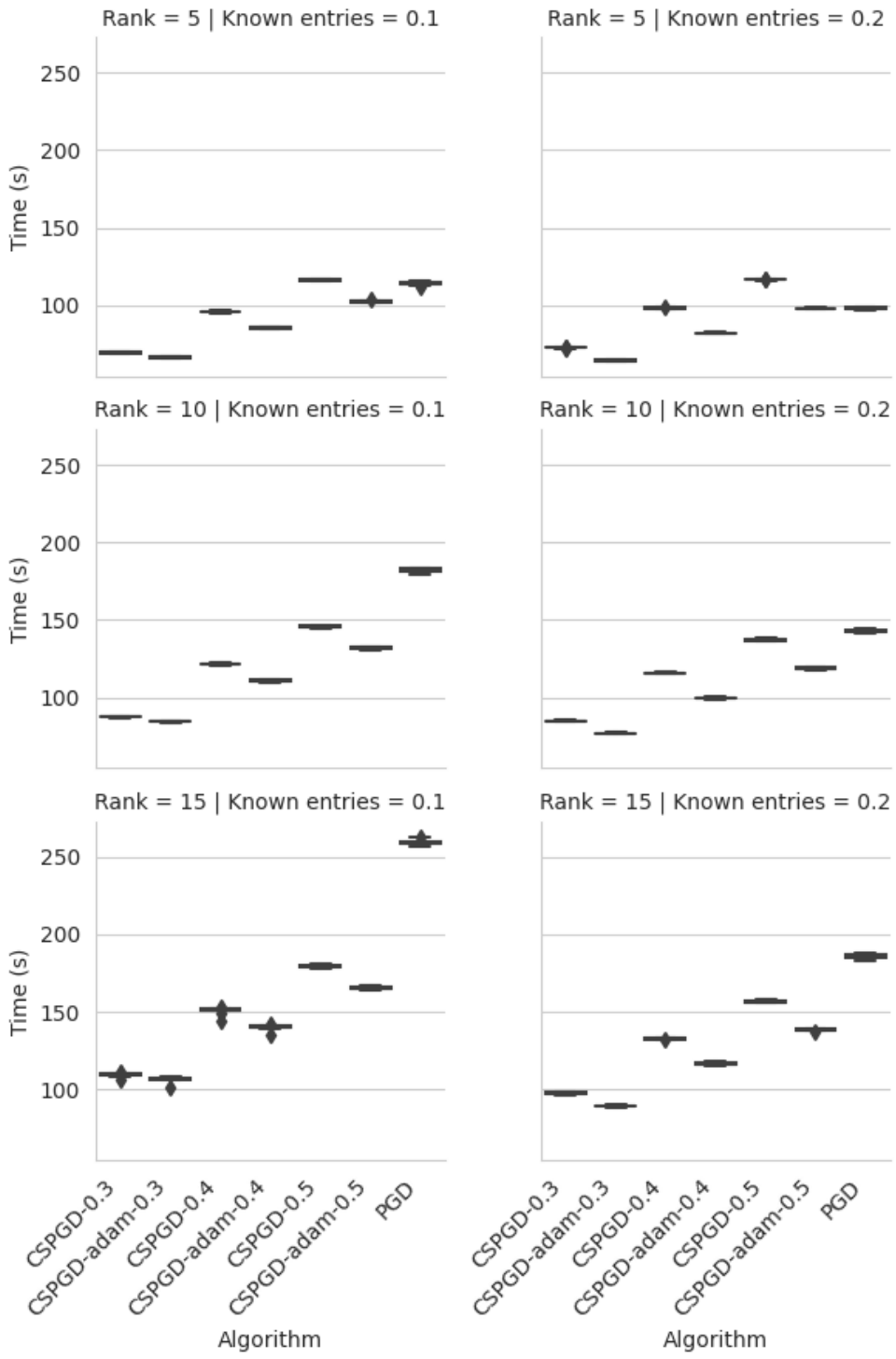


Figure 8.15: Results S IV: Runtime depending on the matrix rank and rate of the known entries for PGD, CSPGD-α and CSPGD-adam-α, $\alpha \in \{0.3, 0.4, 0.5\}$, $\mathbf{M} \in \mathbb{R}^{600\times 10000}$.

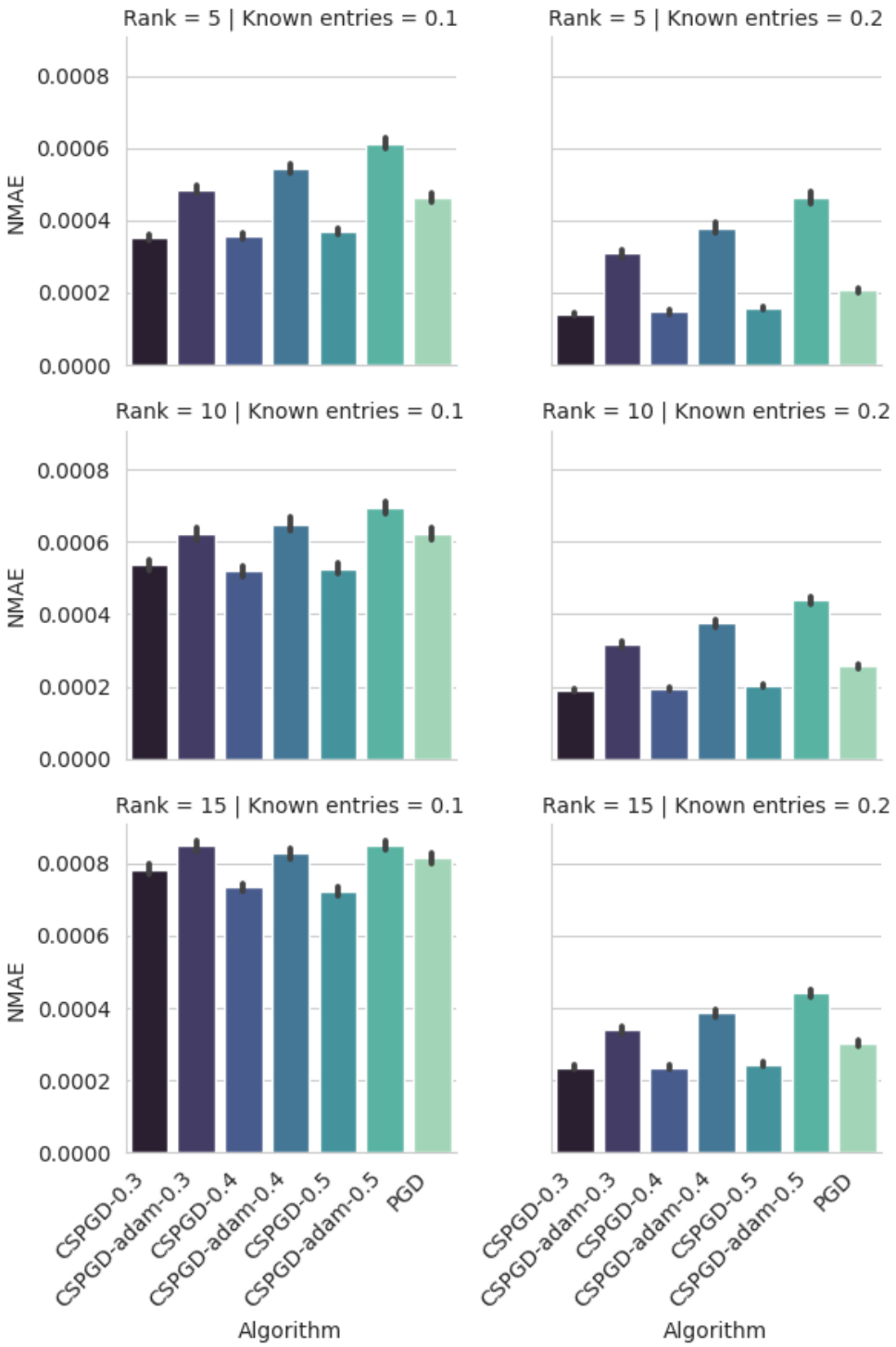


Figure 8.16: Results S IV: NMAE depending on the matrix rank and rate of the known entries for PGD, CSPGD-α and CSPGD-adam-α, $\alpha \in \{0.3, 0.4, 0.5\}$, $\mathbf{M} \in \mathbb{R}^{600\times 10000}$.

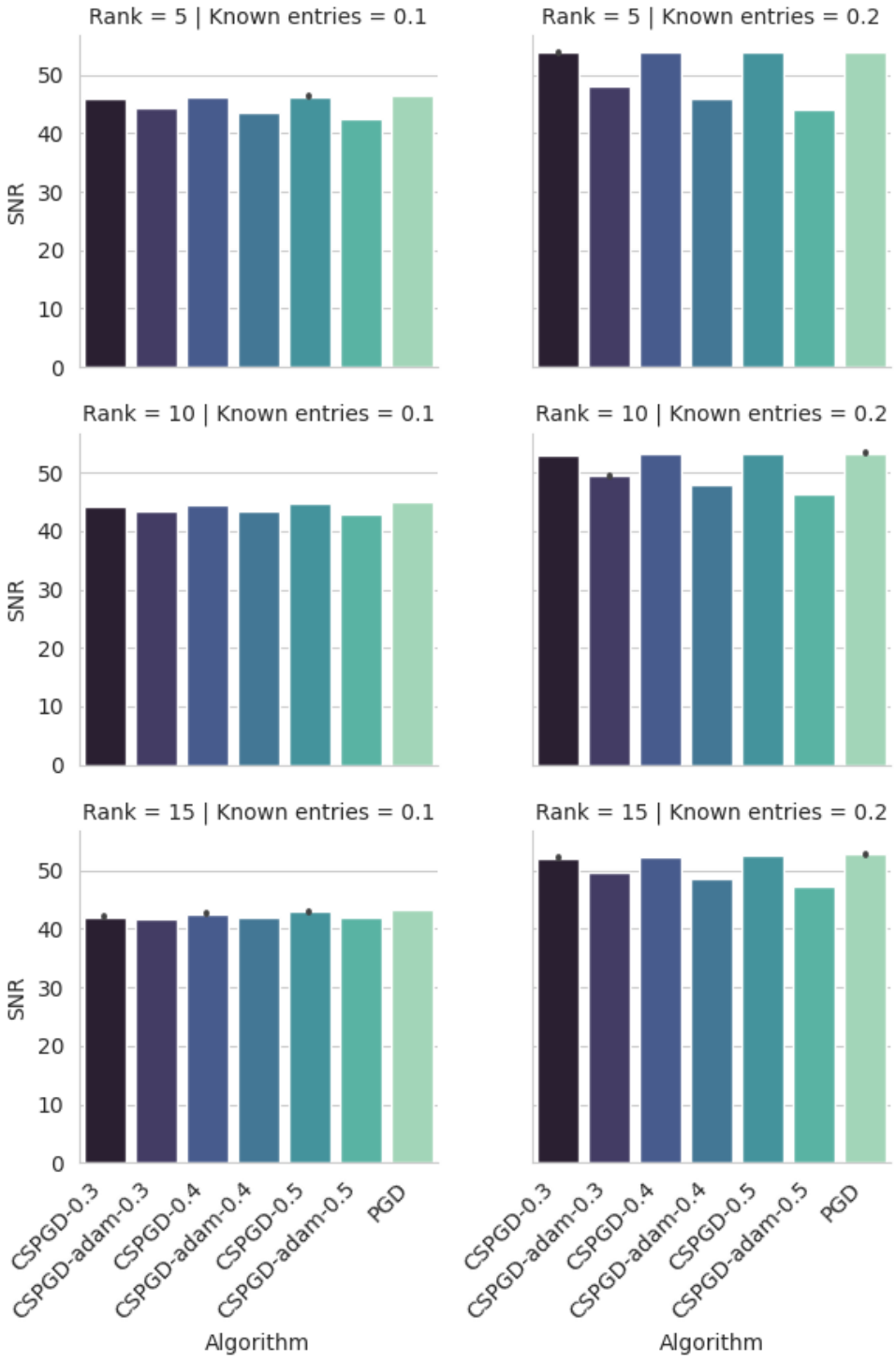


Figure 8.17: Results S IV: SNR depending on the matrix rank and rate of the known entries for PGD, CSPGD-α and CSPGD-adam-α, $\alpha \in \{0.3, 0.4, 0.5\}$, $\mathbf{M} \in \mathbb{R}^{600\times 10000}$.

Known entries ratio (ρ)	Rank / Algorithm	Relative (ϵ) 5	10	15	Time [s] 5	10	15
0.2	CSPGD-0.3	**2.051e-03**	**2.246e-03**	**2.459e-03**	**7.256e+01**	**8.485e+01**	**9.759e+01**
	CSPGD-0.4	2.031e-03	2.202e-03	2.395e-03	9.842e+01	1.158e+02	1.329e+02
	CSPGD-0.5	2.018e-03	2.182e-03	2.363e-03	1.165e+02	1.372e+02	1.568e+02
	CSPGD-adam-0.3	**3.988e-03**	**3.387e-03**	**3.248e-03**	**6.457e+01**	**7.698e+01**	**8.966e+01**
	CSPGD-adam-0.4	5.033e-03	4.047e-03	3.711e-03	8.210e+01	9.965e+01	1.167e+02
	CSPGD-adam-0.5	6.240e-03	4.794e-03	4.276e-03	9.799e+01	1.188e+02	1.386e+02
	PGD	**6.240e-03**	**4.794e-03**	**4.276e-03**	**9.811e+01**	**1.429e+02**	**1.858e+02**
0.1	CSPGD-0.3	**5.036e-03**	**6.190e-03**	**7.859e-03**	**6.964e+01**	**8.752e+01**	**1.098e+02**
	CSPGD-0.4	4.915e-03	5.926e-03	7.387e-03	9.609e+01	1.214e+02	1.512e+02
	CSPGD-0.5	4.851e-03	5.800e-03	7.135e-03	1.164e+02	1.457e+02	1.793e+02
	CSPGD-adam-0.3	**6.066e-03**	**6.694e-03**	**8.197e-03**	**6.664e+01**	**8.449e+01**	**1.067e+02**
	CSPGD-adam-0.4	6.661e-03	6.806e-03	7.909e-03	8.555e+01	1.110e+02	1.405e+02
	CSPGD-adam-0.5	7.460e-03	7.147e-03	7.964e-03	1.025e+02	1.319e+02	1.655e+02
	PGD	**7.460e-03**	**7.147e-03**	7.964e-03	**1.139e+02**	**1.822e+02**	**2.592e+02**

Table 8.5: Results S IV; PGD and CSPGD-α, $\alpha \in \{0.3, 0.4, 0.5\}$), $\mathbf{M} \in \mathbb{R}^{600\times 10000}$.

8.4 Conclusion and discussion

In this chapter, we explored the ability of the CSMC algorithm to recover matrices in various test scenarios. Test matrices fulfilled the coherence and rank constraints from the theorem 5.3.1. Since CSMC algorithms may bring computational gain in recovery rectangular matrices with more columns, all test matrices $\mathbf{M}$ had dimensions $n_1 \times n_2$, $n_2 > n_1$.

To accurately assess the significance of the column subset selection in matrix completion tasks, we compared NN and CSNN-α algorithms. Exact Nuclear Norm minimization is known to produce solutions of the best quality and has theoretical guarantees provided by theorem 5.3.1. However, SDP formulations can be computationally expensive, especially for large matrices. The computational complexity of solving SDP problems overgrows with the size of the matrix, making it challenging to apply SDP-based methods to large-scale matrix completion problems. It is known to be inefficient in the case of large matrices due to the extensive memory cost of the SDP solvers. The results of the S I experiment proved that CSNN-0.3 could maintain the quality of NN solutions while offering considerable runtime savings. The quality of the solutions is appraised by the relative error, NMAE and SNR values. We benchmarked CSNN and NN algorithms with the Matrix Factorization and widely used Iterative SVD in experiment S II. The Iterative SVD was the fastest of algorithms, but it failed in most experiment trials. MF output with the solution of good quality in a short time. However, the relative error was more significant than in the case of CSNN algorithms. Still, CSNN has the same limitations regarding vast amounts of data. The CSNN may be used, in instances where the main focus of the task is to obtain a guaranteed solution, for rectangular matrices.

For large-scale problems, we applied algorithms that utilize proximal gradient

descent as the optimization algorithm. The algorithms are based on the principle of nuclear norm regularization, which promotes low-rank solutions. For larger matrices of dimensions 2000×3000, we compared PGD and CSPGD algorithms in experiment S III. Both algorithms' performance depends on the choice of λ parameter. In our scenario, we chose λ based on the value of relative error. All of th CSPGD-α algorithms for $\alpha \in \{0.3, 0.4, 0.5\}$ succeeded in matrix completion task. Again CSPGD-0.3 brought substantial gain in terms of the runtime. However, we would like to emphasize that the relative error determined the λ parameter's choice. Larger α values could result in solutions of similar quality and lower algorithm runtime in the case of PGD.

Lastly, we addressed the completion of the matrices with thousand of columns in experiment S IV. The closed formula solves the least squares problem in the second stage of the CSMC algorithm. However, it can also be obtained with scalable convex optimization methods like stochastic gradient descent or Adam [143]. As in the S III scenario, CSPGD-0.3 and CSPGD-0.3-adam resulted in considerable runtime savings while maintaining the quality of the solutions of the PGD. CSPGD-adam was slightly faster than the CSPGD algorithm, but the runtime difference was insignificant in this scenario.

Chapter 9

Real data applications

Matrix completion has arisen in a vast range of applications, including signal processing, image processing, recommendation systems, sensor localization, multi-task learning, genomics, multi-task learning and system identification [13, 160, 165–170]. The ability to estimate missing or corrupted entries in matrices has broad implications in various domains where incomplete data is encountered, enabling better decision-making, predictions, and understanding of complex systems. While testing on synthetic data sets provides valuable insights, evaluating numerical algorithms on real-world data sets is crucial to ensure their effectiveness and generalizability. In this chapter, we explore the performance of the CSMC method on three real data examples. Firstly, we evaluate the presented method in collaborative filtering, one of the techniques most commonly used in recommendation systems [171, 172]. Then we analyze results for the application CSMC method in the image processing [160, 167, 168] and the link prediction in a graph [69, 173].

9.1 Recommendation system

Matrix completion is widely used in recommendation systems to predict missing entries in user-item rating matrices. By leveraging the observed ratings, matrix completion algorithms can estimate the unobserved ratings and provide personalized recommendations to users. This is particularly useful in e-commerce, movie recommendations, and social media platforms. The Netflix Prize was the international competition for the best algorithm to predict user ratings for films from previous ratings without any other information about the users or films (Fig. 9.1). By setting up each movie's ratings as a row and each user's ratings as a column, the dataset in a recommendation system may be represented as a matrix with unobserved ratings as missing entries [13]. This section evaluates the CSMC method on the publicly available data set from the Movielens research project [174].

Data set and data preprocessing Since we wanted to benchmark CSNN and CSPGD algorithms, we considered two data sets: Movie Lens Small and

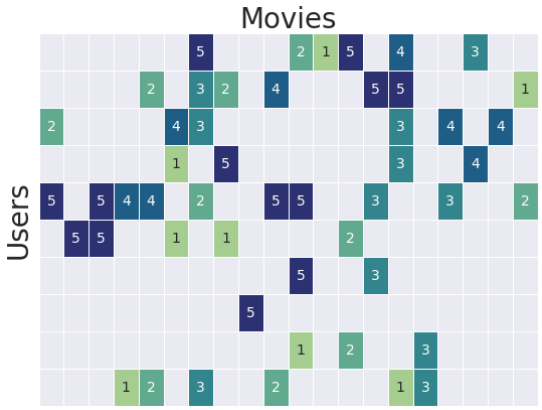
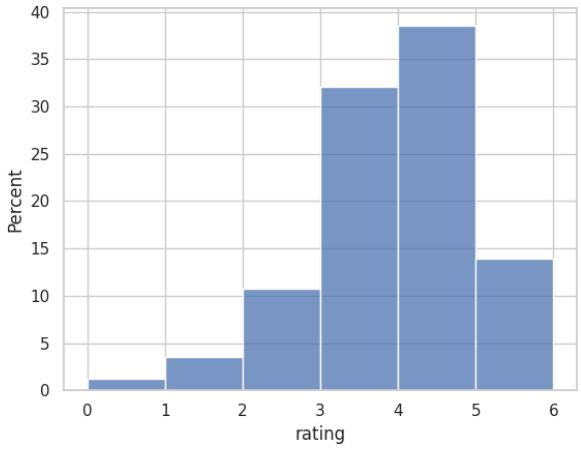


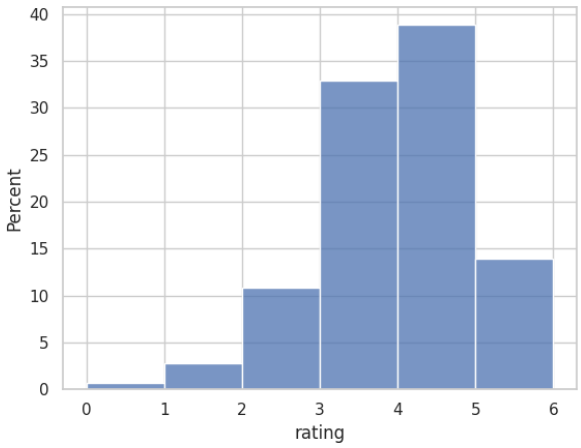
Figure 9.1: Movie recommendation problem as matrix completion

Movie Lens Big. The Movie Lens Small was represented by $\mathbf{M} \in \mathbb{R}^{140\times668}$ matrix obtained from the Movie Lens Small dataset, which contained 100 000 5-star ratings applied to 9742 movies by 610 users. Since the original matrix was too large for SDP solvers in NN and CSNN algorithms, we followed the procedure described in [13, 175] to obtain a submatrix of the desired size. Specifically, we sorted the data by user frequency rate and took data containing rates made by the top 60% users. Then we sorted obtained data by movie frequency rate and selected the top 50% movies. Obtained matrix had $\rho = 0.25$ known entries. In the second test series, we evaluated CSPGD algorithms on the Movielens 25M data set, containing 25 million ratings applied to 62 000 movies by 162 000 users. Again, due to the large size and low observation rate, we extracted a 654×27813 matrix $\mathbf{M}$. The known entries rate ρ was equal to 0.09. As shown in Fig. 9.2, the distribution of the ratings in each of the data sets was highly similar.

Experimental procedures We designed two test scenarios, M I and M II. In the M I scenario, we assessed the performance of the NN and CSNN-α for $\alpha \in \{0.3, 0.4, 0.5, 0.7\}$ on the Movie Lens Small data set. In the M II scenario, we evaluated the performance of the PGD and CSPGD-α for $\alpha \in \{0.3, 0.5\}$ on the Movie Lens Big data set. In the whole experiment, we followed previous work [13] and employed the Cross-Validation method. In each trial of the experiment, Ω set is randomly split into training and testings sets denoted by Ω_{train} and Ω_{test} [176]. Specifically, we randomly selected ρ rate of the observed set, and by assigning the null values to the rest of the entries, we constructed $\mathcal{R}_{\Omega_{\text{train}}}(\mathbf{M})$ matrix. We evaluated each algorithm on the Ω_{test} set. We conducted 20 independent experimental trials under each scenario.



(a) Movie Lens Big



(b) Movie Lens Small

Figure 9.2: Distribution of the rates in Movie Lens Small and Movie Lens Big data sets.

Algorithm	NMAE mean	NMAE std	HR mean	HR std	Time [s] mean	Time [s] std
CSNN-0.3	0.271	0.028	0.172	0.005	10.137	1.118
CSNN-0.4	0.240	0.060	0.191	0.005	17.677	2.005
CSNN-0.5	0.248	0.176	0.208	0.006	28.673	2.419
CSNN-0.7	**0.156**	**0.011**	**0.232**	**0.004**	**39.893**	**8.595**
NN	**0.127**	**0.001**	**0.255**	**0.006**	**56.924**	**2.435**

Table 9.1: Results for M I test scenario; NN and CSNN-α algorithms, $\alpha \in \{0.3, 0.4, 0.5, 0.7\}$.

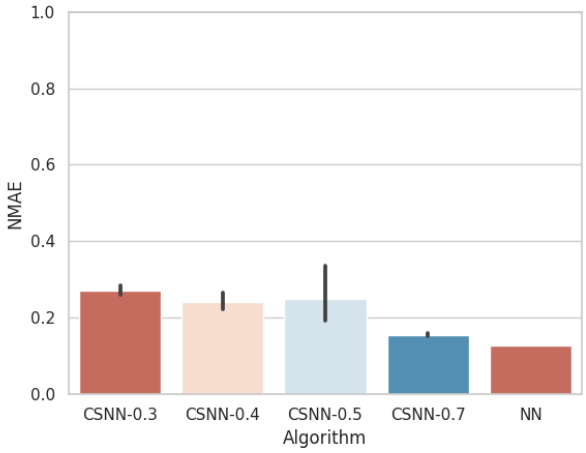
Performance measures Apart from the NMAE (8.5), the quality of the recommendation was measured as hit-rate, defined as

$$\mathrm{HR} = \frac{\#hits}{|\Omega_{\mathrm{test}}|}, \tag{9.1}$$

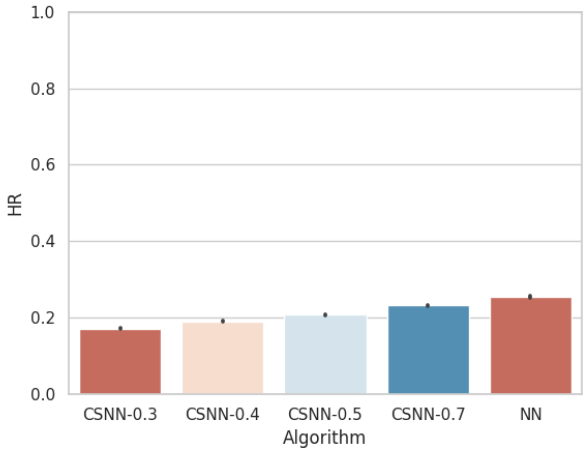
where a predicted rating was considered a *hit* if its rounded value was equal to the actual rating in the test set [13, 176]. As in Chapter 5, we also compared the runtimes of the algorithms.

Results M I As shown in the Fig. 9.3, the predictions of CSNN-0.7 maintained the quality of the NN algorithms in terms of NMAE: (0.16 vs 0.13) and HR (0.23 vs 0.25), and resulted in run time savings from 57 seconds to 40 seconds. CSNN-0.3 required only 10 seconds and output the solution with NMAE: 0.27 and HR: 0.17. Table 9.1 shows results for CSNN algorithms for all test column sampling rates α.

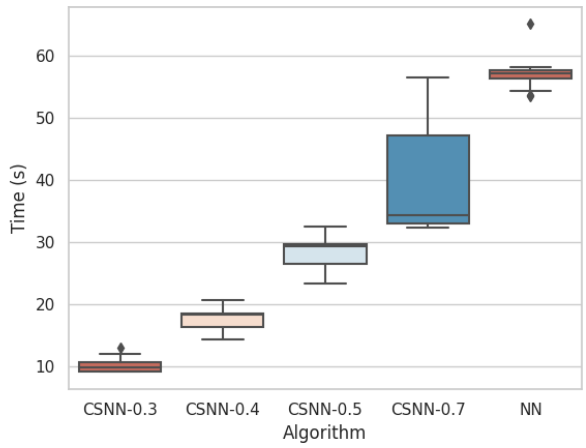
Results M II CSPGD-α for $\alpha = 0.3$ and $\alpha = 0.5$ obtained the best results for λ parameter equal to 10 (NMAE=0.14 and NMAE=0.13), while PGD achieved the lowest NMAE (0.12) for $\lambda = 25$ (see Table 9.2). CSPGD algorithms were much faster and required 23-27 seconds while requiring over 400 seconds (Fig. 9.4). Obtained predictions were slightly better than in M I experiment. The HR values for PGD and CSPGD-0.3 were equal to 0.28 and 0.24, respectively. The good performance for $\alpha = 0.3$ can be caused by the fact that matrices in M II were bigger, while data could be explained using a similar number of latent factors.



(a) NMAE

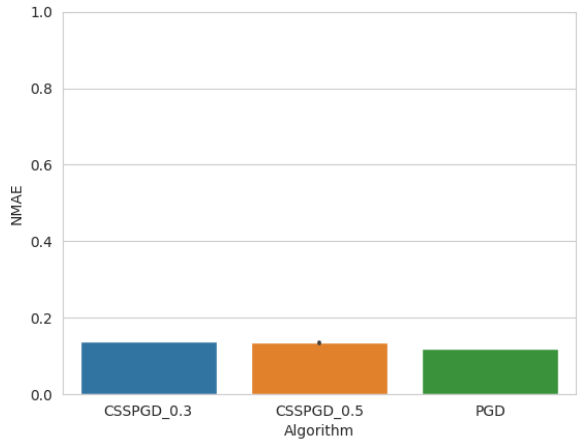


(b) HR

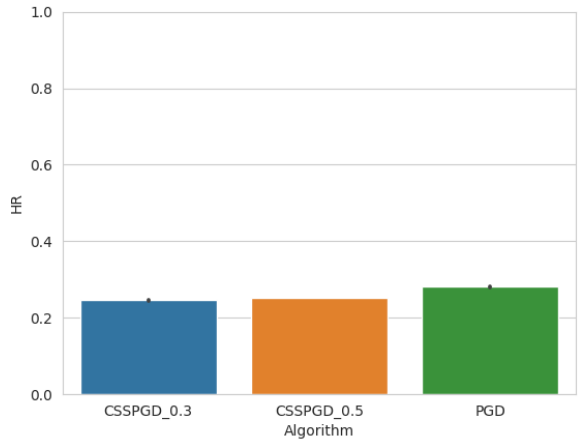


(c) Runtime

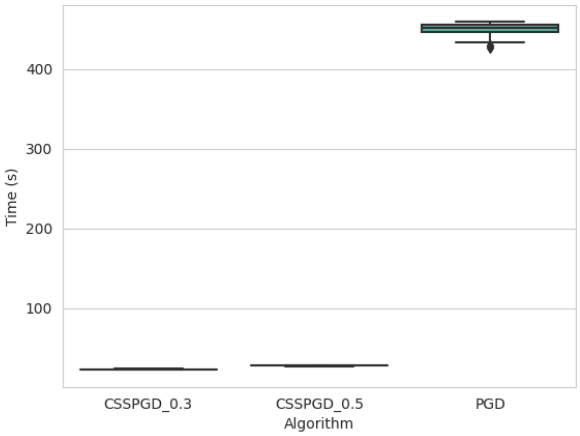
Figure 9.3: Results for M I test scenario; NN and CSNN-α algorithms, $\alpha \in \{0.3, 0.4, 0.5, 0.7\}$.



(a) NMAE



(b) HR



(c) Runtime

Figure 9.4: Results for M II test scenario; PGD and CSPGD-α algorithms for $\alpha \in \{0.3, 0.5\}$.

Algorithm	NMAE mean	NMAE std	HR mean	HR std	Time [s] mean	Time [s] std
CSPGD-0.3	**0.138**	**0.001**	**0.245**	**0.002**	**23.032**	**0.321**
CSPGD-0.5	0.135	0.001	0.252	0.001	27.774	0.299
PGD	**0.119**	**0.001**	**0.281**	**0.001**	**449.026**	**9.502**

Table 9.2: Results for M II test scenario; PGD and CSPGD-α algorithms for $\alpha \in \{0.3, 0.5\}$.

9.2 Image inpainting

Image inpainting is used in image processing and computer vision to fill in missing or corrupted parts. The low-rank models proved tremendously helpful in this task [13, 38, 160, 177]. In this context, the image is treated as a matrix, and missing entries represent the missing or corrupted regions. The basic idea behind low-rank matrix completion is to exploit the assumption that images often possess low-rank structures. The greatest singular values of the image matrix dominate its primary information, whereas the smallest singular values can be set to zero without losing essential details [38]. A low-rank matrix can represent the matrix approximately, indicating underlying patterns and redundancies in the image data. Fig. 9.5 presents the distribution of the singular values for the sample image and its rank-10 approximation with the quality measured by SNR $= 31.50$ (8.7) and the relative error $\epsilon = 0.03$ (8.2). This section compares the performance of the low-rank matrix completion algorithms: CSNN, CSPGD, NN and PGD in the image-inpainting experiments. It is important to note that low-rank matrix completion methods may have limitations when dealing with complex images, textures, or structures that a low-rank matrix cannot adequately represent. More sophisticated techniques, such as deep learning-based approaches, may be more effective for image inpainting tasks [178–183].

Data set and data preprocessing As in the previous example, we constructed two data sets, Bridges Small and Bridges Big. Each one contained ten grey-scaled pictures of bridges downloaded from the public repository [1]. Images from the Bridges Small data set were represented as 240×360 matrices. The mean and standard deviation of the coherence parameter were equal to 1.72 and 0.12, respectively. The Bridges Big data set contained matrices of various sizes with a mean number of rows of 3114.3 and the mean number of columns of 4898.4. For both data sets, we constructed $\mathcal{R}_{\Omega^\rho}(\mathbf{M})$ (Definition 2.0.3) by randomly removing $1 - \rho$ pixels $\rho \in \{0.4, 0.2, 0.1\}$ (Fig. 9.6).

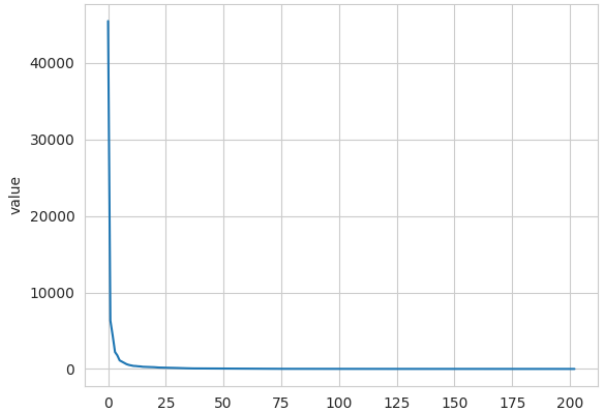
Experimental procedures We constructed two test scenarios: P I and P II. In P I test scenario we evaluated NN and CSNN-α algorithms for $\alpha \in \{0.3, 0.4, 0.5, 0.7\}$ and the known entries rate $\rho \in \{0.4, 0.2, 0.1\}$. We assessed the performance of the algorithms on the Bridges Small data set among 100 independent trials (ten trials per picture). In P II test scenario we compared PGD and CSPGD-α algorithms for $\alpha \in \{0.3, 0.4\}$ and the $\rho \in \{0.4, 0.2, 0.1\}$. We evaluated the performance of the algorithms on the Bridges Big data set among 100 independent trials (10 trials per picture).

Performance measures We assessed the quality of the reconstructed image with the SNR (8.7) and the relative error (8.2).

[1] image source (`https://pxhere.com/pl/`)

(a) Original image

(b) Rank-10 approximation



(c) Singular values distribution

Figure 9.5: Low-rank approximation of the image.

Results P I Fig. 9.7 presents a picture recovered by NN and CSNN-α for $\alpha \in \{0.4, 0.5, 0.7\}$ Table 9.3 presents SNR, approximation error and runtime values for all tested algorithms. The SNR value achieved by the NN algorithm is greater than 20 for the recovery of 0.4 and 0.2 known entries and was equal to 18.59 for inpainting 0.9 of the image. The CSNN-0.7 solutions maintained good quality with the SNR $21.13, 18.52$, and 16.45 for the ρ equal to $0.4, 0.2$, and 0.1, respectively. As shown in Fig. 9.8, CSNN-0.5 offers considerable time savings and good relative error. In the case of 0.4 known entries, relative error ϵ was equal to 0.088 and 0.055 for CSNN-0.7 and NN, respectively, while CSNN-0.5 was significantly faster (113.215 seconds vs 73.67 seconds).

Results P II In this section, we present results for recovering pictures in the Image Big data set. Fig. 9.9 presents image inpainting for a sample picture with 0.1 known entries, SNR, and relative error.

Table 9.4 presents SNR, relative error, and runtime values for the tested algorithms. The CSPGD-0.4 output with a cost-effective solution of similar quality to the PGD algorithm. The SNR values were above 20 for both algorithms in case of 0.6 missing entries. In the recovery of the 80 % of an image, PGD achieved SNR

(a) Original image

(b) Case 2: Image with 10% known entries

Figure 9.6: Image inpainting scenario.

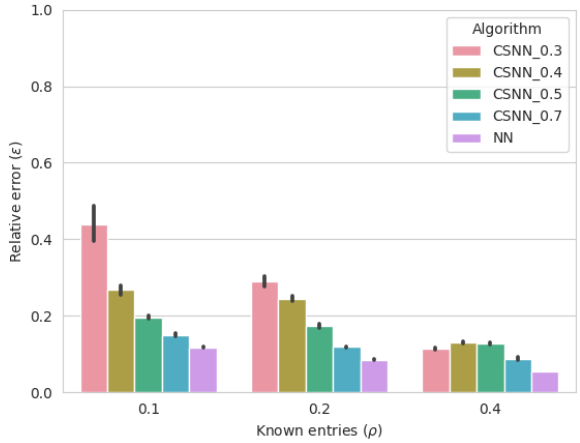
Bridge restored with CSNN-0.4

Bridge restored with CSNN-0.5

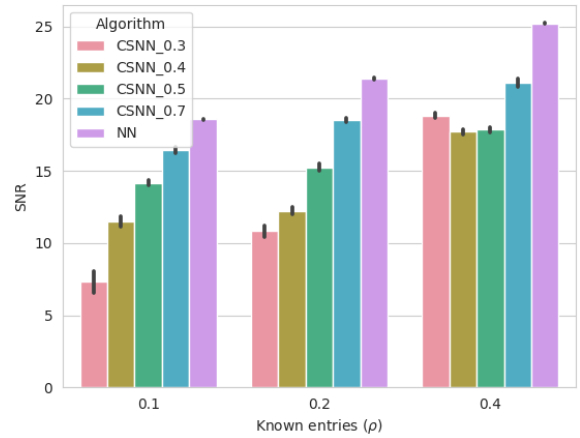
Bridge restored with CSNN-0.7

Bridge restored with NN

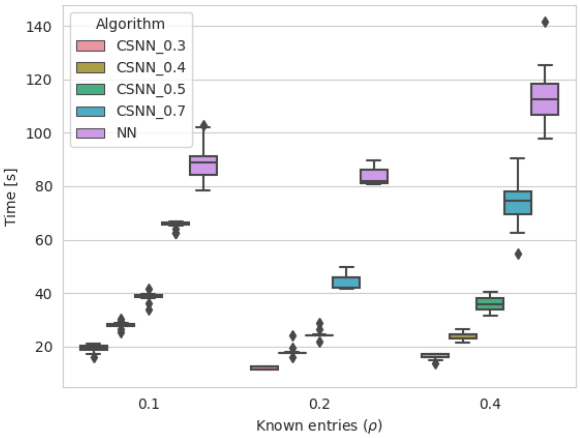
Figure 9.7: Image inpainting for P I experiment; NN and CSNN algorithms, $\rho = 20\%$ known entries.



(a) Relative error (8.2)



(b) SNR (8.7)



(c) Runtime in seconds

Figure 9.8: Results for PI; NN and CSNN-α algorithms for $\alpha \in \{0.3, 0.4, 0.5, 0.7\}$.

(a) Original image

(b) Image with known 10% entries

(c) Image recovered with CSPGD-0.3, SNR = 20.50 (8.7), $\epsilon = 0.09$ (8.2)

(d) Image recovered with CSPGD-0.4, SNR = 20.92 (8.7), $\epsilon = 0.09$(8.2)

(e) Image recovered with PGD, SNR = 21.76(8.7), $\epsilon = 0.08$ (8.2)

Figure 9.9: Image inpainting for P II experiment; PGD and CSPGD algorithms, $\rho = 10\%$ known entries.

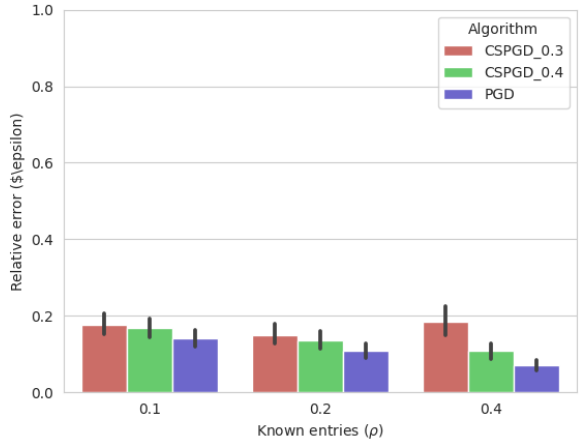
	SNR			Relative error (ϵ)			Time [s]		
Known entries ratio ρ	0.4	0.2	0.1	0.4	0.2	0.1	0.4	0.2	0.1
Algorithm									
CSNN_0.3	18.844	10.829	7.366	0.114	0.289	0.438	16.306	12.026	19.365
CSNN_0.4	17.708	12.245	11.491	0.130	0.245	0.268	24.151	17.997	28.123
CSNN_0.5	17.851	15.245	14.172	0.128	0.173	0.196	35.719	24.424	38.647
CSNN_0.7	**21.134**	**18.526**	**16.458**	**0.088**	**0.119**	**0.151**	**73.670**	**44.948**	**65.639**
NN	**25.221**	**21.377**	**18.582**	**0.055**	**0.085**	**0.118**	**113.215**	**83.196**	**88.966**

Table 9.3: Results for P I experiment; NN and CSNN-α, $\alpha \in \{0.3, 0.4, 0.5, 0.7\}$.

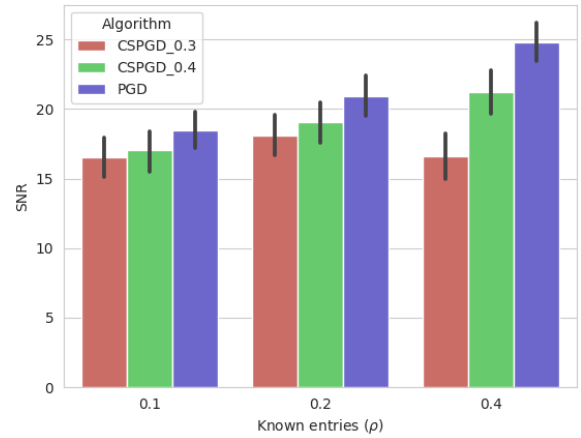
	SNR			Relative error ϵ			Time [s]		
Known entries ratio ρ	0.4	0.2	0.1	0.4	0.2	0.1	0.4	0.2	0.1
Algorithm									
CSPGD_0.3	16.621	18.115	16.537	0.185	0.150	0.176	118.831	84.615	64.262
CSPGD_0.4	**21.237**	**19.049**	**17.014**	**0.108**	**0.135**	**0.168**	**166.478**	**124.155**	**99.823**
PGD	**24.794**	**20.932**	**18.432**	**0.071**	**0.108**	**0.141**	**241.233**	**166.548**	**120.562**

Table 9.4: Results for P II experiment; PGD and CSPGD-α, $\alpha \in \{0.3, 0.4\}$.

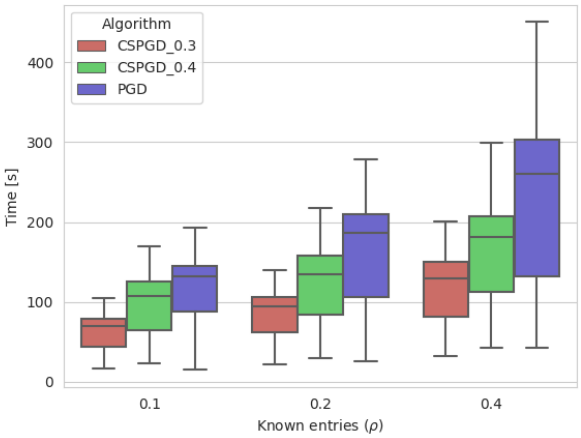
20.93, while the SNR of CSPGD-0.4 output was equal to 19.05, in the recovery of the 90 %, those values were equal to 18.43 and 17.01. In all cases, CSPGD-0.4 was relatively faster than PGD 9.10. The relative error was slightly bigger for PGD than the CSPGD-0.4 algorithm (0.071 vs 0.108, 0.108 vs 0.135, and 0.141 vs 0.168 for known rate ρ equal to 0.4, 0.2 and 0.1, respectively).



(a) Relative error (8.2)



(b) SNR (8.7)



(c) Runtime

Figure 9.10: Results for P II test scenario; PGD and CSPGD-α, $\alpha \in \{0.3, 0.4\}$.

9.3 Link prediction

Link prediction refers to predicting missing or future links in a network. The network is modelled as a graph $G = (V, E)$, with the set V of vertices and set of edges E. It can be represented as an adjacency matrix $\mathbf{A}$. If there exists an observed link between vertices i and j, then $a_{i,j} = 1$; otherwise $a_{i,j} = 0$. The link prediction problem aims to learn the distribution of existing links and, thus, to predict the potential links in the graph [69]. Link prediction has applications in various domains, including social networks, recommendation systems, biological networks, and information retrieval. In scientific collaboration networks or co-authorship networks, link prediction can be used to predict future collaborations between researchers. By analyzing past collaboration patterns, research interests, and affiliations, link prediction algorithms can identify researchers who are likely to collaborate in the future. Link prediction predicts protein-protein interactions, gene regulatory networks, or drug-target interactions in biological networks.

Link prediction techniques can be used in causal inference within graphical models. Graphical models, such as Bayesian networks or causal networks, are graphical representations of variables and their dependencies, where edges represent causal relationships or conditional dependencies between variables [184–187]. Predicted links can provide insights into potential causal relationships or dependencies between variables, which can be further explored in the context of causal inference [188, 189]. Link prediction techniques can be applied to estimate the likelihood or probability of these missing edges, indicating potential causal links that should be investigated further. Link prediction can help evaluate the strength or relevance of potential causal relationships. By predicting the presence or absence of edges, the strength of the predicted links can indicate the strength of the potential causal relationships between variables [190].

Link prediction algorithms employ various techniques, including similarity measures, graph-based methods, latent factor models, and machine learning approaches. These algorithms consider node attributes, network topology, neighbourhood structure, and observed links to estimate the likelihood of missing or future links. Matrix completion algorithms can be employed to predict missing or future links by estimating the missing entries in the adjacency matrix. The basic idea is to leverage the observed links in the network to infer the likelihood of links between pairs of nodes that have not been observed. To apply matrix completion for link prediction, the observed network structure is used to construct a partially observed adjacency matrix. The missing entries in this matrix correspond to the links that need to be predicted. Matrix completion algorithms are then applied to estimate the missing entries, effectively predicting the existence or absence of links. Matrix completion can be a valuable tool for link prediction, particularly in scenarios where the network exhibits certain regularities or follows low-dimensional structures. While matrix completion can be effective in networks with such characteristics, it may not perform optimally in networks with complex and diverse link patterns. In this section, we evaluate the performance

of the CSNN algorithm with state-of-the-art link prediction methods: Common Neighbor Centrality [191], Jaccard Coefficient [192] and Resource Allocation Index [193].

Data sets and data preprocessing We evaluated CSNN on the publicly available data set from the Koblenz Network Collection [194], namely Blogs [2] containing the hyperlinks between blogs in the context of 2004 US election. Based on the Blogs data set, we constructed a Blogs Small set containing 300×300 matrices following the works of [69, 173]. Obtained matrix had 0.9 missing entries.

Experimental procedure We compared NN and CSNN-α with the established link prediction algorithms: Common Neighbor Centrality [191], Jaccard Coefficient [192], and Resource Allocation Index [193]. To evaluate the performance of our methods, we followed the works of [69,173] by randomly dividing the existing links into training and testing samples. From the perspective of matrix completion, this is the same as randomly splitting the observed entries of the adjacency matrix $\mathbf{M}$ into corresponding Ω_{test} and Ω_{train}.

Performance measures Following [13], we adopted two metrics: Precision defined in [195], which focuses on the top predicted links, and Area Under the receiver operating characteristic Curve (AUC) defined in [13,196], which evaluates the entire set of predicted links. The ratio of the number of connected edges to the predicted number of connected edges defines precision. Filled matrix entries are interpreted as the likelihood of unobserved or new edges. The higher likelihood indicates a greater possibility of an unobserved link [69]. We sort the entries of predicted links in descending order and select the top L links. L is chosen to be the cardinality of the testing dataset. Let L_m be the number of links in the top L predicted links that appear in the testing dataset. Precision can be calculated as

$$\text{Precision} = \frac{L_m}{L}, \tag{9.2}$$

where the higher Precision indicates more accurate predictions [69], AUC measures the area under the receiver operating characteristic curve, which is interpreted as the probability that a randomly chosen missing link from the set of predicted Ω test is given a higher likelihood than a randomly chosen potentially non-existing link (which is the set of all unobserved links from G) [13]. Following [13], we calculate AUC as

$$\text{AUC} = \frac{y_m + 0.5y_n}{y}, \tag{9.3}$$

[2]Blogs data set(`https://http://konect.cc/networks/moreno_blogs/`)

Algorithm	AUC mean	AUC std	Precision mean	Precision std	Time [s] mean	Time [s] std
CSNN-0.3	5.85e-01	1.14e-02	2.64e-01	1.48e-02	3.13e+02	3.84e+02
CSNN-0.5	5.92e-01	1.94e-02	2.78e-01	9.80e-03	4.73e+02	2.82e+02
CSNN-0.7	5.97e-01	1.41e-02	2.86e-01	1.13e-02	8.47e+02	8.88e+02
NN	6.01e-01	1.65e-02	2.95e-01	1.05e-02	1.50e+03	1.71e+03
Common neighbour centrality	6.36e-01	1.71e-02	3.18e-01	1.33e-02	8.37e-02	3.86e-03
Jaccard Coefficient	**6.38e-01**	**1.15e-02**	**3.15e-01**	**1.33e-02**	**1.58e-05**	**2.87e-05**
Resource allocation index	**6.39e-01**	**1.17e-02**	**3.12e-01**	**1.23e-02**	**1.55e-05**	**3.72e-05**

Table 9.5: Results link prediction in G I; Resource allocation index, Jaccard Coefficient, Common neighbour centrality, NN, CSNN-α, $\alpha \in \{0.3, 0.5, 0.7\}$.

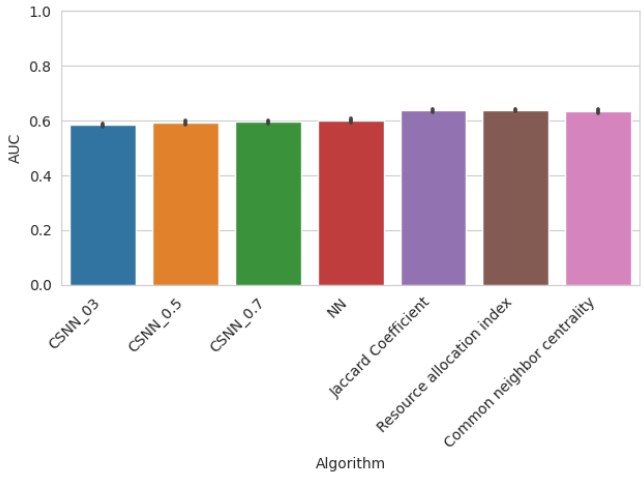
where y is the number of independent comparisons between each randomly picked pair of a missing link and a non-existing link. y_m is the times that the missing links have a higher predicted likelihood than non-existing links while y_n counts the number of times if their likelihoods are equal. We use $y = 1000$. The degree to which the AUC exceeds 0.5 indicates how much better the predictions are compared with a random guess [173].

Results Table 9.5 presents a comparison between NN, CSNN-α for $\alpha \in \{0.3, 0.5, 0.7\}$ with algorithms based on Jaccard Coefficient, Resource Allocation Index and Common Neighbour Centrality. All established algorithms resulted in more accurate predictions than NN and CSNN algorithms. Although the quality of the NN and CSNN algorithms is similar to the established link prediction algorithms, the runtime of the NN and CSNN algorithms is significantly higher 9.11.

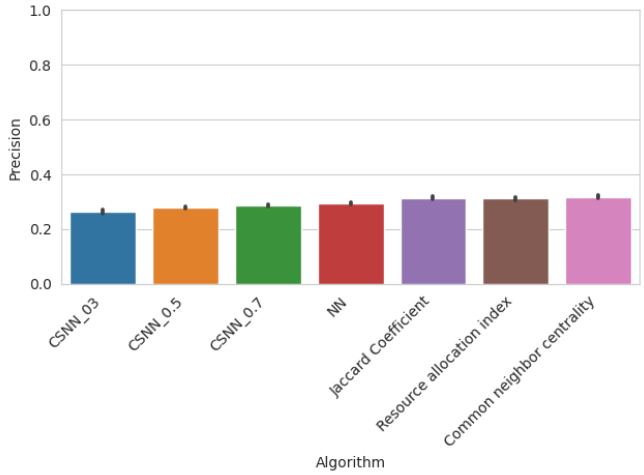
9.4 Conclusion and discussion

The low-rank matrix completion has broad applicability in various domains where incomplete or corrupted matrices must be reconstructed or predicted based on their low-rank structure. Here, we assess the performance of the presented CSMC methods in the recommendation system, image imputation and link prediction problems. For small data sets, we compared exact NN and CSNN-α algorithms using SDP. SDP-based matrix completion methods have provided accurate results in various applications, including recommendation systems, image impainting, and network analysis. While SDP can be a valuable tool, it may not always be the most practical or efficient solution, particularly for large-scale problems. Thus for the large-scale problem, we compare PGD and CSPGD-α algorithms.

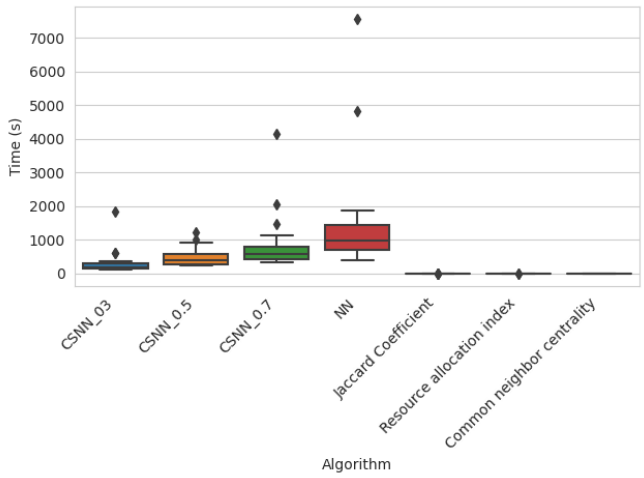
Nuclear norm minimization offers theoretical guarantees for the approximation error of the reconstructed low-rank approximation of the data matrix. The proposed CSNN algorithms output a solution of similar quality and result in good time savings. For the movie recommendation problem, the accuracy of the ob-



AUC



(a) Precision



(b) Runtime

Figure 9.11: Results for G I experiment; various link prediction algorithms.

tained prediction is measured with NMAE and HR. Sampling 0.7 columns in the CSNN algorithm results in predictions of the same quality as NN and is 1.4 times faster. Sampling 0.3 columns reduces mean execution time from almost 1 minute to 10 seconds and outputs predictions with moderately lower quality. We compared PGD and CSPGD-α algorithms for α equal to 0.3 and 0.5. CSPGD-0.3 produced recommendations of the same quality as PGD and was remarkably faster. However, numerous modifications have been developed to enhance the efficiency of PGD in matrix completion [197].

In the image impainting task, the quality of the recovered image was assessed with SNR and approximation error. A higher SNR value indicates a higher quality image with less noise and better fidelity to the original image. CSNN is an effective strategy for recovering missing parts of pictures in scenarios with small pictures. Sampling 0.7 columns allowed us to obtain solutions with comparably good quality to NN and offered computational savings. For images with higher resolution, we applied PGD-based methods. In this experiment, sampling 0.4 columns allow us to recover images with similar SNR and relative approximation error as PGD. The CSPGD algorithms were faster than PGD but offered smaller runtime savings than in the case of recommendation systems. This is caused by the fact that Movie Lens Big data sets consisted of thick matrices with columns number significantly bigger than rows number.

Matrix completion methods can be applied in link prediction by treating the network as an adjacency matrix, where the presence or absence of edges represents the observed or missing links, respectively. While link prediction involves estimating missing values, it often involves considering additional factors such as features, similarity measures, or network properties. On the other hand, matrix completion methods focus on exploiting the low-rank structure assumption of the matrix. The success of matrix completion for link prediction depends on the underlying assumptions of low-rank structure and the availability of informative observed links. Matrix completion techniques can provide reasonable predictions if the network has a low-rank structure and the observed links capture relevant patterns. The conducted experiment has shown that the accuracy of prediction of MC methods was slightly worse than established link prediction methods. However, NN and CSNN, based on SDP, were dramatically slower than link prediction algorithms.

The assumption of uniform sampling in matrix completion problems is simplified and idealized. While it can be helpful in theoretical analyses and algorithm development, it may not always hold in practical scenarios. The observed entries may exhibit specific patterns, biases, or other mechanisms that deviate from uniform sampling in many real-world scenarios. For example, in recommendation systems, user-item ratings may be missing systematically due to user preferences or data collection processes. However, this simplification allows for developing efficient algorithms, theoretical analysis, and understanding the fundamental limits of matrix completion [15]. Nuclear norm minimization techniques deal with some deviations from uniform sampling if the matrix could be completed with

uniformly sampled entries. If the solution matrix with the lowest nuclear norm among matrices matches known entries, it will remain the solution after recovering a non-uniformly sampled subset of its entries.

Chapter 10

Conclusion and discussion

This thesis introduces Column Selected Matrix Completion (CSMC), a two-staged method for the low-rank matrix completion problem. In the first stage, CSMC applies a low-rank matrix completion algorithm for the smaller problem by randomly selecting and filling the column submatrix. In the second stage, CSMC seeks a matrix which minimizes the least squares error calculated for the filled and previously observed entries. We designed three variants of the CSMC depending on the matrix completion algorithm used in the first stage and the least squares minimization method used in the second stage. All presented methods used nuclear norm minimization based matrix completion methods. Nuclear norm minimization provides an efficient and well-understood approach to matrix completion problems, with well-understood strong theoretical guarantees.

Theoretical guarantees provide a foundation of trust and reliability by ensuring that the algorithms meet specific criteria and perform as expected. They assure that the algorithms will generalize and provide reliable predictions or decisions. They provide insights into the underlying principles and assumptions, guiding the design of new models and optimization techniques. Theoretical analysis often reveals fundamental trade-offs and helps identify the best approaches to tackle specific learning problems. For this reason, the important part of this thesis was to provide provable guarantees for CSMC. In particular, we show that under standard assumptions about matrix rank, matrix incoherence, size and distribution of observed entries, solving the least squares problem in the second stage of CSMC perfectly recovers the matrix.

We conducted several numerical simulations to verify the theoretical model and assess the performance of CSNN, CSPGD, and CSPGD-adam algorithms. On small random matrices, we benchmarked CSNN with off-the-shelf nuclear norm minimization and showed that CSNN offers considerable time savings while preserving the quality of the solutions. We have also compared the performance of CSNN with matrix completion based on matrix factorization (MF) and iterative SVD. Although MF was faster, the CSNN solution achieved a smaller relative approximation error. To overcome performance bottlenecks of the SDP solvers,

researchers designed various first-order methods for the relaxed nuclear norm minimization, including algorithms based on the proximal gradient descent (PGD). Since PGD requires calculating the SVD in each iteration, CSPGD may limit the computational burden. We run experiments comparing PGD, CSPGD, and CSPGD-adam on the synthetic data set to verify this hypothesis. As expected, in the case of thick matrices, CSPGD and CSPGD-adam are considerably faster than PGD while preserving the quality of solutions.

Low-rank matrix completion has a wide range of applications across various domains. In this thesis, we assessed the performance and practical significance of the CSMC method in three real-world problems: recommendation systems, image imputation, and link prediction in graphs. Unlike synthetic datasets, real datasets reflect the characteristics and complexities of the actual problem domain. The recommendation systems have significantly contributed to the research on low-rank matrix completion due to their practical importance and economic impact. Thick matrices may often represent such systems.

As expected, in this application, CSNN and CSPGD offered the biggest computational savings. CSNN and CSPGD proved successful and efficient in the image imputation tasks. Matrix completion methods can be applied to link prediction tasks by treating the adjacency matrix of a network as the matrix to be completed. However, it's important to note that link prediction may involve additional considerations specific to network analysis, such as network topology, graph algorithms, and domain-specific knowledge, which may not be present in the general matrix completion problem. The experiments reflected this in which established link-prediction methods outperformed NN and CSNN algorithms.

Presented CSMC methods sampled columns according to the uniform distribution and applied nuclear norm based matrix completion algorithms. However, the presented framework is versatile and allows employing other sampling methods and MC methods. Recent research resulted in more sophisticated Column Subset Selection algorithms in missing data setup. Applying such a method could handle matrices with higher coherence. Another area of research is to bridge matrix completion methods using non-convex optimization with CSS algorithms.

Algorithms for convex optimization have proven efficient in many other related areas after the work on matrix completion. Both robust PCA and signal phase retrieval have benefited from the insights and techniques developed in the field of matrix completion. Researchers have drawn connections between these areas and leveraged ideas such as low-rank matrix recovery, nuclear norm minimization, and convex optimization to develop robust PCA and phase retrieval algorithms. In further research, we will focus on applying Column Subset Selection to those problems.

The success and insights gained from matrix completion methods have motivated researchers to extend these ideas to higher-order tensors, developing low-rank tensor completion techniques. By building upon the foundations established in matrix completion, we will focus on handling tensor data in further research, leveraging the low-rank assumption to recover missing entries and approximate

tensors with low-rank structures.

In summary, the author's original contribution includes:

1. The CSMC - two staged low-rank matrix completion method, dedicated to completing rectangular matrices with one dimension significantly larger than the other.

2. Three algorithms implementing CSMC, namely Column Selected Nuclear Norm (CSNN), Column Selected Proximal Gradient Descent (CSPGD), and Column Selected Proximal Gradient Descent - Adam (CSPGD-adam),

3. Formal analysis CSMC, in particular, the reconstruction error of the second stage

4. The open-source Python 3 library with CSNN, CSPGD, and CSPGD-adam, supporting both CPU and GPU computations,

5. The open-source library for benchmarking matrix completion methods.

6. Numerical evaluation and comparison with methods in the literature.

Data plays a fundamental role in machine learning, as it serves as the basis for training models, evaluating their performance, and making predictions or decisions. The availability of high-quality and relevant data is essential for successful machine learning applications. Many real-world datasets are inherently incomplete, meaning certain entries are unknown or missing. Matrix completion techniques aim to fill in these missing entries, enabling the reconstruction or approximation of the complete matrix.

Bibliography

[1] M. P. Karpowicz and G. Strang, "The pseudoinverse of $A = CR$ is $A^+ = R^+ + C^+(?)$," *arXiv preprint arXiv:2305.01716*, 2023.

[2] A. Ben-Israel and T. Greville, *Adi Ben-Israel and Thomas N.E. Greville, Generalized Inverses: Theory and Applications, ISBN 0-387-00293-6*. 01 2003.

[3] A. Moitra, *Algorithmic Aspects of Machine Learning*. USA: Cambridge University Press, 1st ed., 2018.

[4] M. WEIMIN, "Matrix completion models with fixed basis coefficients and rank regularized problems with hard constraints," 2013.

[5] S. Oh, *Matrix completion: Fundamental limits and efficient algorithms*. Stanford University, 2010.

[6] J. Jafarov, "Survey of matrix completion algorithms," *arXiv preprint arXiv:2204.01532*, 2022.

[7] X. Yu, F. Jiang, J. Du, and D. Gong, "A cross-domain collaborative filtering algorithm with expanding user and item features via the latent factor space of auxiliary domains," *Pattern Recognition*, vol. 94, pp. 96–109, 2019.

[8] Y. Koren, S. Rendle, and R. Bell, "Advances in collaborative filtering," *Recommender systems handbook*, pp. 91–142, 2021.

[9] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.

[10] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 447–456, 2009.

[11] H. Polat and W. Du, "Svd-based collaborative filtering with privacy," in *Proceedings of the 2005 ACM symposium on Applied computing*, pp. 791–795, 2005.

[12] Q. Ba, X. Li, and Z. Bai, "Clustering collaborative filtering recommendation system based on svd algorithm," in *2013 IEEE 4th International Conference on Software Engineering and Service Science*, pp. 963–967, IEEE, 2013.

[13] H. Cai, L. Huang, P. Li, and D. Needell, "Matrix completion with cross-concentrated sampling: Bridging uniform sampling and cur sampling," 08 2022.

[14] A. Ramlatchan, M. Yang, Q. Liu, M. Li, J. Wang, and Y. Li, "A survey of matrix completion methods for recommendation systems," *Big Data Mining and Analytics*, vol. 1, no. 4, pp. 308–323, 2018.

[15] B. Recht, "A simpler approach to matrix completion," 2009.

[16] D. Gross, "Recovering low-rank matrices from few coefficients in any basis," *IEEE Transactions on Information Theory*, vol. 57, p. 1548–1566, Mar 2011.

[17] Y. Chen and Y. Chi, "Harnessing structures in big data via guaranteed low-rank matrix estimation: Recent theory and fast algorithms via convex and nonconvex optimization," *IEEE Signal Processing Magazine*, vol. 35, no. 4, pp. 14–31, 2018.

[18] J.-F. Cai, E. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, pp. 1956–1982, 03 2010.

[19] J.-F. Cai, E. J. Candes, and Z. Shen, "A singular value thresholding algorithm for matrix completion," 2008.

[20] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[21] G. Kutyniok, "Compressed sensing," *Mitteilungen der Deutschen Mathematiker-Vereinigung*, vol. 22, no. 1, pp. 24–29, 2014.

[22] Y. Tsaig and D. L. Donoho, "Extensions of compressed sensing," *Signal processing*, vol. 86, no. 3, pp. 549–571, 2006.

[23] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed sensing mri," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 72–82, 2008.

[24] S. L. Brunton, J. L. Proctor, J. H. Tu, and J. N. Kutz, "Compressed sensing and dynamic mode decomposition," *Journal of computational dynamics*, vol. 2, no. 2, pp. 165–191, 2016.

[25] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010.

[26] E. J. Candes and B. Recht, "Exact matrix completion via convex optimization," 2008.

[27] R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral regularization algorithms for learning large incomplete matrices," *Journal of Machine Learning Research*, vol. 11, no. 80, pp. 2287–2322, 2010.

[28] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[29] J.-F. Cai and S. Osher, "Fast singular value thresholding without singular value decomposition," *Methods and Applications of Analysis*, vol. 20, no. 4, pp. 335–352, 2013.

[30] D. P. Woodruff, "Sketching as a tool for numerical linear algebra," vol. 10, p. 1–157, oct 2014.

[31] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 948–958, 2010.

[32] P. Drineas, M. Mahoney, S. Muthukrishnan, and T. Sarlos, "Faster least squares approximation," *Numerische Mathematik*, vol. 117, 10 2007.

[33] K. Hamm and L. Huang, "Perspectives on cur decompositions," *Applied and Computational Harmonic Analysis*, vol. 48, no. 3, pp. 1088–1099, 2020.

[34] K. Hamm, M. Meskini, and H. Cai, "Riemannian cur decompositions for robust principal component analysis," in *Proceedings of Topological, Algebraic, and Geometric Learning Workshops 2022* (A. Cloninger, T. Doster, T. Emerson, M. Kaul, I. Ktena, H. Kvinge, N. Miolane, B. Rieck, S. Tymochko, and G. Wolf, eds.), vol. 196 of *Proceedings of Machine Learning Research*, pp. 152–160, 25 Feb–22 Jul 2022.

[35] C. Boutsidis, M. W. Mahoney, and P. Drineas, "An improved approximation algorithm for the column subset selection problem," in *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pp. 968–977, SIAM, 2009.

[36] C. Boutsidis and D. P. Woodruff, "Optimal cur matrix decompositions," in *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 353–362, 2014.

[37] M. Xu, R. Jin, and Z.-H. Zhou, "Cur algorithm for partially observed matrices," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 1412–1421, JMLR.org, 2015.

[38] X. P. Li, L. Huang, H. So, and B. Zhao, "A survey on matrix completion: Perspective of signal processing," 01 2019.

[39] A. Mongia and A. Majumdar, *Matrix completion techniques in bioinformatics*. PhD thesis, IIIT-Delhi, 2020.

[40] J. Gillard and K. Usevich, "Hankel low-rank approximation and completion in time series analysis and forecasting: a brief review," *arXiv preprint arXiv:2206.05103*, 2022.

[41] L. Kassab, *Iterative Matrix Completion and Topic Modeling Using Matrix and Tensor Factorizations*. PhD thesis, Colorado State University, 2021.

[42] C. Genes, *Novel matrix completion methods for missing data recovery in urban systems*. PhD thesis, University of Sheffield, 2018.

[43] J.-H. Kim, J.-Y. Sim, and C.-S. Kim, "Video deraining and desnowing using temporal correlation and low-rank matrix completion," *IEEE Transactions on Image Processing*, vol. 24, no. 9, pp. 2658–2670, 2015.

[44] W. Li, L. Zhao, Z. Lin, D. Xu, and D. Lu, "Non-local image inpainting using low-rank matrix completion," in *Computer Graphics Forum*, vol. 34, pp. 111–122, Wiley Online Library, 2015.

[45] W. He, H. Zhang, L. Zhang, and H. Shen, "Total-variation-regularized low-rank matrix factorization for hyperspectral image restoration," *IEEE transactions on geoscience and remote sensing*, vol. 54, no. 1, pp. 178–188, 2015.

[46] M. Le Pendu, X. Jiang, and C. Guillemot, "Light field inpainting propagation via low rank matrix completion," *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1981–1993, 2018.

[47] H. Xue, S. Zhang, and D. Cai, "Depth image inpainting: Improving low rank matrix completion with low gradient regularization," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4311–4320, 2017.

[48] C. Lu, M. Yang, F. Luo, F.-X. Wu, M. Li, Y. Pan, Y. Li, and J. Wang, "Prediction of lncrna–disease associations based on inductive matrix completion," *Bioinformatics*, vol. 34, no. 19, pp. 3357–3364, 2018.

[49] X. Chen, L. Wang, J. Qu, N.-N. Guan, and J.-Q. Li, "Predicting mirna–disease association based on inductive matrix completion," *Bioinformatics*, vol. 34, no. 24, pp. 4256–4265, 2018.

[50] X. Chen, L.-G. Sun, and Y. Zhao, "Ncmcmda: mirna–disease association prediction through neighborhood constraint matrix completion," *Briefings in bioinformatics*, vol. 22, no. 1, pp. 485–496, 2021.

[51] M. Doneva, T. Amthor, P. Koken, K. Sommer, and P. Börnert, "Matrix completion-based reconstruction for undersampled magnetic resonance fingerprinting data," *Magnetic resonance imaging*, vol. 41, pp. 41–52, 2017.

[52] P. J. Shin, P. E. Larson, M. A. Ohliger, M. Elad, J. M. Pauly, D. B. Vigneron, and M. Lustig, "Calibrationless parallel imaging reconstruction based on structured low-rank matrix completion," *Magnetic resonance in medicine*, vol. 72, no. 4, pp. 959–970, 2014.

[53] K.-H. Thung, E. Adeli, P.-T. Yap, and D. Shen, "Stability-weighted matrix completion of incomplete multi-modal data for disease diagnosis," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*, pp. 88–96, Springer, 2016.

[54] B. Sturmfels and C. Uhler, "Multivariate gaussians, semidefinite matrix completion, and convex algebraic geometry," *Annals of the Institute of Statistical Mathematics*, vol. 62, no. 4, pp. 603–638, 2010.

[55] D. Alpago, M. Zorzi, and A. Ferrante, "Identification of sparse reciprocal graphical models," *IEEE control systems letters*, vol. 2, no. 4, pp. 659–664, 2018.

[56] S. Zhang and Y. D. Zhang, "Low-rank hankel matrix completion for robust time-frequency analysis," *IEEE Transactions on Signal Processing*, vol. 68, pp. 6171–6186, 2020.

[57] Y. Chen and Y. Chi, "Spectral compressed sensing via structured matrix completion," in *International Conference on Machine Learning*, pp. 414–422, PMLR, 2013.

[58] K. Xie, X. Ning, X. Wang, D. Xie, J. Cao, G. Xie, and J. Wen, "Recover corrupted data in sensor networks: A matrix completion solution," *IEEE Transactions on Mobile Computing*, vol. 16, no. 5, pp. 1434–1448, 2016.

[59] F. Xiao, W. Liu, Z. Li, L. Chen, and R. Wang, "Noise-tolerant wireless sensor networks localization via multinorms regularized matrix completion," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2409–2419, 2017.

[60] S. Sadeghi Eshkevari and S. N. Pakzad, "Signal reconstruction from mobile sensors network using matrix completion approach," in *Topics in Modal Analysis & Testing, Volume 8: Proceedings of the 37th IMAC, A Conference and Exposition on Structural Dynamics 2019*, pp. 61–75, Springer, 2020.

[61] X. Liu, X. Wang, L. Zou, J. Xia, and W. Pang, "Spatial imputation for air pollutants data sets via low rank matrix completion algorithm," *Environment international*, vol. 139, p. 105713, 2020.

[62] Y. Yu, J. James, V. O. Li, and J. C. Lam, "A novel interpolation-svt approach for recovering missing low-rank air quality data," *IEEE Access*, vol. 8, pp. 74291–74305, 2020.

[63] G. I. Winata, A. Madotto, J. Shin, E. J. Barezi, and P. Fung, "On the effectiveness of low-rank matrix factorization for lstm model compression," *arXiv preprint arXiv:1908.09982*, 2019.

[64] M. P. Karpowicz, A. Krajewska, and I. Okulska, "Resource allocation in clusters of virtual machines," in *2019 International Conference on High Performance Computing & Simulation (HPCS)*, pp. 754–760, IEEE, 2019.

[65] P. Arabas and E. Niewiadomska-Szynkiewicz, "Energy-efficient workload allocation in distributed hpc system," in *2019 International Conference on High Performance Computing & Simulation (HPCS)*, pp. 747–753, IEEE, 2019.

[66] A. Krajewska, "Performance modeling of database systems: a survey," *Journal of Telecommunications and Information Technology*, no. 4, pp. 37–45, 2018.

[67] C. Delimitrou and C. Kozyrakis, "Paragon: Qos-aware scheduling for heterogeneous datacenters," *ACM SIGPLAN Notices*, vol. 48, no. 4, pp. 77–88, 2013.

[68] C. Delimitrou and C. Kozyrakis, "Qos-aware scheduling in heterogeneous datacenters with paragon," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 4, pp. 1–34, 2013.

[69] R. Pech, D. Hao, L. Pan, H. Cheng, and T. Zhou, "Link prediction via matrix completion," *Europhysics Letters*, vol. 117, p. 38002, mar 2017.

[70] G. Mahindre, A. P. Jayasumana, K. Gajamannage, and R. Paffenroth, "On sampling and recovery of topology of directed social networks–a low-rank matrix completion based approach," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, pp. 324–331, IEEE, 2019.

[71] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996.

[72] J. David, "Algorithms for analysis, design of robust controllers," 1994.

[73] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theor.*, vol. 52, p. 489–509, feb 2006.

[74] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Trans. Inf. Theor.*, vol. 52, p. 5406–5425, dec 2006.

[75] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theor.*, vol. 52, p. 1289–1306, apr 2006.

[76] S. L. Brunton, J. L. Proctor, J. H. Tu, and J. N. Kutz, "Compressed sensing and dynamic mode decomposition," *Journal of Computational Dynamics*, vol. 2, no. 2, pp. 165–191, 2015.

[77] M. Jaggi, "Revisiting Frank-Wolfe: Projection-free sparse convex optimization," in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester, eds.), vol. 28 of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 427–435, PMLR, 17–19 Jun 2013.

[78] R. Freund, P. Grigas, and R. Mazumder, "An extended frank–wolfe method with "in-face" directions, and its application to low-rank matrix completion," *SIAM Journal on Optimization*, vol. 27, 11 2015.

[79] N. Rao, P. Shah, and S. Wright, "Forward–backward greedy algorithms for atomic norm regularization," *IEEE Transactions on Signal Processing*, vol. 63, no. 21, pp. 5798–5811, 2015.

[80] N. Boyd, G. Schiebinger, and B. Recht, "The alternating descent conditional gradient method for sparse inverse problems," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 616–639, 2017.

[81] Z. Allen-Zhu, E. Hazan, W. Hu, and Y. Li, "Linear convergence of a frank-wolfe type algorithm over trace-norm balls," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[82] A. Yurtsever, M. Udell, J. Tropp, and V. Cevher, "Sketchy decisions: Convex low-rank matrix optimization with optimal storage," 02 2017.

[83] S. Burer and R. Monteiro, "Local minima and convergence in low-rank semidefinite programming," *Mathematical Programming*, vol. 103, pp. 427–444, 07 2005.

[84] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, (New York, NY, USA), p. 665–674, Association for Computing Machinery, 2013.

[85] M. Hardt, "Understanding alternating minimization for matrix completion," *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 651–660, 2014.

[86] M. Udell, C. Horn, R. Zadeh, and S. Boyd, *Generalized Low Rank Models*. 01 2016.

[87] C. Ma, K. Wang, Y. Chi, and Y. Chen, "Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval, matrix completion and blind deconvolution," *Foundations of Computational Mathematics*, vol. 20, 11 2017.

[88] Y. Chen and M. J. Wainwright, "Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees," *ArXiv*, vol. abs/1509.03025, 2015.

[89] P. Jain, R. Meka, and I. Dhillon, "Guaranteed rank minimization via singular value projection," in *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10, (Red Hook, NY, USA), p. 937–945, Curran Associates Inc., 2010.

[90] P. Jain and P. Netrapalli, "Fast exact matrix completion with finite samples," 11 2014.

[91] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. Altman, "Missing value estimation methods for dna microarrays," *Bioinformatics*, vol. 17 6, pp. 520–5, 2001.

[92] B. Mishra, G. Meyer, S. Bonnabel, and R. Sepulchre, "Fixed-rank matrix factorizations and riemannian low-rank optimization," *Comput. Stat.*, vol. 29, p. 591–621, jun 2014.

[93] B. Vandereycken, "Low-rank matrix completion by riemannian optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1214–1236, 2013.

[94] T. Ngo and Y. Saad, "Scaled gradients on grassmann manifolds for matrix completion," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[95] L. Cambier and P.-A. Absil, "Robust low-rank matrix completion by riemannian optimization," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. S440–S460, 2016.

[96] R. H. Keshavan, S. Oh, and A. Montanari, "Matrix completion from a few entries," ISIT'09, p. 324–328, IEEE Press, 2009.

[97] R. Sun and Z.-Q. Luo, "Guaranteed matrix completion via non-convex factorization," *IEEE Transactions on Information Theory*, vol. 62, pp. 6535–6579, 11 2016.

[98] A. Krishnamurthy and A. Singh, "Low-rank matrix and tensor completion via adaptive sampling," in *Advances in Neural Information Processing Systems* (C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, eds.), vol. 26, Curran Associates, Inc., 2013.

[99] M.-F. Balcan and H. Zhang, "Noise-tolerant life-long matrix completion via adaptive sampling," in *NIPS*, 2016.

[100] A. Krishnamurthy and A. Singh, "On the power of adaptivity in matrix completion and approximation," *ArXiv*, vol. abs/1407.3619, 2014.

[101] T. T. Mai and P. Alquier, "A bayesian approach for noisy matrix completion: Optimal rate under general sampling distribution," 2015.

[102] L. Yang, J. Fang, H. Duan, H. Li, and B. Zeng, "Fast low-rank bayesian matrix completion with hierarchical gaussian prior models," *IEEE Transactions on Signal Processing*, vol. 66, no. 11, pp. 2804–2817, 2018.

[103] P. Alquier, V. Cottet, N. Chopin, and J. Rousseau, "Bayesian matrix completion: prior specification," *arXiv preprint arXiv:1406.1440*, 2014.

[104] S. De, H. Salehi, and A. Gorodetsky, "Efficient mcmc sampling for bayesian matrix factorization by breaking posterior symmetries," *arXiv preprint arXiv:2006.04295*, 2020.

[105] M. Lei, A. Labbe, Y. Wu, and L. Sun, "Bayesian kernelized matrix factorization for spatiotemporal traffic data imputation and kriging," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18962–18974, 2022.

[106] P. Jain and I. S. Dhillon, "Provable inductive matrix completion," *arXiv preprint arXiv:1306.0626*, 2013.

[107] N. Natarajan and I. S. Dhillon, "Inductive matrix completion for predicting gene–disease associations," *Bioinformatics*, vol. 30, no. 12, pp. i60–i68, 2014.

[108] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of the ACM (JACM)*, vol. 58, no. 3, pp. 1–37, 2011.

[109] H. Cai, K. Hamm, L. Huang, and D. Needell, "Robust cur decomposition: Theory and imaging applications," *SIAM J. Imaging Sci.*, vol. 14, pp. 1472–1503, 2021.

[110] Z. Liu, A. Hansson, and L. Vandenberghe, "Nuclear norm system identification with missing inputs and outputs," *Systems & Control Letters*, vol. 62, no. 8, pp. 605–612, 2013.

[111] M. Verhaegen and A. Hansson, "N2sid: Nuclear norm subspace identification of innovation models," *Automatica*, vol. 72, pp. 57–63, 2016.

[112] E. J. Candes, Y. C. Eldar, T. Strohmer, and V. Voroninski, "Phase retrieval via matrix completion," *SIAM review*, vol. 57, no. 2, pp. 225–251, 2015.

[113] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev, "Phase retrieval with application to optical imaging: a contemporary overview," *IEEE signal processing magazine*, vol. 32, no. 3, pp. 87–109, 2015.

[114] G. Odor, Y.-H. Li, A. Yurtsever, Y.-P. Hsieh, Q. Tran-Dinh, M. El Halabi, and V. Cevher, "Frank-wolfe works for non-lipschitz continuous gradient objectives: Scalable poisson phase retrieval," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6230–6234, Ieee, 2016.

[115] M. Fazel, H. Hindi, and S. Boyd, "Rank minimization and applications in system theory," in *Proceedings of the 2004 American Control Conference*, vol. 4, 2004.

[116] J. F. Sturm, "Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones," *Optimization methods and software*, vol. 11, no. 1-4, pp. 625–653, 1999.

[117] K.-C. Toh, M. J. Todd, and R. H. Tütüncü, "Sdpt3—a matlab software package for semidefinite programming, version 1.3," *Optimization methods and software*, vol. 11, no. 1-4, pp. 545–581, 1999.

[118] E. D. Andersen and K. D. Andersen, *The Mosek Interior Point Optimizer for Linear Programming: An Implementation of the Homogeneous Algorithm*, pp. 197–232. Boston, MA: Springer US, 2000.

[119] J. Zhang, B. O'Donoghue, and S. Boyd, "Globally convergent type–I Anderson acceleration for non-smooth fixed-point iterations," *SIAM Journal on Optimization*, vol. 30, no. 4, pp. 3170–3197, 2020.

[120] M. Schmidt, N. Roux, and F. Bach, "Convergence rates of inexact proximal-gradient methods for convex optimization," in *Advances in Neural Information Processing Systems* (J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, eds.), vol. 24, Curran Associates, Inc., 2011.

[121] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[122] Q. Yao and J. T. Kwok, "Accelerated inexact soft-impute for fast large-scale matrix completion," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[123] E. J. Candes and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2053–2080, 2010.

[124] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, "Relative-error cur matrix decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 844–881, 2008.

[125] J. Tropp, "Improved analysis of the subsamples randomized hadamard transform," *Advances in Adaptive Data Analysis*, vol. 03, 11 2010.

[126] M. Gu and S. C. Eisenstat, "Efficient algorithms for computing a strong rank-revealing qr factorization," *SIAM J. Sci. Comput.*, vol. 17, pp. 848–869, 1996.

[127] T. F. Chan, "Rank revealing qr factorizations," *Linear algebra and its applications*, vol. 88, pp. 67–82, 1987.

[128] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang, "Matrix approximation and projective clustering via volume sampling.," *Theory of Computing*, vol. 2, pp. 225–247, 01 2006.

[129] A. Frieze, R. Kannan, and S. Vempala, "Fast monte-carlo algorithms for finding low-rank approximations," *J. ACM*, vol. 51, p. 1025–1041, nov 2004.

[130] P. Drineas, M. Mahoney, and S. Muthukrishnan, "Relative-error *cur* matrix decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, pp. 844–881, 05 2008.

[131] M. P. Karpowicz, "A theory of meta-factorization," 2021.

[132] M. Karpowicz, "The secret life of matrix factorizations: how matrix decompositions reveal and keep secrets of linear equations and what we can do about it," 04 2023.

[133] D. C. Sorensen and M. Embree, "A deim induced cur factorization," *SIAM Journal on Scientific Computing*, vol. 38, no. 3, pp. A1454–A1482, 2016.

[134] S. Voronin and P.-G. Martinsson, "Efficient algorithms for cur and interpolative matrix decompositions," *Advances in Computational Mathematics*, vol. 43, pp. 495–516, 2017.

[135] B. Kramer and A. A. Gorodetsky, "System identification via cur-factored hankel approximation," *SIAM Journal on Scientific Computing*, vol. 40, no. 2, pp. A848–A866, 2018.

[136] E. P. Hendryx, B. M. Rivière, D. C. Sorensen, and C. G. Rusin, "Finding representative electrocardiogram beat morphologies with cur," *Journal of biomedical informatics*, vol. 77, pp. 97–110, 2018.

[137] Y. Wang and A. Singh, "Provably correct algorithms for matrix column subset selection with selectively sampled data," *J. Mach. Learn. Res.*, vol. 18, p. 5699–5740, jan 2017.

[138] A. Eftekhari, M. B. Wakin, and R. A. Ward, "MC2: a two-phase algorithm for leveraged matrix completion," *Information and Inference: A Journal of the IMA*, vol. 7, pp. 581–604, 02 2018.

[139] G. Raskutti and M. Mahoney, "A statistical perspective on randomized sketching for ordinary least-squares," vol. 17, 06 2014.

[140] H. Diao, Z. Song, D. Woodruff, and X. Yang, "Total least squares regression in input sparsity time," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[141] V. Georgiou, C. Boutsikas, P. Drineas, and H. Anzt, "A mixed precision randomized preconditioner for the lsqr solver on gpus," in *High Performance Computing* (A. Bhatele, J. Hammond, M. Baboulin, and C. Kruse, eds.), (Cham), pp. 164–181, Springer Nature Switzerland, 2023.

[142] A. Lydia and S. Francis, "Adagrad—an optimizer for stochastic gradient descent," *Int. J. Inf. Comput. Sci*, vol. 6, no. 5, pp. 566–568, 2019.

[143] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[144] L. Bottou, "Stochastic gradient descent tricks," *Neural Networks: Tricks of the Trade: Second Edition*, pp. 421–436, 2012.

[145] N. Ketkar and N. Ketkar, "Stochastic gradient descent," *Deep learning with Python: A hands-on introduction*, pp. 113–132, 2017.

[146] M. Zinkevich, M. Weimer, L. Li, and A. Smola, "Parallelized stochastic gradient descent," *Advances in neural information processing systems*, vol. 23, 2010.

[147] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford, "Accelerating stochastic gradient descent for least squares regression," in *Conference On Learning Theory*, pp. 545–604, PMLR, 2018.

[148] P. Jain, S. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford, "Parallelizing stochastic gradient descent for least squares regression: mini-batching, averaging, and model misspecification," *Journal of Machine Learning Research*, vol. 18, 2018.

[149] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," *Journal of Optimization Theory and Applications*, vol. 169, pp. 1042–1068, June 2016.

[150] B. O'Donoghue, "Operator splitting for a homogeneous embedding of the linear complementarity problem," *SIAM Journal on Optimization*, vol. 31, pp. 1999–2023, August 2021.

[151] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," vol. 3, p. 1–122, jan 2011.

[152] Y. Nakatsukasa and J. Tropp, "Fast and accurate randomized algorithms for linear systems and eigenvalue problems," 10 2021.

[153] T. Sarlos, "Improved approximation algorithms for large matrices via random projections," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pp. 143–152, 2006.

[154] P.-G. Martinsson and J. A. Tropp, "Randomized numerical linear algebra: Foundations and algorithms," *Acta Numerica*, vol. 29, p. 403–572, 2020.

[155] O. Balabanov and A. Nouy, "Randomized linear algebra for model reduction. part i: Galerkin methods and error estimation," *Adv. Comput. Math.*, vol. 45, p. 2969–3019, dec 2019.

[156] O. Balabanov and L. Grigori, "Randomized gram–schmidt process with application to gmres," *SIAM Journal on Scientific Computing*, vol. 44, no. 3, pp. A1450–A1474, 2022.

[157] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[158] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[159] A. Rubinsteyn and S. Feldman, "fancyimpute: An imputation library for python."

[160] X. Feng, W. Yu, and Y. Li, "Faster matrix completion using randomized svd," in *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 608–615, 2018.

[161] A. W. v. d. Vaart, *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press, 1998.

[162] P. Szynkiewicz, "A comparative study of pso and cma-es algorithms on black-box optimization benchmarks," *Journal of Telecommunications and Information Technology*, vol. 8, 01 2019.

[163] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, pp. 133–151, 2001.

[164] S. Ma, D. Goldfarb, and L. Chen, "Fixed point and bregman iterative methods for matrix rank minimization," *Mathematical Programming*, vol. 128, 05 2009.

[165] P. Chen and D. Suter, "Recovering the missing components in a large noisy low-rank matrix: application to sfm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1051–1063, 2004.

[166] Y. Hu, D. Zhang, J. Ye, X. Li, and X. He, "Fast and accurate matrix completion via truncated nuclear norm regularization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 9, pp. 2117–2130, 2013.

[167] H. Cai, J.-F. Cai, and J. You, "Structured gradient descent for fast robust low-rank hankel matrix completion," 2022.

[168] "Fast and provable algorithms for spectrally sparse signal reconstruction via low-rank hankel matrix completion," *Applied and Computational Harmonic Analysis*, vol. 46, no. 1, pp. 94–121, 2019.

[169] E. Chi, H. Zhou, G. Chen, D. Vecchyo, and K. Lange, "Genotype imputation via matrix completion," *Genome research*, vol. 23, pp. 509–518, 03 2013.

[170] T. Cai, T. Cai, and A. Zhang, "Structured matrix completion with applications to genomic data integration," *Journal of the American Statistical Association*, vol. 111, 04 2015.

[171] J. Bennett, C. Elkan, B. Liu, P. Smyth, and D. Tikk, "Kdd cup and workshop 2007," vol. 9, p. 51–52, dec 2007.

[172] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, p. 61–70, dec 1992.

[173] F. Gao, K. Musial, C. Cooper, and S. Tsoka, "Link prediction methods and their accuracy for different social networks and network metrics," *Scientific Programming*, vol. 2015, 06 2015.

[174] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," vol. 5, dec 2015.

[175] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, "Matrix completion on graphs," 08 2014.

[176] H. Yildirim and M. S. Krishnamoorthy, "A random walk method for alleviating the sparsity problem in collaborative filtering," in *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, (New York, NY, USA), p. 131–138, Association for Computing Machinery, 2008.

[177] T.-H. Oh, Y. Matsushita, Y.-W. Tai, and I. S. Kweon, "Fast randomized singular value thresholding for low-rank optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 2, pp. 376–391, 2018.

[178] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. Efros, "Context encoders: Feature learning by inpainting," pp. 2536–2544, 06 2016.

[179] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 3, 06 2014.

[180] C. Yang, X. Lu, Z. L. Lin, E. Shechtman, O. Wang, and H. Li, "High-resolution image inpainting using multi-scale neural patch synthesis," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4076–4084, 2016.

[181] J. Yu, Z. L. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Free-form image inpainting with gated convolution," *2019 IEEE CVF International Conference on Computer Vision (ICCV)*, pp. 4470–4479, 2018.

[182] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi, "Edgeconnect: Generative image inpainting with adversarial edge learning," 10 2019.

[183] X. Hong, P. Xiong, R. Ji, and H. Fan, "Deep fusion network for image completion," in *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, (New York, NY, USA), p. 2033–2042, Association for Computing Machinery, 2019.

[184] N. Kitson, A. Constantinou, G. Zhigao, Y. Liu, and K. Chobtham, "A survey of bayesian network structure learning," *Artificial Intelligence Review*, pp. 1–94, 01 2023.

[185] T. Koski and J. M. Noble, "A review of bayesian networks and structure learning," *Mathematica Applicanda*, vol. 40, pp. 51–103, 2012.

[186] J. Dai, J. Ren, and W. Du, "Decomposition-based bayesian network structure learning algorithm using local topology information," *Knowledge-Based Systems*, vol. 195, p. 105602, 2020.

[187] F. Gao and D. Huang, "A node sorting method for k2 algorithm in bayesian network structure learning," pp. 106–110, 06 2020.

[188] A. Krajewska, "Learning causal theories with non-reversible mcmc methods," *Control and Cybernetics*, vol. 50, pp. 323–361, 09 2021.

[189] "A probabilistic model of theory formation," *Cognition*, vol. 114, no. 2, pp. 165–196, 2010.

[190] V. K. Mansinghka, C. Kemp, J. B. Tenenbaum, and T. L. Griffiths, "Structured priors for structure learning," in *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'06, (Arlington, Virginia, USA), p. 324–331, AUAI Press, 2006.

[191] I. Ahmad, M. Akhtar, S. Noor, and A. Shahnaz, "Missing link prediction using common neighbor and centrality based parameterized algorithm," *Scientific Reports*, vol. 10, p. 364, 01 2020.

[192] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, (New York, NY, USA), p. 556–559, Association for Computing Machinery, 2003.

[193] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 71, pp. 623–630, 10 2009.

[194] J. Kunegis, "Konect: the koblenz network collection," pp. 1343–1350, 05 2013.

[195] Z. Zhao, Z. Gou, Y. Du, J. Ma, T. Li, and R. Zhang, "A novel link prediction algorithm based on inductive matrix completion," *Expert Systems with Applications*, vol. 188, p. 116033, 2022.

[196] A. Clauset, C. Moore, and M. Newman, "Hierarchical structure and the prediction of missing links in networks," *Nature*, vol. 453, pp. 98–101, 06 2008.

[197] Q. Yao and J. T. Kwok, "Accelerated and inexact soft-impute for large-scale matrix and tensor completion," *IEEE Transactions on Knowledge; Data Engineering*, vol. 31, pp. 1665–1679, sep 2019.