Ph.D. Thesis:

Matrix completion with prior information in reproducing kernel Hilbert spaces

Ph.D. Thesis:

Matrix completion with prior information in reproducing kernel Hilbert spaces

Author:
Pere Joan Giménez Febrer
p.gimenez@upc.edu

Advisor:
Alba Pagès-Zamora
alba.pages@upc.edu

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Departament de Teoria del Senyal i Comunicacions (TSC)
Carrer Jordi Girona 1-3, Campus Nord, Edifici D5
08034 Barcelona, Spain

Barcelona, November 2020

Summary

In matrix completion, the objective is to recover an unknown matrix from a small subset of observed entries. Most successful methods for recovering the unknown entries are based on the assumption that the unknown full matrix has low rank. By having low rank, each of its entries are obtained as a function of a small number of coefficients which can be accurately estimated provided that there are enough available observations. Hence, in low-rank matrix completion the estimate is given by the matrix of minimum rank that fits the observed entries.

Besides low rankness, the unknown matrix might exhibit other structural properties which can be leveraged in the recovery process. In a smooth matrix, it can be expected that entries that are close in index distance will have similar values. Similarly, groups of rows or columns can be known to contain similarly valued entries according to certain relational structures. This relational information is conveyed through different means such as covariance matrices or graphs, with the inconvenient that these cannot be derived from the data matrix itself since it is incomplete. Hence, any knowledge on how the matrix entries are related among them must be derived from prior information. As an example, consider a sensor network taking periodical ambient measurements and storing them into a matrix; the closer two sensors are the more likely it is that they will take similar measurements. Hence, knowing the sensor positions allows to obtain relational information without the need to observe any data. As another example, with a matrix recording the preferences of a group of people on a given topic, e.g., politics or entertainment, knowing details about each person such as age or income can help group people by their preferences. Again, this is solely done using prior information and zero knowledge on the contents of the matrix. By incorporating such prior information into the matrix completion problem, the missing entries can be extrapolated

from those with expected similar value and the recovery error can be reduced.

This thesis deals with matrix completion with prior information, and presents an outlook that generalizes to many situations. In the first part, the columns of the unknown matrix are cast as graph signals with a graph known beforehand. In this, the adjacency matrix of the graph is used to calculate an initial point for a proximal gradient algorithm in order to reduce the iterations needed to converge to a solution. Then, under the assumption that the graph signals are smooth, the graph Laplacian is incorporated into the problem formulation with the aim to enforce smoothness on the solution. This results in an effective denoising of the observed matrix and reduced error, which is shown through theoretical analysis of the proximal gradient coupled with Laplacian regularization, and numerical tests.

The second part of the thesis introduces a framework to exploit prior information through reproducing kernel Hilbert spaces. Since a kernel measures similarity between two points in an input set, it enables the encoding of any prior information such as feature vectors, dictionaries or connectivity on a graph. By associating each column and row of the unknown matrix with an item in a set, and defining a pair of kernels measuring similarity between columns or rows, the missing entries can be extrapolated by means of the kernel functions. A method based on kernel regression is presented, with two additional variants aimed at reducing computational cost, and online implementation. These methods prove to be competitive with existing techniques, especially when the number of observations is very small.

Furthermore, mean-square error and generalization error analyses are carried out, shedding light on the factors impacting algorithm performance. For the generalization error analysis, the focus is on the transductive case, which measures the ability of an algorithm to transfer knowledge from a set of labelled inputs to an unlabelled set. Here, bounds are derived for the proposed and existing algorithms by means of the transductive Rademacher complexity, and numerical tests confirming the theoretical findings are presented.

Finally, the thesis explores the question of how to choose the observed entries of a matrix in order to minimize the recovery error of the full matrix. A passive sampling approach is presented, which entails that no labelled inputs are needed to design the sampling distribution; only the input set and kernel functions are required. The approach is based on building the best Nyström approximation to the kernel matrix by sampling the columns according to their leverage scores, a metric that arises naturally in the theoretical analysis to find an optimal sampling distribution.

Acknowledgments

First and foremost, my deepest gratitude to my advisor, Prof. Alba Pagès-Zamora. Long time has passed since that first meeting about a master's thesis, which ended up sparking my interest in a research career. During these years, Alba has been supportive, kind, very patient, and dedicated. Without her guidance and encouragement, this thesis would have never been possible.

I would also like to thank Prof. Georgios B. Giannakis from University of Minnesota, who welcomed me into his group as one more student. I learned a lot during my time there, and much of it is in this thesis today. I want to thank all the people in the group as well for making me feel at home, with special mention to my dudes Panos and Donghoon.

I also want to extend my gratitude to Prof. Ignacio Santamaría from Universidad de Cantabria, where my recent work as a researcher has developed into fruitful collaborations. Further thanks go to thank Prof. Antonio G. Marqués, Prof. Javier Rodriguez Fonollosa, and Prof. Daniel Romero for agreeing to be part of the thesis committee. Moreover, special thanks to Antonio and Daniel for their time spent reviewing this manuscript and their insightful comments.

I would like to thank my fellow countrymen, colleagues, and friends, Jaume and Miquel with whom I have spent many joyful moments since we met at the university in Mallorca. The road leading to this day has been a long one, and I am happy to have shared it with you.

A n'Alicia, m'has donat amor i suport incondicional, i m'has ensenyat a disfrutar de la vida. Gràcies de tot cor per sempre estar al meu costat. Finalment, agraïr-li als meus pares, Francisco i Magdalena, tot el que han fet per mi. Res d'això ni del que vindrà hagués estat possible sense el seu suport i esforç. Gràcies per donar-me s'oportunitat de tenir una vida millor, vos promet que l'aprofitaré al màxim.

Contents

Notation

Vectors and matrices

$\boldsymbol{x},\ \boldsymbol{X}$	a column vector and a matrix
$\boldsymbol{x}(i)$	the ith entry of $\boldsymbol{x}$
$\boldsymbol{X}_{i,j}$	the entry at the ith row and jth column of $\boldsymbol{X}$
$\boldsymbol{x}^T, \boldsymbol{X}^T$	the transpose of $\boldsymbol{x}$ and $\boldsymbol{X}^T$
$\|\boldsymbol{X}\|_*$	nuclear norm of $\boldsymbol{X}$
$\|\boldsymbol{X}\|_F$	Frobenius norm of $\boldsymbol{X}$
$\mathrm{Tr}(\boldsymbol{X})$	the trace of $\boldsymbol{X}$
$\mathrm{vec}(\boldsymbol{X})$	columnwise vectorization of $\boldsymbol{X}$
$\mathrm{diag}(\boldsymbol{x})$	a diagonal matrix whose entries are the elements of $\boldsymbol{X}$
$\boldsymbol{X} \succeq 0$	$\boldsymbol{X}$ is positive semidefinite
$\boldsymbol{X} \succeq \boldsymbol{Y}$	$\boldsymbol{X} - \boldsymbol{Y}$ is positive semidefinite
$\boldsymbol{X} \otimes \boldsymbol{Y}$	Kronecker product between $\boldsymbol{X}$ and $\boldsymbol{Y}$
$\lambda_i(\boldsymbol{X})$	ith eigenvalue of $\boldsymbol{X}$ in non-descending order

Sets, functions and spaces

$\mathcal{X}$	finite nonempty set of elements
$x \in \mathcal{X}$	x belongs to $\mathcal{X}$
$\{x_n\}_{n=1}^N \subseteq \mathcal{X}$	set of N elements where each belongs to $\mathcal{X}$
$\lvert\mathcal{X}\rvert$	cardinality of $\mathcal{X}$
$\mathbb{R}$	set of real numbers
$\mathcal{H}$	reproducing kernel Hilbert space
$f \in \mathcal{H}$	function f belongs to $\mathcal{H}$
$\text{span}\{f, g\}$	space spanned by the functions f, g
$\text{span}(\boldsymbol{X})$	space spanned by the columns of $\boldsymbol{X}$
κ	kernel function

Acronyms

MC	matrix completion
PG	proximal gradient
RKHS	reproducing kernel Hilbert space
RR	ridge regression
KRR	kernel ridge regression
ALS	alternating least-squares
SGD	stochastic gradient descent
GE	generalization error
RC	Rademacher complexity
iid	independent and identically distributed

1

Introduction

The task of recovering a signal from a few measurements is ubiquitous across disciplines such as signal processing, statistics or image processing. A classical example is Shannon-Nyquist's sampling theorem, which states that a bandlimited continuous time signal can always be reconstructed from a set of sampling points provided that the sampling rate is sufficiently high. Other application examples are the extrapolation of missing measurements in a sensor network [1], or the recovery of a complete image from a subset of pixels [2]. The underlying principle is usually the same: we are given a set of observations each of which is associated with a given input element, e.g., time or location, and we wish to predict the observations at other previously unseen inputs. In today's jargon, this falls under the wide umbrella of machine learning [3, 4]. This encompassing paradigm aims at obtaining a function which is learned through a training method run on the known input-output pairs, such that the function will be able to produce an accurate output for any input quantity within a desired set of inputs. Depending on the availability of data and prediction target, there exist different machine learning training methodologies: supervised, semisupervised and unsupervised. The difference between the approaches lies in whether they use labelled data, i.e., inputs with an associated output measurement, or unlabelled data, i.e., inputs without an associated measurement, to learn the prediction function. In supervised learning, only labelled data are used, whereas semisupervised learning uses a mix of labelled and unlabelled, and in unsupervised learning only the inputs and no labels are known. Thus, the signal reconstruction problem subscribes to the supervised or semisupervised class.

As hinted by Shannon-Nyquist's theorem, there are requirements to a successful and

accurate signal recovery. While a signal might be dense in a domain, it might contain redundancies that become apparent when the signal is mapped to an alternative domain where it appears as sparse or constrained. Notably, this trait is the foundation for the compressed sensing theory. In compressed sensing [5], we are given a small vector of observations with the intent to recover the possibly much larger unknown vector from which the observations were taken. For this to be feasible, it is necessary that the original full vector can be mapped to a space where it is be represented as a linear combination of a very small number of the vectors spanning mapped the space. One example of such a signal meeting this requirement is a time signal whose Fourier transform only takes a non-zero value at a reduced number of frequencies. Then, the problem is solved by finding the frequency coefficients via a regression problem regularized with the $l1$ norm, which is known to promote sparsity.

The notion of sparsity in a transformed domain is also useful when dealing with data available as a matrix. There exist a plethora of applications in which there are missing entries in the data matrix and there is use in recovering them, such as in positioning [6], gene-disease association prediction [7], large-scale network monitoring [8] or medical resonance imaging [9]. Another prevalent example is that of recommender systems [10], where the rows may be assigned to items, the columns to users, and each entry denotes the rating given by a user to an item. Since it is rather unlikely that a user will try and rate every possible item in the database, this type of matrix tends to be very sparse. Hence, if we were to recommend an item to a user, one possible approach is to fill the missing ratings in the matrix with predictions according to the available ratings. This would give us an estimate of the full matrix of item-user ratings, hence having performed an act of matrix completion (MC). There are many approaches to the completion of matrices [11,12], with the one developed in [13] by Candès and Retch having gained the most traction in the recent years. Like most methods, the low-rank MC technique in [13] is centered around the idea of having a low-rank unknown matrix. Thus, as in compressed sensing, the sparsity in the transformed domain boils down to the fact that a low-rank matrix has most of its singular values equal to zero; this implies that it can be constructed from just a few of its singular vectors. Therefore, [13] formulates MC as a convex minimization of the nuclear norm of the partially observed matrix, and demonstrates that an exact recovery of the original matrix is possible whenever the matrix is low rank, incoherent, i.e., the information is evenly distributed across the matrix, and uniformly sampled. An alternative equivalent formulation [10] factorizes the unknown matrix as the product of two low-rank matrices, and replaces the nuclear norm with the sum of the Frobenius norm of the two factor matrices.

While MC has good recovery guarantees for incoherent matrices, it is rare that data are organized with lack of structure. For instance, in the user-item example, some users might

have similar tastes and, likewise, some items might have similar characteristics. Therefore, in this case it would be reasonable to assume an underlying structure such that similar users or items give or receive similar ratings. Moreover, other possible scenarios are previous knowledge about the vector space spanning the matrix columns or vectors, or a known trait such as smoothness. There exist different strategies to incorporate additional information into standard MC [14–16], with the most common being the addition of an ad-hoc regularization term which will enforce the desired structure [17–19]. Additionally, it can be assumed that the columns or rows of the unknown matrix belong to a specific space shaped by prior information. For instance, considering the vectors as signals on a graph, the adjacency and Laplacian matrices of the graph provide relational information that can be used to fill in the missing entries [20, 21].

A more general approach to encode and leverage prior information is via the use of kernel functions and their associated reproducing kernel Hilbert spaces (RKHSs) [22–24]. RKHS are spaces spanned by a kernel function which measures similarity between input points. There exist many kernels, some of which are the linear, Gaussian or polynomial [25]. Moreover, new kernels can be derived provided that they are linear and symmetric positive definite functions. Hence, casting the rows and columns of the unknown matrix as functions in a RKHS, one can encode prior information through kernels and leverage the tools in the RKHS framework.

This thesis explores MC with prior information available in the form of graphs and kernels. From this perspective, it proposes efficient algorithms based on kernel regression which are easily extended to online operation. Moreover, theoretical analysis is conducted on the mean-square error and generalization error, with the latter being a useful metric to assess the ability to predict inputs outside the training set. Finally, noticing that prior information encoded by kernels helps determine which inputs are more important to achieve a low estimation error, optimal kernel-based sampling strategies are presented. On the whole, together with novel propositions, this thesis offers a comprehensive overview on kernel-based methods for MC which serves as a template for future development of new techniques.

1.0.1 Thesis outline and related publications

Including this introduction, this thesis is comprised of six chapters and two appendices. The contents of each chapter and the produced publications are summarized below.

Chapter 2

This chapter introduces the fundamentals of MC and its formulation as the convex minimization of the nuclear norm of the matrix. Algorithms based on proximal gradient (PG) minimization are commonly used in MC due to their simplicity. However, one of their drawbacks is a slow convergence speed and high computational cost per iteration due to the need to perform a singular value decomposition. Two PG-based algorithms are proposed which rely on a variable regularization parameter in order to increase the convergence speed. Moreover, the chapter explores the availability of prior information about the missing entries encoded as a graph mapped to the matrix rows; this prompts a model to viewing the columns of the matrix as graph signals. Hence, an initialization method is introduced that uses the information provided by the graph to reduce the iterations required by the PG-based algorithms to converge. Moreover, the second part of the chapter takes on the recovery of matrices of graph signals contaminated by noise, and presents a theoretical analysis of the performance of the standard PG algorithm with an additional graph-based regularization term.

Publications:

- P. Giménez-Febrer, A. Pagès-Zamora. "Matrix completion of noisy graph signals via proximal gradient minimization", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), March 2017.

Chapter 3

This chapter extends the concept of MC with prior information by introducing the field of reproducing kernel Hilbert spaces as a tool to encode and leverage said information. Aiming at a fast and low-complexity solver, the task is formulated as one of kernel ridge regression with a kernel matrix obtained as the Kronecker product of two smaller kernel matrices. The resulting MC algorithm can also afford online implementation, while the class of kernel functions also encompasses several existing approaches to MC with prior information. Numerical tests on synthetic and real datasets show that the novel approach is faster than widespread methods such as alternating least-squares (ALS) or stochastic gradient descent (SGD), and that the recovery error is reduced, especially when dealing with noisy data. Moreover, theoretical analysis on the mean-square error is presented to give insight into the performance of MC based on kernel regression.

Publications:

- P. Giménez-Febrer, A. Pagès-Zamora and G. B. Giannakis, "Matrix completion and extrapolation via kernel regression", IEEE Transactions on Signal Processing, vol. 67, no. 19, pp. 5004-5017, 1 Oct.1, 2019

Chapter 4

Given the availability of a small percentage of samples, MC algorithms must be able to predict the unseen entries with low error. However, this error can only be measured on the available entries. Attempting an excessive error minimization on the observed entries can lead to overfitting, which entails that the solution is too adjusted to the observations and it is inaccurate elsewhere. In this chapter, the standard and kernel-based MC algorithms are examined in terms of their generalization error. This analysis sheds light on the ability of each algorithm to transfer knowledge from the observed entries onto the unobserved ones, and the influential factors involved.

Publications:

- P. Giménez-Febrer, A. Pagès-Zamora and G. B. Giannakis, "Generalization error bounds for kernel matrix completion and extrapolation", IEEE Signal Processing Letters, January 2020

Chapter 5

This chapter deals with the selection of the training dataset in kernel-based methods. A statistical passive sampling approach is derived which, through minimizing the error between the complete kernel matrix and its Nyström approximation, chooses the set of samples that minimize an upper bound to the mean square error. Furthermore, the analysis and examples are focused the reconstruction of sparsely sampled functions lying in reproducing kernel Hilbert spaces whose kernel matrices adhere to the Kronecker product structure.

Others

Besides the listed publications, during the thesis other research lines have been explored which are not included in this manuscript. Related to MC, the following two publications have been elaborated within the field of array processing:

- V. Garg, P. Giménez-Febrer, A. Pagès-Zamora, I. Santamaría, “Source Enumeration via Toeplitz Matrix Completion”, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2020.

- V. Garg, P. Giménez-Febrer, A. Pagès-Zamora, I. Santamaría, "Energy-Efficient DOA Estimation via Shift Invariance Matrix Completion”, submitted to EURASIP Journal on Advances in Signal Processing, August 2020.

Unrelated to MC but serving as a first contact with graph theory that led to this thesis, the following publications address the topic of distributed estimation in sensor networks with faulty nodes:

- P. Giménez-Febrer, A. Pagès-Zamora. “Matrix completion of noisy graph signals via proximal gradient minimization”, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), March 2017.

- P. Giménez-Febrer, A. Pagès-Zamora, R. López-Valcarce. “Online EM-based distributed estimation in sensor networks with faulty nodes” European Signal Processing Conference (EUSIPCO), August 2016.

2

Matrix completion on graphs

Applications in fields such as sensor networks, data mining and video processing generate large amounts of redundant data that can be compressed or even partially discarded without compromising the performance. Thus, in recent years an increasing attention is being paid to the matrix completion problem, i.e. the problem of recovering a data matrix given a small fraction of its entries. Similar to compressed sensing, in the matrix completion (MC) approach the redundancy reduction is applied at the data collection step, in which the original matrix is sparsely sampled and the recovery of the unknown entries is performed at a later step using these sampled entries.

In [13], Candès and Retch formulated the matrix completion problem as a convex minimization of the nuclear norm of the partially observed matrix, and demonstrated that an exact recovery of the original matrix is possible whenever the matrix is low rank, incoherent and uniformly sampled. MC was solved in [13] using semidefinite programming solvers, which are not suitable for the recovery of large matrices [26] due to high computational costs. Since then, many algorithms based on first order methods such as proximal gradient (PG) minimization or Bregman iterations have been proposed thanks to their simplicity and ease of implementation. Two notable examples are the fixed point iterative (FPI) algorithm in [27], which performs a PG minimization, and the singular value thresholding (SVT) algorithm in [26], which is based on linearized Bregman iterations. While the FPI algorithm minimizes a convex relaxation of the problem in [13], and SVT minimizes the dual problem, both algorithms iteratively use the proximal operator [28] of the nuclear norm to perform a gradient descent minimization. This proximal operator shrinks the eigenvalues of the

matrix, which requires an eigendecomposition that can be computationally intensive for large matrices. Moreover, gradient descent minimization is known to have a slow convergence speed. Therefore, algorithms based on gradient descent often seek to increase the convergence speed and reduce the computational cost. For instance, the fixed point continuation (FPC) algorithm in [27] uses a warm-start technique on the regularization parameter of the nuclear norm, whereas the accelerated PG algorithm in [29] extends the FPC algorithm with an additional interpolation step in order to further speed up the convergence. On the other hand, the iterative partial matrix shrinkage algorithm in [30] reduces the cost of the proximal operator by only shrinking the non-dominant singular values of the matrix at each iteration.

The assumptions of incoherence and uniform sampling imply that the matrix entries are unstructured, which is usually not the case for real data. Therefore, some works have extended the work in [13] by including extra information about hidden matrix structures into the problem formulation. For instance, in [14] an additional restriction is imposed so that the columns of the recovered matrix are a linear combination of the basis elements in a dictionary. In [18], it is observed that the temperature measurements taken by the sensors in a wireless sensor network are temporally stable in the short term, so a regularization term is added to ensure the short term stability of the recovered data. In the problem of predicting an incomplete matrix of ratings in [19], groups of users with similar background are assumed to have similar preferences, thus a penalty term is included to reduce the variability of the predicted ratings within a group. A different approach to include extra information is adopted in [15], where a fraction of the matrix entries are sampled according to their local coherence, which is deemed to be indicative of their relevance to the posterior matrix recovery.

The field of signal processing on graphs provides a framework to model the interdependence between the entries in the matrix, and leverage this information. As described in [31, 32], this novel field extends classical signal processing tools such as filtering or domain transformations and applies them to signals on a graph, so that the vertices in the graph measure signals and the graph edges model the underlying relational structure of those signals. The connections between the vertices are represented by the weighted adjacency matrix of the graph, which can be known beforehand or inferred from the data as, for instance, in [33]. With regards to the matrix completion problem, so far the existing works under the signal processing on graphs perspective take advantage of the fact that the graph signals are known to be smooth on a given graph. For instance, in [21] the signals are assumed to be smooth on a graph when the data from connected vertices have similar values. Hence, the p-Dirichlet norm is added as a regularization function to the problem in [13] to enforce this smoothness on the recovered data. Additionally, a function named total variation is used in [20] as a regularization term to enforce the graph structure described by the adjacency matrix. Although not linked to

the signal processing on graphs framework, there are other works on matrix completion that rely on graph theory concepts, such as [34], where the sampling of the complete matrix is modeled as a bipartite graph in order to derive theoretical recovery guarantees related to the graph Laplacian.

This chapter focuses on the recovery of partially observed graph signals that are arranged in a data matrix; the content is organized as follows. Section 2.1 introduces the MC formulation and theoretical foundation. Section 2.2 presents the proximal gradient algorithm and introduces two variants with reduced convergence time and, therefore, less computational load. Section 2.3, using the extra information provided by the adjacency matrix of the graph, presents an initial approximation that proves to further speed up the convergence of the studied PG-based algorithms. Finally Section 2.4 analyzes the performance of MC when dealing with noisy graph signals and how the Laplacian matrix of the graph improves the recovery result.

2.1 Matrix completion

Let $\boldsymbol{F} \in \mathbb{R}^{N \times L}$ be an unknown matrix which can only be observed through a subset of its entries, as depicted in Fig. 2.1. With $\Omega \subseteq \{1, \dots, N\} \times \{1, \dots, L\}$ denoting the set of indices of the observed entries, and $P_\Omega(\cdot)$ being the projection operator such that

$$P_\Omega(\boldsymbol{F})_{i,j} = \begin{cases} \boldsymbol{F}_{i,j}, & \text{when } (i,j) \in \Omega, \\ 0, & \text{when } (i,j) \notin \Omega. \end{cases} \tag{2.1}$$

The MC problem aims at recovering $\boldsymbol{F}$ from $\boldsymbol{M} = P_\Omega(\boldsymbol{F})$, defined as the observed matrix in absence of noise. Denoting $\text{rank}(\boldsymbol{F}) = r$, and given the singular value decomposition (SVD) $\boldsymbol{F} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, where $\boldsymbol{U} = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_N]$, $\boldsymbol{V} = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_L]$ and $\boldsymbol{\Sigma}$ is rectangular diagonal with $\boldsymbol{\Sigma}_{ii} = \sigma_i \; \forall i \leq r$ and zero elsewhere, we write

$$\boldsymbol{F} = \sum_{n=1}^{r} \sigma_n \boldsymbol{u}_n \boldsymbol{v}_n^T \tag{2.2}$$

and its entries as $\boldsymbol{F}_{i,j} = \sum_{n=1}^{r} \sigma_n \boldsymbol{u}_n(i) \boldsymbol{v}_n^T(j)$. This decomposition evidences that recovering $\boldsymbol{F}$ amounts to obtaining its SVD. Indeed, the SVD has $r(N+L+1)$ variables, such that one would require enough observations to reconstruct the singular values and vectors while satisfying the orthonormality constraints. Hence, since the SVD must satisfy $r(r+1)$ orthonormality constraints, the degrees of freedom (DOF) of a matrix are $r(N+L-r)$ [13].

Since in MC applications the number of observations is usually very low, it is unlikely that

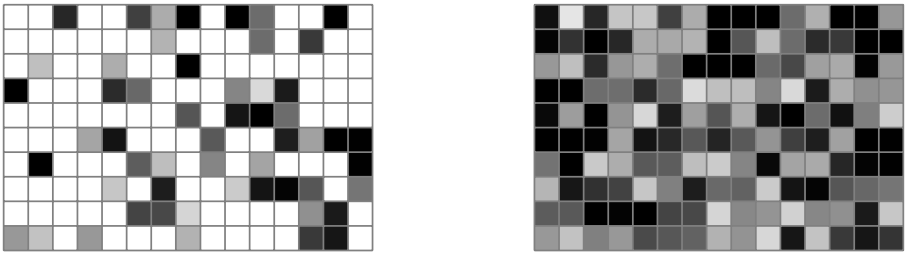


Figure 2.1: Observed matrix (left) and complete matrix (right). Squares in white represent unobserved entries.

the SVD can be recovered since there may exist an infinite number of possible SVDs solving the undetermined equation system $P_\Omega(\boldsymbol{F})_{i,j} = \sum_{n=1}^{r} \sigma_n \boldsymbol{u}_n(i)\boldsymbol{v}_n^T(j) \; \forall \, (i,j) \in \Omega$. Hence, one needs to establish a prior on $\boldsymbol{F}$ so that the DOF are reduced. In low-rank matrix completion this prior amounts to assuming that $r \ll \min\{N, L\}$, where hereafter we will assume that $L > N$ without loss of generality. Then, the recovery problem is formulated as

$$\begin{aligned} \hat{\boldsymbol{F}} = \underset{\boldsymbol{F} \in \mathbb{R}^{N \times L}}{\arg\min} \quad & \operatorname{rank}(\boldsymbol{F}) \\ \text{s.t.} \quad & \boldsymbol{M}_{ij} = P_\Omega(\boldsymbol{F})_{ij} \text{ for } (i,j) \in \Omega. \end{aligned} \tag{2.3}$$

The optimization problem (2.3) amounts to finding the matrix with the lowest rank that fits the already observed entries indexed by Ω. Due to its combinatorial nature, solving (2.3) is NP-hard, meaning that it cannot be solved in polynomial time. Nevertheless, there exist theoretical guarantees. In order for (2.3) to have a unique solution, the map $P_\Omega(\boldsymbol{F})$ must be injective on the solution space to (2.3) of matrices with limited rank; this is accomplished with high probability when the number of observations is $|\Omega| \geq 4Lr - 4r^2$ and $r \leq L/2$ [35]. To overcome issues with (2.3), the approach taken in [36] is to replace the rank function with the nuclear norm as a convex surrogate. The ensuing problem is

$$\begin{aligned} \hat{\boldsymbol{F}} = \underset{\boldsymbol{F} \in \mathbb{R}^{N \times L}}{\arg\min} \quad & \|\boldsymbol{F}\|_* \\ \text{s.t.} \quad & \boldsymbol{M}_{ij} = P_\Omega(\boldsymbol{F})_{ij} \text{ for } (i,j) \in \Omega, \end{aligned} \tag{2.4}$$

where $\|\boldsymbol{F}\|_* = \operatorname{Tr}(\sqrt{\boldsymbol{F}^T\boldsymbol{F}})$ is the sum of the singular values of $\boldsymbol{F}$. This approach stems from the least-squares problem coupled with $l1$ regularization used in compressed sensing [5]. Like the $l1$ norm, the nuclear norm promotes sparsity in the singular values of $\hat{\boldsymbol{F}}$. Indeed, the nuclear norm is equivalent to the $l1$ norm applied to the singular values, i.e., $\|\boldsymbol{F}\|_* = \|\boldsymbol{\Sigma}\mathbf{1}\|_1$. Therefore the nuclear norm minimization in (2.3) causes the smaller singular values to approach or become zero, hence resulting in a low-rank solution.

To illustrate the sparsifying effect of the nuclear norm, let us consider the $l1$-regularized least-squares problem

$$\arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{B}\boldsymbol{x}\|_2^2 + \mu_1 \|\boldsymbol{x}\|_1 \tag{2.5}$$

and compare it to its $l2$-regularized counterpart

$$\arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{B}\boldsymbol{x}\|_2^2 + \mu_2 \|\boldsymbol{x}\|_2^2 \tag{2.6}$$

where $\mu_1, \mu_2 \geq 0$. In MC terms, this would be a comparison between the nuclear and Frobenius norms as regularizers. Due to its nonlinearity, the $l2$ norm penalizes the largest terms more, whereas the $l1$ penalizes all entries in the vector equally. Consequently, the $l1$ norm is more prone to induce sparse solutions. The two problems (2.5) and (2.6) are equivalently written [37] as

$$\begin{aligned} \arg\min_{\boldsymbol{x}} \quad & \|\boldsymbol{y} - \boldsymbol{B}\boldsymbol{x}\|_2^2 \\ \text{s.t.} \quad & \|\boldsymbol{x}\|_1 \leq \beta_1 \end{aligned} \tag{2.7}$$

$$\begin{aligned} \arg\min_{\boldsymbol{x}} \quad & \|\boldsymbol{y} - \boldsymbol{B}\boldsymbol{x}\|_2^2 \\ \text{s.t.} \quad & \|\boldsymbol{x}\|_2 \leq \beta_2 \end{aligned} \tag{2.8}$$

for adequate parameters $\beta_1, \beta_2 \geq 0$. Fig. 2.2 shows an example of the solution obtained by both problems for a bidimensional $\boldsymbol{x}$. Due to the shape of the feasible set, the $l1$-constrained problem is more likely to have one of the estimated components be zero.

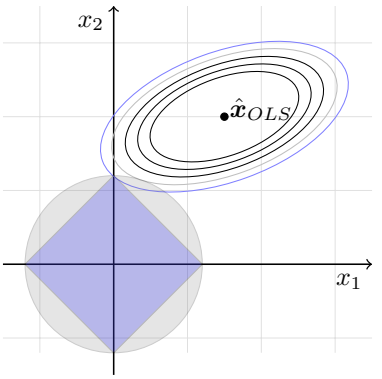


Figure 2.2: Minimization of a function constrained with the $l1$ and $l2$ norms, where $\hat{\boldsymbol{x}}_{OLS}$ denotes the solution obtained through ordinary least-squares regression. The ellipses represent the level curves of the loss function, and the colored square and circle represent the feasible area of the $l1$ and $l2$ constraints, respectively.

It has been proven [13] that the solution to (2.4) matches that of (2.3) when the following conditions are met: $\boldsymbol{F}$ is incoherent, the samples are taken uniformly at random, and the number of samples is large enough. The notion of incoherence points to a matrix with no

hidden underlying structure and information evenly spread across the matrix, such that all entries are equally important for the recovery. More specifically, an incoherent matrix has singular vectors whose entries have similar magnitude instead of being concentrated in a few large coordinates. The coherence [36] is measured on the space spanned by the columns or rows of a matrix and, given a vector space $U \subseteq \mathbb{R}^N$ of dimension r, it is expressed as

$$\tau(U) = \frac{N}{r} \max_{1 \leq i \leq N} \|\boldsymbol{P}_U \boldsymbol{e}_i\|_2 \tag{2.9}$$

where $\boldsymbol{P}_U$ is the orthogonal projection matrix onto U, $\boldsymbol{e}_i$ is the ith vector of the canonical Euclidean basis, and $\max_{1 \leq i \leq N} \|\boldsymbol{P}_U \boldsymbol{e}_i\|_2 \leq 1$. Letting U and V correspond to the column and row spaces of $\boldsymbol{F}$ spanned by the singular vector matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively, it is assumed [36] that there exist constants τ_1 and τ_0 such that

$$\max(\tau(U), \tau(V)) \leq \tau_0 \text{ and} \tag{2.10}$$

$$\|\sum_{i=n}^{r} \boldsymbol{u}_n \boldsymbol{v}_n^T\|_\infty \leq \tau_1. \tag{2.11}$$

With $\boldsymbol{P}_U = \sum_{i=n}^{r} \boldsymbol{u}_n \boldsymbol{u}_n^T$ and $\boldsymbol{P}_V = \sum_{n=1}^{r} \boldsymbol{v}_n \boldsymbol{v}_n^T$, τ_0 bounds the maximum correlation with the standard basis of the rows and columns of $\boldsymbol{F}$. Similarly, τ_1 bounds the maximum correlation between the rows of $\boldsymbol{U}$ and $\boldsymbol{V}$. The lower the coherence, the lower the chance that there exists an entry which must be observed in order to be recovered. Take for instance the $N \times N$ matrix

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & 1 & \dots & 1 \end{bmatrix} \tag{2.12}$$

This matrix has two left singular vectors: $[1, 0, \dots, 0]^T$ and $\frac{1}{\sqrt{N-1}}[0, 1, \dots, 1]$ with associated singular values 1 and $N - 1$, respectively. Having $r = 2$, the matrix attains a maximum coherence value (2.9) of $N/2$. At its maximum value, the coherence indicates that one singular vector is equal to a vector in the standard basis. Hence, maximum coherence means that there are entries which must be observed; otherwise they cannot be recovered. This is the case for the entry at $(1, 1)$ in the example matrix, since not observing it would result in an estimated all-zeros row or column. On the other hand, the second singular vector has more

evenly spread values. Hence, reconstructing the second block of ones in (2.12) only requires a few observations per row.

As previously stated, the choice of observation set plays an important role in MC. If the matrix is highly coherent, some entries will need to be observed in order to be recovered accurately. On the other hand, a matrix with low coherence allows for a less structured sampling scheme. Hence, in the MC theory it is usually required that the matrix be incoherent so that the observations can be taken uniformly at random. This is an ideal sampling distribution since it requires no prior knowledge about the matrix and can be easily implemented. When τ_0 is small and the observations are taken uniformly at random, $|\Omega| \geq CL^{\frac{6}{5}} r \log L$ for a constant C observations are required for exact recovery with high probability [13].

While an exact recovery is possible with (2.4), observations are often contaminated with noise that will induce an estimation error due to the equality constraint. For instance, in the movie-user scenario each rating might have an associated indecisiveness factor which makes the rating not reflect the actual opinion of the user. Let us update the model for the observed matrix to

$$\boldsymbol{M} = P_\Omega(\boldsymbol{F}) + \boldsymbol{E} \tag{2.13}$$

where $\boldsymbol{E}$ is a noise matrix with zeros at the entries $(i,j) \notin \Omega$. Given this model, allowing for some change in the observed entries will prevent overfitting to the noise and reduce its impact. Thus, (2.4) can be modified [36] to

$$\begin{aligned} \hat{\boldsymbol{F}} = \underset{\boldsymbol{F} \in \mathbb{R}^{N \times L}}{\arg\min} \quad & \|\boldsymbol{F}\|_* \\ \text{s.t.} \quad & \|\boldsymbol{M} - P_\Omega(\boldsymbol{F})\|_F \leq \theta \text{ for } (i,j) \in \Omega. \end{aligned} \tag{2.14}$$

The switch to inequality constraints controls the maximum change in the observed entries, which will percolate to the estimates of the unobserved entries and can result in an overall reduced estimation error. Equations (2.4) and (2.14) can be cast as semidefinite programming (SDP) problems and solved with SDP solvers such as SeDuMi and SDPT3 [27]. Still, these solvers become very slow for matrices with $N, L \gg 100$ [27, 38].

There exist alternative formulations to (2.4) and (2.14) which focus on reducing the computational cost while maintaining accuracy [39]. Instead of directly solving the convex optimization problem, these approaches solve its Lagrangian version

$$\underset{\boldsymbol{F} \in \mathbb{R}^{N \times L}}{\arg\min} \frac{1}{2} \|\boldsymbol{M} - P_\Omega(\boldsymbol{F})\|_{\mathrm{F}}^2 + \mu \|\boldsymbol{F}\|_* \tag{2.15}$$

Given a correct adjustment for parameters θ in (2.14) and μ in (2.15), both problems can be made equivalent [38]. While (2.4) and (2.15) are not equivalent, the solution to (2.15) serves as an approximation to that of (2.4) when $\boldsymbol{E} = \boldsymbol{0}$. The optimal choice of μ depends mainly on the number of observed entries, the amount of noise, and the rank of the unknown matrix. Setting a larger μ will yield a lower-rank solution with a poor fit to the observations, i.e., $\|\boldsymbol{M} - P_\Omega(\boldsymbol{F})\|_F^2$ will be large. On the other hand, a small μ will provide a better fit to the observations and less restriction on the rank. When there are very few observations or high noise, it is preferable to set a large μ in order to prevent overfitting to $\boldsymbol{M}$ and achieve better accuracy for the unknown entries. As toy example, Fig. 2.3 depicts the underfitting and overfitting problems in polynomial regression. In MC terms, choosing a higher degree polynomial is analogous to allowing a higher rank solution, i.e., setting a small μ. In real applications, μ is typically chosen by cross-validation: the MC algorithm is run on a training dataset for several values, and the value that gives the smallest error on a different testing dataset is chosen.

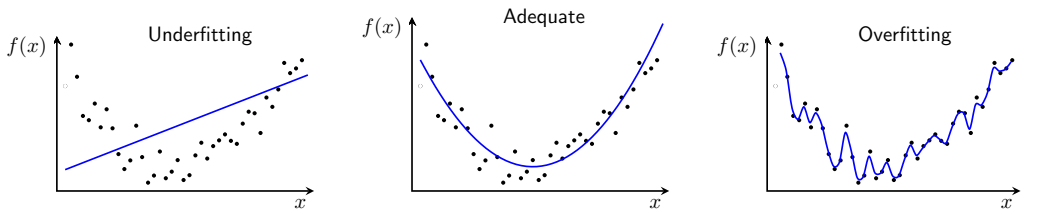


Figure 2.3: Fitting a noisy quadratic function with a low (left), adequate (middle), and high (right) degree polynomial. The overfitted polynomial attains a very small error on the training data but it will have a high error on any new data since it does not correctly estimate the quadratic function.

Among the algorithms for minimizing (2.15), two popular options are those based on matrix factorization [39, 40], and proximal gradient [27, 38, 41]. While factorization-based algorithms are faster, they solve a nonconvex version of (2.15). On the other hand, proximal gradient solves the convex (2.15), which has a guaranteed unique minimum. Factorization-based approaches will be discussed in Chapter 3, and the next section introduces proximal gradient minimization for MC.

2.2 Proximal gradient minimization for matrix completion

Proximal algorithms are a type of methods that solve optimization problems by means of a proximal operator [28, 42]. This operator is especially useful when dealing with nonsmooth objective functions since it turns them into smooth ones. Its combination with gradient

descent yields the proximal gradient (PG) algorithm, which can lead to a simple and efficient iterative minimization scheme. PG is used in MC as an alternative to SDP which allows for the recovery of larger matrices in simple repeating steps. This section first reviews concepts on proximal operators, and then details the derivation of the PG algorithm for MC.

The proximal operator associated with a convex function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is a Hilbert space, is defined as

$$\text{prox}_f(v) = \arg\min_x \frac{1}{2t}\|x - v\|_2^2 + f(x) \tag{2.16}$$

where $t > 0$ is a weight parameter. There are different interpretations for the proximal operator, such as the Moreau-Yosida envelope or as a trust region problem [43]. The latter is the simplest, and it boils down to the fact that the proximal operator finds the minimum of f around a point v. Take for instance the indicator function of a set $\mathcal{Y}$

$$\mathcal{I}(x) = \begin{cases} 0, & \text{when } x \in \mathcal{Y}, \\ \infty, & \text{when } x \notin \mathcal{Y}. \end{cases} \tag{2.17}$$

Its proximal operator $\text{prox}_{\mathcal{I}}(v) = \arg\min_{x\in\mathcal{Y}} \|x - v\|_2^2$ turns out to be the orthogonal projector onto $\mathcal{Y}$. The proximal operator has the following important properties:

- Convexity.
- Firm nonexpansiveness: $\|\text{prox}_f(v) - \text{prox}_f(y)\|_2^2 \leq \langle v - y, \text{prox}_f(v) - \text{prox}_f(y)\rangle \forall x, y \in \mathcal{X}$.
- A fixed point of prox_f is also a minimizer of f.

These properties guarantee that consecutive applications of (2.16) will reach a fixed point which will minimize f. Moreover, this is also true for nonsmooth f, since the proximal operator is smooth. Finally, the evaluation of (2.16) can often yield a function that is easy to evaluate and hence simplify the minimization process; see Fig. 2.4 for an example depiction.

In MC, the objective in (2.15) is a convex nonsmooth function which can be written as

$$f(\boldsymbol{F}) = g(\boldsymbol{F}) + h(\boldsymbol{F}) \tag{2.18}$$

where $g(\boldsymbol{F}) = \frac{1}{2}\|P_\Omega(\boldsymbol{F}) - \boldsymbol{M}\|_F^2$ is a smooth component and $h(\boldsymbol{F}) = \mu\|\boldsymbol{F}\|_*$ is nonsmooth. Therefore, a proximal algorithm will be able to find its minimum. The PG algorithm for MC which minimizes (2.18) is an iterative scheme that executes two steps repeatedly: first, a gradient step towards the minimum of g is taken and second, the proximal operator of the

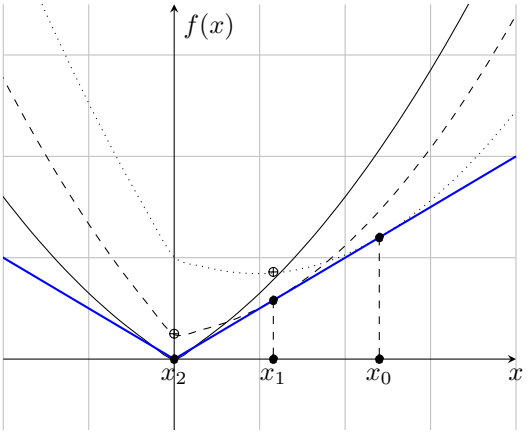


Figure 2.4: Minimization of the nonsmooth $f(x) = |x|$, colored in blue, via successive application of a proximal operator at x_k, $k = 0, 1, 2$. The lines in black represent $\frac{1}{2t}\|x - v\|_2^2 + f(x)$ in (2.16) for $v = x_k$, and the symbol $\oplus$ marks its minimum, i.e., $\text{prox}_f(x_k)$.

nuclear norm is applied. The two steps are executed as follows

$$\boldsymbol{F}' = \boldsymbol{F}_{k-1} - t\nabla g(\boldsymbol{F}_{k-1}) \tag{2.19}$$

$$\boldsymbol{F}_k = S_{t\mu}(\boldsymbol{F}'). \tag{2.20}$$

where $S_{t\mu}$ is the proximal operator of the nuclear norm, also known as matrix shrinkage operator (MSO). Following the formulation in (2.16), given $\boldsymbol{Z}$ with SVD $\boldsymbol{Z} = \boldsymbol{U}_Z\boldsymbol{\Sigma}_Z\boldsymbol{V}_Z$ the MSO is

$$S_{t\mu}(\boldsymbol{Z}) = \arg\min_{\boldsymbol{F}} \frac{1}{2t}\|\boldsymbol{F} - \boldsymbol{Z}\|_F^2 + \mu\|\boldsymbol{F}\|_* \tag{2.21}$$

$$= \boldsymbol{U}_Z D_\mu(\boldsymbol{\Sigma}_Z)\boldsymbol{V}_Z \tag{2.22}$$

where

$$D_\mu(\boldsymbol{\Sigma}_Z)_{n,n} = \begin{cases} (\boldsymbol{\Sigma}_Z)_{n,n} - \mu, & \text{when } (\boldsymbol{\Sigma}_Z)_{n,n} \geq \mu, \\ 0, & \text{otherwise.} \end{cases} \tag{2.23}$$

for $n \leq \text{rank}(\boldsymbol{Z})$. Actually, D_μ is the vector shrinkage operator [44] used to solve $l1$-regularized problems as the proximal operator of the $l1$ norm. This operator shrinks every singular value of $\boldsymbol{Z}$ by an amount μ, hence making the smaller singular values zero when applied repeatedly. Combining (2.19) and (2.20) into a single operation, the PG estimate of $\boldsymbol{F}$ at iteration k is

$$\boldsymbol{F}_k = S_{t\mu}(\boldsymbol{F}_{k-1} - t\nabla g(\boldsymbol{F}_{k-1})) = S_{t\mu}(\boldsymbol{F}_{k-1} - t(P_\Omega(\boldsymbol{F}_{k-1}) - \boldsymbol{M})). \tag{2.24}$$

Since each iteration of (2.24) requires an SVD decomposition, its cost is $\mathcal{O}(N^2L)$ per iteration. The complete derivation of the PG algorithm for MC is detailed below.

Proximal gradient derivation for MC

To begin with, we will focus on minimizing $g(\boldsymbol{F})$ and show how the gradient descent method, which generates a sequence $\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots$ of estimates of $\boldsymbol{F}$, results from the reiterated application of a proximal operator.

Let $q_g(\boldsymbol{F}, \boldsymbol{Z})$ denote the quadratic approximation to $g(\boldsymbol{F})$ around a point $\boldsymbol{Z}$, defined as

$$q_g(\boldsymbol{F}, \boldsymbol{Z}) = g(\boldsymbol{Z}) + \langle \nabla g(\boldsymbol{Z}), (\boldsymbol{F} - \boldsymbol{Z}) \rangle + \frac{1}{2t} \|\boldsymbol{F} - \boldsymbol{Z}\|_{\mathrm{F}}^2 . \tag{2.25}$$

Then, the kth gradient iteration $\check{\boldsymbol{F}}_k = \check{\boldsymbol{F}}_{k-1} - t\nabla g(\check{\boldsymbol{F}}_{k-1})$ for minimizing $g(\boldsymbol{F})$ is obtained by optimizing (2.25) for $\boldsymbol{Z} = \check{\boldsymbol{F}}_{k-1}$ as shown below.

$$\begin{aligned}
\check{\boldsymbol{F}}_k &= \underset{\boldsymbol{F}\in\mathbb{R}^{N\times L}}{\arg\min}\; q_g(\boldsymbol{F}, \check{\boldsymbol{F}}_{k-1}) \\
&= \underset{\boldsymbol{F}\in\mathbb{R}^{N\times L}}{\arg\min}\; g(\check{\boldsymbol{F}}_{k-1}) + \langle \nabla g(\check{\boldsymbol{F}}_{k-1}), (\boldsymbol{F} - \check{\boldsymbol{F}}_{k-1}) \rangle + \frac{1}{2t} \left\|\boldsymbol{F} - \check{\boldsymbol{F}}_{k-1}\right\|_{\mathrm{F}}^2 \qquad (2.26) \\
&= \underset{\boldsymbol{F}\in\mathbb{R}^{N\times L}}{\arg\min}\; g(\check{\boldsymbol{F}}_{k-1}) + \mathrm{Tr}(\nabla g(\check{\boldsymbol{F}}_{k-1})^T \boldsymbol{F} - \nabla g(\check{\boldsymbol{F}}_{k-1})^T \check{\boldsymbol{F}}_{k-1}) \\
&\qquad\qquad + \frac{1}{2t} \mathrm{Tr}(\boldsymbol{F}^T\boldsymbol{F} - 2\boldsymbol{F}^T\check{\boldsymbol{F}}_{k-1} + \check{\boldsymbol{F}}_{k-1}^T\check{\boldsymbol{F}}_{k-1}) \\
&= \underset{\boldsymbol{F}\in\mathbb{R}^{N\times L}}{\arg\min}\; \mathrm{Tr}\Big[\nabla g(\check{\boldsymbol{F}}_{k-1})^T \boldsymbol{F} - \nabla g(\check{\boldsymbol{F}}_{k-1})^T \check{\boldsymbol{F}}_{k-1} + \frac{1}{2t}(\boldsymbol{F}^T\boldsymbol{F} - 2\boldsymbol{F}^T\check{\boldsymbol{F}}_{k-1} + \check{\boldsymbol{F}}_{k-1}^T\check{\boldsymbol{F}}_{k-1}) \\
&\qquad\qquad + \frac{t^2}{2}\nabla g(\check{\boldsymbol{F}}_{k-1})^T \nabla g(\check{\boldsymbol{F}}_{k-1})\Big] \\
&= \underset{\boldsymbol{F}\in\mathbb{R}^{N\times L}}{\arg\min}\; \frac{1}{2t} \left\|\boldsymbol{F} - (\check{\boldsymbol{F}}_{k-1} - t\nabla g(\check{\boldsymbol{F}}_{k-1}))\right\|_{\mathrm{F}}^2 . \qquad (2.27)
\end{aligned}$$

Comparing with (2.16), we observe that the operation in (2.26) is actually the proximal operator of the linear approximation to $g(\boldsymbol{F})$ around $\check{\boldsymbol{F}}_{k-1}$. Moreover, (2.27) is minimized for $\boldsymbol{F} = \check{\boldsymbol{F}}_{k-1} - t\nabla g(\check{\boldsymbol{F}}_{k-1})$. Therefore, successively applying the proximal operator yields the gradient descent algorithm and, given a correct choice for t it will reach a fixed point which will also be a minimizer of $g(\boldsymbol{F})$.

Since (2.25) is an upper bound on $g(\boldsymbol{F})$ when $t \in (0, 1/C]$ [43], where C is the Lipschitz constant of $g(\boldsymbol{F})$, the minimization of $q_g(\boldsymbol{F}, \boldsymbol{Z})$ in (2.26) is a majorization-minimization type algorithm. That is, we first approximate a function with an upper bound and then minimize

this bound. Applying this same idea to (2.18), we obtain the approximation

$$q_f(\boldsymbol{F}, \boldsymbol{Z}) = q_g(\boldsymbol{F}, \boldsymbol{Z}) + \mu\|\boldsymbol{F}\|_*. \tag{2.28}$$

Minimizing (2.28) around a $\boldsymbol{F}_{k-1}$ following a similar process as in (2.27) yields

$$\boldsymbol{F}_k = \underset{\boldsymbol{F}\in\mathbb{R}^{N\times L}}{\arg\min}\, q_f(\boldsymbol{F}, \boldsymbol{F}_{k-1}) = \underset{\boldsymbol{F}\in\mathbb{R}^{N\times L}}{\arg\min}\, \frac{1}{2t}\left\|\boldsymbol{F} - (\boldsymbol{F}_{k-1} - t\nabla g(\boldsymbol{F}_{k-1})\right\|_{\mathrm{F}}^2 + \mu\left\|\boldsymbol{F}\right\|_*. \tag{2.29}$$

We have that (2.29), which is solved by (2.24), is the proximal operator of $\mu\|\boldsymbol{F}\|_*$ applied around $\boldsymbol{F}_{k-1} - t\nabla g(\boldsymbol{F}_{k-1})$. Hence, this iterative scheme is known as proximal gradient descent.

2.2.1 Proximal gradient algorithms with varying regularization parameter

The regularization parameter μ in (2.15) controls the balance between the error with respect to $\boldsymbol{M}$ and the nuclear norm. Moreover, it also controls the convergence speed of the iterative algorithm in (2.24). A large μ leads to fasert convergence [43] to a solution of lower nuclear norm, whereas with a small μ the algorithm takes longer to converge but it reaches a solution with a better fit to the observed entries. Thus, while there is an optimum μ to minimize the error in recovering $\boldsymbol{F}$, its value must also be chosen to optimize the overall performance of the PG.

Typical implementations of the proximal gradient algorithm use a warm start technique to increase the convergence speed when MC is to be solved for a given regularization parameter $\bar{\mu}$. This is shown in Algorithm 1, where (2.15) is sequentially solved with (2.24) for a list of N_μ values $\mu_1 > \mu_2 > \ldots > \mu_{N_\mu} = \bar{\mu}$, with the solution obtained for a μ_i being used as a starting point for the proximal gradient with the next μ_{i+1}. A switching criterion $\Phi(\cdot)$ is evaluated at each iteration to determine whether a solution has been reached and, if it yields true, a switch to the next μ is made. For instance, in the fixed point continuation (FPC) algorithm from [27] the switch is made once the proximal gradient converges to a fixed point, which is controlled by the switching criterion

$$\Phi_{FPC}(\boldsymbol{F}_k, \boldsymbol{F}_{k-1}) = \left(\frac{\|\boldsymbol{F}_k - \boldsymbol{F}_{k-1}\|_{\mathrm{F}}^2}{\|\boldsymbol{F}_{k-1}\|_{\mathrm{F}}^2} < \varepsilon_{FPC}\right), \tag{2.30}$$

where ε_{FPC} is a small positive constant. Moreover, the FPC algorithm sets the next regularization parameter as

$$\mu_{i+1} = \max\{\mu_i \cdot \eta_{FPC}, \bar{\mu}\}, \tag{2.31}$$

where $\eta_{FPC} \in (0, 1)$ is the reduction factor. The algorithm stops when a maximum number of iterations is reached or when the proximal gradient has converged for $\bar{\mu}$.

Algorithm 1: proximal gradient with varying μ

Input : $\boldsymbol{M}, \boldsymbol{F}_0, \mu_0, \bar{\mu}, t$
Output : $\hat{\boldsymbol{F}}$
function $\text{pg}(\boldsymbol{M}, \boldsymbol{F}_0, \mu_0, \bar{\mu}, t)$
$\boldsymbol{F} = \boldsymbol{F}_0$
for $\mu = \mu_1, \mu_2, \ldots, \bar{\mu}$ **do**
  do
    $\boldsymbol{F} \leftarrow S_{t\mu}(\boldsymbol{F} - t(P_\Omega(\boldsymbol{F}) - \boldsymbol{M}))$
  while not $\Phi(\cdot)$
end
return $\boldsymbol{F}$

For the sake of convenience, let us define the error of a matrix $\boldsymbol{Z}$ with respect to $\boldsymbol{F}$ as $e_{\boldsymbol{F}}(\boldsymbol{Z}) = \|\boldsymbol{Z} - \boldsymbol{F}\|_{\text{F}}^2$, and the error with respect to the observed entries as $e_{\boldsymbol{M}}(\boldsymbol{Z}) = \|P_\Omega(\boldsymbol{Z}) - \boldsymbol{M}\|_{\text{F}}^2$. Moreover, let us assume that $\bar{\mu}$ minimizes $e_{\boldsymbol{F}}(\hat{\boldsymbol{F}})$ and, therefore, any solution obtained for $\mu_i > \bar{\mu}$ will have a larger estimation error. Under this assumption, the FPC algorithm can become inefficient since reaching a fixed point for a given μ_i might require extra iterations that do not contribute significantly to lowering the error. Therefore, the switch to the next μ can be made earlier in order to further reduce the iterations to arrive at the solution for $\bar{\mu}$. In this section two proximal gradient algorithms are proposed, namely steered proximal gradient (SPG) and vanishing proximal gradient (VPG), that have a varying regularization parameter and rely on a criterion other than the convergence of the proximal gradient to decide when to switch to a lower μ. Thus, while the final solution obtained by each algorithm is the same as the standard proximal gradient for $\mu = \bar{\mu}$, it is reached in a smaller number of iterations.

Given the assumption that a smaller μ_i leads to a more accurate solution since it is closer to $\bar{\mu}$, and that it also leads to a smaller error with respect to $\boldsymbol{M}$, the SPG algorithm is based on Algorithm 1 and uses the following switching criterion:

$$\Phi_{SPG}(\boldsymbol{F}_k, \boldsymbol{F}_{k-1}) = \left(\frac{e_{\boldsymbol{M}}(\boldsymbol{F}_{k-1}) - e_{\boldsymbol{M}}(\boldsymbol{F}_k)}{e_{\boldsymbol{M}}(\boldsymbol{F}_{k-1})} < \varepsilon_{SPG} \right) \tag{2.32}$$

where ε_{SPG} is a small positive constant. With this criterion, when the error in the observed entries increases between two consecutive iterations it is assumed that the proximal gradient is moving away from the optimum solution and μ should be made smaller in order to reduce the error and steer the proximal gradient towards $\boldsymbol{F}$. The sequence of regularization parameters

Algorithm	$\Phi(\cdot)$	μ_{i+1}
Conv. PG	true	$\bar{\mu}$
FPC	$\frac{\|\boldsymbol{F}_k-\boldsymbol{F}_{k-1}\|_{\mathrm{F}}^2}{\|\boldsymbol{F}_{k-1}\|_{\mathrm{F}}^2} < \varepsilon_{FPC}$	$\max\{\mu_i \cdot \eta_{FPC}, \bar{\mu}\}$
SPG	$\frac{e_{\boldsymbol{M}}(\boldsymbol{F}_{k-1})-e_{\boldsymbol{M}}(\boldsymbol{F}_k)}{e_{\boldsymbol{M}}(\boldsymbol{F}_{k-1})} < \varepsilon_{SPG}$	$\max\{\mu_i \cdot \eta_{SPG}, \bar{\mu}\}$
VPG	true	$\max\{\mu_0 \cdot \eta_{VPG}^k, \bar{\mu}\}$

Table 2.1: Switching criteria and regularization parameters for the PG-based algorithms.

for this algorithm is given as

$$\mu_{i+1} = \max\{\mu_i \cdot \eta_{SPG}, \bar{\mu}\}. \tag{2.33}$$

As a simpler alternative to the SPG algorithm, the VPG algorithm implements the conventional proximal gradient algorithm in (2.24) and exponentially reduces the value of μ at each iteration by setting its value as

$$\mu_k = \max\{\mu_0 \cdot \eta_{VPG}^k, \bar{\mu}\}. \tag{2.34}$$

For the sake of clarity, note that the iterative solution given in Algorithm 1 is general for the PG-based algorithms, namely the conventional PG, FPC, SPG and VPG algorithms with switching criteria and regularization parameters given in Table 2.1.

2.3 Matrix completion for graph signals

As mention in the introduction to the chapter, $\boldsymbol{F}$ might have an underlying structure which can leveraged to improve the recovery result. One possible approach is to model this structure through a graph derived or known beforehand. In this section, the columns of the unknown matrix are assumed to be signals lying on a known graph; for an introduction to the topic of graph signals, see Appendix A. Using the graph adjacency matrix, an initialization method for the PG algorithm is presented, which gives a starting point which is closer to the minimum and therefore improves the overall convergence speed. Moreover, numerical results comparing the performance of the algorithms introduced in the previous section are provided, while also showing the benefits of using a graph-based initialization method.

2.3.1 Graph-based proximal gradient initialization

Proximal gradient algorithms are usually initialized to the all-zero matrix given the lack of previous information about the data. With the aim to increase the convergence speed, in this section the PG-based algorithms in Table I are initialized to a point hopefully closer to the optimum solution by making use of the fact that the columns of $\boldsymbol{F}$ are graph signals lying on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{A})$.

Leveraging the graph signal inpainting method in [45], a preliminary estimate of $\boldsymbol{F}$ can be obtained as:

$$\begin{aligned} \check{\boldsymbol{F}} = \underset{\boldsymbol{F}\in\mathbb{R}^{N\times L}}{\arg\min} \quad & \|\boldsymbol{F} - \boldsymbol{A}\boldsymbol{F}\|_{\mathrm{F}}^2 \\ \text{s.t.} \quad & P_{\Omega}(\boldsymbol{F}) = \boldsymbol{M} \end{aligned} \tag{2.35}$$

and set it as the initialization $\boldsymbol{F}_0$ in Algorithm 1. This optimization problem recovers the matrix that, given the observed $\boldsymbol{M}$, is smoothest with respect to the graph with weighted adjacency matrix $\boldsymbol{A}$. Indeed, the norm

$$T(\boldsymbol{F}) = \|\boldsymbol{F} - \boldsymbol{A}\boldsymbol{F}\|_{\mathrm{F}}^2 \tag{2.36}$$

is known as the total variation function and was used in [20] as a regularization term added to (2.15) to enforce the graph structure on the recovered data. When the total variation is small, $\boldsymbol{A}$ accurately describes the underlying relational structure of the graph signals in $\boldsymbol{F}$.

The solution to (2.35), derived in [45] for the vector case, i.e. $L = 1$, can be used to solve (2.35) when $L > 1$ since the problem is decoupled columnwise. For completeness, we reproduce the solution in [45] to recover the l^{th} column of $\check{\boldsymbol{F}}$ in (2.35), i.e. $\check{\boldsymbol{f}}_l$, using the weighted adjacency matrix $\boldsymbol{A}$ and the l^{th} column of $\boldsymbol{M}$, denoted by $\boldsymbol{m}_l$.

Let us first define the subset of observed entries in column l as $\Omega_l = \{i \,|\, (i,l) \in \Omega\}$ and the subset of unobserved entries as $\Omega_l^c = \{i \,|\, (i,l) \in \Omega^c\}$. Then, given a graph signal $\boldsymbol{z} \in R^{N\times 1}$, we also define the operators $(\cdot)^{\Omega_l}$ and $(\cdot)^{\Omega_l^c}$ as

$$\boldsymbol{z}^{\Omega_l} = [z(i) \,|\, i \in \Omega_l], \tag{2.37}$$

$$\boldsymbol{z}^{\Omega_l^c} = [z(i) \,|\, i \in \Omega_l^c]. \tag{2.38}$$

The reordered graph signal is defined as

$$\boldsymbol{z}_l^o = \begin{bmatrix} \boldsymbol{z}^{\Omega_l} \\ \boldsymbol{z}^{\Omega_l^c} \end{bmatrix} \tag{2.39}$$

a vector whose $|\Omega_l|$ first entries are indexed by the observed entries of $\boldsymbol{m}_l$, and the $|\Omega_l^c|$ last entries are indexed by the zero entries of $\boldsymbol{m}_l$. Similarly, let us define the reordered weighted adjacency matrix $\boldsymbol{A}_l^o$ as the matrix satisfying $\boldsymbol{A}_l^o \boldsymbol{z}_l^o = (\boldsymbol{A}\boldsymbol{z})_l^o$, where $(\boldsymbol{A}\boldsymbol{z})_l^o$ denotes the reordered version of $\boldsymbol{A}\boldsymbol{z}$ as in (2.39). Then, according to [45], the reordered l^{th} column of $\check{\boldsymbol{F}}$ in (2.35) is given by $\check{\boldsymbol{f}}_l^o = [\boldsymbol{m}_l^{\Omega_l}; \check{\boldsymbol{f}}_l^{\Omega_l^c}]$, where

$$\check{\boldsymbol{f}}_l^{\Omega_l^c} = -\boldsymbol{D}_{\Omega_l^c,\Omega_l^c}^{-1} \cdot \boldsymbol{D}_{\Omega_l^c,\Omega_l} \cdot \boldsymbol{m}_l^{\Omega_l}, \tag{2.40}$$

$\boldsymbol{D}_l = (\boldsymbol{I} - \boldsymbol{A}_l^o)^T(\boldsymbol{I} - \boldsymbol{A}_l^o)$ is partitioned as

$$\boldsymbol{D}_l = \begin{bmatrix} \boldsymbol{D}_{\Omega_l,\Omega_l} & \boldsymbol{D}_{\Omega_l,\Omega_l^c} \\ \boldsymbol{D}_{\Omega_l^c,\Omega_l} & \boldsymbol{D}_{\Omega_l^c,\Omega_l^c} \end{bmatrix} \tag{2.41}$$

and each submatrix $\boldsymbol{D}_{\nu,\phi}$ has dimensions $|\nu| \times |\phi|$.

Calculating the exact solution to (2.35) using (2.40) can be very computationally intensive for large matrices. Indeed, assuming $\boldsymbol{M}$ has s observed entries per column on average, the computational cost of (2.40) is $O(L((N-s)^3 + (N-s)s)) \approx O(L(N-s)^3)$[1].

Therefore, and since we are simply looking for a convenient initialization $\boldsymbol{F}_0$ for the PG-based algorithms, we opt for an alternative expression based on an approximate solution to (2.35) that is less computationally demanding and performs excellently as it will be shown in the simulations.

Let the eigendecomposition of the adjacency matrix be $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, and Ω^c denote the complementary set of Ω. Knowing that the best rank P approximation to a matrix in terms of mean-square error is built using its first P singular vectors, the initialization matrix is

$$\boldsymbol{F}_0 = P_{\Omega^c}(\boldsymbol{Q}_0\boldsymbol{C}_0) + \boldsymbol{M}, \tag{2.42}$$

where $\boldsymbol{Q}_0 = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_P]$, and $\boldsymbol{C}_0 \in \mathbb{R}^{P\times L}$ is a coefficient matrix obtained as

$$\boldsymbol{C}_0 = \arg\min_{\boldsymbol{C}} \|\boldsymbol{M} - P_\Omega(\boldsymbol{Q}_0\boldsymbol{C})\|_F^2. \tag{2.43}$$

Eq. (2.43) finds an approximation to $\boldsymbol{M}$ considering only the non-zero entries. Since the problem is columnwise decoupled, if we denote $\boldsymbol{C}_0 = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_L]$, then (2.43) is equivalent to solving

$$\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_L\} = \arg\min_{\boldsymbol{c}_1,\ldots,\boldsymbol{c}_L} \sum_{l=1}^{L} \|\bar{\boldsymbol{m}}_l - \boldsymbol{S}_l\boldsymbol{Q}_0\boldsymbol{c}_l\|_2^2, \tag{2.44}$$

[1]For each column, the inversion of $\tilde{\boldsymbol{D}}_{\Omega_l^C,\Omega_l^C}$ requires $O(N-s)^3$ operations [46], whereas the products in (2.40) need $O(N(N-s))$.

where $\bar{\boldsymbol{m}}_l$ is a vector of length $|\Omega_l|$ containing the observed samples of the lth column of $\boldsymbol{M}$, and $\boldsymbol{S}_l$ is a $|\Omega_l| \times N$ sampling matrix. The matrix $\boldsymbol{S}_l$ has a single non-zero element per row equal to 1, and it is built so that $\bar{\boldsymbol{m}}_l = \boldsymbol{S}_l \boldsymbol{m}_l$. The solution to (2.44) is

$$\boldsymbol{c}_l = (\boldsymbol{Q}_0^T \boldsymbol{S}_l^T \boldsymbol{S}_l \boldsymbol{Q}_0)^{-1} \boldsymbol{Q}_0^T \boldsymbol{S}_l^T \bar{\boldsymbol{m}}_l \;\; \forall l = 1, \dots, L. \tag{2.45}$$

At this point, we want to compare the computational cost of using (2.40), which is $O(L(N-s)^3)$ to the cost of obtaining (2.42). Assuming again an average of s observed entries per column, the computational cost of (2.42) is the cost of calculating $\boldsymbol{C}_0$, which entails the eigendecomposition of $\boldsymbol{A}$ with cost $O(N^3)$ [46], plus the cost of the multiplications involved in (2.45) and (2.42), which results in a total cost $O(N^3 + N^2 P + L(P^3 + P^2 + P^2 s + Ps))$ that can be approximated to $O(N^3)$ for small P and large N. This is a smaller cost than one PG iteration for $N < L$.

When the number of observations is low, we can safely assume that the cost of the proposed approximation (2.42) is lower than the cost of using (2.40). Moreover, we can just calculate the top P eigenvectors to obtain $\boldsymbol{Q}_0$, which would greatly reduce the computational cost. For instance for a 500×500 matrix with $s = 100$ and a weighted adjacency matrix with $P = 1$, the computational cost of calculating the approximation is $O(125 \cdot 10^6)$. If we used (2.40) instead, the cost is $O(32 \cdot 10^9)$, which is 256 times higher than the cost of our initialization.

2.3.2 Numerical tests

The matrix completion algorithms have been tested on a synthetic dataset and a real dataset of temperature measurements. We compare the following algorithms: a) the conventional PG, b) the FPC algorithm in [27], c) the SPG and VPG algorithms proposed in Section 2.2.1, and d) the graph signal completion via total variation regularization (GMRC) algorithm in [20]. GMRC adds the total variation (2.36) to (2.15) as a regularization term and performs a PG minimization. All the algorithms have been tested initializing them to $\boldsymbol{F}_0 = \boldsymbol{0}$, to which we will refer as zero-initialized (ZI), and to the solution in (2.42), to which we will refer as graph-initialized (GI).

Table 2.2 shows the values of the algorithm parameters used in the simulations. The conventional PG and GMRC algorithms have a constant nuclear norm regularization parameter of value $\bar{\mu}$. The algorithms have been run for $K = 500$ iterations over $N_{rea} = 50$ realizations with different percentages of observed samples, denoted by $P_s = \frac{|\Omega|}{N \cdot L} \cdot 100$. As a performance

Parameter	Dataset	
	Synthetic	Real
$\bar{\mu}$	50	0.03
μ_0^{ZI}	$10\bar{\mu}$	$10\bar{\mu}$
μ_0^{GI}	$5\bar{\mu}$	$5\bar{\mu}$
η_{FPC}	0.75	0.65
η_{SPG}	0.65	0.55
η_{VPG}	0.85	0.65
ε_{FPC}	10^{-4}	10^{-4}
ε_{SPG}	0.06	0.06
t	1	1

Table 2.2: Algorithm parameters used in the simulations.

metric for the algorithms, we use the normalized mean square error at iteration k

$$\mathrm{NMSE}_k = \frac{1}{N_{rea}} \sum_{n=1}^{N_{rea}} \frac{\|\boldsymbol{F}_{k,n} - \boldsymbol{F}\|_{\mathrm{F}}^2}{\|\boldsymbol{F}\|_{\mathrm{F}}^2}, \tag{2.46}$$

where $\boldsymbol{F}_{k,n}$ is the estimate of $\boldsymbol{F}$ at iteration k and realization n. We say that an algorithm has converged when the difference in the NMSE between two consecutive iterations is below 10^{-4}, that is, when

$$NMSE_k - NMSE_{k-1} < 10^{-4}. \tag{2.47}$$

Additionally, we use the normalized total variation

$$\bar{T}(\boldsymbol{F}) = \frac{\|\boldsymbol{F} - \boldsymbol{A}\boldsymbol{F}\|_{\mathrm{F}}^2}{\|\boldsymbol{F}\|_{\mathrm{F}}^2} \tag{2.48}$$

as a measure of the smoothness of the graph signals for the given weighted adjacency matrix.

Synthetic dataset

Similar to the dataset used in [26], the synthetic dataset $\boldsymbol{H}$ is a 500×515 matrix of rank 10 generated as $\boldsymbol{H} = \frac{\boldsymbol{V}_1\boldsymbol{V}_2}{\|\boldsymbol{V}_1\boldsymbol{V}_2\|_{\mathrm{F}}^2}$, where $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ are 500×10 and 10×515 matrices respectively with random independent and identically distributed entries with distribution $\mathcal{N}(0, 1)$. Let us define the 500×15 matrix

$$\boldsymbol{T} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{15}] + \boldsymbol{E} \tag{2.49}$$

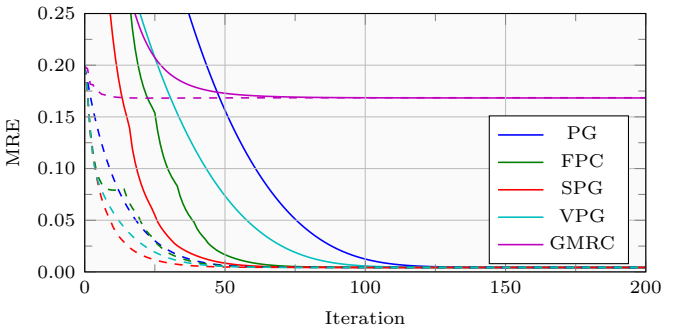


Figure 2.5: NMSE vs. iterations for $P_s = 20\%$ for the synthetic dataset with the algorithms initialized to $\boldsymbol{F}_0 = \boldsymbol{0}$ (solid lines) and with Eq. (2.42) (dashed lines).

where $\boldsymbol{E}$ is a matrix of noise with distribution $\mathcal{N}(\boldsymbol{0}, \frac{\|\boldsymbol{F}\|_{\mathrm{F}}^2}{7NL}\boldsymbol{I})$. This noisy $\boldsymbol{T}$ is used to build the adjacency $\boldsymbol{A} = \boldsymbol{U}_T\boldsymbol{U}_T^T$, where $\boldsymbol{U}_T$ denotes the left singular vector matrix of $\boldsymbol{T}$. The algorithms have been run on the 500×500 matrix $\boldsymbol{F} = [\boldsymbol{h}_{16}, \dots, \boldsymbol{h}_{515}]$. For the graph-based initialization, we use (2.42) with $P = 10$. The normalized total variation (2.48) for this dataset $\boldsymbol{F}$ and $\boldsymbol{A}$ is 0.2013.

Fig. 2.5 shows the evolution of the NMSE for all the algorithms when $P_s = 20\%$, with the zero-initialized algorithms shown in continuous lines and the graph-initialized (2.42) in dashed lines. This graphical distinction between the zero-initialized and graph-initialized algorithms will be used in all of the figures in this section. We observe in Fig. 2.5 that, for both the zero-initialized and graph-initialized cases, all the algorithms except GMRC converge to a point with the same NMSE since they end with the same μ, although at different speeds. The NMSE at iteration 0 is the error of $\boldsymbol{F}_0$, which is 1 for the zero-initialized algorithms (not shown in the plot) and 0.2 for the graph initialized. Hence, it is clear that the proposed initialization method reduces the convergence time for all the algorithms, and that the SPG algorithm has the best performance in this case. This figure also illustrates the importance of choosing a smaller μ_0 for the graph-initialized algorithms. Note that for $t = 1$ the observed matrix $\boldsymbol{M}$, which is high-rank, serves as a starting point for the algorithms when $\boldsymbol{F}_0 = \boldsymbol{0}$. When $\boldsymbol{F}_0$ is calculated using (2.42) instead, the power of the resulting matrix is more concentrated in the top eigenvalues, which allows for a lower μ_0. Indeed, if μ_0 were too large, the algorithms would begin by moving away from the minimum of (2.15), thus increasing the error, and would not start approaching the minimum until a small enough μ were chosen. This is especially critical for the FPC algorithm since its switching criterion is the convergence of the proximal gradient, which would spend many iterations reaching a fixed point with a higher error than the starting point.

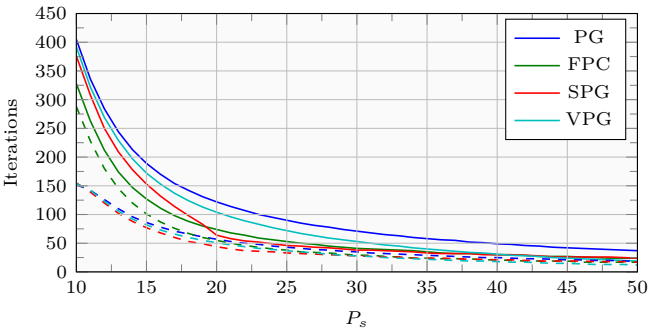


Figure 2.6: Iterations vs. P_s for the synthetic dataset.

Fig. 2.6 shows the iterations required to reach the solution for $\bar{\mu}$ for all the algorithms except GMRC, which we have excluded due to its poor performance, at different percentages of observed entries. We observe that as the number of samples increases, the number of iterations is reduced for all the algorithms. In the zero-initialized case, the algorithms with varying μ outperform the conventional PG, although which one is fastest depends on the percentage of observed entries. Thus, FPC is faster for low percentages, SPG for middle, and VPG for high. The convergence speed of the three algorithms largely depends on the choice of initial μ_0 and the rate at which μ decays for each algorithm. When P_s is low, the rank of $\boldsymbol{M}$ is higher and the initial soft-thresholding operations (2.21) are applied to higher rank matrices. Therefore, a larger μ is required and the algorithm for which μ decays at a slower rate obtains better results. On the other hand, when P_s grows the rank of $\boldsymbol{M}$ decreases and μ needs not be as large. Hence, the VPG algorithm, which has the fastest decaying μ, performs better.

When the algorithms are initialized using (2.42), Fig. 2.6 shows a reduction in the number of iterations for all the algorithms with respect to the zero-initialized case. Since the starting point is closer to the minimum, less iterations are required to reach it. On the other hand, the gap between the zero-initialized and graph-initialized version of the algorithms gets smaller as the percentage of observed samples increases. This is due to the fact that the error of $\boldsymbol{M}$, which serves as a starting point for the zero-initialized algorithms, with respect to $\boldsymbol{F}$ decreases. The graph-initialized SPG algorithm is the fastest for low values of P_s, whereas VPG is the fastest medium and high values.

From the simulations with the synthetic dataset we conclude that the initialization method proposed in Section 2.3.1 always improves the convergence speed of the proximal gradient algorithms, especially when the percentage of observed entries is low. Likewise, the graph-initialized SPG and VPG algorithms proposed in Section 2.2.1 have a higher

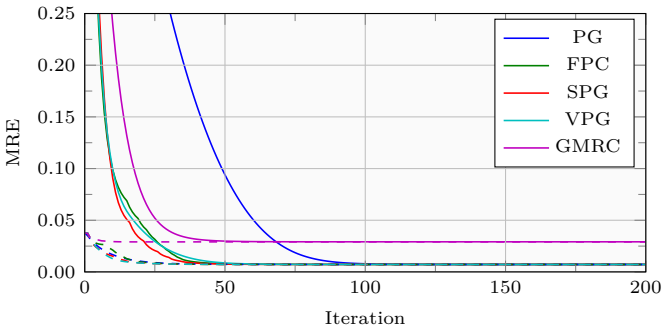


Figure 2.7: NMSE vs iterations for the real dataset.

convergence speed than the conventional PG and FPC algorithms. It should be noted that, although in the simulations we have tuned their parameters for a wide range of values of P_s, the performance of the algorithms with varying μ can be improved by setting the μ_0 and η parameters accordingly to the percentage of observed samples. Even so, the graph-initialized SPG and VPG algorithms are better suited for cases when μ_0 cannot be tightly adjusted in order to prevent the error of the proximal gradient from rising in the first iterations.

Real dataset

The dataset $\boldsymbol{F}$ is a 150×365 matrix of temperature readings taken by 150 stations over 365 days in 2002 in the United States [20]. The graph signals $\boldsymbol{x}_l$ are the temperature values measured at each station. In order to obtain the weighted adjacency matrix $\boldsymbol{A}$, first a graph $\mathcal{G}'$ with unweighted adjacency matrix $\boldsymbol{P}'$ is generated as in [20] for the stations. In this graph, each station is a vertex and is connected to the 8 geographically closest stations. Next, we obtain the undirected graph $\mathcal{G}$ with symmetric adjacency matrix $\boldsymbol{P} = \text{sign}(\boldsymbol{P}'^T + \boldsymbol{P}')$. Finally, the entries of $\boldsymbol{A}$ are calculated as $\boldsymbol{A}_{i,j} = \exp(-\frac{N^2 d_{i,j}}{\sum_{i,j} d_{i,j}})$, where $d_{i,j}$ are the geodesic distances on $\mathcal{G}$. The normalized total variation for the dataset $\boldsymbol{F}$ is 0.0383, and the observation matrix is $\boldsymbol{M} = P_\Omega(\boldsymbol{F})$.

Fig. 2.7 shows the evolution of the NMSE for $P_s = 20\%$. We observe that the starting point of the initialized algorithms is fairly close to the minimum since it has a very low error. Fig. 2.8 shows the iterations to convergence for all the algorithms at different percentages of observed entries. Among the zero-initialized algorithms, SPG is faster for low percentages and VPG is faster for high percentages. We observe a significant reduction in the number of iterations for the graph-initialized algorithms, especially when P_s is low. Since the total variation for this dataset is very low, the initialization is more accurate. Similar to the results with synthetic data, the graph-initialized SPG algorithm is the fastest for low values of P_s,

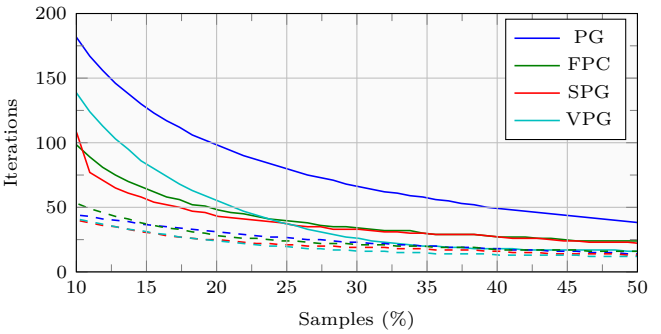


Figure 2.8: Iterations vs P_s for the real dataset.

and the graph-initialized VPG for middle-to-high values.

2.4 Matrix completion for noisy graph signals

This section addresses the recovery of partially observed graph signals that are arranged in a data matrix when the observations are noisy. The problem is solved by adding the Laplacian quadratic form as a regularization term as in [21, 47], and using the standard PG method, which facilitates the derivation theoretical bounds on the recovery error and give insight into the effect of the regularization. Moreover, numerical results are presented showing the performance of the regularized PG compared to its non-regularized counterpart.

2.4.1 Regularized proximal gradient minimization

When a graph signal is smooth on its graph, the values corresponding to connected vertices have similar values. On the other hand, if the signal is contaminated by noise, the noise does not necessarily satisfy the smoothness condition. Therefore, we can reduce the impact of the noise and take advantage of the structural information provided by the graph by incorporating its Laplacian matrix into the MC formulation. The quadratic Laplacian form measures the smoothness of a signal on the graph and is defined as

$$\mathcal{L}(\boldsymbol{F}) = \mathrm{Tr}(\boldsymbol{F}^T \boldsymbol{L} \boldsymbol{F}) = \frac{1}{2} \sum\nolimits_{l=1}^{L} \sum\nolimits_{(i,j)\in\mathcal{E}} \boldsymbol{A}_{i,j} (\boldsymbol{F}_{i,l} - \boldsymbol{F}_{j,l})^2, \tag{2.50}$$

Thus, the unknown matrix can be recovered by solving

$$\underset{\boldsymbol{F}\in\mathbb{R}^{N\times L}}{\arg\min} \frac{1}{2} \left\| P_\Omega(\boldsymbol{F}) - \boldsymbol{M} \right\|_{\mathrm{F}}^2 + \mu \left\| \boldsymbol{F} \right\|_* + \alpha \mathcal{L}(\boldsymbol{F}), \tag{2.51}$$

where the Laplacian quadratic form has been incorporated as an additional term weighted by a parameter α in order to promote similarity between the vertices connected on the graph. Note that, although here only the columns of $\boldsymbol{F}$ are considered to be graph signals, both columns and rows can simultaneously be graph signals on two different graphs. Hence, a second regularization term can be added to (2.51) to also enforce the graph structure on the rows. This approach will be detailed in Chapter 3.

Since the Laplacian quadratic form is a smooth function, (2.51) can be solved via the PG algorithm with the iterative scheme

$$\boldsymbol{F}_k = S_{t\mu}\left(\boldsymbol{F}_{k-1} - t\nabla\left(\|P_\Omega(\boldsymbol{F}_{k-1}) - \boldsymbol{M}\|_F^2 + \alpha\mathcal{L}(\boldsymbol{F}_{k-1})\right)\right) \tag{2.52}$$

$$= S_{t\mu}\left(\boldsymbol{F}_{k-1} - t(P_\Omega(\boldsymbol{F}_{k-1}) - \boldsymbol{M} + 2\alpha\boldsymbol{L}\boldsymbol{F}_{k-1})\right). \tag{2.53}$$

Section 2.3.1 proposed an initialization method based on the adjacency matrix. Similarly, $\boldsymbol{L}$ can be used to obtain an initialization point. Let $\boldsymbol{Q}_0 \in \mathbb{R}^{N\times P}$ be the matrix containing the P eigenvectors of $\boldsymbol{L}$ with associated eigenvalue 0. As explained in Appendix A, $\boldsymbol{Q}_0$ contains the graph signals that are smoothest on the graph so that $\mathcal{L}(\boldsymbol{Q}_0) = 0$. Therefore, an initial point can be obtained as in (2.44) using $\boldsymbol{Q}_0$ as a basis in the least-squares problem.

2.4.2 Error analysis of proximal gradient minimization

This section analyzes the recovery error of the PG algorithm and how the addition of the Laplacian quadratic form as a regularization term impacts on the noise. To avoid notation clutter, in this section the sampling operator is denoted as $(\cdot)^\Omega = P_\Omega(\cdot)$, and the matrix shrinkage operator is redefined as

$$S_{t\mu}(\boldsymbol{F}) = \boldsymbol{U}(\boldsymbol{\Sigma} - t\boldsymbol{\beta}_{\boldsymbol{F}})\boldsymbol{V} \tag{2.54}$$

where $\boldsymbol{\beta}_{\boldsymbol{F}}$ denotes a diagonal $N \times L$ matrix with main diagonal

$$\boldsymbol{d}_{\boldsymbol{F}} = [\mu, \ldots, \mu, \frac{\sigma_d}{t}, \ldots, \frac{\sigma_D}{t}] \tag{2.55}$$

with $D = \min(N, L)$, and $\{\sigma_d, \ldots, \sigma_D\}$ are the singular values of $\boldsymbol{F}$ smaller than $t\mu$, that is, $\sigma_i < t\mu \ \forall i \geq d$. Before introducing the main results, the following lemmas and assumptions are defined and made:

Assumption 1. Similar to Assumption 7 in [17], we assume that, given small enough α and t, there exists a constant $0 \leq \gamma < 1$ such that for any $\boldsymbol{F}$

$$\left\|(\boldsymbol{I} - 2\alpha t\boldsymbol{L})\boldsymbol{F} - t\boldsymbol{F}^\Omega\right\|_{\mathrm{F}} \leq \gamma \left\|\boldsymbol{F}\right\|_{\mathrm{F}}. \tag{2.56}$$

Lemma 2.1. *Given a pair of matrices $\boldsymbol{F}$ and $\boldsymbol{Z}$,*

$$\|S_{t\mu}(\boldsymbol{F}) - S_{t\mu}(\boldsymbol{Z})\|_F \leq \|\boldsymbol{F} - \boldsymbol{Z}\|_F . \tag{2.57}$$

Proof. This property is due to the non-expansiveness of the MSO, and its proof can be found in [27]. □

Lemma 2.2. *For any $\boldsymbol{F}$*

$$\|S_{t\mu}((\boldsymbol{I} - 2\alpha t\boldsymbol{L})\boldsymbol{F}) - \boldsymbol{F}\|_F \leq \sqrt{r}t\mu + 2\alpha t \|\boldsymbol{L}\boldsymbol{F}\|_F . \tag{2.58}$$

Proof. We begin by showing that

$$\begin{aligned} &\|S_{t\mu}((\boldsymbol{I} - 2\alpha t\boldsymbol{L})\boldsymbol{F}) - \boldsymbol{F}\|_{\mathrm{F}} \\ &= \|S_{t\mu}((\boldsymbol{I} - 2\alpha t\boldsymbol{L})\boldsymbol{F}) - \boldsymbol{F} + 2\alpha t\boldsymbol{L}\boldsymbol{F} - 2\alpha t\boldsymbol{L}\boldsymbol{F}\|_{\mathrm{F}} \\ &\leq \|S_{t\mu}((\boldsymbol{I} - 2\alpha t\boldsymbol{L})\boldsymbol{F}) - (\boldsymbol{I} - 2\alpha t\boldsymbol{L})\boldsymbol{F}\|_{\mathrm{F}} + 2\alpha t \|\boldsymbol{L}\boldsymbol{F}\|_{\mathrm{F}} . \end{aligned} \tag{2.59}$$

Next, let us define $\boldsymbol{F}' = (\boldsymbol{I} - 2\alpha t\boldsymbol{L})\boldsymbol{F}$ with SVD $\boldsymbol{F}' = \boldsymbol{U}'\boldsymbol{\Sigma}'\boldsymbol{V}'$. Then,

$$\begin{aligned} \|S_{t\mu}((\boldsymbol{I} - 2\alpha t\boldsymbol{L})\boldsymbol{F}) - \boldsymbol{F}\|_{\mathrm{F}} &\leq \|S_{t\mu}(\boldsymbol{F}') - S_{t\mu}(\boldsymbol{F}' + t\mu\boldsymbol{I})\|_{\mathrm{F}} + 2\alpha t \|\boldsymbol{L}\boldsymbol{F}\|_{\mathrm{F}} \\ &\leq \|S_{t\mu}(\boldsymbol{F}') - \boldsymbol{F}'\|_{\mathrm{F}} + 2\alpha t \|\boldsymbol{L}\boldsymbol{F}\|_{\mathrm{F}} \end{aligned} \tag{2.60}$$

which, after applying (2.54) and noticing that $\operatorname{rank}(\boldsymbol{F}') = r$, leads to (2.58). □

We now introduce the following theorem, which bounds the recovery error of the original matrix when the observed entries are noiseless:

Theorem 2.1. *Let $\hat{\boldsymbol{F}}$ be the rank r estimate obtained with (2.53) after the PG has converged, with a noiseless observed matrix $\boldsymbol{F}^{\Omega}$. Then*

$$\left\|\hat{\boldsymbol{F}} - \boldsymbol{F}\right\|_F \leq \frac{\sqrt{r}t\mu + 2\alpha t \|\boldsymbol{L}\boldsymbol{F}\|_F}{1 - \gamma}. \tag{2.61}$$

Proof. Since $f(\boldsymbol{F}) = \frac{1}{2}\left\|\boldsymbol{F}^{\Omega} - \boldsymbol{F}^{\Omega}\right\|_{\mathrm{F}}^2 + \alpha\mathcal{L}(\boldsymbol{F})$ is convex and continuously differentiable, the proximal gradient converges to a solution in the optimal solution set of (2.51) which is also a

fixed point of (2.53) [48]. Hence, since $\hat{\boldsymbol{F}}$ is a fixed point, we have that

$$\begin{aligned}
&\left\|\hat{\boldsymbol{F}}-\boldsymbol{F}\right\|_{\mathrm{F}} = \left\|S_{t\mu}(\hat{\boldsymbol{F}}-t(\hat{\boldsymbol{F}}^{\Omega}-\boldsymbol{F}^{\Omega}+2\alpha\boldsymbol{L}\hat{\boldsymbol{F}}))-\boldsymbol{F}\right\|_{\mathrm{F}} \\
&= \left\|S_{t\mu}((\boldsymbol{I}-2\alpha t\boldsymbol{L})\hat{\boldsymbol{F}}-t(\hat{\boldsymbol{F}}^{\Omega}-\boldsymbol{F}^{\Omega}))-S_{t\mu}((\boldsymbol{I}-2\alpha t\boldsymbol{L})\boldsymbol{F})+S_{t\mu}((\boldsymbol{I}-2\alpha t\boldsymbol{L})\boldsymbol{F})-\boldsymbol{F}\right\|_{\mathrm{F}} \\
&\leq \left\|S_{t\mu}((\boldsymbol{I}-2\alpha t\boldsymbol{L})\hat{\boldsymbol{F}}-t(\hat{\boldsymbol{F}}^{\Omega}-\boldsymbol{F}^{\Omega}))-S_{t\mu}((\boldsymbol{I}-2\alpha t\boldsymbol{L})\boldsymbol{F})\right\|_{\mathrm{F}} + \left\|S_{t\mu}((\boldsymbol{I}-2\alpha t\boldsymbol{L})\boldsymbol{F})-\boldsymbol{F}\right\|_{\mathrm{F}}.
\end{aligned} \tag{2.62}$$

Applying Lemma 1:

$$\left\|\hat{\boldsymbol{F}}-\boldsymbol{F}\right\|_{\mathrm{F}} \leq \left\|(\boldsymbol{I}-2\alpha t\boldsymbol{L})(\hat{\boldsymbol{F}}-\boldsymbol{F})-t(\hat{\boldsymbol{F}}-\boldsymbol{F})^{\Omega}\right\|_{\mathrm{F}} + \left\|S_{t\mu}((\boldsymbol{I}-2\alpha t\boldsymbol{L})\boldsymbol{F})-\boldsymbol{F}\right\|_{\mathrm{F}}. \tag{2.63}$$

Next, we apply Lemma 2 and Assumption 1 and obtain

$$\left\|\hat{\boldsymbol{F}}-\boldsymbol{F}\right\|_{\mathrm{F}} \leq \gamma\left\|\hat{\boldsymbol{F}}-\boldsymbol{F}\right\|_{\mathrm{F}} + \sqrt{r}t\mu + 2\alpha t\left\|\boldsymbol{L}\boldsymbol{F}\right\|_{\mathrm{F}} \tag{2.64}$$

which leads to the error bound in Theorem 1. □

Intuitively, Theorem 1 shows that the recovery error with noiseless observations can be reduced by setting a lower μ, as it was anticipated in [49], as well as by choosing a graph on which the graph signals in $\boldsymbol{F}$ are smooth. Nevertheless, whether the error can be driven to 0 by having $\mu \to 0$ depends on the number of observed entries and their distribution, which is implicit through the constant γ.

In order to assess the effect of the regularization on the noise, let us switch to another perspective. In the signal processing on graphs theory [31, 32], $\boldsymbol{Q}^T$ obtained from the eigendecomposition of $\boldsymbol{L}$ is referred to as the Graph Fourier Transform (GFT) matrix. The eigenvectors in $\boldsymbol{Q}$ can be viewed as frequencies on the graph, with the eigenvectors associated to the smaller eigenvalues corresponding to the lower frequencies, that is, signals that vary slowly on the graph. Given a vector $\boldsymbol{f}$, the product $\boldsymbol{L}\boldsymbol{f} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T\boldsymbol{f}$ projects the vector onto the graph frequency domain of $\boldsymbol{L}$, scales the frequency coefficients by $\boldsymbol{\Lambda}$, and projects the result back to the graph domain given by $\boldsymbol{Q}$. Therefore, the result of $\boldsymbol{L}\boldsymbol{f}$ is a highpass filtered version of the vector since the lowest frequency coefficients in $\boldsymbol{Q}^T\boldsymbol{f}$ associated with the smallest eigenvalue, which is equal to 0, are eliminated and the rest are scaled according to $\boldsymbol{\Lambda}$. Then, we can introduce the following theorem:

Theorem 2.2. *Given $\boldsymbol{I}_0 = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\dagger}$, where $\boldsymbol{\Lambda}^{\dagger}$ is the pseudoinverse of $\boldsymbol{\Lambda}$, the frequency content of $\hat{\boldsymbol{F}}$ on the nonzero frequencies of the graph is bounded as*

$$\left\|\boldsymbol{I}_0\boldsymbol{Q}^T\hat{\boldsymbol{F}}\right\|_F \leq \frac{1}{2\alpha}\left(\left\|\boldsymbol{\Lambda}^{\dagger}\boldsymbol{Q}^T(\boldsymbol{F}^{\Omega}-\hat{\boldsymbol{F}}^{\Omega})\right\|_F + \left\|\boldsymbol{\Lambda}^{\dagger}\boldsymbol{Q}^T\boldsymbol{E}\right\|_F + \left\|\boldsymbol{\Lambda}^{\dagger}\boldsymbol{Q}^T\boldsymbol{U}'\boldsymbol{\beta}_{\hat{\boldsymbol{F}}'}\boldsymbol{V}')\right\|_F\right) \tag{2.65}$$

where matrices $\boldsymbol{U}'$,$\boldsymbol{V}'$ are the left and right singular vector matrices of

$$\hat{\boldsymbol{F}}' = \hat{\boldsymbol{F}} - t(\hat{\boldsymbol{F}}^{\Omega} - \boldsymbol{M}) - 2\alpha t \boldsymbol{L}\hat{\boldsymbol{F}}. \tag{2.66}$$

Proof. Let $\hat{\boldsymbol{F}}'$ in (2.66) have SVD $\hat{\boldsymbol{F}}' = \boldsymbol{U}'\boldsymbol{\Sigma}'\boldsymbol{V}'$. Given an estimate $\hat{\boldsymbol{F}}$, which is a fixed point of (2.53), we have that $S_{t\mu}(\hat{\boldsymbol{F}}') = \hat{\boldsymbol{F}}$. Expanding the MSO in this last expression we obtain

$$\hat{\boldsymbol{F}}' - t\boldsymbol{U}'\boldsymbol{\beta}_{\hat{\boldsymbol{F}}'}\boldsymbol{V}' = \hat{\boldsymbol{F}}. \tag{2.67}$$

Now, if we replace $\hat{\boldsymbol{F}}'$ by its definition, substitute $\boldsymbol{M} = \boldsymbol{F}^{\Omega} + \boldsymbol{E}$ and rearrange the resulting equation, we have

$$2\alpha\boldsymbol{L}\hat{\boldsymbol{F}} = \boldsymbol{F}^{\Omega} - \hat{\boldsymbol{F}}^{\Omega} + \boldsymbol{E} - \boldsymbol{U}'\boldsymbol{\beta}_{\hat{\boldsymbol{F}}'}\boldsymbol{V}'. \tag{2.68}$$

Note that if we multiply $\boldsymbol{L}\hat{\boldsymbol{F}}$ by $\boldsymbol{\Lambda}^{\dagger}\boldsymbol{Q}^T$ we are left with a GFT matrix of the columns of $\hat{\boldsymbol{F}}$, containing only the coefficients of the nonzero frequencies. Hence, multiplying both sides of (2.68) by $\boldsymbol{\Lambda}^{\dagger}\boldsymbol{Q}^T$ we obtain the bound in (2.65). □

Theorem 2 shows that the Laplacian regularization has a lowpass filtering effect on the solution since the right-hand side of (2.65) vanishes with α. Moreover, the second term on the right-hand side of (2.65) also shows the filtering effect on the noise $\boldsymbol{E}$ since the coefficients of its GFT are multiplied by $\boldsymbol{\Lambda}^{\dagger}$. If the noise is not smooth on the graph, that is, $\mathcal{L}(\boldsymbol{E})$ is large and the GFT coefficients are concentrated around the higher frequencies, its contribution to the spectrum of $\hat{\boldsymbol{F}}$ will be small.

2.4.3 Numerical tests

We have tested the PG algorithm for the Laplacian-regularized matrix completion problem on the real dataset of temperature measurements. The algorithm has been run until convergence over $N_{rea} = 20$ realizations with different percentages of observed samples, denoted by $P_s = \frac{|\Omega|}{N \cdot L} \cdot 100$. The parameters of the PG are $\mu = 200$, which is set relative to the singular values of the observed dataset, $t = 0.05$ and $\alpha = \{0, 0.5\}$. As a performance metric, we use the NMSE (2.46). The initial estimates are calculated using (2.45) and the Laplacian.

Fig. 2.9 shows the NMSE after convergence for different percentages of samples and parameter values. We observe that in the noiseless case the NMSE is reduced as the percentage increases. Moreover, the PG shows a larger error with $\alpha = 0.5$ than with $\alpha = 0$. This is because there are enough samples and μ is small enough to allow the non-regularized PG to attain a low recovery error. In the noisy case, Fig. 2.9 shows that the error also decreases with P_s for $\alpha = 0.5$, although it rises for $\alpha = 0$ after $P_s = 25\%$. This is due to the absence of regularization, which causes the PG to overfit to the noisy entries thus resulting in a larger

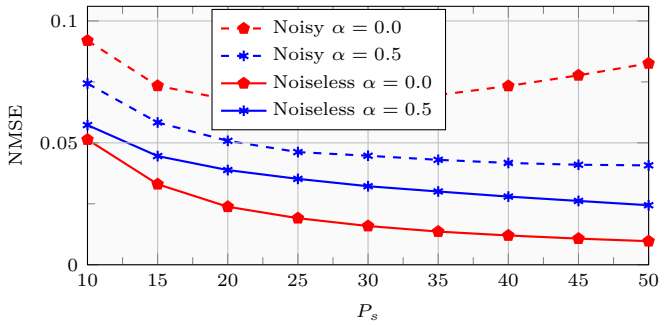


Figure 2.9: NMSE vs. P_s for noisy and noiseless observations.

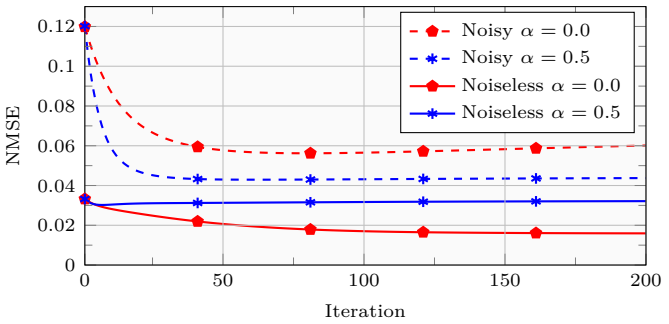


Figure 2.10: NMSE vs. iterations with $P_s = 30\%$ for noisy and noiseless observations.

error as more observations and noise are added.

Fig. 2.10 shows the evolution of the NMSE for $P_s = 30\%$ for the first 200 iterations for noisy and noiseless observations with different parameter values. We observe that the starting point of the PG with noiseless observations is fairly close to the minimum since the initial estimate calculated with (2.42) has a very low NMSE of 0.033. For comparison, if we initialize to $\boldsymbol{M}$ the NMSE is 0.699 and it takes 400 iterations to converge with $\alpha = 0$. For $\alpha = 0.5$, the PG converges to a point close to the initial error. In the noisy case, we observe a smaller error at convergence for $\alpha = 0.5$ than for $\alpha = 0$, which converges to NMSE $= 0.069$ (see Fig. 2.9), since the graph regularization helps filter out some of the noise. Moreover, the regularization also increases the convergence speed with noisy observations since the PG with $\alpha = 0$ needs 1000 iterations to converge at $P_s = 30\%$.

2.5 Conclusions

In this chapter we have addressed the completion of partially observed matrices of graph signals and the convergence speed of the proximal gradient algorithm. We have first proposed

two algorithms: SPG, which switches to a lower regularization parameter whenever the proximal gradient moves aways from the optimal solution, and VPG, a simpler implementation that reduces the regularization parameter at each iteration. Second, we have proposed an initialization method for the proximal gradient algorithms that makes use of the weighted adjacency matrix of the graph to calculate a starting point closer to the solution. We have shown in simulations with the synthetic and real datasets that the proposed initialization method always reduces the number of iterations required for convergence, and that the SPG and VPG algorithms are faster than the conventional PG and FPC algorithms when initialized using this method. Moreover, this initialization can be useful for any iterative matrix completion algorithm, even those with an already fast convergence since they tend to rely on more complicated and computationally expensive operations to achieve higher speeds. We have also studied the performance of the Laplacian-regularized PG. From the simulations we can conclude that, if the random sampling is able to capture the underlying structure of the dataset and μ is small enough, the Laplacian regularization does not reduce the recovery error when the observations are noiseless. On the other hand, with noisy observations the regularization improves the error for datasets which are smooth on the graph since it helps filter out the noise, prevents the overfitting to the observed noisy entries, and it also reduces the iterations to convergence.

3

Matrix completion via kernel regression

Existing MC approaches rely on some form of rank minimization or low-rank matrix factorization. Specifically, [13] proves that when MC is formulated as the minimization of the nuclear norm subject to the constraint that the observed entries remain unchanged, exact recovery is possible under mild assumptions. Alternatively, [10] replaces the nuclear norm by a product of two low-rank factor matrices that are identified in order to recover the complete matrix. These are the two most widespread approaches, which yield good results in general situations where there is little extra knowledge about the unknown matrix.

As introduced in Chapter 1, while the low-rank assumption can be sufficient for reliable recovery, prior information about the unknown matrix can be also accounted for to improve the completion outcome. Forms of prior information can include sparsity [17], local smoothness [18], and interdependencies encoded by graphs [20, 21, 47, 50]. These approaches exploit the available similarity information or prior knowledge of the bases spanning the column or row spaces of the unknown matrix. In this regard, reproducing kernel Hilbert spaces (RKHSs) constitute a powerful tool for leveraging available prior information thanks to the kernel functions, which measure the similarity between pairs of points in an input set. Prompted by this, [22, 24, 51, 52] postulate that columns of the factor matrices belong to a pair of RKHSs spanned by their respective kernels. In doing so, a given structure or similarity between rows or columns is effected on the recovered matrix. Upon choosing a suitable kernel function, [17] as well as [18, 20, 21, 47, 50] can be cast into the RKHS framework. In addition to improving MC performance, kernel-based approaches also enable extrapolation of rows and columns, even when all their entries are missing - a task impossible by the standard MC approaches

in e.g. [13] and [10] relying on the basic formulation.

One major hurdle in MC is the computational cost as the matrix size grows. In its formulation as a rank minimization task, MC can be solved via semidefinite programming [13], or proximal gradient minimization [20,26,27,41], which entails a singular value decomposition of the recovered matrix per iteration. Instead, algorithms with lower computational cost are available for the bi-convex formulation based on matrix factorization [10]. These commonly rely on iterative minimization schemes such as alternating least-squares (ALS) [40, 53] or stochastic gradient descent (SGD) [24,54]. With regard to kernel-based MC, the corresponding algorithms rely on alternating convex minimization and semidefinite programming [51], block coordinate descent [22], and SGD [24]. However, algorithms based on alternating minimization only converge to the minimum after infinite iterations. In addition, existing kernel-based algorithms adopt a specific sampling pattern, as in [52] where the observations are arranged in grid fashion, or do not effectively make use of the Representer Theorem for RKHSs that will turn out to be valuable in further reducing the complexity, especially when the number of observed entries is small.

This chapter introduces an RKHS-based approach to matrix completion and extrapolation (MCEX) that also unifies and broadens the scope of MC approaches, while offering reduced complexity algorithms that scale well with the matrix size. Specifically, a novel MC solver via kernel ridge regression as a convex alternative to the nonconvex factorization-based formulation that offers a closed-form solution is presented. Through an explicit sampling matrix, the proposed method offers an encompassing sampling pattern, which further enables the derivation of upper bounds on the mean-square error. Moreover, an approximate solution to the MCEX regression formulation that also enables online implementation using SGD is developed . Finally, means of incorporating prior information through kernels are discussed in the RKHS framework.

The rest of the chapter is organized as follows. Section 3.1 outlines the RKHS formulation and the kernel regression task. Section 3.2 unifies the existing methods for MC under the RKHS umbrella, while Section 3.3 introduces our proposed Kronecker kernel MCEX (KKMCEX) approach. Section 3.4 develops our ridge regression MCEX (RRMCEX) algorithm, an accelerated version of KKMCEX, and its online variant. Section 3.5 deals with the construction of kernel matrices. Finally, Section 4.3 presents numerical tests, and Section 3.7 concludes the chapter.

3.1 Kernel regression and reproducing kernel Hilbert spaces

Consider a set of s input-observation pairs $\{(\bar{x}_i, m_i)\}_{i=1}^s$ in $\mathcal{X} \times \mathbb{R}$, where $\mathcal{X}$ is the input set of cardinality $|\mathcal{X}| = N$, $\bar{\mathcal{X}} := \{\bar{x}_1, \ldots, \bar{x}_s\} \subseteq \mathcal{X}$ contains the sampled inputs, and measurements obey the model

$$m_i = f(\bar{x}_i) + e_i \tag{3.1}$$

where $f : \mathcal{X} \to \mathbb{R}$ is an unknown function and $e_i \in \mathbb{R}$ is noise. In pattern recognition, the objective is to learn the function f so that the output $f(x)$ for points $x \notin \bar{\mathcal{X}}$ can be predicted. The variable x can be any entity susceptible of mathematical representation as an element within a set. For instance, in the user-movie example x_i could be the ith user and m_i the received average rating across all movies, with $f(x_i)$ being the true average prior to the indecisiveness induced by e_i. There exist a myriad of methods to learn f from the observations, such as linear regression, kernel regression, support vector machines, or neural networks [4]. These methods fall within the so-called supervised learning techniques: we are given a set of inputs each with an associated measurement or label. In unsupervised learning, on the other hand, we are only given the inputs and none of the labels; this category applies to, e.g., clustering. As an in-between category, semisupervised learning deals with a set containing inputs with associated measurement, and single inputs with no associated measurement.

Adapting the standard MC formulation (2.15) to the notation in (3.1), the input set is $\mathcal{X} := \{1, \ldots, N\} \times \{1, \ldots, L\}$, the training set with associated observations is $\bar{\mathcal{X}} = \Omega$, and the matrix itself is a mapping $\mathcal{X} \to \mathbb{R}$ such that each index pair (i, j) represents an input point, and the indexed entry the output $f(i, j) = \boldsymbol{F}_{i,j}$. Then, MC as explained in Chapter 2 falls within the semisupervised learning class since obtaining a solution involves the whole matrix, including zero-valued entries. Nevertheless, MC can also be implemented as a supervised learning problem, i.e., the learning stage uses only labelled inputs, by means of the kernel-based, and factorization-based approaches that will be detailed in this chapter. This section lays the foundations for kernel ridge regression (KRR) and reproducing kernel Hilbert spaces (RKHS).

3.1.1 Nonlinear kernel regression

Given an input set $\mathcal{X} \subseteq \mathbb{R}^D$ with $|\mathcal{X}| \leq N$, in linear regression applications the evaluation of $f : \mathcal{X} \to \mathbb{R}$ is a linear combination of the elements of the input vector weighted by some coefficients $\boldsymbol{u}$, i.e., $f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{u}$. With a set of input-observation pairs $\{(\bar{\boldsymbol{x}}_i, m_i)\}_{i=1}^s$, where

the inputs constitute the set $\bar{\mathcal{X}} = \{\bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_s\}$, these can be represented in compact form as

$$\boldsymbol{m} = \bar{\boldsymbol{X}}^T \boldsymbol{u} + \boldsymbol{e} \tag{3.2}$$

where $\boldsymbol{m} := [m_1, \dots, m_s]$, $\bar{\boldsymbol{X}} := [\bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_s] \in \mathbb{R}^{D \times s}$ and $\boldsymbol{e} := [e_1, \dots, e_s]$. The coefficient vector $\boldsymbol{u}$ in (3.2) can be found by solving the ridge regression (RR) problem

$$\arg\min_{\boldsymbol{u} \in \mathbb{R}^D} \|\boldsymbol{m} - \bar{\boldsymbol{X}}^T \boldsymbol{u}\|_2^2 + \mu \|\boldsymbol{u}\|_2^2 \tag{3.3}$$

which gives the estimate $\hat{\boldsymbol{u}} = \bar{\boldsymbol{X}}(\bar{\boldsymbol{X}}^T \bar{\boldsymbol{X}} + \mu \boldsymbol{I})^{-1} \boldsymbol{m}$. The RR solution approximates $\boldsymbol{f} := [f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_N)]$ with $\hat{\boldsymbol{f}} = \boldsymbol{X}^T \hat{\boldsymbol{u}}$, which is a linear combination of the columns in $\boldsymbol{X}^T = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_N]^T \in \mathbb{R}^{N \times D}$. The quality of the solution is impacted by the level of noise and by the distance from $\boldsymbol{f}$ to the space spanned by the columns of $\boldsymbol{X}^T$. Moreover, another possible impediment is a wrongful model assumption in (3.2) when the relationship between $\boldsymbol{f}$ and the columns of $\boldsymbol{X}^T$ is nonlinear, i.e., the linear model $f(\boldsymbol{x}_i) = \boldsymbol{x}_i^T \boldsymbol{u}\ \forall \boldsymbol{x}_i \in \mathcal{X}$ is not accurate.

A method to overcome nonlinearities in regression is to map the regressors $\boldsymbol{x}_i$ onto a higher dimensional space where $\boldsymbol{f}$ can be obtained as a linear combination of a basis in this extended space. Let $\phi : \mathcal{X} \to \mathbb{R}^{D_\phi}$, where $D_\phi > 0$, be a feature map lifting elements in $\mathcal{X}$ into a higher dimensional space and inducing the model $f(\boldsymbol{x}) = \phi(\boldsymbol{x})^T \boldsymbol{u}$, where $\boldsymbol{u}$ now has dimension D_ϕ. Moreover, let $\boldsymbol{\Phi} := [\phi(\bar{\boldsymbol{x}}_1), \dots, \phi(\bar{\boldsymbol{x}}_s)] \in \mathbb{R}^{D_\phi \times s}$. The feature map turns (3.3) into

$$\arg\min_{\boldsymbol{u} \in \mathbb{R}^{D_\phi}} \|\boldsymbol{m} - \boldsymbol{\Phi}^T \boldsymbol{u}\|_2^2 + \mu \|\boldsymbol{u}\|_2^2 \tag{3.4}$$

which yields $\hat{\boldsymbol{u}} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \mu \boldsymbol{I})^{-1} \boldsymbol{m}$ and $\hat{f}(\boldsymbol{x}) = \phi(\boldsymbol{x})^T \hat{\boldsymbol{u}}\ \forall \boldsymbol{x} \in \mathcal{X}$. With an appropriate feature map choice, solving (3.4) in $\mathbb{R}^{D_\phi}$ yields a more accurate coefficient vector. See Fig. 3.1 for an example application. While the feature map might improve the RR solution, it can result in huge computational cost. Since D_ϕ can be very large or even infinite, the cost of calculating the products $\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$ to obtain $\boldsymbol{\Phi}^T \boldsymbol{\Phi}$ can become prohibitive. Fortunately, some feature maps allow to completely bypass the computation of the vector products and obtain their values via a cheaper function.

Given ϕ, let us define the function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle \tag{3.5}$$

which evaluates the inner product between two elements in $\mathbb{R}^{D_\phi}$. This κ is known as kernel function, and the process of mapping $\boldsymbol{x}$ to a higher dimensional space with ϕ and evaluating the inner product with (3.5) is known as kernel trick. Since $\hat{\boldsymbol{u}} \in \text{span}(\boldsymbol{\Phi})$, making the variable

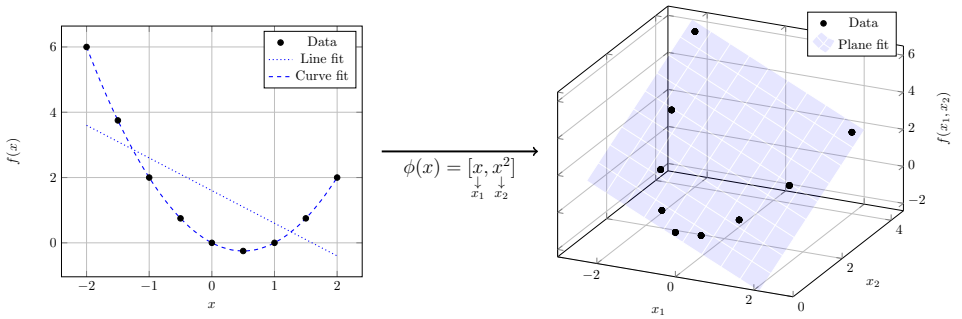


Figure 3.1: 2D linear and polynomial regression for $f(x) = x^2 - x$ vs. plane fitting on points lifted to 3D space. The polynomial regression can be interpreted as mapping x to $\mathbb{R}^2$ via $\phi(x) = [x, x^2]$ and obtaining the ridge regression solution, which yields $\hat{\boldsymbol{u}} = [-1, 1]$ and $\hat{f}(x_1, x_2) = x_2 - x_1$ for $\mu = 0$.

change $\boldsymbol{u} := \boldsymbol{\Phi}\bar{\boldsymbol{\alpha}}$ the RR in (3.4) becomes

$$\arg\min_{\bar{\boldsymbol{\alpha}} \in \mathbb{R}^s} \|\boldsymbol{m} - \boldsymbol{\Phi}^T \boldsymbol{\Phi}\bar{\boldsymbol{\alpha}}\|_2^2 + \mu \|\boldsymbol{\Phi}\bar{\boldsymbol{\alpha}}\|_2^2. \tag{3.6}$$

Then, taking advantage of (3.5), (3.6) can be rewritten as

$$\arg\min_{\bar{\boldsymbol{\alpha}} \in \mathbb{R}^s} \|\boldsymbol{m} - \bar{\boldsymbol{K}}\bar{\boldsymbol{\alpha}}\|_2^2 + \mu \bar{\boldsymbol{\alpha}}^T \bar{\boldsymbol{K}}\bar{\boldsymbol{\alpha}} \tag{3.7}$$

where $\bar{\boldsymbol{K}}$ is $s \times s$ with entries $\bar{\boldsymbol{K}}_{i,j} = \kappa(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_j)$. The resulting problem (3.7) is known as kernel ridge regression (KRR), and it is the dual problem [55] of RR in (3.4). Indeed, the solution to (3.7) is $\hat{\bar{\boldsymbol{\alpha}}} = (\bar{\boldsymbol{K}} + \mu \boldsymbol{I})^{-1}\boldsymbol{m}$, and multiplying $\boldsymbol{\Phi}\hat{\bar{\boldsymbol{\alpha}}}$ yields $\hat{\boldsymbol{u}}$ from (3.4). One advantage granted by KRR is that it is not necessary to know the map ϕ since (3.7) only requires knowledge of κ. There exist many possible kernel functions, the choice of which depends on the data in $\mathcal{X}$. A commonly used one is the Gaussian kernel, detailed in the example below.

Example: Gaussian kernel. Assuming an input set $\mathcal{X} \subseteq \mathbb{R}^D$, the Gaussian kernel function is defined as

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\eta \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2\right) \tag{3.8}$$

with η being a free parameter. Actually, this kernel evaluates an inner product in an infinite dimensional space as shown next. Given $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^D$, (3.8) can be expanded as

$$\exp\left(-\eta \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2\right) = \exp\left(-\eta \|\boldsymbol{x}_i\|_2^2\right) \exp\left(-\eta \|\boldsymbol{x}_j\|_2^2\right) \exp\left(-2\eta \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle\right). \tag{3.9}$$

Taking the Taylor series expansion of the last term [56], we have

$$\exp\left(-2\eta\langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle\right) = \sum_{l=0}^{\infty} \frac{1}{l!}(2\eta\langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle)^l = \sum_{l=0}^{\infty} \frac{1}{l!}\left(2\eta \sum_{d=1}^{D} \boldsymbol{x}_i(d)\boldsymbol{x}_j(d)\right)^l$$
$$= \sum_{l=0}^{\infty} \sum_{\sum n_d = l} \binom{l}{n_1, \ldots, n_D} \left(\sqrt{\frac{(2\eta)^l}{l!}} \boldsymbol{x}_i(1)^{n_1} \cdots \boldsymbol{x}_i(D)^{n_D}\right) \left(\sqrt{\frac{(2\eta)^l}{l!}} \boldsymbol{x}_j(1)^{n_1} \cdots \boldsymbol{x}_j(D)^{n_D}\right) \tag{3.10}$$

where the last equality uses the multinomial theorem, and the subindex $\sum n_d = l$ iterates over all D-tuples with positive integer elements $(n_1, \ldots, n_D)$ that add up to l. Substituting (3.10) into (3.9), the Gaussian kernel is

$$\begin{aligned} \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = &\left(\exp\left(-\eta \left\|\boldsymbol{x}_i\right\|_2^2\right) \sum_{l=0}^{\infty} \sqrt{(2\eta)^l} \sum_{\sum n_d = l} \prod_{d=1}^{D} \frac{\boldsymbol{x}_i(d)^{n_d}}{\sqrt{n_d!}}\right) \\ &\left(\exp\left(-\eta \left\|\boldsymbol{x}_j\right\|_2^2\right) \sum_{l=0}^{\infty} \sqrt{(2\eta)^l} \sum_{\sum n_d = l} \prod_{d=1}^{D} \frac{\boldsymbol{x}_j(d)^{n_d}}{\sqrt{n_d!}}\right) \\ &= \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)\rangle \end{aligned} \tag{3.11}$$

where the feature map $\phi : \mathbb{R}^D \to \mathbb{R}^\infty$ is defined as

$$\phi(\boldsymbol{x}_i) = \exp\left(-\eta \left\|\boldsymbol{x}_i\right\|_2^2\right) \left[\sqrt{(2\eta)^l} \prod_{d=1}^{D} \frac{\boldsymbol{x}_i(d)^{n_d}}{\sqrt{n_d!}}\right]_{l=0,\ldots\infty, \sum_{d=1}^{l} n_d = l} \tag{3.12}$$

□

The derivation of the Gaussian kernel highlights how much simpler it can be to evaluate $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ than to perform the mapping into a feature space and take the inner product. Moreover, as evidenced by the Gaussian kernel, kernel functions can also be thought of as a measure of similarity between items in a set. For MC, this premise provides useful tools to encode prior information about the columns and rows of the matrix, and facilitates efficient algorithms based on kernel regression.

3.1.2 Kernel regression in reproducing kernel Hilbert spaces

While KRR has been first introduced as a way to perform nonlinear regression efficiently, it is a problem that stands on its own in trying to learn f from a measurement set, as is depicted in Fig. 3.2. This section introduces KRR from a more formal perspective using the

reproducing kernel Hilbert space (RKHS) framework.

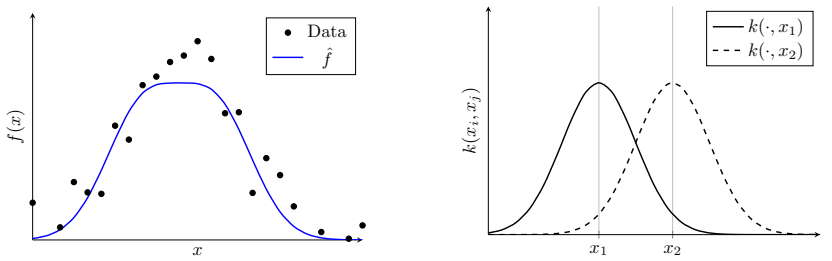


Figure 3.2: KRR (left) using the Gaussian kernel. The right figure shows the Gaussian kernel centered at two different observations $x_1, x_2 \in \mathbb{R}$.

Assuming that f in (3.1) satisfies $f \in \mathcal{H}_f$, where $\mathcal{H}_f$ is a RKHS, KRR seeks to obtain

$$\hat{f} = \underset{f \in \mathcal{H}_f}{\arg\min} \frac{1}{s} \sum_{i=1}^{s} (m_i - f(\bar{x}_i))^2 + \mu' \|f\|^2_{\mathcal{H}_f} \tag{3.13}$$

where $\mu' \in \mathbb{R}^+$ is the regularization parameter, and $\|f\|_{\mathcal{H}_f}$ the norm in $\mathcal{H}_f$. Note the factor $\frac{1}{s}$, which keeps the balance between the loss and regularization terms as new observations are added and s increases. Let us define the RKHS as

$$\mathcal{H}_f := \{f : f(x) = \sum_{i=1}^{N} \alpha_i \kappa_f(x, x_i), \quad \alpha_i \in \mathbb{R}\} \tag{3.14}$$

where $\{\alpha_i\}_{i=1}^N$ are weight coefficients, and $\kappa_f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the kernel function that spans $\mathcal{H}_f$. The kernel can also be defined through the map $\phi_x : \mathcal{X} \to \mathcal{C}$, where $\mathcal{C}$ is a feature space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{C}} : \mathcal{C} \times \mathcal{C} \to \mathbb{R}$, such that

$$\kappa_f(x_i, x_j) = \langle \phi_x(x_i), \phi_x(x_j) \rangle_{\mathcal{C}} \tag{3.15}$$

For an input set of text files, for example, the files could be mapped to a feature vector that tracks the number of words, lines, and blank spaces in the file. The norm in $\mathcal{H}_f$ is $\|f\|_{\mathcal{H}_f} := \langle f, f \rangle_{\mathcal{H}_f}$, where $\langle f, f \rangle_{\mathcal{H}_f}$ is the inner product in $\mathcal{H}_f$. With $\{\alpha_i\}_{i=1}^N$ and $\{\alpha'_i\}_{i=1}^N$ denoting the coefficients of $f, f' \in \mathcal{H}_f$, respectively, we define $\langle f, f' \rangle_{\mathcal{H}_f} := \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha'_j \kappa_x(x_i, x_j)$. Moreover, given the kernel matrix $\boldsymbol{K}_f \in \mathbb{R}^{N \times N}$ with $(\boldsymbol{K}_f)_{i,j} = \kappa_f(x_i, x_j)$, we have

$$\langle f, f' \rangle_{\mathcal{H}_f} = \boldsymbol{\alpha}^T \boldsymbol{K}_f \boldsymbol{\alpha}' \tag{3.16}$$

where $\boldsymbol{\alpha} := [\alpha_1, \ldots, \alpha_N]^T$ and $\boldsymbol{\alpha}' := [\alpha'_1, \ldots, \alpha'_N]^T$. RKHSs are complete and separable linear spaces endowed with an inner product that satisfies the reproducing property [25]. The

significance of these properties is explained as follows.

A space is complete if every Cauchy sequence in it converges to a point within the space. A Cauchy sequence is a sequence $f_1, f_2, f_3, \dots$ such that for any $\epsilon \in \mathbb{R}^+$ there exists a positive integer c_d such that the distance $\|f_i - f_j\|_{\mathcal{H}_f} < \epsilon$ for all $i, j > c_d$; this implies that the terms of the sequence get closer as it advances, and that it converges to a point in the space. One example of non complete space is the set of rational numbers $\mathbb{Q}$: there exists an infinite sequence within $\mathbb{Q}$ that approaches $\sqrt{2}$, but its convergence point is outside $\mathbb{Q}$. A space is separable if it contains a dense countable subset, i.e., there exists a countable subset $\mathcal{H}'_f \subseteq \mathcal{H}_f$ such that, for all $f \in \mathcal{H}_f$ and $\epsilon > 0$, there exists $f' \in \mathcal{H}'_f$ that satisfies $\|f - f'\| < \epsilon$. For instance, the set $\mathbb{Q}$ is a countable dense subset of $\mathbb{R}$ since 1) its elements can be ordered with one-to one correspondence with the set of natural numbers $\mathbb{N}$, i.e., it is countable, and 2) for any number in $\mathbb{R}$ there is a number in $\mathbb{Q}$ arbitrarily close to it, i.e., it is dense. The properties of completeness and separability ensure that $\mathcal{H}_f$ is isomorphic to $\mathbb{R}^N$ for $N < \infty$ [25].

The reproducing trait of $\mathcal{H}_f$ is granted by its inner product, and it states that $f(x) = \langle f, \kappa_f(\cdot, x)\rangle_{\mathcal{H}_f}$; that is, f in $\mathcal{H}_f$ can be evaluated at any $x \in \mathcal{X}$ by taking the inner product between f and $\kappa_f(\cdot, x)$. This also applies to the kernel function, where $\kappa_f(x_i, x_j) = \langle \kappa_f(\cdot, x_i), \kappa_f(\cdot, x_j)\rangle_{\mathcal{H}_f}$. In order for $\langle \cdot, \cdot \rangle_{\mathcal{H}_f}$ in (3.16) to be an inner product, κ_f must be symmetric and positive semidefinite, meaning $\langle f, f\rangle_{\mathcal{H}_f} \geq 0 \, \forall f \in \mathcal{H}_f$. As a consequence, $\boldsymbol{K}_f$ will be symmetric positive semidefinite since $\boldsymbol{\alpha}^T \boldsymbol{K}_f \boldsymbol{\alpha} \geq 0 \ \forall \boldsymbol{\alpha} \in \mathbb{R}^N$.

Using $\boldsymbol{K}_f$, consider without loss of generality expressing the vector $\boldsymbol{f} := [f(x_1), \dots, f(x_N)]^T$ as $\boldsymbol{f} = \boldsymbol{K}_f \boldsymbol{\alpha}$, where $\boldsymbol{\alpha} := [\alpha_1, \dots, \alpha_N]^T$. Then, KRR (3.13) boils down to solving

$$\arg\min_{f \in \mathcal{H}_f} \sum_{i=1}^{s} (m_i - \langle f, \kappa_f(\cdot, \bar{x}_i)\rangle_{\mathcal{H}_f})^2 + \mu \|f\|^2_{\mathcal{H}_f}. \tag{3.17}$$

For $\boldsymbol{K}_f$ invertible, $\boldsymbol{f}$ is estimated as

$$\hat{\boldsymbol{f}} = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \|\boldsymbol{m} - \boldsymbol{S}\boldsymbol{f}\|_2^2 + \mu \, \boldsymbol{f}^T \boldsymbol{K}_f^{-1} \boldsymbol{f} \tag{3.18}$$

where $\boldsymbol{S}$ is a $s \times N$ binary sampling matrix with a single nonzero element per row equal to 1. Moreover, the coefficient vector estimate is

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \|\boldsymbol{m} - \boldsymbol{S}\boldsymbol{K}_f \boldsymbol{\alpha}\|_2^2 + \mu \, \boldsymbol{\alpha}^T \boldsymbol{K}_f \boldsymbol{\alpha}. \tag{3.19}$$

The Representer Theorem, detailed in Appendix B.1, states that the ith component of $\hat{\boldsymbol{\alpha}}$ in (3.19) is equal to 0 if $x_i \notin \bar{\mathcal{X}}$; this means that $\hat{f} \in \text{span}\{k(\cdot, \bar{x}_i)\}_{i=1}^s$. Therefore, one only needs to optimize for the coefficients associated with the inputs $\bar{x}_i \in \bar{\mathcal{X}}$. Doing so leads to

the estimator

$$\hat{\bar{\boldsymbol{\alpha}}} = \underset{\bar{\boldsymbol{\alpha}} \in \mathbb{R}^s}{\arg\min} \|\boldsymbol{m} - \boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T\bar{\boldsymbol{\alpha}}\|_2^2 + \mu\bar{\boldsymbol{\alpha}}^T\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T\bar{\boldsymbol{\alpha}} \tag{3.20}$$

which upon setting $\bar{\boldsymbol{K}} = \boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T$ is equal to what was obtained in the nonlinear RR problem (3.7). Then, $\boldsymbol{f}$ can be recovered through $\boldsymbol{f} = \boldsymbol{K}_f\boldsymbol{S}^T\hat{\bar{\boldsymbol{\alpha}}}$.

Due to their interpretability as a similarity function, kernel functions give rise to the notion of smoothness in RKHSs: if $\kappa_f(x_i, x_j)$ is large, then $f(x_i)$ and $f(x_j)$ are expected to have similar values. This is akin to what was is explained in Appendix A for graph signal processing, where the similarity was indicated by a graph. Indeed, the Laplacian quadratic form is not unlike the regularization term in (3.19), which promotes smoothness and similarity according to k_f and prevents overfitting to the observations. Therefore, we will leverage the smoothness assumption in Section 3.3 to extrapolate the missing values in MC with an algorithm based on KRR.

3.2 Kernel-based matrix completion

This section presents an overview of factorization and kernel-based approaches to MC that exist in the literature. MC considers $\boldsymbol{F} \in \mathbb{R}^{N\times L}$ of rank r observed through an $N \times L$ matrix of noisy observations

$$\boldsymbol{M} = P_\Omega(\boldsymbol{F} + \boldsymbol{E}) \tag{3.21}$$

where $\Omega \subseteq \{1, \ldots, N\} \times \{1, \ldots, L\}$ is the sampling set of cardinality $s = |\Omega|$ containing the indices of the observed entries; $P_\Omega(\cdot)$ is a projection operator that sets to zero the entries with index $(i, j) \notin \Omega$ and leaves the rest unchanged; and, $\boldsymbol{E} \in \mathbb{R}^{N\times L}$ is a noise matrix. According to [36], one can recover a low-rank $\boldsymbol{F}$ from $\boldsymbol{M}$ by solving the following problem:

$$\min_{\boldsymbol{F}\in\mathbb{R}^{N\times L}} \|P_\Omega(\boldsymbol{M} - \boldsymbol{F})\|_{\mathrm{F}}^2 + \mu \|\boldsymbol{F}\|_* . \tag{3.22}$$

Because $\boldsymbol{F}$ is low rank, it is always possible to factorize it as $\boldsymbol{F} = \boldsymbol{W}\boldsymbol{H}^T$ as shown in Fig. 3.3, where $\boldsymbol{W} \in \mathbb{R}^{N\times p}$ and $\boldsymbol{H} \in \mathbb{R}^{L\times p}$ are the latent factor matrices with $p \geq r$. This factorization allows expressing the nuclear norm as [57] $\|\boldsymbol{F}\|_* = \min_{\boldsymbol{F}=\boldsymbol{W}\boldsymbol{H}^T} \frac{1}{2}\left(\|\boldsymbol{W}\|_{\mathrm{F}}^2 + \|\boldsymbol{H}\|_{\mathrm{F}}^2\right)$, which allows reformulating (3.22) as

$$\{\hat{\boldsymbol{W}}, \hat{\boldsymbol{H}}\} = \underset{\substack{\boldsymbol{W}\in\mathbb{R}^{N\times p}\\ \boldsymbol{H}\in\mathbb{R}^{L\times p}}}{\arg\min} \left\|P_\Omega(\boldsymbol{M} - \boldsymbol{W}\boldsymbol{H}^T)\right\|_{\mathrm{F}}^2 + \frac{\mu}{2}\left(\|\boldsymbol{W}\|_{\mathrm{F}}^2 + \|\boldsymbol{H}\|_{\mathrm{F}}^2\right) \tag{3.23}$$

and yields $\hat{\boldsymbol{F}} = \hat{\boldsymbol{W}}\hat{\boldsymbol{H}}^T$. While (3.22) and (3.23) yield the same $\hat{\boldsymbol{F}}$ when the rank of the matrix minimizing (3.22) is smaller than p [40], solving (3.22) can be costlier since it involves

the computation of the singular values of the matrix. On the other hand, since (3.23) is bi-convex it can be solved by alternately optimizing $\boldsymbol{W}$ and $\boldsymbol{H}$, e.g. via ALS [53] or SGD iterations [54]. Moreover, leveraging the structure of (3.23), it is also possible to optimize one row from each factor matrix at a time instead of updating the full factor matrices, which enables faster and also online and distributed implementations [58]. Still, one important drawback of (3.23) is that it is unable to provide an estimate for columns or rows with zero observations since an all-zero column or row in $\boldsymbol{M}$ would have a corresponding all-zero coefficient row in $\hat{\boldsymbol{H}}$ or $\hat{\boldsymbol{W}}$, respectively. This issue can be overcome by incorporating prior information in MC through kernels.

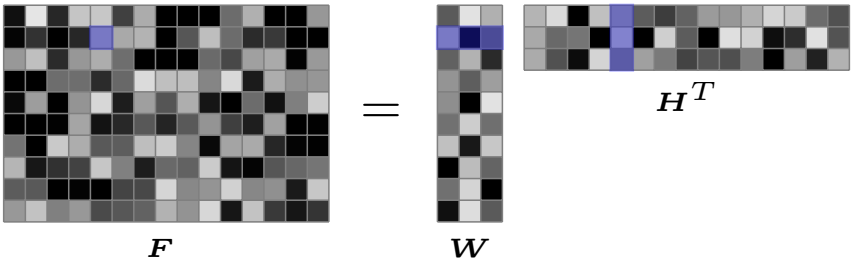


Figure 3.3: Factorization of a rank 3 matrix.

Aiming at a kernel-based matrix completion (KMC), we model the columns and rows of $\boldsymbol{F}$ as functions that belong to two different RKHSs. To this end, consider the input sets $\mathcal{X} := \{x_1, \ldots, x_N\}$ and $\mathcal{Y} := \{y_1, \ldots, y_L\}$ for the column and row functions, respectively. The content of each space depends on the application; in the user-movie ratings paradigm, $\mathcal{X}$ could be the set of users and $\mathcal{Y}$ the set of movies. Nevertheless, one can always resort to the basic setting with $\mathcal{X}$ and $\mathcal{Y}$ denoting the index sets $\{1, \ldots, N\}$ and $\{1, \ldots, L\}$, respectively. Then, $\boldsymbol{F} := [\boldsymbol{f}_1, \ldots, \boldsymbol{f}_L]$ is formed with columns $\boldsymbol{f}_l := [f_l(x_1), \ldots, f_l(x_N)]^T$ with $f_l : \mathcal{X} \to \mathbb{R}$. Likewise, we rewrite $\boldsymbol{F} := [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_N]^T$, with rows $\boldsymbol{g}_n := [g_n(y_1), \ldots, g_n(y_L)]^T$ and $g_n : \mathcal{Y} \to \mathbb{R}$. We further assume that $f_l \in \mathcal{H}_w \; \forall l = 1, \ldots, L$ and $g_n \in \mathcal{H}_h \; \forall n = 1, \ldots, N$, where

$$\mathcal{H}_w := \{w : w(x) = \sum_{n=1}^{N} \alpha_n \kappa_w(x, x_n), \quad \alpha_n \in \mathbb{R}\} \tag{3.24}$$

$$\mathcal{H}_h := \{h : h(y) = \sum_{l=1}^{L} \beta_l \, \kappa_h(y, y_l), \quad \beta_l \in \mathbb{R}\} \tag{3.25}$$

and $\kappa_w : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $\kappa_h : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ are the kernels forming $\boldsymbol{K}_w \in \mathbb{R}^{N \times N}$ and $\boldsymbol{K}_h \in \mathbb{R}^{L \times L}$, respectively.

Since $\boldsymbol{W}$ and $\boldsymbol{H}$ span the column and row spaces of $\boldsymbol{F}$, their columns belong to $\mathcal{H}_w$ and $\mathcal{H}_h$ as well. Thus, the m^{th} column of $\boldsymbol{W}$ is

$$\boldsymbol{w}_m := [w_m(x_1), \ldots, w_m(x_N)]^T \tag{3.26}$$

where $w_m : \mathcal{X} \to \mathbb{R}$ and $w_m \in \mathcal{H}_w \; \forall m = 1, \ldots, p$, and the m^{th} column of $\boldsymbol{H}$ is

$$\boldsymbol{h}_m := [h_m(y_1), \ldots, h_m(y_L)]^T \tag{3.27}$$

where $h_m : \mathcal{Y} \to \mathbb{R}$ and $h_m \in \mathcal{H}_h \; \forall m = 1, \ldots, p$. Hence, instead of simply promoting a small Frobenius norm for the factor matrices as in (3.23), we can also promote smoothness on their respective RKHS. The kernel-based formulation in [22] estimates the factor matrices by solving

$$\{\hat{\boldsymbol{W}}, \hat{\boldsymbol{H}}\} = \underset{\substack{\boldsymbol{W} \in \mathcal{H}_w \\ \boldsymbol{H} \in \mathcal{H}_h}}{\arg\min} \left\| P_\Omega(\boldsymbol{M} - \boldsymbol{W}\boldsymbol{H}^T) \right\|_{\text{F}}^2 + \mu \text{Tr}(\boldsymbol{W}^T \boldsymbol{K}_w^{-1} \boldsymbol{W}) + \mu \text{Tr}(\boldsymbol{H}^T \boldsymbol{K}_h^{-1} \boldsymbol{H}). \tag{3.28}$$

Note that (3.28) is equivalent to (3.23) for $\boldsymbol{K}_w = 2\boldsymbol{I}$ and $\boldsymbol{K}_h = 2\boldsymbol{I}$. Alternatively, we can instead find the coefficients that generate $\boldsymbol{W}$ and $\boldsymbol{H}$ in their respective RKHSs. Thus, if we expand $\boldsymbol{W} = \boldsymbol{K}_w \boldsymbol{B}$ and $\boldsymbol{H} = \boldsymbol{K}_h \boldsymbol{C}$, where $\boldsymbol{B} \in \mathbb{R}^{N \times p}$ and $\boldsymbol{C} \in \mathbb{R}^{L \times p}$ are coefficient matrices, (3.28) becomes

$$\{\hat{\boldsymbol{B}}, \hat{\boldsymbol{C}}\} = \underset{\substack{\boldsymbol{B} \in \mathbb{R}^{N \times p} \\ \boldsymbol{C} \in \mathbb{R}^{L \times p}}}{\arg\min} \left\| P_\Omega(\boldsymbol{M} - \boldsymbol{K}_w \boldsymbol{B}\boldsymbol{C}^T \boldsymbol{K}_h) \right\|_{\text{F}}^2 + \mu \text{Tr}(\boldsymbol{B}^T \boldsymbol{K}_w \boldsymbol{B}) + \mu \text{Tr}(\boldsymbol{C}^T \boldsymbol{K}_h \boldsymbol{C}). \tag{3.29}$$

Nevertheless, with nonsingular kernel matrices, $\boldsymbol{B}$ and $\boldsymbol{C}$ can be found by solving (3.28) with Algorithm 2[1] and substituting $\hat{\boldsymbol{B}} = \boldsymbol{K}_w^{-1} \hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{C}} = \boldsymbol{K}_h^{-1} \hat{\boldsymbol{H}}$ [22].

Alternating minimization schemes that solve the bi-linear MC formulation (3.23) tend to the solution to the convex problem (3.22) in the limit [53], thus convergence to the global optimum is not guaranteed unless the number of iterations is infinite. Since algorithms for kernel-based MC [22] solving (3.28) rely on such alternating minimization schemes, they lack convergence guarantees given finite iterations as well. In addition to that, their computational cost scales with the size of $\boldsymbol{F}$. On the other hand, online implementations have a lower cost [24], but only guarantee convergence to a stationary point [59]. The ensuing section develops a convex kernel-based reformulation of KMC that enables a closed-form solver which purely exploits the extrapolation facilitated by the kernels.

[1]The symbols Ω_n^w and Ω_l^h in Algorithm 2 denote, respectively, the indices of the observations in the nth row and lth column of $\boldsymbol{M}$.

Algorithm 2: Alternating least squares for kernel-based MC.

Input : $\boldsymbol{M}, \boldsymbol{K}_w, \boldsymbol{K}_h, \mu,$ iterations
Output : $\hat{\boldsymbol{W}}, \hat{\boldsymbol{H}}$
function ALS($\boldsymbol{M}, \boldsymbol{K}_w, \boldsymbol{K}_h, \mu,$ iterations)
for $t = 1, \ldots,$ *iterations* **do**
  for $n = 1, \ldots, N$ **do**
    $\boldsymbol{w}_n \leftarrow (\sum_{j \in \Omega_n^w} \boldsymbol{h}_j \boldsymbol{h}_j^T + \mu(\boldsymbol{K}_w^{-1})_{i,i}\boldsymbol{I})^{-1}(\sum_{j \in \Omega_n^w} m_{i,j}\boldsymbol{h}_j - \sum_{i \neq j}(\boldsymbol{K}_w^{-1})_{i,j}\boldsymbol{w}_j)$
  end
  for $l = 1, \ldots, L$ **do**
    $\boldsymbol{h}_l \leftarrow (\sum_{j \in \Omega_l^h} \boldsymbol{w}_i \boldsymbol{w}_i^T + \mu(\boldsymbol{K}_h^{-1})_{j,j}\boldsymbol{I})^{-1}(\sum_{i \in \Omega_l^h} m_{i,j}\boldsymbol{w}_i - \sum_{j \neq i}(\boldsymbol{K}_h^{-1})_{i,j}\boldsymbol{h}_i)$
  end
end
return $\hat{\boldsymbol{W}}, \hat{\boldsymbol{H}}$

3.3 Kronecker kernel MC and extrapolation

In the previous section, we viewed the columns and rows of $\boldsymbol{F}$ as functions evaluated at the points of the input sets $\mathcal{X}$ and $\mathcal{Y}$ in order to unify the state-of-the-art on MC using RKHSs. Instead, we now postulate entries of $\boldsymbol{F}$ as the output of a function lying on an RKHS evaluated at a tuple $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Given the spaces $\mathcal{X}$ and $\mathcal{Y}$, consider the space $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$ with cardinality $|\mathcal{Z}| = NL$ along with the two-dimensional function $f : \mathcal{Z} \to \mathbb{R}$ as $f(x_i, y_j) = \boldsymbol{F}_{i,j}$, which belongs to the RKHS

$$\mathcal{H}_f := \{f : f(x, y) = \sum_{n=1}^{N}\sum_{l=1}^{L} \gamma_{n,l}\kappa_f((x, y), (x_n, y_l)), \quad \gamma_{n,l} \in \mathbb{R}\} \tag{3.30}$$

with $\kappa_f : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$. While one may choose any kernel to span $\mathcal{H}_f$, we will construct one adhering to the bilinear factorization $\boldsymbol{F} = \boldsymbol{W}\boldsymbol{H}^T$ whose (i, j)th entry yields

$$\boldsymbol{F}_{ij} = f(x_i, y_j) = \sum_{m=1}^{p} w_m(x_i)h_m(y_j) \tag{3.31}$$

with w_m and h_m functions capturing m^{th} column vector of $\boldsymbol{W}$ and $\boldsymbol{H}$ as in (3.26) and (3.27). Since $w \in \mathcal{H}_w$ and $h \in \mathcal{H}_h$, we can write $w_m(x) = \sum_{n=1}^{N} b_{n,m}\kappa_w(x, x_n)$ and $h_m(y) = \sum_{l=1}^{L} c_{l,m}\kappa_h(y, y_l)$, where $b_{n,m}$ and $c_{l,m}$ are the entries at (n, m) and (l, m) of the factor

matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ from (3.29), respectively. Therefore, (3.31) can be rewritten as

$$\begin{aligned} f(x_i, y_j) &= \sum_{m=1}^{p} \sum_{n=1}^{N} b_{n,m} \kappa_w(x_i, x_n) \sum_{l=1}^{L} c_{l,m} \kappa_h(y_j, y_l) \\ &= \sum_{n=1}^{N} \sum_{l=1}^{L} \left(\sum_{m=1}^{p} b_{n,m} c_{l,m} \right) \kappa_w(x_i, x_n) \kappa_h(y_j, y_l) \\ &= \sum_{n=1}^{N} \sum_{l=1}^{L} \gamma_{n,l} \kappa_f((x_i, y_j), (x_n, y_l)) \end{aligned} \tag{3.32}$$

where $\gamma_{n,l} = \sum_{m=1}^{p} b_{m,n} c_{m,l}$, and $\kappa_f((x_i, y_j), (x_n, y_l)) = \kappa_w(x_i, x_n)\kappa_h(y_j, y_l)$ since a product of kernels is itself a kernel [37]. Using the latter, (3.32) can be written compactly as

$$f(x_i, y_j) = \boldsymbol{k}_{i,j}^T \boldsymbol{\gamma} \tag{3.33}$$

where $\boldsymbol{\gamma} := [\gamma_{1,1}, \gamma_{2,1}, \ldots, \gamma_{N,1}, \gamma_{1,2}, \gamma_{2,2}, \ldots, \gamma_{N,L}]^T$, and correspondingly,

$$\begin{aligned} \boldsymbol{k}_{i,j} =& [\kappa_w(x_i, x_1)\kappa_h(y_j, y_1), \ldots, \kappa_w(x_i, x_N)\kappa_h(y_j, y_1), \\ & \kappa_w(x_i, x_1)\kappa_h(y_j, y_2), \ldots, \kappa_w(x_i, x_N)\kappa_h(y_j, y_L)]^T \\ =& (\boldsymbol{K}_h)_{:,j} \otimes (\boldsymbol{K}_w)_{:,i} \end{aligned} \tag{3.34}$$

where a subscript $(:, j)$ denotes the j^{th} column of a matrix, and we have used that $\boldsymbol{K}_w$ and $\boldsymbol{K}_h$ are symmetric matrices. In accordance with (3.34), the kernel matrix of $\mathcal{H}_f$ in (3.30) is

$$\boldsymbol{K}_f = \boldsymbol{K}_h \otimes \boldsymbol{K}_w. \tag{3.35}$$

Clearly, $\boldsymbol{k}_{i,j}$ in (3.34) can also be expressed as $\boldsymbol{k}_{i,j} = (\boldsymbol{K}_f)_{:,(j-1)N+i}$. Moreover, note that the entries of the kernel matrix are $(\boldsymbol{K}_f)_{i',j'} = \kappa_w(x_i, x_n)\kappa_h(y_j, y_l)$, where $n = j' \bmod N, i = i' \bmod N, l = \lceil \frac{j'}{N} \rceil$, and $j = \lceil \frac{i'}{N} \rceil$. This together with (3.33), and recalling that $\boldsymbol{K}_f$ is symmetrical, implies that

$$\boldsymbol{f} = [f(x_1, y_1), f(x_2, y_1), \ldots, f(x_N, y_1), f(x_1, y_2), f(x_2, y_2), \ldots, f(x_N, y_N)]^T \tag{3.36}$$

can be expressed in matrix-vector form as

$$\boldsymbol{f} = \boldsymbol{K}_f \boldsymbol{\gamma} \tag{3.37}$$

or, equivalently, $\boldsymbol{f} = \text{vec}(\boldsymbol{F})$.

Since the eigenvalues of $\boldsymbol{K}_f$ are the product of eigenvalues of $\boldsymbol{K}_h$ and $\boldsymbol{K}_w$, it follows

that $\boldsymbol{K}_f$ is positive semidefinite and thus a valid kernel matrix. With the definition of the function f and its vector form we have transformed the matrix of functions specifying $\boldsymbol{F}$ into a function that lies on the RKHS $\mathcal{H}_f$. Hence, we are ready to formulate MC as a kernel regression task for recovering $\boldsymbol{f}$ from the observed entries of $\boldsymbol{m} = \text{vec}(\boldsymbol{M})$. Given $\{((x_i, y_j), m_{i,j})\}_{(i,j)\in\Omega}$ in $\mathcal{Z} \times \mathbb{R}$, the goal is to recover f as

$$\hat{f} = \underset{f\in\mathcal{H}_f}{\arg\min} \sum_{(i,j)\in\Omega} (m_{i,j} - f(x_i, y_j))^2 + \mu\|f\|^2_{\mathcal{H}_f} \tag{3.38}$$

where $\|f\|^2_{\mathcal{H}_f} := \boldsymbol{\gamma}^T \boldsymbol{K}_f \boldsymbol{\gamma}$. See Fig. 3.4 for an example depiction of the recovery of f from a few observations.

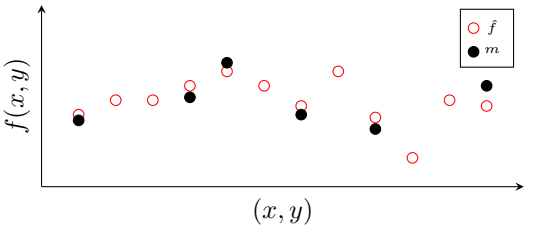


Figure 3.4: Recovery of the unobserved entries in $\boldsymbol{F}$ after formulating MC as a function estimation problem.

Define next $\boldsymbol{e} := \text{vec}(\boldsymbol{E})$ and $\bar{\boldsymbol{m}} := \boldsymbol{S}\boldsymbol{m}$, where $\boldsymbol{S}$ is an $S \times NL$ binary sampling matrix also used to specify the sampled noise vector $\bar{\boldsymbol{e}} := \boldsymbol{S}\boldsymbol{e}$. With these definitions and (3.37), the model in (3.21) becomes

$$\bar{\boldsymbol{m}} = \boldsymbol{S}\boldsymbol{f} + \boldsymbol{S}\boldsymbol{e} = \boldsymbol{S}\boldsymbol{K}_f\boldsymbol{\gamma} + \bar{\boldsymbol{e}} \tag{3.39}$$

which can be solved to obtain

$$\hat{\boldsymbol{\gamma}} = \underset{\boldsymbol{\gamma}\in\mathbb{R}^{NL}}{\arg\min} \|\bar{\boldsymbol{m}} - \boldsymbol{S}\boldsymbol{K}_f\boldsymbol{\gamma}\|_2^2 + \mu\boldsymbol{\gamma}^T\boldsymbol{K}_f\boldsymbol{\gamma} \tag{3.40}$$

in closed form

$$\hat{\boldsymbol{\gamma}} = (\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{K}_f + \mu\boldsymbol{I})^{-1}\boldsymbol{S}^T\bar{\boldsymbol{m}} \tag{3.41}$$

under the assumption that $\boldsymbol{K}_f$ is invertible. Then, we obtain the estimate $\hat{\boldsymbol{F}} = \text{unvec}(\boldsymbol{K}_f\hat{\boldsymbol{\gamma}})$. Since the size of $\boldsymbol{K}_f$ is $NL \times NL$, the inversion in (3.41) can be very computationally intensive. To alleviate this, we will leverage the Representer Theorem (see Appendix B.1 for a formal proof), which allows a reduction in the the number of degrees of freedom of the regression problem. In our setup, this theorem is as follows.

Theorem 3.1. ***Representer Theorem.*** *Given the set of input-observations pairs*

$\{(x_i, y_j), m_{i,j})\}_{(i,j)\in\Omega} \in \mathcal{Z} \times \mathbb{R}$ and the function f as in (3.32), the solution to

$$\arg\min_{f\in\mathcal{H}_f} \sum_{(i,j)\in\Omega} (m_{i,j} - f(x_i, y_j))^2 + \mu\|f\|^2_{\mathcal{H}_f} \tag{3.42}$$

is an estimate $\hat{f}$ that satisfies

$$\hat{f} = \sum_{(n,l)\in\Omega} \tau_{n,l}\kappa_f((\cdot,\cdot),(x_n, y_l)) \tag{3.43}$$

for some coefficients $\tau_{n,l} \in \mathbb{R}$, $\forall(n,l) \in \Omega$.

Theorem 3.1 asserts that $\hat{\boldsymbol{\gamma}}$ in (3.40) satisfies $\hat{\gamma}_{n,l} = 0 \; \forall \; (n,l) \notin \Omega$. Therefore, we only need to optimize $\{\gamma_{n,l} : (n,l) \in \Omega\}$ which correspond to the observed pairs. In fact, for our vector-based formulation, the Representer Theorem boils down to applying on (3.41) the matrix inversion lemma (MIL), which asserts the following.

Lemma 3.1. ***MIL*** *[60]. Given matrices $\boldsymbol{A}, \boldsymbol{U}$ and $\boldsymbol{V}$ of conformal dimensions, with $\boldsymbol{A}$ invertible, it holds that*

$$(\boldsymbol{U}\boldsymbol{V} + \boldsymbol{A})^{-1}\boldsymbol{U} = \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U} + \boldsymbol{I})^{-1}. \tag{3.44}$$

With (3.41) $\boldsymbol{A} = \mu\boldsymbol{I}, \boldsymbol{U} = \boldsymbol{S}^T$ and $\boldsymbol{V} = \boldsymbol{S}\boldsymbol{K}_f$, application of (3.44) to (3.41) yields

$$\hat{\boldsymbol{\gamma}} = \boldsymbol{S}^T(\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T + \mu\boldsymbol{I})^{-1}\bar{\boldsymbol{m}}. \tag{3.45}$$

Subsequently, we reconstruct $\boldsymbol{f}$ as

$$\hat{\boldsymbol{f}}_K = \boldsymbol{K}_f\boldsymbol{S}^T(\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T + \mu\boldsymbol{I})^{-1}\bar{\boldsymbol{m}} \tag{3.46}$$

and we will henceforth refer to it as the *Kronecker kernel MC and extrapolation* (KKMCEX) estimate of $\boldsymbol{f}$. Regarding the computational cost incurred by (3.45), inversion costs $\mathcal{O}(S^3)$, since the size of the matrix to be inverted is reduced from NL to S. Clearly, there is no need to compute $\boldsymbol{K}_f = \boldsymbol{K}_h \otimes \boldsymbol{K}_w$. As $\boldsymbol{S}$ has binary entries, $\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T$ is just a selection of S^2 entries in $\boldsymbol{K}_f$; and, given that $\kappa_f((x_i, y_j),(x_n, y_l)) = \kappa_w(x_i, x_n)\kappa_h(y_j, y_l)$, it is obtained at cost $\mathcal{O}(S^2)$. Overall, the cost incurred by (3.45) is $\mathcal{O}(S^3)$. Compared to the MC approach in (3.28), the KKMCEX method is easier to implement since it only involves a matrix inversion. Moreover, since it admits a closed-form solution, it facilitates deriving bounds on the estimation error of $\hat{\boldsymbol{f}}_K$.

One meaningful advantage of KKMCEX over the standard MC formulation is that it is able to predict any output as long as its corresponding input belongs to the domain of

the kernel function. Hence, it is able to estimate an entry in a column or row of $\boldsymbol{M}$ with zero observations. Notice that the solution in (3.45) depends only on the observed entries. Hence, in order to generalize to a point $(\tilde{x}, \tilde{y}) \notin \mathcal{X} \times \mathcal{Y}$, one simply needs to build the augmented $\boldsymbol{K}_f$ including the new point. This is possible provided that the kernel function can be evaluated at $(\tilde{x}, \tilde{y})$ in order to obtain $\kappa_f((\cdot,\cdot),(\tilde{x},\tilde{y}))$. Consider for instance an input set $\{\mathcal{X} \times \mathcal{Y}\} \subseteq \mathbb{R}^D$, where $D > 0$, and a Gaussian kernel. While KKMCEX in (3.46) only tries to predict $f(x)\ \forall x \in \mathcal{X}$, it is easy to obtain its value for any input as long as it is a vector in $\mathbb{R}^D$ since the kernel function can be evaluated at any input in this vector space.

Insights on KKMCEX vs. other kernel-based methods

The KKMCEX solution in (3.46), differs from that obtained as the solution of (3.29). On the one hand, the loss in (3.40) can be derived from the factorization-based one in (3.29) by using the Kronecker product kernel $\boldsymbol{K}_h \otimes \boldsymbol{K}_w$ and $\boldsymbol{\gamma} = \text{vec}(\boldsymbol{B}\boldsymbol{C}^T)$ to arrive at

$$\left\|P_\Omega(\boldsymbol{M} - \boldsymbol{K}_w\boldsymbol{B}\boldsymbol{C}^T\boldsymbol{K}_h)\right\|_{\text{F}}^2 = \left\|\bar{\boldsymbol{m}} - \boldsymbol{S}(\boldsymbol{K}_h \otimes \boldsymbol{K}_w)\text{vec}(\boldsymbol{B}\boldsymbol{C}^T)\right\|_2^2. \tag{3.47}$$

One difference between the two loss functions is that (3.40) does not explicitly limit the rank of the recovered matrix $\hat{\boldsymbol{F}} = \text{unvec}(\hat{\boldsymbol{f}}_K)$ since it has NL degrees of freedom through $\hat{\boldsymbol{\gamma}}$, while in (3.29) the rank of $\hat{\boldsymbol{F}}$ cannot exceed p since $\boldsymbol{B}$ and $\boldsymbol{C}$ are of rank p at most. In fact, the low-rank property is indirectly promoted in (3.40) through the kernel matrices. Since $\text{rank}(\boldsymbol{F}) \leq \min(\text{rank}(\boldsymbol{K}_w), \text{rank}(\boldsymbol{K}_h))$, we can limit the rank of $\hat{\boldsymbol{F}}$ by selecting rank deficient kernels. On the other hand, the regularization terms in (3.29) and (3.40) play a different role in each formulation. The regularization in (3.29) promotes smoothness on the columns of the estimated factor matrices $\{\hat{\boldsymbol{W}}, \hat{\boldsymbol{H}}\}$; or, in other words, similarity between the rows of $\{\hat{\boldsymbol{W}}, \hat{\boldsymbol{H}}\}$ as measured by κ_w and κ_h. On the contrary, the regularization in (3.40) promotes smoothness on $\hat{\boldsymbol{f}}$, which is tantamount to promoting similarity between the entries of $\hat{\boldsymbol{F}}$ in accordance with κ_f. Hence, (3.40) allows the design of entrywise relationships in $\boldsymbol{F}$, whereas (3.29) is restricted to row or columnwise design.

Comparing the KKMCEX formulation with the Kronecker product kernel matrix to existing ones, matrices built via the Kronecker product have been used in regression for different purposes. In particular, [61] proposes the use of Kronecker kernels in the recovery of time-varying graph signals via kernel regression. Related to MC, [47] leverages Kronecker product structures to efficiently solve the Sylvester equations that arise in alternating minimization iterations to find $\{\hat{\boldsymbol{W}}, \hat{\boldsymbol{H}}\}$ in (3.28). On the other hand, [52, 62] propose a Kronecker kernel ridge regression method that can be used to extrapolate missing entries in a matrix. However, the methods in [52, 62] assume a complete training set and Kronecker

structure for the regression matrix; this implies that the observed entries in $\boldsymbol{M}$ can be permuted to form a full submatrix. Still, KKMCEX introduces $\boldsymbol{S}$ which encompasses any sampling pattern in Ω. Thus, the properties of the Kronecker product used in [47, 52, 62] cannot be applied to solve (3.46) since $\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T$ must not necessarily be the Kronecker product of two smaller matrices.

3.3.1 KKMCEX error analysis

The performance of KKMCEX is assessed by the mean-square error

$$MSE := \mathbb{E}_{\boldsymbol{e}}\{\|\boldsymbol{f} - \hat{\boldsymbol{f}}_K\|_2^2\} \tag{3.48}$$

where $\mathbb{E}_{\boldsymbol{e}}\{\cdot\}$ denotes the expectation with respect to $\boldsymbol{e}$. This assessment and the analysis in this section also applies to standard KRR in (3.19). Before we proceed, we will outline Nyström's approximation.

Definition 1. Given a kernel matrix $\boldsymbol{K}$ and a binary sampling matrix $\boldsymbol{S}$ of appropriate dimensions, the Nyström approximation [63] of $\boldsymbol{K}$ is $\boldsymbol{T} = \boldsymbol{K}\boldsymbol{S}^T(\boldsymbol{S}\boldsymbol{K}\boldsymbol{S}^T)^\dagger\boldsymbol{S}\boldsymbol{K}$, and the regularized Nyström approximation is

$$\tilde{\boldsymbol{T}} = \boldsymbol{K}\boldsymbol{S}^T(\boldsymbol{S}\boldsymbol{K}\boldsymbol{S}^T + \mu\boldsymbol{I})^{-1}\boldsymbol{S}\boldsymbol{K}. \tag{3.49}$$

□

Nyström's approximation is employed to reduce the complexity of standard kernel regression problems. Instead of $\boldsymbol{K}$, the low-rank approximation $\tilde{\boldsymbol{T}}$ is used to reduce the cost of inverting large-size matrices using the MIL [64]. While it is known that the best low-rank approximation to a matrix in the Frobenius distance sense is obtained from its top eigenvectors, Nyström's approximation is cheaper. Using Def. 1, the following lemma provides the bias and variance of the KKMCEX estimator in (3.46) in terms of $\tilde{\boldsymbol{T}}$:

Lemma 3.2. *Given the kernel matrix $\boldsymbol{K}_f$ and its regularized Nyström approximation $\tilde{\boldsymbol{T}}_f$ with $\mu > 0$, assuming $\mathbb{E}\{\boldsymbol{e}\} = \boldsymbol{0}$ the MSE of the KKMCEX estimator is*

$$MSE = \left\|(\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f)\boldsymbol{\gamma}\right\|_2^2 + \mathbb{E}_{\boldsymbol{e}}\{\frac{1}{\mu^2}\left\|(\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f)\boldsymbol{S}^T\bar{\boldsymbol{e}}\right\|_2^2\} \tag{3.50}$$

where the first term accounts for the bias and the second term accounts for the variance.

Lemma 3.2 shows that the MSE of the KKMCEX can be expressed in terms of $\tilde{\boldsymbol{T}}_f$; see proof in Appendix B.2. Knowing that, for a matrix $\boldsymbol{A}$ and vector $\boldsymbol{x}$, $\|\boldsymbol{A}\boldsymbol{x}\|_2 \leq \|\boldsymbol{A}\|_2\|\boldsymbol{x}\|_2$

and that the 2-norm satisfies $\|\boldsymbol{A}\|_2^2 \leq \|\boldsymbol{A}\|_{\mathrm{F}}^2$, we have

$$\left\|(\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f)\boldsymbol{\gamma}\right\|_2^2 + \mathbb{E}_{\boldsymbol{e}}\left\{\frac{1}{\mu^2}\left\|(\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f)\boldsymbol{S}^T\bar{\boldsymbol{e}}\right\|_2^2\right\} \leq \left\|\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f\right\|_{\mathrm{F}}^2 \left(\|\boldsymbol{\gamma}\|_2^2 + \frac{1}{\mu^2}\operatorname{Tr}(\mathbb{E}_{\boldsymbol{e}}\{\bar{\boldsymbol{e}}\bar{\boldsymbol{e}}^T\})\right). \tag{3.51}$$

Consequently, the upper bound on the MSE is proportional to the approximation error of $\tilde{\boldsymbol{T}}_f$ to $\boldsymbol{K}_f$. This suggests selecting $\{m_{i,j}\}_{(i,j)\in\Omega}$ so that this approximation error is minimized; see also [64] where Ω is chosen according to the so-called leverage scores of $\boldsymbol{K}_f$ in order to minimize the regression error. The next theorem uses Lemma 3.2 to upper bound the MSE in (3.50); see Appendix B.3 for its proof.

Theorem 3.2. *Let σ_{NL} be the maximum eigenvalue of a nonsingular $\boldsymbol{K}_f$, and $\tilde{\boldsymbol{\gamma}} := \boldsymbol{L}^T\boldsymbol{\gamma}$, where $\boldsymbol{L}$ is the eigenvector matrix of $\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f$ with eigenvectors ordered in ascending order. If $\boldsymbol{e}$ is a zero-mean vector of iid Gaussian random variables with covariance matrix $\nu^2\boldsymbol{I}$, the MSE of the KKMCEX estimator is bounded as*

$$MSE \leq \frac{\mu^2\sigma_{NL}^2}{(\sigma_{NL}+\mu)^2}\sum_{i=1}^{s}\tilde{\gamma_i}^2 + \sigma_{NL}^2\sum_{i=s+1}^{NL}\tilde{\gamma}_i^2 + \frac{s\nu^2\sigma_{NL}^2}{\mu^2}. \tag{3.52}$$

Considering the right-hand side of (3.52), the first two terms correspond to the bias, while the last term is related to the variance. In order to assess how the MSE bound behaves as s increases, let us recall the variable change $\mu = s\mu'$ from Section 3.1. Considering this change and fixed values in $(0,\infty)$ for μ', $\|\tilde{\gamma}_i\|^2$ and ${\sigma_{NL}}^2$[2], the bias term reduces to

$$\frac{s^2\mu'^2\sigma_{NL}^2}{(\sigma_{NL}+s\mu')^2}\sum_{i=1}^{s}\tilde{\gamma_i}^2 + \sigma_{NL}^2\sum_{i=s+1}^{NL}\tilde{\gamma}_i^2\,. \tag{3.53}$$

It can be seen in (3.53) that as s increases, terms move from the second summation to the first. Therefore, whether the bias term grows or diminishes depends on the multiplication factors in front of the two summations. Since $\frac{s^2\mu'^2}{(\sigma_{NL}+s\mu')^2} \leq 1$ the bias term in (3.53) decreases with s. Moreover, when $\boldsymbol{M}$ is fully observed, that is, $s = NL$, the bias can be made arbitrarily small by having $\mu' \to 0$. On the other hand, the variance term becomes $\frac{\nu^2\sigma_{NL}^2}{s\mu'^2}$ and decays with s as well. As a result, the MSE bound in Theorem 3.2 decays up until $s = NL$.

[2]Note that $\|\tilde{\gamma}_i\|^2$ and σ_{NL} depend on the selected kernel $\boldsymbol{K}_f$ and matrix $\boldsymbol{F}$, and do not depend on $\boldsymbol{S}$.

3.4 Ridge regression MCEX

Although the KKMCEX algorithm is fast when s is small, the size of the matrix to be inverted in (3.45) grows with s, hence increasing the computational cost. Available approaches to reducing the computational cost of kernel regression methods are centered around the idea of approximating the kernel matrix. For instance, [64] uses Nyström's approximation, that the performance analysis in Section 3.3.1 was based on, whereas [65] relies on a sketch of $\boldsymbol{K}_f$ formed by a subset of its columns, hence reducing the number of regression coefficients; see also [66], where the kernel function is approximated by the inner product of random finite-dimensional feature maps, which also speeds up the matrix inversion. In this section, we reformulate the KKMCEX of Section 3.3 to incorporate a low-rank approximation of $\boldsymbol{K}_f$ in order to obtain a reduced complexity estimate for $\boldsymbol{f}$. Moreover, we also develop an online method based on this reformulation.

Recall from Eq. (3.15) that a kernel can be viewed as the inner product of vectors mapped to a feature space $\mathcal{C}_z$, namely $\kappa_f((x_i, y_j), (x_n, y_l)) = \langle \phi_z(x_i, y_j), \phi_z(x_n, y_l) \rangle_{\mathcal{C}_z}$. Let $\tilde{\phi}_z : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ be a feature map approximating κ_f so that

$$\kappa_f((x_i, y_j), (x_n, y_l)) \simeq \langle \tilde{\phi}_z(x_i, y_j), \tilde{\phi}_z(x_n, y_l) \rangle. \tag{3.54}$$

Then, we define the $NL \times d$ feature matrix

$$\tilde{\boldsymbol{\Phi}}_z := [\tilde{\phi}_z(x_1, y_1), \tilde{\phi}_z(x_2, y_1), \ldots, \tilde{\phi}_z(x_N, y_L)]^T \tag{3.55}$$

and form $\tilde{\boldsymbol{K}}_f = \tilde{\boldsymbol{\Phi}}_z \tilde{\boldsymbol{\Phi}}_z^T$. Note that $\tilde{\boldsymbol{K}}_f$ is a rank-d approximation of $\boldsymbol{K}_f$, and that the equality $\boldsymbol{K}_f = \tilde{\boldsymbol{K}}_f$ is only feasible when $\text{rank}(\boldsymbol{K}_f) \leq d$. Consider $\tilde{\boldsymbol{\Phi}}_x := [\tilde{\phi}_x(x_1), \ldots, \tilde{\phi}_x(x_N)]^T$ and $\tilde{\boldsymbol{\Phi}}_y = [\tilde{\phi}_y(y_1), \ldots, \tilde{\phi}_y(y_L)]^T$, where $\tilde{\phi}_x : \mathcal{X} \to \mathbb{R}^{d_x}$ and $\tilde{\phi}_y : \mathcal{Y} \to \mathbb{R}^{d_y}$, as the feature matrices forming low-rank approximations to $\boldsymbol{K}_w$ and $\boldsymbol{K}_h$, respectively. Since $\boldsymbol{K}_f = \boldsymbol{K}_h \otimes \boldsymbol{K}_w$ in KKMCEX, a prudent choice is $\tilde{\boldsymbol{\Phi}}_z = \tilde{\boldsymbol{\Phi}}_y \otimes \tilde{\boldsymbol{\Phi}}_x$. The next section will present means of constructing $\{\tilde{\boldsymbol{\Phi}}_x, \tilde{\boldsymbol{\Phi}}_y, \tilde{\boldsymbol{\Phi}}_z\}$ maps.

Since $\tilde{\boldsymbol{K}}_f$ is a valid kernel matrix, upon replacing $\boldsymbol{K}_f$ in (3.39) with $\tilde{\boldsymbol{K}}_f$, the observation model reduces to

$$\bar{\boldsymbol{m}} = \boldsymbol{S}\tilde{\boldsymbol{\Phi}}_z \tilde{\boldsymbol{\Phi}}_z^T \boldsymbol{\gamma} + \tilde{\boldsymbol{e}}, \tag{3.56}$$

where $\tilde{\boldsymbol{e}} = \bar{\boldsymbol{e}} + \boldsymbol{S}(\boldsymbol{K}_f - \tilde{\boldsymbol{K}}_f)\boldsymbol{\gamma}$. With this model, the weights in (3.40) are obtained as

$$\hat{\boldsymbol{\gamma}} = \underset{\boldsymbol{\gamma} \in \mathbb{R}^{NL}}{\arg\min} \left\| \bar{\boldsymbol{m}} - \boldsymbol{S}\tilde{\boldsymbol{\Phi}}_z \tilde{\boldsymbol{\Phi}}_z^T \boldsymbol{\gamma} \right\|_2^2 + \mu \boldsymbol{\gamma}^T \tilde{\boldsymbol{\Phi}}_z \tilde{\boldsymbol{\Phi}}_z^T \boldsymbol{\gamma}. \tag{3.57}$$

Letting $\boldsymbol{\xi} := \tilde{\boldsymbol{\Phi}}_z^T \boldsymbol{\gamma}$ and substituting into (3.57), we arrive at

$$\hat{\boldsymbol{\xi}} = \underset{\boldsymbol{\xi} \in \mathbb{R}^d}{\arg\min} \left\| \bar{\boldsymbol{m}} - \boldsymbol{S}\tilde{\boldsymbol{\Phi}}_z \boldsymbol{\xi} \right\|_2^2 + \mu \left\| \boldsymbol{\xi} \right\|_2^2 \tag{3.58}$$

which admits the closed-form solution

$$\hat{\boldsymbol{\xi}} = (\tilde{\boldsymbol{\Phi}}_z^T \boldsymbol{S}^T \boldsymbol{S} \tilde{\boldsymbol{\Phi}}_z + \mu \boldsymbol{I})^{-1} \tilde{\boldsymbol{\Phi}}_z^T \boldsymbol{S}^T \bar{\boldsymbol{m}}. \tag{3.59}$$

Using $\hat{\boldsymbol{\xi}}$, we obtain $\hat{\boldsymbol{f}}_R := \tilde{\boldsymbol{\Phi}}_z \hat{\boldsymbol{\xi}}$ as the *ridge regression MCEX* (RRMCEX) estimate. Using the MIL (3.44) on (3.59), it follows that

$$\hat{\boldsymbol{\xi}} = \tilde{\boldsymbol{\Phi}}_z^T \boldsymbol{S}^T (\boldsymbol{S} \tilde{\boldsymbol{\Phi}}_z \tilde{\boldsymbol{\Phi}}_z^T \boldsymbol{S}^T + \mu \boldsymbol{I})^{-1} \bar{\boldsymbol{m}} \tag{3.60}$$

and thus,

$$\hat{\boldsymbol{f}}_R = \tilde{\boldsymbol{\Phi}}_z \hat{\boldsymbol{\xi}} = \tilde{\boldsymbol{K}}_f \boldsymbol{S}^T (\boldsymbol{S} \tilde{\boldsymbol{K}}_f^T \boldsymbol{S}^T + \mu \boldsymbol{I})^{-1} \bar{\boldsymbol{m}}. \tag{3.61}$$

Therefore, (3.61) shows that $\hat{\boldsymbol{f}}_R$ is equivalent to the KKMCEX solution $\hat{\boldsymbol{f}}_K$ in (3.46) after replacing $\boldsymbol{K}_f$ by its low-rank approximation $\tilde{\boldsymbol{K}}_f$. For error-free approximation, $\boldsymbol{K}_f = \tilde{\boldsymbol{\Phi}}_z \tilde{\boldsymbol{\Phi}}_z^T$, while $\hat{\boldsymbol{\xi}}$ in (3.59) can be viewed as the primal solution to the optimization problem in (3.58), and $\hat{\boldsymbol{\gamma}}$ in (3.45) as its dual [25]. Still, obtaining $\hat{\boldsymbol{\xi}}$ requires multiplying two $d \times s$ matrices and inverting a $d \times d$ matrix, which incurs computational cost $\mathcal{O}(d^2 s)$ when $s \geq d$, and $\boldsymbol{S}\tilde{\boldsymbol{\Phi}}_z$ is obtained at cost $\mathcal{O}(ds)$. Thus, the cost of RRMCEX grows linearly with s in contrast to KKMCEX which increases with s^3.

By choosing an appropriate feature map so that $d \ll s$, it is possible to control the computational cost of calculating $\hat{\boldsymbol{\xi}}$. However, reduced computational cost by selecting a small d might come at the price of an approximation error to $\boldsymbol{K}_f$, which correspondingly increases the estimation error of $\hat{\boldsymbol{f}}_R$. The selection of a feature matrix to minimize this error and further elaboration on the computational cost are given in Section 3.5.

3.4.1 Online RRMCEX

Online methods learn a model by processing one datum at a time. An online algorithm often results when the objective can be separated into several subfunctions, each depending on one or multiple data. In the context of MC, online implementation updates $\hat{\boldsymbol{F}}$ every time a new entry $\boldsymbol{M}_{i,j}$ becomes available. If we were to solve (3.45) each time a new observation becomes available, inverting an $s \times s$ matrix per iteration would result in an overall prohibitively high computational cost. Still, the cost of obtaining an updated solution per observation can stay manageable using online kernel regression solvers that fall into three categories [67]: dictionary

learning, recursive regression and stochastic gradient descent based. Akin to [68, 69], we will pursue here the SGD.

Consider rewriting (3.58) entrywise as

$$\hat{\boldsymbol{\xi}} = \underset{\boldsymbol{\xi}\in\mathbb{R}^d}{\arg\min} \sum_{(i,j)\in\Omega} \left[m_{i,j} - \tilde{\phi}_z^T(x_i, y_j)\boldsymbol{\xi}\right]^2 + \mu \left\|\boldsymbol{\xi}\right\|_2^2. \tag{3.62}$$

With n denoting each scalar observation, SGD iterations form a sequence of estimates

$$\hat{\boldsymbol{\xi}}^n = \hat{\boldsymbol{\xi}}^{n-1} - t_n \left[\tilde{\phi}_z(x_i, y_j)(\tilde{\phi}_z^T(x_i, y_j)\hat{\boldsymbol{\xi}}^{n-1} - m_{i,j}) + \mu\hat{\boldsymbol{\xi}}^{n-1}\right] \tag{3.63}$$

where t_n is the step size, $n = 1, \ldots, S$ and the tuple (i, j) denotes the indices of the entry revealed at iteration n. With properly selected t_n, the sequence $\hat{\boldsymbol{\xi}}^n$ will converge to (3.62) at per iteration cost $\mathcal{O}(d)$ [70]. Apart from updating all entries in the matrix, (3.63) can also afford a simple distributed implementation using e.g., the algorithms in [71].

Remark 3.1. *Online algorithms for MC can be designed to solve the factorization-based formulation from (3.23) rewritten as*

$$\underset{\substack{\boldsymbol{W}\in\mathbb{R}^{N\times p}\\ \boldsymbol{H}\in\mathbb{R}^{N\times p}}}{\arg\min} \sum_{(i,j)\in\Omega} \left((m_{i,j} - \boldsymbol{w}_i^T\boldsymbol{h}_j)^2 + \frac{\mu}{|\Omega_i^w|}\left\|\boldsymbol{w}_i\right\|_2^2 + \frac{\mu}{|\Omega_j^h|}\left\|\boldsymbol{h}_j\right\|_2^2\right) \tag{3.64}$$

where $\boldsymbol{w}_i^T$ and $\boldsymbol{h}_j^T$ denote the i^{th} and j^{th} rows of $\boldsymbol{H}$ and $\boldsymbol{W}$ respectively, $\Omega_i^w = \{j : (i, j) \in \Omega\}$, and $\Omega_j^h = \{i : (i, j) \in \Omega\}$. When $m_{i,j}$ becomes available, algorithms such as SGD and online ALS update the rows $\{\boldsymbol{w}_i^T, \boldsymbol{h}_j^T\}$ of the coefficient matrices. This procedure can also be applied to the kernel MCEX formulation in (3.28), that solves for $\boldsymbol{W}$ and $\boldsymbol{H}$. Then, all entries in the i^{th} row and j^{th} column of $\hat{\boldsymbol{F}}$ are also updated per iteration, as opposed to our method, which updates the whole matrix. □

Table 3.1 summarizes the main contributions in this chapter, namely the KKMCEX, RRMCEX, and online RRMCEX algorithms.

KKMCEX	$\hat{\boldsymbol{f}}_K = \boldsymbol{K}_f\boldsymbol{S}^T(\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T + \mu\boldsymbol{I})^{-1}\bar{\boldsymbol{m}}$
RRMCEX	$\hat{\boldsymbol{f}}_R = \tilde{\boldsymbol{\Phi}}_z(\tilde{\boldsymbol{\Phi}}_z^T\boldsymbol{S}^T\boldsymbol{S}\tilde{\boldsymbol{\Phi}}_z + \mu\boldsymbol{I})^{-1}\tilde{\boldsymbol{\Phi}}_z^T\boldsymbol{S}^T\bar{\boldsymbol{m}}$
(o)RRMCEX	$\hat{\boldsymbol{f}}_O^n = \tilde{\boldsymbol{\Phi}}_z\hat{\boldsymbol{\xi}}^n,\ \hat{\boldsymbol{\xi}}^n = \hat{\boldsymbol{\xi}}^{n-1} - t_n\left[\tilde{\phi}_z(x_i, y_j)(\tilde{\phi}_z^T(x_i, y_j)\hat{\boldsymbol{\xi}}^{n-1} - m_{i,j}) + \mu\hat{\boldsymbol{\xi}}^{n-1}\right]$

Table 3.1: Algorithms proposed in the chapter.

3.5 Choosing the kernel matrices

This section provides pointers on how to build matrices $\boldsymbol{K}_f$ for KKMCEX and $\tilde{\boldsymbol{\Phi}}_z$ for RRMCEX in Table 3.1 when prior information about either the matrix $\boldsymbol{F}$, or the input sets $\mathcal{X}$ and $\mathcal{Y}$, is available. In the first considered scenario, $\boldsymbol{F}$ is a matrix of graph signals and the kernels are built using the Laplacian matrix of the row and column graphs of $\boldsymbol{F}$. Second, the availability of feature vectors associated with the elements in the input sets is leveraged to generate kernels. Finally, several approaches on how to construct the approximate feature matrix $\boldsymbol{\Phi}_z$ while aiming at reducing computational costs are presented. To avoid notational clutter, the symbols referring to the input sets and associated feature vectors will be reused throughout the section, with their purpose being deducible from context.

3.5.1 Kernels based on the graph Laplacian

Suppose that the columns and rows of $\boldsymbol{F}$ lie on a graph, that is, each entry of a column or row vector is associated with a node on a graph that encodes the interdependencies among entries in the same vector. Specifically, we define an undirected weighted graph $\mathcal{G}_x = (\mathcal{X}, \mathcal{E}_x, \boldsymbol{A}_x)$ for the columns of $\boldsymbol{F}$, where $\mathcal{X}$ is the set of vertices with $|\mathcal{X}| = N$, $\mathcal{E}_x \subseteq \mathcal{X} \times \mathcal{X}$ is the set of edges connecting the vertices, and $\boldsymbol{A}_x \in \mathbb{R}^{N\times N}$ is a weighted adjacency matrix. Then, functions $\{f_l : \mathcal{X} \to \mathbb{R}\}_{l=1}^{L}$ are graph signals [31], that is, a map from the set $\mathcal{X}$ of vertices into the set of real numbers. Likewise, we define a graph $\mathcal{G}_y = (\mathcal{Y}, \mathcal{E}_y, \boldsymbol{A}_y)$ for the rows of $\boldsymbol{F}$, i.e., $\{g_n : \mathcal{Y} \to \mathbb{R}\}_{n=1}^{N}$, which are also graph signals. In a matrix of user-movie ratings for instance, we would have two graphs: one for the users and one for the movies. The graphs associated with the columns and rows yield the underlying structure of $\boldsymbol{F}$ that can be used to generate a pair of kernels.

Using $\boldsymbol{A}_x$ and $\boldsymbol{A}_y$, we can form the corresponding graph Laplacian as $\boldsymbol{L}_x := \mathrm{diag}(\boldsymbol{A}_x\mathbf{1}) - \boldsymbol{A}_x$ and likewise for $\boldsymbol{L}_y$, that can serve as kernels. A family of graphical kernels results from using a monotonic inverse function $r^{\dagger}(\cdot)$ on the Laplacian eigendecomposition as [72]

$$\boldsymbol{K} = \boldsymbol{Q} r^{\dagger}(\boldsymbol{\Lambda}) \boldsymbol{Q}^T. \tag{3.65}$$

A possible choice of $r(\cdot)$ is the Gaussian radial basis function, which generates the diffusion kernel $r(\lambda_i) = e^{\eta\lambda_i}$, where λ_i is the i^{th} eigenvalue of $\boldsymbol{L}$, and η a weight parameter. Alternatively, one can choose just the linear function $r(\lambda_i) = 1 + \eta\lambda_i$, which results in the regularized Laplacian kernel. By applying different weighting functions to the eigenvalues of $\boldsymbol{L}_x$ and $\boldsymbol{L}_y$, we promote smoother or more rapidly changing functions for the columns and rows of $\hat{\boldsymbol{F}}$ [61]. While $\boldsymbol{K}_w$ and $\boldsymbol{K}_h$ are chosen as Laplacian kernels, this would not be the case

for $\boldsymbol{K}_f = \boldsymbol{K}_h \otimes \boldsymbol{K}_w$ used in our KKMCEX context since it does not result from applying $r^\dagger(\cdot)$ to a Laplacian matrix. However, since $\boldsymbol{K}_w = \boldsymbol{Q}_w \boldsymbol{\Sigma}_w \boldsymbol{Q}_w^T$ and $\boldsymbol{K}_h = \boldsymbol{Q}_h \boldsymbol{\Sigma}_h \boldsymbol{Q}_h^T$, the eigendecomposition of $\boldsymbol{K}_f$ is $\boldsymbol{K}_f = (\boldsymbol{Q}_h \otimes \boldsymbol{Q}_w)(\boldsymbol{\Sigma}_h \otimes \boldsymbol{\Sigma}_w)(\boldsymbol{Q}_h^T \otimes \boldsymbol{Q}_w^T)$, and the notions of frequency and smoothness carry over. In other words, we are still promoting similarity through $\boldsymbol{K}_f$ among entries that are connected on the row and column graphs.

A key attribute in graph signal processing is that of "graph bandlimitedness", which arises when a signal can be generated as a linear combination of a few eigenvectors of the Laplacian matrix. Therefore, a bandlimited graph signal belongs to an RKHS that is spanned by a bandlimited kernel [23] that suppresses some of the frequencies of the graph. A bandlimited kernel is derived from the Laplacian matrix of a graph as in (3.65), using

$$r(\lambda_i) = 0 \quad \forall i \notin \Psi, \tag{3.66}$$

where $\Psi \subseteq \mathbb{N}$ contains the indices of frequencies not to be suppressed. As mentioned earlier, we define a graph for the columns and a graph for the rows of $\boldsymbol{F}$. Therefore, graph signals contained in the columns and rows may be bandlimited with different bandwidths. In order to form $\boldsymbol{K}_f$ our KKMCEX approach, we will need to apply different weighting functions akin to the one in (3.66) for kernel matrices $\boldsymbol{K}_w$ and $\boldsymbol{K}_h$.

3.5.2 Kernels from known bases or features

In some applications, the basis that spans the columns or rows of the unobserved matrix is assumed known, although this basis matrix needs not be a kernel matri. In order to be able to include such a basis into the kernel framework, we need to generate kernel functions that span the same spaces as the columns and rows of $\boldsymbol{F}$.

Consider the input sets $\{\mathcal{X}, \mathcal{Y}\}$ whose entries can be mapped into an Euclidean space through feature extraction functions $\theta_x : \mathcal{X} \to \mathbb{R}^{t_x}$ and $\theta_y : \mathcal{Y} \to \mathbb{R}^{t_y}$ defined as $\theta_x(x_i) := \boldsymbol{x}_i$ and $\theta_y(y_j) := \boldsymbol{y}_j$. For instance, in a movie recommender system where the users are represented in $\mathcal{X}$ and the movies in $\mathcal{Y}$, each coordinate of $\boldsymbol{y}_j$ could denote the amount of action, drama and nudity in the movie, and $\boldsymbol{x}_i$ would contain weights according to the user's preference for each attribute. We may then use the feature vectors to determine the similarities among entries in $\mathcal{X}$ and $\mathcal{Y}$ by means of kernel functions.

Let $\boldsymbol{X} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^T$ and $\boldsymbol{Y} := [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_L]^T$. If $\text{span}(\boldsymbol{F}) \subseteq \text{span}(\boldsymbol{X})$ and $\text{span}(\boldsymbol{F}^T) \subseteq \text{span}(\boldsymbol{Y})$, we may conveniently resort to the linear kernel. The linear kernel amounts to the dot product in Euclidean spaces, which we use to define the pair $\kappa_w(x_i, x_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j$ and $\kappa_h(y_i, y_j) = \boldsymbol{y}_i^T \boldsymbol{y}_j$. This leads to a straightforward construction of the kernel matrices for KKMCEX as $\boldsymbol{K}_w = \boldsymbol{X}\boldsymbol{X}^T$ and $\boldsymbol{K}_h = \boldsymbol{Y}\boldsymbol{Y}^T$.

Besides the linear kernel, it is often necessary to use a different kernel class for each κ_w and κ_h chosen to better fit the spaces spanned by the rows and columns of $\boldsymbol{F}$. For instance, the Gaussian kernel defined as $\kappa_w(x_i, x_j) = \exp\{-\eta \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2\}$, is a widely used alternative in the regression of smooth functions.

3.5.3 Feature maps for RRMCEX

Aiming to construct $\tilde{\boldsymbol{\Phi}}_z$ in (3.55) that approximates $\boldsymbol{K}_f$ at reduced complexity, we choose $\tilde{\phi}_z$ with $d \ll S$. To approximate linear kernels, let $\tilde{\phi}_x(x_i) = \theta_x(x_i) = \boldsymbol{x}_i$ and $\tilde{\phi}_y(y_j) = \theta_y(x_i) = \boldsymbol{y}_j$ so that we can set $\tilde{\phi}_z(x_i, y_j) = \tilde{\phi}_y(y_j) \otimes \tilde{\phi}_x(x_i)$ and $\tilde{\boldsymbol{\Phi}}_z = \boldsymbol{Y} \otimes \boldsymbol{X}$. Note that in this case $\tilde{\boldsymbol{\Phi}}_z \tilde{\boldsymbol{\Phi}}_z^T$ yields a zero-error approximation to $\boldsymbol{K}_f = (\boldsymbol{Y} \otimes \boldsymbol{X})(\boldsymbol{Y} \otimes \boldsymbol{X})^T$, which renders the KKMCEX and RRMCEX solutions equivalent.

On occasion, $\boldsymbol{X}$ and $\boldsymbol{Y}$ may have a large column dimension, thus rendering $\boldsymbol{Y} \otimes \boldsymbol{X}$ undesirable as a feature matrix in RRMCEX. In order to overcome this hurdle, we build an approximation to the column space of $\boldsymbol{Y} \otimes \boldsymbol{X}$ from the SVD of $\boldsymbol{X}$ and $\boldsymbol{Y}$. Consider the SVDs of matrices $\boldsymbol{X} = \boldsymbol{U}_x \boldsymbol{D}_x \boldsymbol{V}_x^T$ and $\boldsymbol{Y} = \boldsymbol{U}_y \boldsymbol{D}_y \boldsymbol{V}_y^T$, to obtain $\boldsymbol{Y} \otimes \boldsymbol{X} = (\boldsymbol{U}_y \otimes \boldsymbol{U}_x)(\boldsymbol{D}_y \otimes \boldsymbol{D}_x)(\boldsymbol{V}_y^T \otimes \boldsymbol{V}_x^T)$. Let $\tilde{\boldsymbol{\Phi}}_z = \boldsymbol{U}_d \boldsymbol{D}_d$, where $\boldsymbol{U}_d$ and $\boldsymbol{D}_d$ respectively hold the top d singular vectors and singular values of $\boldsymbol{Y} \otimes \boldsymbol{X}$. The SVD has cost $\mathcal{O}(N t_x^2)$ for $\boldsymbol{X}$ and $\mathcal{O}(L t_y^2)$ for $\boldsymbol{Y}$. Comparatively, the cost of building $\boldsymbol{K}_w$ and $\boldsymbol{K}_h$ for the linear kernel is $\mathcal{O}(N^2 t_x)$ and $\mathcal{O}(L^2 t_y)$, respectively. Therefore, choosing RRMCEX over KKMCEX in this case incurs no extra overhead.

When a function other than the linear kernel is selected, obtaining an approximation is more complex. To approximate a Gaussian kernel on $\mathcal{X} \times \mathcal{X}$, the vectors $\{\tilde{\phi}_x(x_i)\}_{i=1}^N$ can be obtained by means of Taylor series expansion [73] or random Fourier features [66], which can also approximate Laplacian, Cauchy and polynomial kernels [66, 74]. Therefore, the maps $\tilde{\phi}_x$ and $\tilde{\phi}_y$ must be designed according to the chosen kernels.

In some instances, such as when dealing with graph Laplacian kernels, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are not available and we are only given $\boldsymbol{K}_w$ and $\boldsymbol{K}_h$. We are then unable to derive approximations to the kernel matrices by means of maps $\tilde{\phi}_x$ and $\tilde{\phi}_y$. Nevertheless, we can still derive an adequate $\tilde{\boldsymbol{\Phi}}_z$ to approximate $\boldsymbol{K}_f$. Indeed, Mercer's Theorem asserts that there are eigenfunctions $\{q_n\}_{n=1}^{NL}$ in $\mathcal{H}_f$ along with a sequence of nonnegative real numbers $\{\sigma_n\}_{n=1}^{NL}$, such that

$$\kappa_f((x_i, y_j), (x_n, y_l)) = \sum_{n=1}^{NL} \sigma_n q_n(x_i, y_j) q_n(x_n, y_l). \tag{3.67}$$

We can find (3.67) from the eigendecomposition $\boldsymbol{K}_f = \boldsymbol{Q}_f \boldsymbol{\Sigma}_f \boldsymbol{Q}_f^T$, where q_n is the n^{th} eigenvector in $\boldsymbol{Q}_f$ and σ_n the n^{th} eigenvalue in $\boldsymbol{\Sigma}_f$. If $\boldsymbol{K}_f$ is low rank, we can construct

$\tilde{\boldsymbol{\Phi}}_z = \boldsymbol{Q}_d \boldsymbol{\Sigma}_d^{\frac{1}{2}}$, where $\boldsymbol{Q}_d$ and $\boldsymbol{\Sigma}_d$ respectively hold the top d eigenvectors and eigenvalues of $\boldsymbol{K}_f$. Note that, since $\boldsymbol{K}_f = (\boldsymbol{Q}_h \otimes \boldsymbol{Q}_w)(\boldsymbol{\Sigma}_h \otimes \boldsymbol{\Sigma}_w)(\boldsymbol{Q}_h^T \otimes \boldsymbol{Q}_w^T)$, we only need to eigendecompose smaller matrices $\boldsymbol{K}_w$ and $\boldsymbol{K}_h$ at complexity $\mathcal{O}(N^3 + L^3)$. In some cases however, such as when using Laplacian kernels, the eigendecompositions are readily available, and $\boldsymbol{\Phi}_z$ can be built at a markedly reduced cost.

3.6 Numerical tests

In this section, we test the performance of the KKMCEX, RRMCEX and online (o)RRMCEX algorithms developed in Sections 3.3, 3.4 and 3.4.1, respectively; and further compare them to the solution of (3.28) obtained with ALS [53] and SGD [24]. We run the tests on synthetic and real datasets, with and without noise, and measure the signal-to-noise-ratio (SNR) as $snr = \frac{\|\boldsymbol{F}\|_{\mathrm{F}}^2}{\|\boldsymbol{E}\|_{\mathrm{F}}^2}$. The algorithms are run until convergence over $N_r = 50$ realizations with different percentages of observed entries, denoted by $P_s = 100s/(NL)$, which are taken uniformly at random per realization. As figure of merit, we use

$$\mathrm{NMSE} = \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{\left\| \hat{\boldsymbol{F}}_i - \boldsymbol{F} \right\|_{\mathrm{F}}^2}{\|\boldsymbol{F}\|_{\mathrm{F}}^2} \tag{3.68}$$

where $\hat{\boldsymbol{F}}_i$ is the estimate at realization i. We show results for the optimal combination of regularization and kernel parameters, found via grid search. Finally, ALS and SGD are initialized by a product of two random factor matrices.

3.6.1 Synthetic data

We first test the algorithms on synthetic data. A 250×250 data matrix is generated as $\boldsymbol{F} = \boldsymbol{K}_w \boldsymbol{\Gamma} \boldsymbol{K}_h$, where $\boldsymbol{\Gamma}$ is a 250×250 matrix of Gaussian random deviates. For $\boldsymbol{K}_w$ and $\boldsymbol{K}_h$ we use Laplacian diffusion kernels with $\eta = 1$ based on Erdös-Rényi graphs, whose binary adjacency matrices are unweighted and any two vertices are connected with probability 0.03. The resulting $\boldsymbol{F}$ is approximately low-rank, with the sum of the first 10 eigenvalues accounting for 96% of the total eigenvalue sum. Therefore, we set the rank bound p to 10 for the ALS and SGD algorithms. Whether $\boldsymbol{F}$ is approximately low-rank or exactly low-rank did not affect our results, as they were similar for matrices with an exact rank of 10. For KKMCEX, $\boldsymbol{K}_f = \boldsymbol{K}_h \otimes \boldsymbol{K}_w$, and for RRMCEX $\tilde{\boldsymbol{\Phi}}_z = \boldsymbol{Q}_d \boldsymbol{\Sigma}_d^{\frac{1}{2}}$, where $\boldsymbol{Q}_d$ contains the top 250 eigenvectors of $\boldsymbol{K}_f$, and $\boldsymbol{\Sigma}_d$ the corresponding top 250 eigenvalues.

Fig. 3.5 shows the simulated NMSE when $\boldsymbol{M}$ is noiseless (a) or noisy (b). We deduce from Fig. 3.5a that all algorithms except SGD achieve a very small NMSE, below 0.003

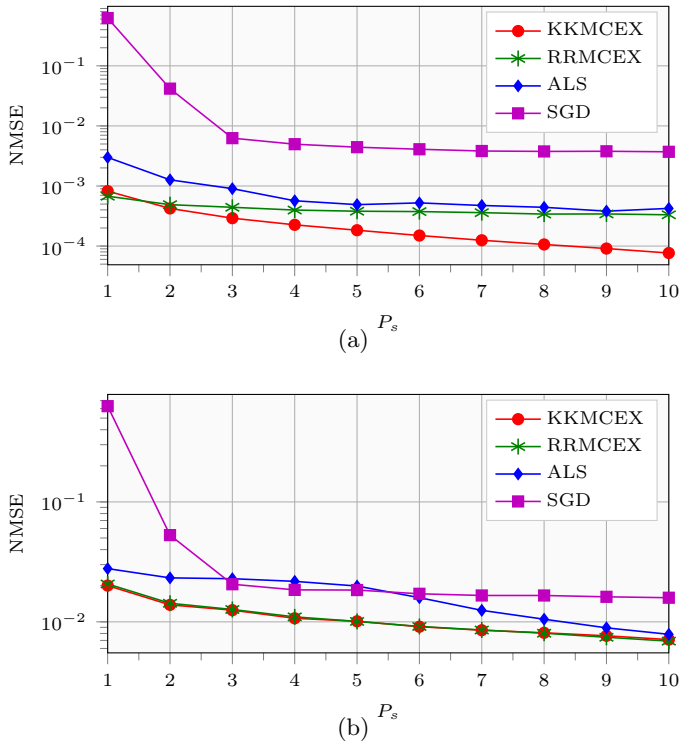


Figure 3.5: NMSE vs P_s for (a) synthetic noiseless matrix; and (b) synthetic noisy matrix.

at $P_s = 1\%$ that falls to 0.0007 at $P_s = 10\%$. Of the three algorithms, KKMCEX has the smallest error except at $P_s = 1\%$, where RRMCEX performs best. Although the error drops for SGD at $P_s > 4\%$, it is outperformed by the other algorithms by an order of magnitude. Fig. 3.5b shows the same results when Gaussian noise is added to $\boldsymbol{F}$ at snr $= 1$. We observe that KKMCEX and RRMCEX are matched and attain the lowest error, whereas ALS and SGD have larger errors across P_s. This corroborates that thanks to the regularization term that smoothes over all the entries instead of row or column-wise, the noise effect is reduced. Interestingly, RRMCEX is able to reduce the noise effect despite the bias it suffers because it only uses the top 250 eigenvalues of $\boldsymbol{K}_f$ from a total of 62,500. This is mainly due to the additive noise being evenly distributed across the eigenspace of $\boldsymbol{K}_f$. Therefore, by keeping only the eigenvectors associated with the top 250 eigenvalues in $\boldsymbol{K}_f$, we are discarding those dimensions in which the SNR is lower.

Fig. 3.6 depicts the time needed for the algorithms to perform the simulations reported in Fig. 3.5. We observe in Fig. 3.6a that RRMCEX has an almost constant computation time, whereas the time for KKMCEX grows with P_s as expected since the size of the matrix to be

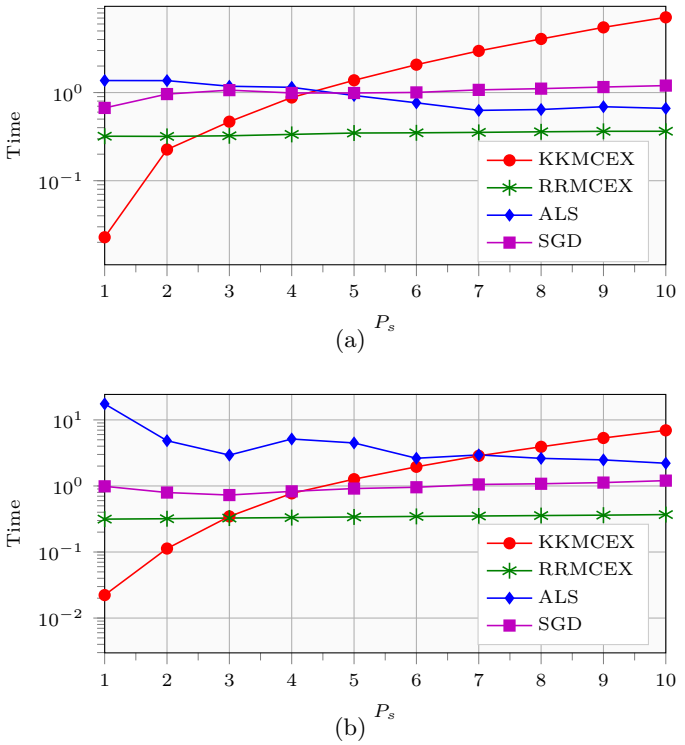


Figure 3.6: Time vs P_s for (a) synthetic noiseless matrix; and (b) synthetic noisy matrix.

inverted increases with s. On the other hand, ALS and SGD require less time than KKMCEX for the larger values of P_s, but are always outperformed by RRMCEX. Moreover, the ALS time is reduced as P_s increases because the number of iterations required to converge to the minimum is smaller. Fig. 3.6b suggests that the noise only impacts ALS, which has its computation time rise considerably across all P_s. Overall, Figures 3.5 and 3.6 illustrate that RRMCEX has the best performance for the synthetic matrix both in terms of NMSE and computational cost.

3.6.2 Temperature measurements

In this case, $\boldsymbol{F}$ has size 150×365 comprising temperature readings taken by 150 stations over 365 days in 2002 in the United States[3]. This is the same dataset used in Section 2.4.3, and the columns and rows of $\boldsymbol{F}$ are modeled as graph signals with $\boldsymbol{A}_x$ and $\boldsymbol{A}_y$ for the graphs formed by the stations and the days of the year, respectively. We use Laplacian diffusion kernels both for $\boldsymbol{K}_w$ and $\boldsymbol{K}_h$, while $\boldsymbol{K}_f$ and $\tilde{\boldsymbol{\Phi}}_z$ are obtained as in the tests on synthetic data, except that

[3] http://earthpy.org/ulmo.html

$\tilde{\boldsymbol{\Phi}}_z$ is constructed with the top 150 eigenvectors of $\boldsymbol{K}_f$. The steps to construct the adjacency matrix $\boldsymbol{A}_x$ are the same as in Section 2.4.3, and are detailed here again for conveniency. The matrix $\boldsymbol{A}_x$ is obtained as in [20], where a graph $\mathcal{G}$ with unweighted adjacency matrix $\boldsymbol{P}$ is generated for the stations, and each station is a vertex connected to the 8 geographically closest stations. Next, we obtain the undirected graph $\mathcal{G}'$ with symmetric adjacency matrix $\boldsymbol{P}' = \text{sign}(\boldsymbol{P}^T + \boldsymbol{P})$. Finally, the entries of $\boldsymbol{A}_x$ are constructed as $(\boldsymbol{A}_x)_{i,j} = \exp(-\frac{N^2 d_{i,j}}{\sum_{i,j} d_{i,j}})$, where $\{d_{i,j}\}$ are geodesic distances on $\mathcal{G}$. In order to form $\boldsymbol{A}_y$, we adopt a graph on which each day is a vertex and each day is connected to the 10 past and future days.

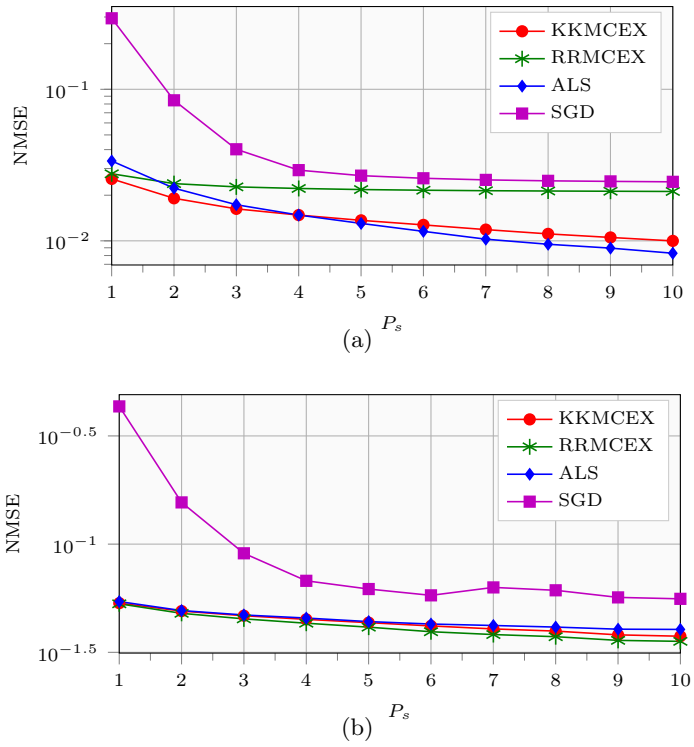


Figure 3.7: NMSE vs P_s for the matrix of temperature measurements (a) without noise, and (b) with noise.

Fig. 3.7 shows the simulated tests for (a) the matrix of temperature readings, and (b) the same matrix with additive Gaussian noise at snr = 1. Fig. 3.7a demonstrates that KKMCEX achieves the lowest error for the first three data points, while afterwards ALS has a slight edge over KKMCEX. The real data matrix $\boldsymbol{F}$ is approximately low rank, since the sum of the first 10 singular values accounts for 75% of the total sum. This explains why RRMCEX fares worse than KKMCEX. Because it only contains the top 150 eigenvectors of $\boldsymbol{K}_f$, which is full

rank, the vectorized data $\boldsymbol{m}$ lies in part outside the space spanned by $\tilde{\boldsymbol{\Phi}}_z$. Indeed, increasing the number of eigenvectors in $\tilde{\boldsymbol{\Phi}}_z$ results in a lower error, although the computational cost increases accordingly. Fig. 3.7b further demonstrates that the addition of noise has the least impact on RRMCEX, which attains the lowest error slightly below KKMCEX. On the other hand, ALS has a marginally higher error whereas the gap between SGD and the other three methods remains. Fig. 3.8 depicts the computational time for the results in Fig. 3.7a, which follow the same trends as those obtained for the synthetic dataset.

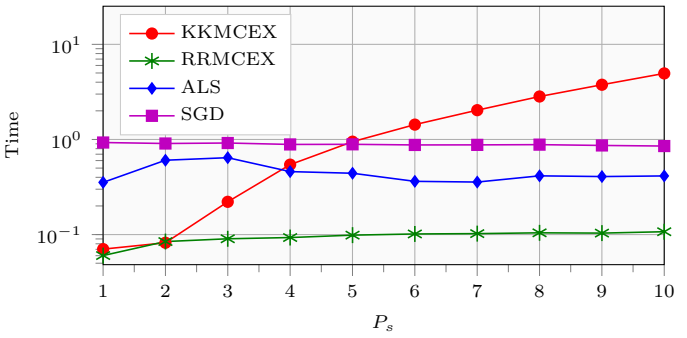


Figure 3.8: Time vs P_s for the noiseless matrix of temperature measurements.

3.6.3 Mushroom dataset

The Mushroom dataset[4] comprises 8,124 labels and as many feature vectors. Each label indicates whether a sample is edible or poisonous, and each vector has 22 entries describing the shape, color, and other features of the mushroom sample. After removing items with missing features, we are left with 5,643 labels and feature vectors. Here, a clustering problem is solved in which $\boldsymbol{F}$ is a $5,643 \times 5,643$ adjacency matrix, where $\boldsymbol{F}_{i,j} = 1$ if the i^{th} and the j^{th} mushroom samples belong to the same class (poisonous or edible), and $\boldsymbol{F}_{i,j} = -1$ otherwise.

We encode the matrix stacking the feature vectors via one-hot encoding to produce a $5,643 \times 98$ binary feature matrix analogous to $\boldsymbol{X}$ in Section 3.5.2. The kernel matrix $\boldsymbol{K}_w$ is built from the Pearson correlation coefficients of the rows of $\boldsymbol{X}$, and let $\boldsymbol{K}_h = \boldsymbol{K}_w$. The feature matrix $\tilde{\boldsymbol{\Phi}}_z$ for RRMCEX is built using the top 3,000 left singular vectors of $\boldsymbol{X} \otimes \boldsymbol{X}$.

Fig. 3.9a shows the test results on the mushroom adjacency matrix from $s = 2,000$ ($P_s = 0.006\%$) to $s = 20,000$ ($P_s = 0.06\%$) in steps of 1,000 observations. KKMCEX and RRMCEX achieve similar NMSE, while SGD has an error one order of magnitude higher, and ALS outperforms both by around one order of magnitude. This difference with ALS is because regression-based methods restrict the solution to belong to the space spanned by the basis matrix. On the other hand, when solving (3.28), we do not enforce the constraints

[4] http://archive.ics.uci.edu/ml

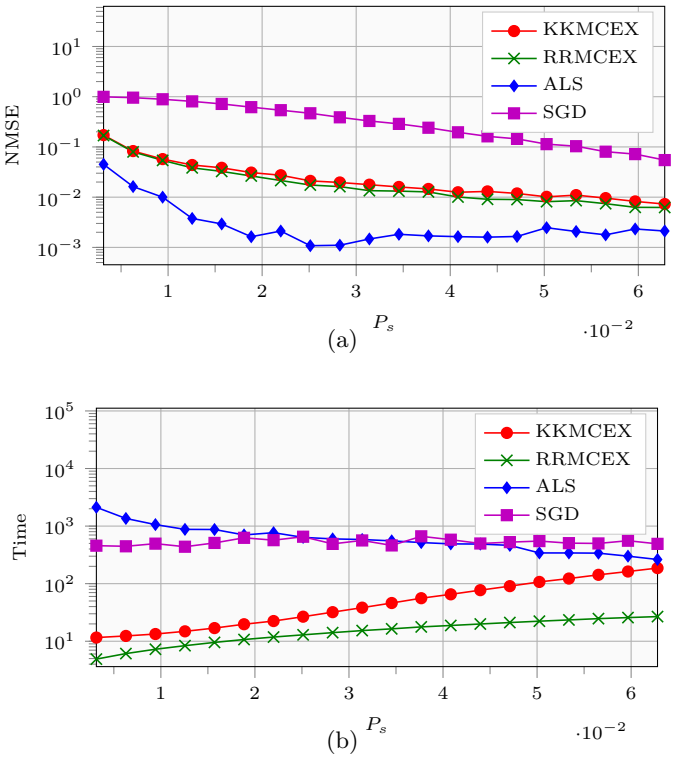


Figure 3.9: results for the mushroom adjacency matrix as (a) NMSE vs P_s, and (b) time vs P_s.

$\boldsymbol{W} \in \mathcal{H}_x$, $\boldsymbol{H} \in \mathcal{H}_y$ [22, 24]. Therefore, when the prior information encoded in the kernel matrices is imperfect, ALS might be able to find a factorization that fits better the data at the cost of having $\hat{\boldsymbol{W}} \notin \mathcal{H}_x$ and $\hat{\boldsymbol{H}} \notin \mathcal{H}_y$. However, in Fig. 3.9b we see that the computational cost for ALS and SGD is much higher than for KKMCEX and RRMCEX for the smaller P_s. On the other hand, the time for ALS decreases with s due to requiring les iterations to converge, whereas for KKRRMCEX and RRMCEX it increases with s.

3.6.4 Online MC

In the online scenario, we compare the (o)RRMCEX algorithm with online (o)ALS, implemented after Algorithm 2 and SGD. One observation is revealed per iteration at random, and all three algorithms process a single observation per iteration in a circular fashion. The (o)ALS algorithm is based on Algorithm 2, and it updates only the row corresponding to the newly revealed entry in each factor matrix. Per realization, we run tests on both synthetic and temperature matrices with $P_s = 10\%$, that is, $s = 6{,}250$ and $s = 5{,}475$ observations for

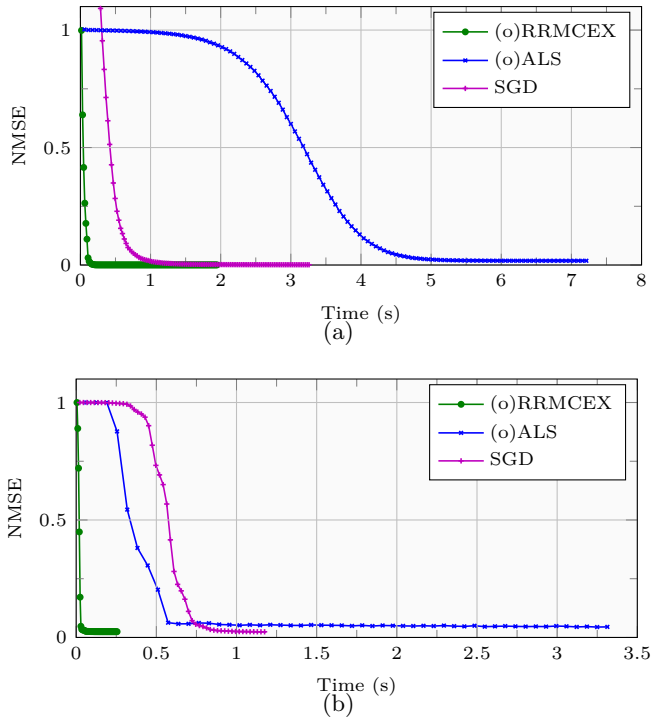


Figure 3.10: NMSE vs time for the online algorithms on the (a) synthetic noiseless matrix; and (b) matrix of noiseless temperature measurements. Each mark denotes 1000 iterations have passed.

the synthetic and temperature matrices, respectively, for a single realization.

Fig. 3.10a depicts the tests for the noiseless synthetic matrix. Clearly, (o)RRMCEX converges much faster than SGD and (o)ALS. Indeed, as opposed to SGD and (o)ALS, which require several passes over the data, (o)RRMCEX approaches the minimum in around 6,000 iterations. Moreover, it achieves the smallest NMSE of 0.0004, which is slightly below the 0.0011 obtained by SGD. Fig. 3.10b shows the results for the temperature matrix without noise. Again, we observe that (o)RRMCEX converges the fastest to the minimum, whereas SGD requires many passes through the data before it starts descending, while (o)ALS converges much faster than with the synthetic data. Regarding the NMSE, (o)RRMCEX and SGD achieve the same minimum value.

The tests on the Mushroom dataset are run with $s = 10{,}000$ ($P_s = 0.033\%$) and $s = 20{,}000$ ($P_s = 0.036\%$) observations following the same procedure as with the synthetic and temperature datasets. Fig. 3.11 shows results for the Mushroom adjacency matrix with the error for $s = 20{,}000$ plotted in solid lines, and for $s = 10{,}000$ in dotted lines. We observe that for $s = 20{,}000$, (o)RRMCEX crosses the minimum of (o)ALS and SGD in 7 seconds,

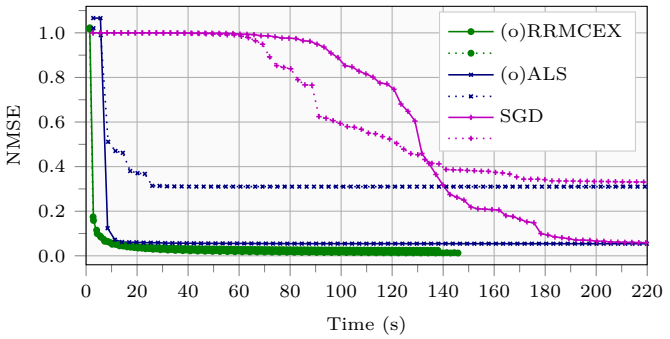


Figure 3.11: NMSE vs time for the mushroom adjacency matrix. Solid lines denote $s = 20{,}000$, and dotted lines denote $s = 10{,}000$. Each mark denotes 10,000 iterations have passed.

whereas (o)ALS and SGD converge to this minimum in 12 and 200 seconds, respectively. Afterwards, the line for (o)RRMCEX keeps descending until an error of 0.012 is reached. For $s = 10{,}000$ the convergence time of (o)RRMCEX and SGD remains almost unchanged, whereas for (o)ALS it increases to 26 seconds. Moreover, the error of both (o)ALS and SGD grows much larger, whereas (o)RRMCEX exhibits just a small increase.

3.7 Conclusions

This chapter has taken a comprehensive look at MC under the framework of RKHS. It has viewed columns and rows of the data matrix as functions from an RKHS, and leveraged kernel theory to account for the available prior information on the contents of the sought matrix. Moreover, two estimation algorithms have been developed which offer simplicity and speed as their main advantages. When the number of observed data is small, KKMCEX obtains the full matrix estimate by inverting a reduced-size matrix thanks to the Representer Theorem. On the other hand, when the number of observations is too large for KKMCEX to handle, RRMCEX can be employed instead in order to lower the computational cost with no impact on the recovery error when noise is present. In addition, RRMCEX can be easily turned into an online method implemented via SGD iterations. Compared to mainstream methods designed for the factorization-based formulation, namely ALS and SGD, RRMCEX exhibited improved performance in simulated and real data sets.

By casting aside the low-rank constraints, the proposed algorithms prove to be efficient and equally accurate as existing kernel-based approaches. Moreover, theoretical analysis has assessed the MSE of the estimator, and revealed an approach to sample selection through the Nyström approximation. This avenue is explored in depth in Chapter 5, which presents efficient methods to design optimal sampling strategies in RKHSs.

4

Generalization error bounds for matrix completion and extrapolation

When analyzing the performance of MC algorithms, several works, e.g. [26, 36, 47, 75], provide sample complexity bounds; that is, the evolution of the distance to the optimum across the number of samples and iterations. Other analyses are based on the generalization error (GE) [39, 76, 77], which reports the estimation error to be expected when predicting the output for an input not seen in the training set.

In the process of training a machine learning model, the objective is to obtain a function able to predict labels for unseen data $\boldsymbol{x} \notin \bar{\mathcal{X}}$, where $\bar{\mathcal{X}} \subseteq \mathcal{X}$ denotes the training input set, with low error. Ideally, this error should be small across the whole $\mathcal{X}$ or different subsets of $\mathcal{X}$. If true, this indicates a prediction function with good generalization properties, which means that one can expect a similar overall performance when the function is applied to two different sets. There exist two settings in which the generalization properties are to be assessed: inductive and transductive. Both settings are usually cast in the semisupervised learning class [78], with the following differences between them. The inductive setting adheres to a probabilistic characterization of the inputs, which are all drawn from a possibly known distribution $\mathcal{D}$. Moreover, its aim is to learn a general map $\mathcal{X} \to \mathbb{R}$ that can handle inputs not seen during training. In this case, the GE measures the difference between the expected value of the loss function, and the loss on a given random dataset drawn from $\mathcal{D}$ [25]. On the other hand, the transductive setting assumes no known distribution for any input set and strives to predict only a set of known inputs. Indeed, transductive learning [79] describes how most machine learning is performed: a training set, which may include unlabelled inputs, is used to learn a function which is then evaluated on a testing set on which the prediction

error must be minimized. Hence, the purpose is to transfer knowledge from the training set onto the testing set and, consequently, the transductive GE is defined as the difference between the loss function evaluated on the training and testing datasets [76, 79].

Since in MC there is no assumed known $\mathcal{D}$ for the matrix entries, and the set to be predicted is always known beforehand, it falls within transductive learning. The present chapter deals with transductive GE analysis in MC with prior information, and derives GE bounds based on the transductive Rademacher complexity [79]. Moreover, it presents numerical tests demonstrating that the transductive GE of KKMCEX is less dependent on matrix size, thus making it more reliable when dealing with large matrices with a few observations.

This chapter is organized as follows. Section 4.1 presents an introduction to inductive GE analysis, while Section 4.2 applies the transductive GE analysis to the standard MC, kernel MC (KMC) and KKKMCEX algorithms. Then, Section 4.3 describes the numerical tests, and Section 4.4 offers conclusions.

4.1 Inductive generalization error

Consider a loss function $L : \mathcal{X}^u \to \mathbb{R}$, where $\mathcal{X}^u$ denotes the Cartesian power $\mathcal{X}^u := \{(x_1, \ldots, x_u) : x_i \in \mathcal{X}\ \forall i = 1, \ldots, u\}$ and $u \leq |\mathcal{X}|$. Moreover, let $\mathcal{X}_u := \{x_1, \ldots, x_u\} \subseteq \mathcal{X}$ denote a set of u random variable inputs drawn each from $\mathcal{X}$ under the distribution $\mathcal{D}$ with replacement. The function $L(\cdot)$ is defined as the function that, after having learned a function f on a training set, measures the loss over the set of random variables $\mathcal{X}_u$. For instance, one could use the average loss $L(\mathcal{X}_u) = \frac{1}{u}\sum_{i=1}^{u} l(x_i)$ or weighted loss $L(\mathcal{X}_u) = \sum_{i=1}^{u} v_i l(x_i)$, where $l : \mathcal{X} \to \mathbb{R}$ is an element-wise function such as the square loss and $v_i \in \mathbb{R}$. Assuming that $L(\cdot), l(\cdot) > 0$, it is desirable to have $L(\mathcal{X}_u) - \mathbb{E}\{L(\mathcal{X}_u)\} \approx 0$ so that $L(\cdot)$ is statistically stable and its measured loss over any random draw is close to its expected value. The theorem below provides a probabilistic bound for this difference.

Theorem 4.1. ***McDiarmid's Inequality*** *[25]. Assume that $L(\cdot)$ satisfies*

$$\sup_{x_1,\ldots,x_u,\tilde{x}_j} |L(x_1, \ldots, x_u) - L(x_1, \ldots, x_{j-1}, \tilde{x}_j, x_{j+1}, \ldots, x_u)| \leq c_j, \forall 1 \leq j \leq u, \tilde{x}_j \in \mathcal{X}. \quad (4.1)$$

Then, for all $\epsilon > 0$

$$P\{L(\mathcal{X}_u) - \mathbb{E}\{L(\mathcal{X}_u)\} \geq \epsilon\} \leq \exp \frac{-2\epsilon^2}{\sum_{i=1}^{u} c_i^2}. \quad (4.2)$$

McDiarmid's Inequality first assumes a bound on the change when one item in the drawn

input set is swapped with a newly drawn input from $\mathcal{X}$, and then applies a concentration inequality to bound the difference between the loss on $\mathcal{X}_u$ and its expected value. A concentration inequality provides a bound on the difference between a random variable and its expected value [80], and these are used in many contexts [39, 64] to obtain algorithm performance bounds. Note that (4.2) does not account for the possibility that $L(\mathcal{X}_u) - \mathbb{E}\{L(\mathcal{X})\} < 0$ since this would be a desirable outcome. The example below shows how McDiarmid's Inequality can be used to bound the GE[1] for a solution to the least squares regression problem.

Example: least squares GE. Let us assume a function $f^* : \mathcal{X} \subseteq \mathbb{R}^D \to \mathbb{R}$ defined as $f^*(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{w}^*$, where $\boldsymbol{w}^*$ is fixed and has been obtained via least squares regression on a training dataset. Given the input set $\mathcal{X}_u := \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_u\} \subseteq \mathcal{X}$ and observation set $\{y_i\}_{i=1}^u$ with $y_i \in \mathbb{R}$ associated with $\boldsymbol{x}_i$, the loss function of the least squares problem $L^* : \mathcal{X}^u \to \mathbb{R}$ for the optimal coefficient vector $\boldsymbol{w}^*$ is

$$L^*(\mathcal{X}_u) = \frac{1}{u} \sum_{i=1}^{u} (y_i - \boldsymbol{x}_i^T \boldsymbol{w}^*)^2. \tag{4.3}$$

For any draw $\mathcal{X}_u$, assume the bound $\left|(y_j - \boldsymbol{x}_j^T \boldsymbol{w}^*)^2 - (\tilde{y}_j - \tilde{\boldsymbol{x}}_j^T \boldsymbol{w}^*)^2\right| \leq B \; \forall j = 1, \ldots, u$, $\tilde{\boldsymbol{x}}_j \in \mathcal{X}$, $B \geq 0$ and bounded $y_j, \tilde{y}_j \in \mathbb{R}$. Then, the upper bound for the loss difference after an element swap in $\mathcal{X}_u$ is

$$\sup_{\mathcal{X}_u, \tilde{\boldsymbol{x}}_j} |L^*(\mathcal{X}_u) - L^*(\mathcal{X}_u \setminus \{\boldsymbol{x}_j\} \cup \{\tilde{\boldsymbol{x}}_j\})| = \sup_{\mathcal{X}_u, \tilde{\boldsymbol{x}}_j} \frac{1}{u} \left|(y_j - \boldsymbol{x}_j^T \boldsymbol{w}^*)^2 - (\tilde{y}_j - \tilde{\boldsymbol{x}}_j^T \boldsymbol{w}^*)^2\right| \leq \frac{B}{u} \tag{4.4}$$

$\forall j = 1, \ldots, u$. Applying McDiarmid's Inequality, it holds that

$$P\{L^*(\mathcal{X}_u) - \mathbb{E}\{L^*(\mathcal{X}_u)\} \geq \epsilon\} \leq \exp \frac{-2u\epsilon^2}{B^2}. \tag{4.5}$$

Equating the right-hand side of (4.5) to δ, which denotes the probability of failure with $\delta \in (0, 1)$, we rewrite $\epsilon = \frac{B}{\sqrt{2u}} \sqrt{\ln \frac{1}{\delta}}$. Finally, the difference between the empirical loss and its expected value is bounded as

$$L^*(\mathcal{X}_u) - \mathbb{E}\{L^*(\mathcal{X}_u)\} \leq \frac{B}{\sqrt{2u}} \sqrt{\ln \frac{1}{\delta}} \tag{4.6}$$

with probability $1 - \delta$. □

[1] GE refers to inductive GE, whereas TGE refers to transductive GE.

As shown in the example, McDiarmid's Inequality enables the derivation of a bound on the deviation from the mean for a single function. However, in optimization problems the solution is chosen among a set of possible solutions, the size of which may depend on several parameters. For instance, the least squares minimization problem chooses from a set of candidates in a hypothesis class

$$\mathcal{F} := \{f : f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{w},\, \boldsymbol{w} \in \mathbb{R}^D\} \tag{4.7}$$

which induces the class of loss functions $\mathcal{L} = \{l : l(\boldsymbol{x}) = (y_i - \boldsymbol{x}^T \boldsymbol{w})^2, \boldsymbol{w} \in \mathbb{R}^D\}$. Therefore, it would be practical to be able to analyze the generalization properties of any possible solution in the class without having to solve the problem first to find $\boldsymbol{w}^*$. To this end, we have the bound given by the theorem below.

Theorem 4.2. *[25] Let $\mathcal{L}$ be a class of mappings from $\mathcal{X}$ to $[0,1]$ with $x \in \mathcal{X} \sim \mathcal{D}$. Then, w.p. $1-\delta$ over a random draw $\mathcal{X}_u \subseteq \mathcal{X}$, every $l \in \mathcal{L}$ satisfies*

$$\mathbb{E}\{l(x)\} - \frac{1}{u}\sum_{x \in \mathcal{X}_u} l(x) \leq R_u(\mathcal{L}) + \sqrt{\frac{\ln \frac{2}{\delta}}{2u}} \tag{4.8}$$

$$\leq \hat{R}_u(\mathcal{L}) + 3\sqrt{\frac{\ln \frac{2}{\delta}}{2u}} \tag{4.9}$$

where $\hat{\mathcal{R}}_u(\mathcal{L})$ is the so-called empirical Rademacher complexity of $\mathcal{L}$ over $\mathcal{X}_u$[2], $R_u(\mathcal{L}) = \mathbb{E}\{\hat{R}_u(\mathcal{L})\}$ is the Rademacher complexity, and the difference $\mathbb{E}\{l(x)\} - \frac{1}{u}\sum_{x \in \mathcal{X}_u} l(x)$ is the inductive GE for $\mathcal{L}$.

Theorem 4.2 shows that the difference between the expected value and the empirical mean of the loss function $l(\cdot)$, i.e. the GE, is bounded by the Rademacher complexity (RC) plus a term that decays with u. The fact that the range of $\mathcal{L}$ is restricted to $[0,1]$ is not relevant since l can be scaled to adapt its range to a new interval. Moreover, note that the left-hand side of (4.8) is the same as $\mathbb{E}\{L(\mathcal{X}_u)\} - L(\mathcal{X}_u)$ when $L(\mathcal{X}_u) = \frac{1}{u}\sum_{i=1}^{u} l(x_i)$ provided that the variables in $\mathcal{X}_u$ are iid. The empirical RC is an empirical measure of the capacity of a function class, and is defined as

$$\hat{R}_u(\mathcal{L}) = \mathbb{E}_\theta \left\{ \sup_{l \in \mathcal{L}} \left| \frac{2}{u} \sum_{i=1}^{u} \theta_i l(x_i) \right| \right\},\ x_i \in \mathcal{X}_u \tag{4.10}$$

where $\{\theta_i\}_{i=1}^u$ are the so-called Rademacher variables, which take values in $\{-1, 1\}$ with probability 0.5 each, and $\mathbb{E}_\theta$ denotes the expected value over the Rademacher variables.

[2]The shorthand notation $\hat{\mathcal{R}}_u(\mathcal{L}) := \hat{\mathcal{R}}(\mathcal{L}, \mathcal{X}_u)$, where $\hat{\mathcal{R}}$ also denotes empirical RC, is used as in [25].

Eq. (4.10) measures the maximum correlation between a function in $\mathcal{L}$ and the sequence of noise $\theta_1, \theta_2, \ldots$, with a high correlation indicating high capacity for $\mathcal{L}$. A high capacity amounts to having a higher chance of finding a function $l \in \mathcal{L}$ that fits any set of observations [25]. Take for instance polynomial regression: allowing a high degree polynomial can fit any set of data points, whereas with a lower degree the fitting error is increased. See Fig. 4.1 for a depiction of this example. If the capacity of $\mathcal{L}$ is too high, then optimizing within it will yield solutions with a very small error on any training dataset, but poor performance when tested on a different testing set. Hence, it is necessary to limit the capacity of the class in order to prevent overfitting; this is done through the addition of regularization terms to the optimization function.

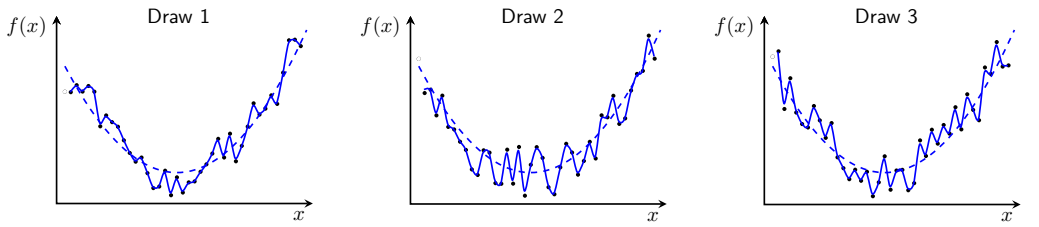


Figure 4.1: Optimization over three different data draws for the classes of quadratic (dashed line) and high degree polynomial (solid line) functions. The polynomial class has such capacity that it always contains a function that closely fits the observations. Thus, applying the function from the first draw onto the second or third would yield high error. The quadratic class has lower capacity but does not overfit, hence any solution can be applied to any draw and good performance is expected.

Note that the class $\mathcal{L}$ is actually a composite $l \circ \mathcal{F}$ since in optimization problems one usually tries to find

$$\min_{f \in \mathcal{F}} \frac{1}{s} \sum_{\bar{x} \in \bar{\mathcal{X}}} l(f(\bar{x})) \tag{4.11}$$

where f is the function to be learned within a class $\mathcal{F}$, and $\bar{\mathcal{X}} \subseteq \mathcal{X}$ is the set of training data with $|\bar{\mathcal{X}}| = s$. Instead of obtaining the RC for $\mathcal{L}$, one can leverage the Ledoux-Talagrand contraction inequality for $l \circ \mathcal{F}$ with $l(\cdot)$ Lipschitz-continuous with constant C and $l(0) = 0$, which states that

$$\hat{R}_u(l \circ \mathcal{F}) \leq 2C\hat{R}_u(\mathcal{F}). \tag{4.12}$$

Hence, the RC measure of the class $\mathcal{F}$ is also valuable and often easier to calculate such as in the case of the least squares hypothesis class[3] (4.7). Then, assuming that $\mathcal{L} := l \circ \mathcal{F}$, the

[3]For the loss function class induced by (4.7) to be Lipschitz continuous, $|y_i|$ and $|\boldsymbol{x}^T\boldsymbol{w}|$ must be bounded.

GE bound in (4.9) becomes

$$\mathbb{E}\{l(x)\} - \frac{1}{u}\sum_{x\in\mathcal{X}_u} l(x) \leq 2C\hat{R}_u(\mathcal{F}) + 3\sqrt{\frac{\ln\frac{2}{\delta}}{2u}}. \tag{4.13}$$

This section has introduced the necessary tools to evaluate the GE of a learned function f in a class $\mathcal{F}$ in the inductive setting. However, in real applications the learning process is carried out differently and is split into a training and testing phase; this would fall within the transductive learning approach. Moreover, the distribution $\mathcal{D}$ is usually unknown, which renders Theorem 4.2 unable to measure the GE. Therefore, the next section presents the transductive approach to GE and its application to MC.

4.2 Generalization error in MC

The inductive GE does not fit the MC framework because it requires that: i) the data distribution is known; and, ii) the entries are sampled with replacement. In order to come up with distribution-free claims for MC, one may resort to the transductive GE analysis [79]. In the transductive scenario, the set of s observations Ω is redefined as the set $\Omega = \Omega_t \cup \Omega_v$ comprising the union of the training set Ω_t and the testing set Ω_v, where $|\Omega_t| = s_t$, $|\Omega_v| = s_v$ and $s = s_t + s_v$. These data are taken without repetition, and the objective is to minimize the loss on the testing set; with a diverse enough testing set, good performance on it should carry over to unseen points not in Ω. This section derives bounds for the transductive GE of base MC in (4.14), kernel-based MC (KMC) in (4.15) and KKMCEX in (4.16) algorithms shown in Table 4.1 in the transductive setting. Additionally, while this work focuses on MC, the KMC formulation is similar to tensor completion [81], which has a regularization term per dimension. Similarly, KKMCEX can be formulated with tensors in mind by forming $\boldsymbol{K}_f$ as the Kronecker product of three or more matrices. Therefore, the analysis of the ensuing section readily carries over to tensors as well, although this is out of the scope of this thesis.

Consider rewriting MC in the general form

$$\hat{\boldsymbol{F}} = \arg\min_{\boldsymbol{F}\in\mathcal{F}} \frac{1}{s_t}\sum\nolimits_{(i,j)\in\Omega_t} l(\boldsymbol{M}_{i,j}, \boldsymbol{F}_{i,j}) \tag{4.17}$$

where $l : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ denotes the loss function, and $\mathcal{F}$ is the hypothesis class shaped by constraints or regularization terms. For instance, choosing the square loss and setting the class to the set of matrices with a nuclear norm smaller than a constant t results in the base MC formulation (2.15). Thus, the transductive GE (TGE) is the difference between the

MC	$\underset{\substack{\boldsymbol{W}\in\mathbb{R}^{N\times p}\\ \boldsymbol{H}\in\mathbb{R}^{L\times p}}}{\arg\min} \left\|P_{\Omega_t}(\boldsymbol{M}-\boldsymbol{W}\boldsymbol{H}^T)\right\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \|\boldsymbol{W}\|_{\mathrm{F}}^2+\|\boldsymbol{H}\|_{\mathrm{F}}^2 \leq t$ (4.14)
KMC	$\underset{\substack{\boldsymbol{B}\in\mathbb{R}^{N\times p}\\ \boldsymbol{C}\in\mathbb{R}^{L\times p}}}{\arg\min} \left\|P_{\Omega_t}(\boldsymbol{M}-\boldsymbol{K}_w\boldsymbol{B}\boldsymbol{C}^T\boldsymbol{K}_h)\right\|_{\mathrm{F}}^2 \quad \text{s.t. } \mathrm{Tr}(\boldsymbol{B}^T\boldsymbol{K}_w\boldsymbol{B})+\mathrm{Tr}(\boldsymbol{C}^T\boldsymbol{K}_h\boldsymbol{C}) \leq t_B$ (4.15)
KKMCEX	$\underset{\bar{\boldsymbol{\gamma}}\in\mathbb{R}^{s_t}}{\arg\min} \left\|\bar{\boldsymbol{m}}-\bar{\boldsymbol{K}}_f\bar{\boldsymbol{\gamma}}\right\|_2^2 \quad \text{s.t. } \bar{\boldsymbol{\gamma}}^T\bar{\boldsymbol{K}}_f\bar{\boldsymbol{\gamma}} \leq b$ (4.16)

Table 4.1: MC algorithms rewritten as equivalent convex optimization problems for the TGE analysis, where $t, t_B, b \in \mathbb{R}^+$. KKMCEX uses the sampling matrix $\boldsymbol{S}_t$ which selects the training observations in Ω_t so that $\bar{\boldsymbol{K}}_f = \boldsymbol{S}_t\boldsymbol{K}_f\boldsymbol{S}_t^T$.

testing and training loss functions

$$\frac{1}{s_v}\sum\nolimits_{(i,j)\in\Omega_v} l(\boldsymbol{M}_{i,j},\hat{\boldsymbol{F}}_{i,j}) - \frac{1}{s_t}\sum\nolimits_{(i,j)\in\Omega_t} l(\boldsymbol{M}_{i,j},\hat{\boldsymbol{F}}_{i,j}). \tag{4.18}$$

By making this difference small, it is ensured that $\hat{\boldsymbol{F}}$ has good generalization properties, meaning a similar empirical loss on a different testing set of samples can be expected. Since MC algorithms find their solution among a class of matrices under different restrictions or hypotheses, we are interested in bounding (4.18) for any matrix in the solution space. Moreover, the analysis of (4.18) provides insight into how the parameters of each MC formulation impact on the TGE. Before presenting such results, the notion of transductive Rademacher complexity (TRC) must be introduced as follows.

Definition 4.1. ***Transductive Rademacher complexity*** *[79] Given a set $\Omega = \Omega_t \cup \Omega_v$ with $q := \frac{1}{s_t} + \frac{1}{s_v}$, the TRC of a matrix class $\mathcal{F}$ is*

$$R_s(\mathcal{F}) = q\mathbb{E}_{\boldsymbol{\theta}}\left\{\sup_{\boldsymbol{F}\in\mathcal{F}}\sum\nolimits_{(i,j)\in\Omega}\theta_{i,j}\boldsymbol{F}_{i,j}\right\} \tag{4.19}$$

where $\theta_{i,j}$ is a Rademacher random variable that takes values in $\{-1,1\}$ with prob. 0.5 each. We may also write (4.19) in vectorized form as $R_s(\mathcal{F}) = q\mathbb{E}_{\boldsymbol{\theta}}\left\{\sup_{\boldsymbol{F}\in\mathcal{F}}\boldsymbol{\theta}^T\mathrm{vec}(\boldsymbol{F})\right\}$, where $\boldsymbol{\theta} = \mathrm{vec}(\boldsymbol{\Theta})$, and $\boldsymbol{\Theta}\in\mathbb{R}^{N\times L}$ has entries $\boldsymbol{\Theta}_{i,j} = \theta_{i,j}$ if $(i,j)\in\Omega$, and $\boldsymbol{\Theta}_{i,j} = 0$ otherwise. □

TRC measures the expected maximum correlation between any function in the class and the random vector $\boldsymbol{\theta}$. Intuitively, the greater this correlation is, the greater is the chance of finding a solution in the hypothesis class that will fit any observation draw, that is, $\hat{\boldsymbol{F}}_{i,j} \simeq \boldsymbol{M}_{i,j} \forall\,(i,j)\in\Omega$. For instance, for a hypothesis class containing very smooth matrices only, the TRC would be low since all entries would have similar values and thus

the expectation in (4.19) would be close to 0. On the other hand, for an unconstrained class containing every possible matrix in $\mathbb{R}^{N\times L}$, the TRC would be infinite. Although TRC measures the ability to fit both the testing and training data at once, a model for $\boldsymbol{F}$ is learnt using only the training data indexed by Ω_t. Moreover, while having a small loss across all entries in Ω is desirable, making it too small can lead to overfitting, and an increased error when predicting entries outside Ω. Using the TRC, the TGE is bounded as follows.

Theorem 4.3. *[79] Let $\mathcal{F}$ be a matrix hypothesis class. For a loss function l with Lipschitz constant γ, and any $\boldsymbol{F}\in\mathcal{F}$, it holds with probability of at least $1-\delta$ that*

$$\begin{aligned}&\frac{1}{s_v}\sum\nolimits_{(i,j)\in\Omega_v} l(\boldsymbol{M}_{i,j},\boldsymbol{F}_{i,j}) - \frac{1}{s_t}\sum\nolimits_{(i,j)\in\Omega_t} l(\boldsymbol{M}_{i,j},\boldsymbol{F}_{i,j})\\ &\leq R_s(l\circ\mathcal{F}) + 5.05q\sqrt{\min(s_t,s_v)} + \sqrt{2q\ln(1/\delta)}\,.\end{aligned} \tag{4.20}$$

Theorem 4.3 asserts that, in order to bound the TGE, it suffices to bound the TRC. Moreover, using the contraction property in [79], Lemma 5, which states that $R_s(l\circ\mathcal{F})\leq CR_s(\mathcal{F})$, where C is the Lipschitz constant of $l(\cdot)$, only the TRC of $\mathcal{F}$ is needed. Given that the same loss function is used in MC, KMC and KKMCEX, in order to assess the TGE upper bound of the three methods, we will pursue the TRC, i.e., $R_s(\mathcal{F})$, for the hypothesis class of each algorithm. Moreover, since KKMCEX has been proven specially useful for large matrices with a very small number of samples, the analysis will focus on how the matrix size and size of the training and testing sets affect the TGE.

4.2.1 Generalization error for base MC

In the base MC formulation (4.14), the hypothesis class contains the set of matrices with bounded nuclear norm, hence defined as

$$\mathcal{F}_{MC} := \{\boldsymbol{F} : \|\boldsymbol{F}\|_* \leq t,\ t\in\mathbb{R}^+\} \tag{4.21}$$

As derived in [76], the TRC for this class of matrices is bounded as

$$R_s(\mathcal{F}_{MC}) \leq q\mathbb{E}_{\boldsymbol{\theta}}\left\{\sup_{\boldsymbol{F}\in\mathcal{F}_{MC}} ||\boldsymbol{\Theta}||_2\,\|\boldsymbol{F}\|_*\right\} \leq Gqt(\sqrt{N}+\sqrt{L}) \tag{4.22}$$

where G is a universal constant.

Since $q=\frac{1}{s_t}+\frac{1}{s_v}$, the bound in (4.22) decays as $\mathcal{O}(\frac{1}{s_t}+\frac{1}{s_v})\subseteq\mathcal{O}\left(1/\min(s_t,s_v)\right)$ for fixed t, N and L. However, the TGE does not since the sum of the second and third terms on the right-hand side of (4.20) decays as $\mathcal{O}(1/\sqrt{\min\left(s_t,s_v\right)})$. Thus, the size of the training and

testing datasets should be proportional, e.g., $s = 2s_t = 2s_v$, for the TGE bound to diminish with the number of samples as $\mathcal{O}(1/\sqrt{s})$.

For non-fixed N and L, the TRC bound (4.22) also scales with the matrix dimensions. Moreover, note that the nuclear norm is $\mathcal{O}(\sqrt{NL})$ since $\|\boldsymbol{F}\|_{\mathrm{F}} \leq \|\boldsymbol{F}\|_* \leq \sqrt{r}\,\|\boldsymbol{F}\|_{\mathrm{F}}$. Therefore, t should also grow with N and L in order to match the hypothesis class to the matrix size, and obtain a good estimate of $\boldsymbol{F}$. Hence, for varying N, L and s with $s_t = s_v$, the TGE bound is $\mathcal{O}(\frac{1}{\sqrt{s}} + \frac{N\sqrt{L}+L\sqrt{N}}{s})$. This implies that increasing N or L results in a larger TGE bound regardless of the value of s, whereas increasing s_t and s_v in the same proportion results in a smaller TGE bound.

Although the TRC bound indicates growth with matrix size, we observe in (4.20) that a constant TGE bound is feasible when the set $\{\boldsymbol{F}_{i,j}\}_{(i,j)\in\Omega}$ does not change across different hypothesis classes; this is also observable in the definition of the TRC (4.19). For base MC, this happens when an empty column or row is added to $\boldsymbol{M}$ since the new row in $\hat{\boldsymbol{W}}$ or $\hat{\boldsymbol{H}}$ will be an all-zero vector. Therefore, if Ω does not change between sizes, the non-zero coefficients in $\hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{H}}$ will be the same and so will be the TGE. An example of such scenario is a recommender system, where new items are added without having been rated by any user; this is also known as zero-shot learning. Nevertheless, note that base MC cannot recover all-zero vectors, which will increase the overall prediction error regardless of the TGE on Ω.

4.2.2 Generalization error for KMC

Unlike base MC that minimizes the nuclear norm of the data matrix, KMC does not directly employ the rank in its objective function. Instead, it imposes constraints on the maximum norm of the factor matrices in their respective RKHSs. Hence, the TRC for KMC is bounded as follows.

Theorem 4.4. *If the KMC hypothesis class is*

$$\mathcal{F}_K := \left\{\boldsymbol{F} : \boldsymbol{F} = \boldsymbol{K}_w\boldsymbol{B}\boldsymbol{C}^T\boldsymbol{K}_h, \mathrm{Tr}(\boldsymbol{B}^T\boldsymbol{K}_w\boldsymbol{B}) + \mathrm{Tr}(\boldsymbol{C}^T\boldsymbol{K}_h\boldsymbol{C}) < t_B,\ t_B \in \mathbb{R}^+\right\} \tag{4.23}$$

then

$$R_s(\mathcal{F}_K) \leq \lambda_{\max} G q t_B \left(\sqrt{N} + \sqrt{L}\right) \tag{4.24}$$

where $\lambda_{\max}$ is the largest eigenvalue of $\boldsymbol{K}_w$ and $\boldsymbol{K}_h$.

Proof. Given the factorization $\boldsymbol{F} = \boldsymbol{W}\boldsymbol{H}^T$, rewrite the nuclear norm in (4.22) as

$$\begin{aligned}\|\boldsymbol{F}\|_* &\leq \frac{1}{2}\left(\|\boldsymbol{W}\|_{\mathrm{F}}^2 + \|\boldsymbol{H}\|_{\mathrm{F}}^2\right) = \frac{1}{2}(\mathrm{Tr}(\boldsymbol{B}^T\boldsymbol{K}_w^2\boldsymbol{B}) + \mathrm{Tr}(\boldsymbol{C}^T\boldsymbol{K}_h^2\boldsymbol{C})) \\ &\leq \frac{\lambda_{\max}}{2}[\mathrm{Tr}(\boldsymbol{B}^T\boldsymbol{K}_w\boldsymbol{B}) + \mathrm{Tr}(\boldsymbol{C}^T\boldsymbol{K}_h\boldsymbol{C})] \leq \frac{\lambda_{\max}t_B}{2} \end{aligned} \tag{4.25}$$

where we used that $\mathrm{Tr}(\boldsymbol{B}^T\boldsymbol{K}_w^2\boldsymbol{B}) = \sum_{i=1}^N \boldsymbol{b}_i^T\boldsymbol{K}_w^2\boldsymbol{b}_i$ with $\boldsymbol{b}_i$ denoting the i^{th} column of $\boldsymbol{B}$, and $\boldsymbol{b}_i^T\boldsymbol{K}_w^{\frac{1}{2}}\boldsymbol{K}_w\boldsymbol{K}_w^{\frac{1}{2}}\boldsymbol{b}_i \leq \lambda_{\max}\boldsymbol{b}_i^T\boldsymbol{K}_w\boldsymbol{b}_i$. □

Theorem 4.4 establishes that the TRC bound expressions of KMC and base MC are identical within a scale. With $t_B = t$, $\lambda_{\max}$ controls whether KMC has a larger or smaller TRC bound than base MC. Thus, according to Theorem 4.4, the TGE bound for KMC shrinks with s and grows with N, L and $\lambda_{\max}$. Next, an alternative bound is derived in order to gain further insights about the factors affecting the TGE.

Consider the factorizations $\boldsymbol{K}_w = \boldsymbol{\Phi}_x\boldsymbol{\Phi}_x^T$ and $\boldsymbol{K}_h = \boldsymbol{\Phi}_y\boldsymbol{\Phi}_y^T$, where $\boldsymbol{\Phi}_x \in \mathbb{R}^{N\times d_x}$ and $\boldsymbol{\Phi}_y \in \mathbb{R}^{L\times d_y}$. Plugging these into (4.15) yields

$$\min \left\|P_{\Omega_t}(\boldsymbol{M} - \boldsymbol{\Phi}_x\boldsymbol{\Phi}_x^T\boldsymbol{B}\boldsymbol{C}^T\boldsymbol{\Phi}_y\boldsymbol{\Phi}_y^T)\right\|_{\mathrm{F}}^2 \quad \text{s.t. } \mathrm{Tr}(\boldsymbol{B}^T\boldsymbol{\Phi}_x\boldsymbol{\Phi}_x^T\boldsymbol{B}) + \mathrm{Tr}(\boldsymbol{C}^T\boldsymbol{\Phi}_y\boldsymbol{\Phi}_y^T\boldsymbol{C}) \leq t_B \tag{4.26}$$

$$= \min \left\|P_{\Omega_t}(\boldsymbol{M} - \boldsymbol{\Phi}_x\boldsymbol{A}_x\boldsymbol{A}_y^T\boldsymbol{\Phi}_y^T)\right\|_{\mathrm{F}}^2 \quad \text{s.t. } \|\boldsymbol{A}_x\|_{\mathrm{F}}^2 + \|\boldsymbol{A}_y\|_{\mathrm{F}}^2 \leq t_B \tag{4.27}$$

where $\boldsymbol{A}_x = \boldsymbol{\Phi}_x^T\boldsymbol{B}$ and $\boldsymbol{A}_y = \boldsymbol{\Phi}_y^T\boldsymbol{C}$ are coefficient matrices of size $d_x \times p$ and $d_y \times p$, respectively. Optimizing for $\{\boldsymbol{B}, \boldsymbol{C}\}$ in (4.26) or for $\{\boldsymbol{A}_x, \boldsymbol{A}_y\}$ in (4.27) yields the same $\hat{\boldsymbol{F}}$ provided that $\{\boldsymbol{\Phi}_x^T, \boldsymbol{\Phi}_y^T\}$ have full column rank. Under this assumption, consider the hypothesis class

$$\mathcal{F}_I := \left\{\boldsymbol{F} : \boldsymbol{F} = \boldsymbol{\Phi}_x\boldsymbol{A}_x\boldsymbol{A}_y^T\boldsymbol{\Phi}_y^T, \|\boldsymbol{A}_x\|_{\mathrm{F}}^2 + \|\boldsymbol{A}_y\|_{\mathrm{F}}^2 \leq t_B,\ t_B \in \mathbb{R}^+\right\} \tag{4.28}$$

which satisfies $\mathcal{F}_I = \mathcal{F}_K$. This leads to the following result.

Theorem 4.5. *If $\boldsymbol{K} = (\boldsymbol{\Phi}_y \otimes \boldsymbol{\Phi}_x)(\boldsymbol{\Phi}_y \otimes \boldsymbol{\Phi}_x)^T$, and $\boldsymbol{S}_s$ is a binary sampling matrix that selects the entries in Ω, then*

$$R_s(\mathcal{F}_I) \leq qt_B\mathrm{Tr}\left(\sqrt{\boldsymbol{S}_s\boldsymbol{K}\boldsymbol{S}_s^T}\right). \tag{4.29}$$

Proof. With $\boldsymbol{\theta} := \mathrm{vec}(\boldsymbol{\Theta})$, $b_w := \|\boldsymbol{A}_x\|_{\mathrm{F}}^2$, and $b_h := \|\boldsymbol{A}_y\|_{\mathrm{F}}^2$, we have that

$$
\begin{aligned}
R_s(\mathcal{F}_I) &= q\mathbb{E}_{\boldsymbol{\theta}}\left\{\sup_{b_w+b_h\leq t_B} \boldsymbol{\theta}^T \mathrm{vec}(\boldsymbol{\Phi}_x \boldsymbol{A}_x \boldsymbol{A}_y^T \boldsymbol{\Phi}_y^T)\right\} \\
&= q\mathbb{E}_{\boldsymbol{\theta}}\left\{\sup_{b_w+b_h\leq t_B} \boldsymbol{\theta}^T (\boldsymbol{\Phi}_y \otimes \boldsymbol{\Phi}_x)\mathrm{vec}(\boldsymbol{A}_x \boldsymbol{A}_y^T)\right\} \\
&\leq q\mathbb{E}_{\boldsymbol{\theta}}\left\{\sup_{b_w+b_h\leq t_B} \left\|\boldsymbol{\theta}^T (\boldsymbol{\Phi}_y \otimes \boldsymbol{\Phi}_x)\right\| \left\|\mathrm{vec}(\boldsymbol{A}_x \boldsymbol{A}_y^T)\right\|\right\} \\
&= q\mathbb{E}_{\boldsymbol{\theta}}\left\{\sup_{b_w+b_h\leq t_B} \sqrt{\boldsymbol{\theta}^T \boldsymbol{K}\boldsymbol{\theta}}\left\|\boldsymbol{A}_x \boldsymbol{A}_y^T\right\|_{\mathrm{F}}\right\} \\
&\leq q\mathbb{E}_{\boldsymbol{\theta}}\left\{\sup_{b_w+b_h\leq t_B} \sqrt{\boldsymbol{\theta}^T \boldsymbol{K}\boldsymbol{\theta}}\left\|\boldsymbol{A}_x\right\|_{\mathrm{F}}\left\|\boldsymbol{A}_y^T\right\|_{\mathrm{F}}\right\} \\
&\leq q t_B \sqrt{\mathbb{E}_{\boldsymbol{\theta}}\left\{\boldsymbol{\theta}^T \boldsymbol{K}\boldsymbol{\theta}\right\}} = q t_B \sqrt{\mathrm{Tr}(\boldsymbol{S}_s \boldsymbol{K} \boldsymbol{S}_s^T)}
\end{aligned}
$$

where we have used the Cauchy-Schwarz inequality, the sub-multiplicative property of the Frobenius norm, and Jensen's inequality in the first, second and third inequalities. Moreover, the last equality uses the independency between the Rademacher variables in $\boldsymbol{\theta}$, and $\boldsymbol{S}_s \boldsymbol{K} \boldsymbol{S}_s$ selects the entries in the diagonal of $\boldsymbol{K}$ corresponding to Ω. □

Theorem 4.5 shows through $\boldsymbol{S}_s$ how the choice of sampling and testing datasets impacts the TRC bound, which can be leveraged to develop optimal sampling strategies [82]. Moreover, the derived bound is also valid for the inductive MC (IMC) [53] algorithm, as detailed ahead in Remark 4.1. Finally, the theorem reveals the conditions under which the TGE bound does not grow with N and L, which are as follows.

If c denotes the maximum value of the sampled entries in the diagonal of $\boldsymbol{K}$, and $s_t = s_v$, then Theorem 4.5 provides a bound that decays as $\mathcal{O}(t_B\sqrt{\frac{c}{s}})$. The definition of $\mathcal{F}_I$ implies that t_B is determined by the Frobenius norms of $\{\boldsymbol{A}_x, \boldsymbol{A}_y\}$. Since $||\boldsymbol{A}_x||_F^2$ and $||\boldsymbol{A}_y||_F^2$ are $\mathcal{O}(d_x p)$ and $\mathcal{O}(d_y p)$, respectively, assuming an adaptive parameter t_B we have that t_B is $\mathcal{O}\left((d_x + d_y)p\right)$. Hence the TRC bound is limited by the rank of the kernel matrices, given by d_x and d_y. Therefore, the TGE bound for KMC in (4.20) scales as $\mathcal{O}\left((d_x + d_y)p\sqrt{\frac{c}{s}}\right)$, which is maintained through different N and L so long as the kernel matrices have constant rank, and c does not change.

Remark 4.1. KMC generalizes the formulation of inductive MC (IMC) in [75]. In IMC, the observation model is assumed $\boldsymbol{F}_{i,j} = \boldsymbol{x}_i \boldsymbol{Z} \boldsymbol{y}_j$, where $\boldsymbol{x}_i, \boldsymbol{x}_j$ are feature vectors and $\boldsymbol{Z} \in \mathbb{R}^{d_x \times d_y}$ is a rank p coefficient matrix. Then, $\boldsymbol{Z}$ is recovered through factorizing $\boldsymbol{Z} = \boldsymbol{A}_x \boldsymbol{A}_y^T$ and

solving

$$\underset{\substack{\boldsymbol{A}_x \in \mathbb{R}^{d_x \times p} \\ \boldsymbol{A}_y \in \mathbb{R}^{d_y \times p}}}{\arg\min} = \left\| P_{\Omega_t}(\boldsymbol{M} - \boldsymbol{X}\boldsymbol{A}_x\boldsymbol{A}_y^T\boldsymbol{Y}^T) \right\|_{\mathrm{F}}^2 + \mu \left(\|\boldsymbol{A}_x\|_{\mathrm{F}}^2 + \|\boldsymbol{A}_y\|_{\mathrm{F}}^2 \right), \tag{4.30}$$

where $\boldsymbol{X}$ and $\boldsymbol{Y}$ are the matrices stacking the feature vectors $\{\boldsymbol{x}_i\}_{i=1}^N$ and $\{\boldsymbol{y}_j\}_{j=1}^L$, respectively. Clearly, the objective function in (4.30) is (4.27) with $\boldsymbol{\Phi}_x = \boldsymbol{X}$ and $\boldsymbol{\Phi}_y = \boldsymbol{Y}$. Therefore, the objective in IMC (4.30) is the same as in KMC (4.15) when κ_w and κ_h are linear kernels with feature matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$. □

4.2.3 Generalization error for KKMCEX

KMC (4.15) and KKMCEX (4.16) provide an estimate within the same RKHS since $\hat{\boldsymbol{f}}_K = \mathrm{vec}(\boldsymbol{K}_w\hat{\boldsymbol{\Gamma}}\boldsymbol{K}_h)$, where $\hat{\boldsymbol{\Gamma}} = \mathrm{unvec}(\boldsymbol{S}_t^T\hat{\bar{\boldsymbol{\gamma}}})$ and $\boldsymbol{S}_t$ selects the observations in Ω_t. However, the complexity of the hypothesis spaces differs due to the different regularization terms: KKMCEX restricts the smoothness in $\mathcal{H}_f$, whereas KMC does so along $\mathcal{H}_w$ and $\mathcal{H}_h$. This results in a TRC bound for KKMCEX that is given by the theorem next.

Theorem 4.6. *Given the hypothesis space for KKMCEX as*

$$\mathcal{F}_R := \{\boldsymbol{F} : \boldsymbol{F} = \mathrm{unvec}(\boldsymbol{K}_f\boldsymbol{S}_t^T\bar{\boldsymbol{\gamma}}), \bar{\boldsymbol{\gamma}}^T\bar{\boldsymbol{K}}_f\bar{\boldsymbol{\gamma}} \leq b^2,\ b \in \mathbb{R}^+\} \tag{4.31}$$

it holds that

$$R_s(\mathcal{F}_R) \leq qb\sqrt{\mathrm{Tr}(\boldsymbol{S}_s\boldsymbol{K}_f\boldsymbol{S}_t^T\bar{\boldsymbol{K}}_f^{-1}\boldsymbol{S}_t\boldsymbol{K}_f\boldsymbol{S}_s^T)}. \tag{4.32}$$

Proof.

$$\begin{aligned}
R_s(\mathcal{F}_R) &= q\mathbb{E}_{\boldsymbol{\theta}} \left\{ \sup_{\bar{\boldsymbol{\gamma}}^T\boldsymbol{K}_f\bar{\boldsymbol{\gamma}} \leq b} \boldsymbol{\theta}^T\boldsymbol{K}_f\boldsymbol{S}_t^T\bar{\boldsymbol{\gamma}} \right\} \\
&= q\mathbb{E}_{\boldsymbol{\theta}} \left\{ \sup_{\bar{\boldsymbol{\gamma}}^T\bar{\boldsymbol{K}}_f\bar{\boldsymbol{\gamma}} \leq b} \boldsymbol{\theta}^T\boldsymbol{K}_f\boldsymbol{S}_t^T\bar{\boldsymbol{K}}_f^{-\frac{1}{2}}\bar{\boldsymbol{K}}_f^{\frac{1}{2}}\bar{\boldsymbol{\gamma}} \right\} \\
&\leq q\mathbb{E}_{\boldsymbol{\theta}} \left\{ \sup_{\bar{\boldsymbol{\gamma}}^T\bar{\boldsymbol{K}}_f\bar{\boldsymbol{\gamma}} \leq b} \left\| \boldsymbol{\theta}^T\boldsymbol{K}_f\boldsymbol{S}_t^T\bar{\boldsymbol{K}}_f^{\frac{-1}{2}} \right\| \left\| \bar{\boldsymbol{K}}_f^{\frac{1}{2}}\bar{\boldsymbol{\gamma}} \right\| \right\} \\
&\leq qb\mathbb{E}_{\boldsymbol{\theta}} \left\{ \left\| \boldsymbol{\theta}^T\boldsymbol{K}_f\boldsymbol{S}_t^T\bar{\boldsymbol{K}}_f^{-\frac{1}{2}} \right\| \right\} \\
&= qb\sqrt{\mathrm{Tr}(\boldsymbol{S}_s\boldsymbol{K}_f\boldsymbol{S}_t\bar{\boldsymbol{K}}_f^{-1}\boldsymbol{S}_t^T\boldsymbol{K}_f\boldsymbol{S}_s^T)}.
\end{aligned} \tag{4.33}$$

□

Supposing that $||\boldsymbol{K}_f||_\infty \leq c$, the bound in (4.32) decays as $\mathcal{O}(b\sqrt{sc}/\min(s_t, s_v))$. For $s_t = s_v$, this yields a rate $\mathcal{O}(b\sqrt{\frac{c}{s}})$. Note that, if c is constant for different N and L, then b can also be kept constant. Thus, in this case the TGE bound induced by (4.32) only scales

with the number of samples as $\mathcal{O}(\frac{1}{\sqrt{s}})$. This is feasible when the feature maps are independent of the matrix size so that for any N and L it is satisfied that $\langle \phi_z((x_i, y_j)), \phi_z(x_n, y_l) \rangle_{\mathcal{C}_z} \leq c \; \forall \; (i,j),(n,l) \in \{1,\ldots,N\} \times \{1,\ldots,L\}$; such is the case with, e.g., the linear kernel. Interestingly, although the degrees of freedom of KKMCEX (and hence the risk of overfitting) grow with s_t since $\bar{\boldsymbol{\gamma}} \in \mathbb{R}^{s_t}$, the TGE does not increase because the number of samples increases proportionally. Thus, different from baseline MC and KMC, similar performance is expected on the testing dataset regardless of the data matrix size.

4.3 Numerical tests

This section compares the TGE of base MC and KMC, solved via alternating least-squares (ALS) in Algorithm 2, with KKMCEX solved with (3.46). Besides comparing the TGE of these algorithms, the simulations assess how the matrix size impacts the TGE. To this end, first a fixed-rank synthetic data matrix with $N = L$ is generated as $\boldsymbol{F} = \boldsymbol{K}_w \boldsymbol{B}\boldsymbol{C}^T \boldsymbol{K}_h$. The kernel matrices are $\boldsymbol{K}_w = \boldsymbol{K}_h = \text{abs}(\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T)$, where $\boldsymbol{R} \in \mathbb{C}^{N \times N}$ is the DFT basis and $\boldsymbol{D} \in \mathbb{R}^{N \times N}$ is a positive diagonal matrix with decreasing values on its diagonal. The coefficient matrices $\{\boldsymbol{B}, \boldsymbol{C}\}$ have $p = 30$ columns, with entries drawn from a zero-mean Gaussian distribution with variance 1. The tests are run over 1,000 realizations. A new matrix $\boldsymbol{F}$ is generated per realization with $s_t = 1,000$ entries drawn uniformly at random, and the remaining $s_v = N^2 - s$ forming the testing dataset. The parameter μ is chosen by cross-validation for each size.

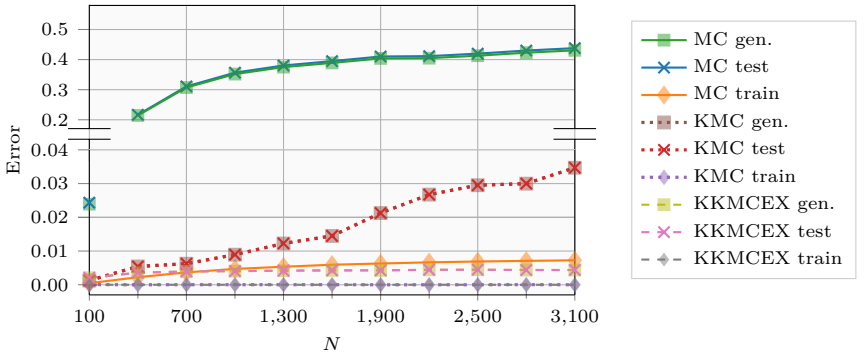


Figure 4.2: Training loss, testing loss, and generalization error vs. matrix size for noiseless synthetic data.

Fig. 4.2 shows the training, testing, and TGEs from Eq. (4.18) for the synthetic matrices. We observe for base MC that the training loss is small, whereas it is much larger on the testing dataset, and also it grows with N. Moreover, since the training loss is minimal, the

TGE coincides with the testing loss. Clearly, the base MC solution (4.14) is not able to predict the unobserved entries due to the lack of prior information that would allow for extrapolation in rows or columns with no observed entries. In addition, the TGE approaches saturation for large matrix sizes since most entries in the estimated matrix are 0, and the testing loss tends to the average $\frac{1}{s_v}\sum_{(i,j)\in\Omega_v} \boldsymbol{M}_{i,j}^2$. Regarding KMC and KKMCEX, we observe that both algorithms achieve a constant training loss. Although not visible on the plot, the training loss of KKMCEX is one order of magnitude smaller than that of KMC. On the other hand, the testing and TGE of KKMCEX are constant unlike in KMC for which both are higher and grow with N. These results confirm what was asserted by the TGE bounds in Section 4.2.

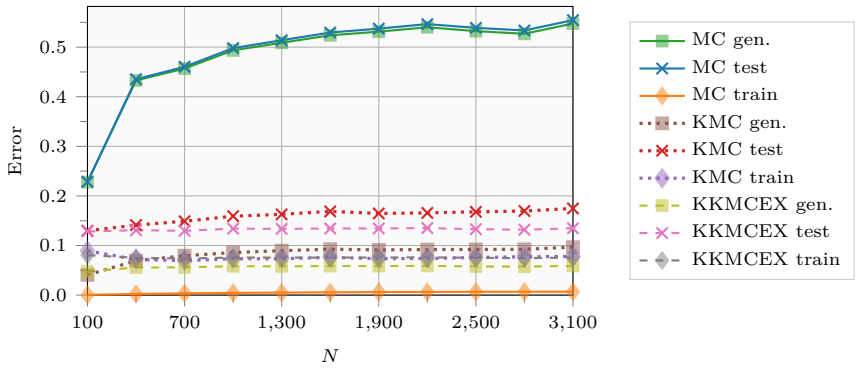


Figure 4.3: Training loss, testing loss, and generalization error vs. matrix size for noisy synthetic data.

Fig. 4.3 shows the same simulation results as Fig. 4.2, but with noisy data at $snr = 4$. We observe that MC overfits the noisy observations since the training loss is, again, very small, while the testing loss is much larger. For KMC and KKMCEX, the presence of noise increases the training and testing losses. Due to the noise, a larger μ is selected to prevent overfitting at the cost of a higher training loss. Nevertheless, the testing loss of KMC slightly grows with N. In terms of TGE, KKMCEX outperforms KMC with a lower value that tends to a constant.

Fig. 4.4 shows the numerical tests with the $150 \times L$ matrix of temperature measurements taken in 2002 by 150 weather stations in the US. The kernel matrices are the row and column covariances of the same data from 2001, and $s_t = 500$ with $s_v = 150L - s_t$. We observe that for KMC and KKMCEX both training and testing errors grow with L. However, KMC is unstable both in training and testing, whereas KKMCEX is smooth. Thus, although the TGE grows slightly for KKMCEX, it is more reliable when the number of samples is very small.

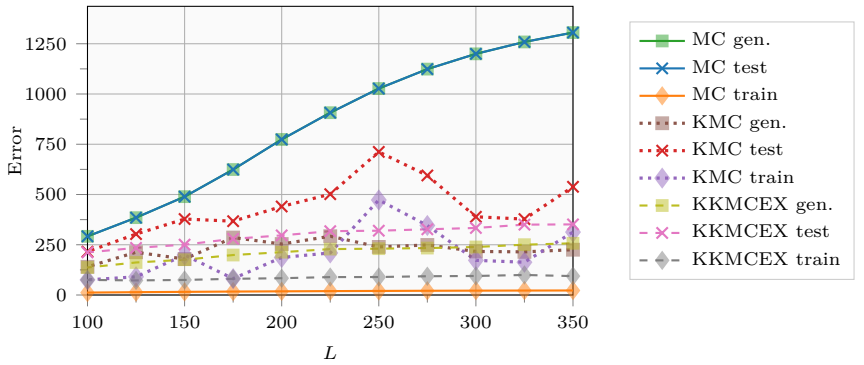


Figure 4.4: Training loss, testing loss, and generalization error vs. matrix size for temperature data.

4.4 Conclusions

This chapter has analyzed the TGE for MC with prior information following a procedure that can be utilized to additional data imputation methods after properly defining a loss function and corresponding hypothesis class. Bounds on the TRC have established that baseline MC and KMC become less reliable as the size of the matrix increases when the number of samples remains constant. On the other hand, KKMCEX offers improved analytical guarantees with a TGE that scales only with the number of samples. Moreover, numerical tests support the theoretical findings for synthetic data with known kernel matrices, and have demonstrated improved performance for KKMCEX with real data. Finally, since the RKHS framework generalizes several MC settings with prior information, the analysis herein applies also to these settings.

5

Optimal sampling in RKHSs

While in learning applications the focus is usually on achieving a small prediction error, there is also the cost of label acquisition for training. Often, the cost of labelling a training input is comparable or superior to the prediction cost. For instance, in recommender systems, a request needs to be sent to the user asking them to rate an item; this does not even guarantee a rating since the user might just ignore the request. In other areas such as sensor networks, medicine or any scenario with human annotators, the time to label an input is also non-negligible. Besides the cost of label acquisition, there is also the fact that the prediction error on new inputs is affected by the chosen training set as it was elucidated in Chapter 4. Hence, the design of optimal sample selection strategies is of paramount importance in order to improve algorithm performance.

Active sampling provides a protocol under which to choose the most promising set of inputs to label. It implements an iterative scheme that, provided the availability of a small starting training set, executes two steps repeatedly: first, the function is learned or updated using the current training set and, second, a criterion is evaluated to decide which next input should be labelled and added to the training set. There exist a variety of criteria, the most common being the evaluation of prediction uncertainty. For instance, in Gaussian processes, the posterior pdf serves as an uncertainty measure. Another approach is the query by committee, in which the function is learned with several algorithms, and the input on which the disagreement is highest is chosen as the next one to be labelled. Other approaches are based on minimizing the expected error, or choosing the input that will induce the biggest change in the function. With regards to MC, active sampling approaches follow the

aforementioned procedures adapted to the MC formulation [83–86].

While active sampling generally achieves good performance results, it may not be the case in some situations e.g., when the samples are too noisy or when online operation is not possible. There also exist batch versions of active learning which acquire the labels in groups, but the issue with noisy samples still prevails. Moreover, most active learning methods are designed for classification tasks, which often have built-in uncertainty measures, with the availability of methods for regression being more limited [87, 88].

An alternative to active sampling is provided by the passive sampling method. In passive sampling, the set of inputs to be labelled is selected observing uniquely the geometry of the input space. For instance, the greedy sampling approach in [87], iteratively adds new input points to the training set by choosing the one with the largest distance to the set. By relying only on the input space, passive sampling avoids having to iteratively recompute the learned function for every additional input; the training set can be labelled all at once when it is deemed complete. Furthermore, the impact of noisy samples is diminished

Since the KKMCEX algorithm is regression-based and operates on a fully known, although not fully labelled, set of inputs $\mathcal{X}$, passive sampling is a sensible choice to reduce the prediction error. As hinted in Lemma 3.2, the MSE depends on the difference between the kernel matrix and its Nyström approximation. Since the kernel matrix is built needing only the inputs in $\mathcal{X}$ to evaluate the kernel function, a passive sampling approach is constituted by first building the Nyström approximation with an optimal set of columns of $\boldsymbol{K}_f$. Then, the inputs in $\mathcal{X}$ corresponding to the chosen columns are labelled to form the training set. Unlike [87], this kernel-based approach enables passive sampling on any type of input set $\mathcal{X}$, including non-metric spaces. That is, as long as there exists a kernel function measuring similarity between inputs, e.g., files, users or vertices on a graph, a passive sampling strategy can be devised. This chapter introduces a probabilistic approach to passive sampling based around building an accurate Nyström approximation to the kernel matrix.

The chapter is organized as follows. Section 5.1 provides a theoretical analysis of the sampling procedure in RKHSs and proves that Nyström-based passive sampling is optimal, in the absolute error sense, for noiseless samples. Section 5.2 introduces the passive sampling approach for KKMCEX and discusses different sampling strategies to accelerate the algorithm. Finally, Section 5.3 presents numerical tests and Section 5.4 offers conclusions.

5.1 Optimal sampling in RKHSs

Let $f : \mathcal{X} \to \mathbb{R}$ be a function lying on an RKHS $\mathcal{H}_f$, $\bar{\mathcal{X}} = \{\bar{x}_i\}_{i=1}^s$ be a set of sampled points in $\mathcal{X}$ where $|\mathcal{X}| = N$, and assume noiseless observations $\{y_i\}_{i=1}^s$ such that $y_i = f(\bar{x}_i) \forall i = 1, \dots, s$.

Moreover, consider a slight departure from the train/test protocol in Chapter 4 and assume that the objective is simply to learn $f \in \mathcal{H}_f$ from $\bar{\mathcal{X}}$ and $\{y_i\}_{i=1}^s$ as in Chapter 3. This section takes the functional analysis approach on the signal reconstruction process from [89] and derives results valid for any algorithm operating in a finite dimensional RKHS.

The recovery of f from the observation set can be thought of as the "inversion" of

$$\bar{\boldsymbol{f}} = \mathcal{A}f \tag{5.1}$$

where $\bar{\boldsymbol{f}} = [f(\bar{x}_1), \ldots, f(\bar{x}_s)]^T$, and $\mathcal{A} : \mathcal{H}_f \to \mathbb{R}^s$ is a sampling operator evaluating f at $\{\bar{x}_i\}_{i=1}^s$. The reproducing property states that any functional evaluation in $\mathcal{H}_f$ obeys $f(x) = \langle f, \kappa_f(\cdot, x)\rangle_{\mathcal{H}_f}$. Thus, the sampling operator is defined as

$$\mathcal{A}f = \sum_{n=1}^{s} \langle f, \kappa_f(\cdot, \bar{x}_n)\rangle_{\mathcal{H}_f} \boldsymbol{e}_n \tag{5.2}$$

where $\boldsymbol{e}_n$ is the nth vector of the standard basis, i.e., the nth column of an $s \times s$ identity matrix. Let $\mathcal{S}$ be the subspace spanned by $\{\kappa_f(\cdot, \bar{x}_n)\}_{n=1}^s$. Then, $f = f_{\mathcal{S}} + f_{\mathcal{S}^\perp}$, where $f_{\mathcal{S}}$ is the projection of f onto $\mathcal{S}$ and $f_{\mathcal{S}^\perp}$ is the projection onto the subspace orthogonal to $\mathcal{S}$, denoted by $\mathcal{S}^\perp$. Thus, for $\bar{x} \in \bar{\mathcal{X}}$,

$$f(\bar{x}) = f_{\mathcal{S}}(\bar{x}) + f_{\mathcal{S}^\perp}(\bar{x}) = \langle f_{\mathcal{S}}, \kappa_f(\cdot, \bar{x})\rangle_{\mathcal{H}_f} + \langle f_{\mathcal{S}^\perp}, \kappa_f(\cdot, \bar{x})\rangle_{\mathcal{H}_f} = \langle f_{\mathcal{S}}, \kappa_f(\cdot, \bar{x})\rangle_{\mathcal{H}_f} = f_{\mathcal{S}}(\bar{x}). \tag{5.3}$$

Since $f_{\mathcal{S}} \in \mathcal{S}$, it can be written as $\sum_{n=1}^s \bar{\alpha}_n \kappa_f(\cdot, \bar{x}_n)$ for some real coefficients $\{\bar{\alpha}_n\}_{n=1}^s$. In turn, $f(\bar{x}) = \sum_{n=1}^s \bar{\alpha}_n \kappa_f(\bar{x}, \bar{x}_n)$. Therefore, since $\bar{\boldsymbol{f}}(i) := f(\bar{x}_i)$, $\mathcal{A}$ maps f into a new RKHS $\mathcal{H}_g \subseteq \mathbb{R}^s$ defined as

$$\mathcal{H}_g := \{\bar{\boldsymbol{f}} : \bar{\boldsymbol{f}}(i) = \sum_{n=1}^{s} \bar{\alpha}_n \kappa_f(\bar{x}_i, \bar{x}_n), \;\; \bar{\alpha}_n \in \mathbb{R}\} \tag{5.4}$$

with kernel matrix $\bar{\boldsymbol{K}} \in \mathbb{R}^{s\times s}$ where $\bar{\boldsymbol{K}}_{i,j} = \kappa_f(\bar{x}_i, \bar{x}_j)$ so that $\bar{\boldsymbol{f}} = \bar{\boldsymbol{K}}\bar{\boldsymbol{\alpha}}$, where $\bar{\boldsymbol{\alpha}} = [\bar{\alpha}_1, \ldots, \bar{\alpha}_s]^T$, and $\langle \bar{\boldsymbol{f}}, \bar{\boldsymbol{f}}'\rangle_{\mathcal{H}_g} = \bar{\boldsymbol{f}}^T \bar{\boldsymbol{K}}^{-1} \bar{\boldsymbol{f}}'$ for $\bar{\boldsymbol{f}}, \bar{\boldsymbol{f}}' \in \mathcal{H}_g$ and assuming $\bar{\boldsymbol{K}}$ invertible. With $\mathcal{A} : \mathcal{H}_f \to \mathcal{H}_g$, its adjoint operator is $\mathcal{A}^* : \mathcal{H}_g \to \mathcal{H}_f$, and it satisfies

$$\langle \mathcal{A}f, \bar{\boldsymbol{f}}\rangle_{\mathcal{H}_g} = \langle f, \mathcal{A}^*\bar{\boldsymbol{f}}\rangle_{\mathcal{H}_f} \tag{5.5}$$

which leads to the definition

$$\mathcal{A}^*\bar{\boldsymbol{f}} = \sum_{n=1}^{s} \kappa_f(\cdot, \bar{x}_n)\langle \bar{\boldsymbol{f}}, \boldsymbol{e}_n\rangle_{\mathcal{H}_g} \tag{5.6}$$

where $\langle \bar{\boldsymbol{f}}, \boldsymbol{e}_n \rangle_{\mathcal{H}_g} = \bar{\boldsymbol{f}}^T \bar{\boldsymbol{K}}^{-1} \boldsymbol{e}_n = \bar{\alpha}_n$. The operator $\mathcal{A}^*$ maps $\bar{\boldsymbol{f}}$ back into $\mathcal{H}_f$ using only the basis functions in $\mathcal{S}$, hence reversing the sampling in (5.1) and yielding an estimate for f.

After the sampling operator and its adjoint have been defined, the reconstruction of a function in $\mathcal{H}_f$ from its samples can be achieved through the consecutive application of each operator, i.e., $\hat{f} = \mathcal{A}^*\mathcal{A}f$. Hence, $\mathcal{P} = \mathcal{A}^*\mathcal{A}$ is an orthogonal projector onto $\mathcal{S}$ since it is a self-adjoint operator, i.e., $\langle \mathcal{P}f, f' \rangle_{\mathcal{H}_f} = \langle f, \mathcal{P}f' \rangle_{\mathcal{H}_f}$, and it satisfies $\mathcal{P}\mathcal{P} = \mathcal{P}$. Then, the estimate $\hat{f}$ is equal to $f_{\mathcal{S}}$ as shown below

$$\begin{aligned}
\hat{f} = \mathcal{P}f &= \mathcal{A}^* \sum_{n=1}^{s} \langle f, \kappa_f(\cdot, \bar{x}_n) \rangle_{\mathcal{H}_f} \boldsymbol{e}_n \\
&= \sum_{m=1}^{s} \kappa_f(\cdot, \bar{x}_m) \langle \sum_{n=1}^{s} \langle f, \kappa_f(\cdot, \bar{x}_n) \rangle_{\mathcal{H}_f} \boldsymbol{e}_n, \boldsymbol{e}_m \rangle_{\mathcal{H}_g} \\
&= \sum_{m=1}^{s} \kappa_f(\cdot, \bar{x}_m) \langle \bar{\boldsymbol{f}}, \boldsymbol{e}_m \rangle_{\mathcal{H}_g} \\
&= \sum_{m=1}^{s} \kappa_f(\cdot, \bar{x}_m) \bar{\boldsymbol{\alpha}}^T \bar{\boldsymbol{K}} \bar{\boldsymbol{K}}^{-1} \boldsymbol{e}_m \\
&= \sum_{m=1}^{s} \bar{\alpha}_m \kappa_f(\cdot, \bar{x}_m) = f_{\mathcal{S}}.
\end{aligned} \tag{5.7}$$

Since an orthogonal projection onto a space yields an element with minimum distance to the original function, the reconstruction error at $x \in \mathcal{X}$ is upper bounded by

$$\begin{aligned}
|f(x) - \mathcal{P}f(x)| &= |\langle f, \kappa_f(\cdot, x) \rangle_{\mathcal{H}_f} - \langle \mathcal{P}f, \kappa_f(\cdot, x) \rangle_{\mathcal{H}_f}| \\
&= |\langle f, \kappa_f(\cdot, x) \rangle_{\mathcal{H}_f} - \langle f, \mathcal{P}\kappa_f(\cdot, x) \rangle_{\mathcal{H}_f}| \\
&= |\langle f, \kappa_f(\cdot, x) - \mathcal{P}\kappa_f(\cdot, x) \rangle_{\mathcal{H}_f}| \\
&\leq \|f\|_{\mathcal{H}_f} \|\kappa_f(\cdot, x) - \mathcal{P}\kappa_f(\cdot, x)\|_{\mathcal{H}_f}
\end{aligned} \tag{5.8}$$

It can be seen that the error upper bound at a point x depends on the distance between the kernel function $\kappa_f(\cdot, x)$ and its projection onto $\mathcal{S}$. Note that zero error is obtained at the sampled points since $\mathcal{P}\kappa_f(\cdot, \bar{x}) = \kappa_f(\cdot, \bar{x})\ \forall \bar{x} \in \bar{\mathcal{X}}$. For $x \notin \bar{\mathcal{X}}$, the error can only be zero if $\kappa_f(\cdot, x) \in \mathcal{S}$; this is the case when a kernel function is duplicated such that $\kappa_f(\cdot, \bar{x}_i) = \kappa_f(\cdot, x_j)$ where $\bar{x}_i \in \bar{\mathcal{X}}$ and $x_j \notin \bar{\mathcal{X}}$. See Fig. 5.1 for an example. Assuming no such duplicities exist,

$$|f(x) - \mathcal{P}f(x)| \leq \begin{cases} 0 & \text{if } x \in \bar{\mathcal{X}} \\ \|f\|_{\mathcal{H}_f} \|\kappa_f(\cdot, x) - \mathcal{P}\kappa_f(\cdot, x)\|_{\mathcal{H}_f} & \text{if } x \notin \bar{\mathcal{X}} \end{cases} \tag{5.9}$$

$$\begin{bmatrix} 1 & 2 & 3 & 2 \\ 2 & 1 & 2 & 1 \\ 3 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 \end{bmatrix}$$

Figure 5.1: Kernel matrix for a space $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$ where $\bar{\mathcal{X}} = \{x_1, x_2, x_3\}$ and $x_4 \notin \bar{\mathcal{X}}$. The error $|f(x_4) - \mathcal{P}f(x_4)| = 0$ since $\kappa_f(\cdot, x_4) = \kappa_f(\cdot, x_2)$ and $x_2 \in \bar{\mathcal{X}}$.

The total error over the function estimate is denoted by $|f - \hat{f}| = \sum_{i=1}^{N} |f(x_i) - \hat{f}(x_i)|$, and (5.9) shows that it can only be zero when $f \in \mathcal{S}$ so that $\mathcal{P}f = f$. Moreover, since according to (5.7) $\hat{f} = f_{\mathcal{S}}$, then $\|f - \hat{f}\|_{\mathcal{H}_f} = \|f_{\mathcal{S}^\perp}\|_{\mathcal{H}_f}$. Therefore, in order to have zero error for any $f \in \mathcal{H}_f$, then $\mathcal{H}_f = \mathcal{S}$ must be true.

The second norm in (5.8) denotes the distance in $\mathcal{H}_f$ between a kernel function and its closest approximation built with the functions in $\mathcal{S}$. This fact exposes useful properties of $\mathcal{A}$ such as the one in the lemma below.

Lemma 5.1. *$\mathcal{A}$ is a norm-preserving transform in $\mathcal{H}_g$ with respect to $\mathcal{H}_f$ such that it holds that $\|f\|_{\mathcal{H}_f} = \|\mathcal{A}f\|_{\mathcal{H}_g}\ \forall\ f \in \mathcal{S}$.*

Proof. Let us expand the second term in (5.8) as

$$\begin{aligned} \|\kappa_f(\cdot, x) - \mathcal{P}\kappa_f(\cdot, x)\|^2_{\mathcal{H}_f} &= \langle \kappa_f(\cdot, x) - \mathcal{P}\kappa_f(\cdot, x), \kappa_f(\cdot, x) - \mathcal{P}\kappa_f(\cdot, x)\rangle_{\mathcal{H}_f} \\ &= \kappa_f(x, x) - 2\langle \kappa_f(\cdot, x), \mathcal{P}\kappa_f(\cdot, x)\rangle_{\mathcal{H}_f} + \langle \mathcal{P}\kappa_f(\cdot, x), \mathcal{P}\kappa_f(\cdot, x)\rangle_{\mathcal{H}_f} \\ &= \kappa_f(x, x) - 2\langle \kappa_f(\cdot, x), \mathcal{P}\kappa_f(\cdot, x)\rangle_{\mathcal{H}_f} + \langle \mathcal{A}\kappa_f(\cdot, x), \mathcal{A}\kappa_f(\cdot, x)\rangle_{\mathcal{H}_g} \\ &= \kappa_f(x, x) - 2\langle \mathcal{A}\kappa_f(\cdot, x), \mathcal{A}\kappa_f(\cdot, x)\rangle_{\mathcal{H}_g} + \langle \mathcal{A}\kappa_f(\cdot, x), \mathcal{A}\kappa_f(\cdot, x)\rangle_{\mathcal{H}_g} \\ &= \kappa_f(x, x) - \langle \mathcal{A}\kappa_f(\cdot, x), \mathcal{A}\kappa_f(\cdot, x)\rangle_{\mathcal{H}_g} \\ &= \|\kappa_f(\cdot, x)\|_{\mathcal{H}_f} - \|\mathcal{A}\kappa_f(\cdot, x)\|_{\mathcal{H}_g} \end{aligned} \tag{5.10}$$

where, using $\langle f, \mathcal{A}^*\bar{\boldsymbol{f}}\rangle_{\mathcal{H}_f} = \langle \mathcal{A}f, \bar{\boldsymbol{f}}\rangle_{\mathcal{H}_g}$ and $\mathcal{A}\mathcal{A}^* = \mathcal{I}$, the identity $\langle \mathcal{A}^*\mathcal{A}f, \mathcal{A}^*\mathcal{A}f\rangle_{\mathcal{H}_f} = \langle \mathcal{A}\mathcal{A}^*\mathcal{A}f, \mathcal{A}f\rangle_{\mathcal{H}_g}$ has been applied on the third equality, and $\langle f, \mathcal{A}^*\mathcal{A}f\rangle_{\mathcal{H}_f} = \langle \mathcal{A}f, \mathcal{A}f\rangle_{\mathcal{H}_g}$ on the fourth. The resulting expression measures the difference between the norm of $\kappa_f(\cdot, x)$ in $\mathcal{H}_f$ and its sampled counterpart in $\mathcal{H}_g$. Thus, since (5.10) must be 0 for $\kappa_f \in \mathcal{S}$, $\mathcal{A}$ is a norm-preserving transform in $\mathcal{H}_g\ \forall\ f \in \mathcal{S}$. □

The expansion in (5.10) has further implications as shown in the theorem below.

Theorem 5.1. *Let $\boldsymbol{f} = [f(x_1), \ldots, f(x_N)]^T$, and $\boldsymbol{S}$ be an $s \times N$ binary sampling matrix with a nonzero element per row equal to 1 such that $\boldsymbol{S}\boldsymbol{f} := \mathcal{A}f$. Moreover, let $\boldsymbol{K}_f$ be the kernel*

matrix of $\mathcal{H}_f$*, and* $\boldsymbol{T}_f = \boldsymbol{K}_f \boldsymbol{S}^T \bar{\boldsymbol{K}}_f^{-1} \boldsymbol{S} \boldsymbol{K}_f$*. Then, the total error across* f *is bounded as*

$$|f - \mathcal{P}f| \leq \|f\|_{\mathcal{H}_f} \operatorname{Tr}\{\boldsymbol{K}_f - \boldsymbol{T}_f\} \tag{5.11}$$

Proof. Using (5.8) and (5.10), it holds that

$$\begin{aligned}
|f - \mathcal{P}f| &= \sum_{m=1}^{N} |f(x_m) - \mathcal{P}f(x_m)| \\
&\leq \|f\|_{\mathcal{H}_f} \sum_{m=1}^{N} \|\kappa_f(\cdot, x_m) - \mathcal{P}\kappa_f(\cdot, x_m)\|_{\mathcal{H}_f} \\
&\leq \|f\|_{\mathcal{H}_f} \sum_{m=1}^{N} \left(\|\kappa_f(\cdot, x_m)\|_{\mathcal{H}_f} - \|\mathcal{A}\kappa_f(\cdot, x_m)\|_{\mathcal{H}_g} \right) \qquad (5.12) \\
&= \|f\|_{\mathcal{H}_f} \sum_{m=1}^{N} \left(\kappa_f(x_m, x_m) - \sum_{i=1}^{s} \sum_{j=1}^{s} \kappa_f(\bar{x}_i, x_m) (\bar{\boldsymbol{K}}^{-1})_{i,j} \kappa_f(\bar{x}_j, x_m) \right) \\
&= \|f\|_{\mathcal{H}_f} \operatorname{Tr}(\boldsymbol{K}_f - \boldsymbol{K}_f \boldsymbol{S}^T \bar{\boldsymbol{K}}^{-1} \boldsymbol{S} \boldsymbol{K}_f) = \|f\|_{\mathcal{H}_f} \operatorname{Tr}(\boldsymbol{K}_f - \boldsymbol{T}_f). \qquad (5.13)
\end{aligned}$$

□

Matrix $\boldsymbol{T}_f$ is the so-called Nyström approximation[1] to $\boldsymbol{K}_f$. Thus, (5.13) shows that the error upper bound is proportional to the difference between $\boldsymbol{K}_f$ and $\boldsymbol{T}_f$ on its diagonal entries. Moreover, $\operatorname{Tr}(\boldsymbol{K}_f - \boldsymbol{T}_f) \geq 0$ since in (5.10) $\|\kappa_f(\cdot, x)\|_{\mathcal{H}_f} \geq \|\mathcal{A}\kappa_f(\cdot, x)\|_{\mathcal{H}_g}$. Therefore, designing $\mathcal{A}$ to maximize $\sum_{n=1}^{N} \|\mathcal{A}\kappa_f(\cdot, x_n)\|_{\mathcal{H}_g} = \operatorname{Tr}(\boldsymbol{T}_f)$ yields the smallest bound in Theorem 5.1.

5.2 Passive sampling for KKMCEX

Section 5.1 regards the reconstruction of a sampled function by means of projection operators. Moreover, it shows that the recovery error is proportional to the difference between the kernel matrix and its Nyström approximation. Since this approximation is built from a subset of columns, which columns are chosen is crucial to keep the approximation accurate. While there exist deterministic [90] methods which order the columns according to a metric and then choose the top performers, most implementations of the Nyström approximation opt for statistical approaches. In these, a sampling probability is assigned to each column and the required number of columns is sampled according to the resulting distribution. One example

[1] $\boldsymbol{T}$ is different to the regularised Nyström approximation $\tilde{\boldsymbol{T}}_f = \boldsymbol{K}_f \boldsymbol{S}^T (\bar{\boldsymbol{K}}_f + \mu \boldsymbol{I})^{-1} \boldsymbol{S} \boldsymbol{K}_f$, except for $\mu = 0$.

of such a technique is in [64, 91], which measures column importance through the so-called leverage scores and provides strong theoretical guarantees on the approximation error.

In KKMCEX, the function to be recovered is an NL vector $\boldsymbol{f}$. The sampling operator in Section 5.1 takes the form of the sampling matrix, i.e., $\mathcal{A} := \boldsymbol{S}$, and its adjoint derived following (5.5) is $\mathcal{A}^* := \boldsymbol{K}_f \boldsymbol{S}^T (\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T)^{-1}$, which maps $\mathbb{R}^s \to \mathbb{R}^{NL}$. Therefore, following the same process as in Section 5.1, the projection operator is $\mathcal{P} = \boldsymbol{K}_f \boldsymbol{S}^T (\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T)^{-1}\boldsymbol{S}$. Looking at the KKMCEX solution in (3.46),

$$\hat{\boldsymbol{f}}_K = \boldsymbol{K}_f \boldsymbol{S}^T (\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S} + \mu\boldsymbol{I})^{-1}\boldsymbol{S}\boldsymbol{m} \tag{5.14}$$

one observes that it is not obtained by directly applying the projection operator onto $\boldsymbol{m}$ unless $\mu = 0$. However, since (5.14) lies on the column span of $\boldsymbol{K}_f\boldsymbol{S}^T$, the result is a non-orthogonal projection of $\boldsymbol{m}$ onto $\mathcal{S} = \text{span}\{\boldsymbol{K}_f\boldsymbol{S}^T\}$, i.e., the space spanned by the kernel functions evaluated at the sampled inputs. Still, a direct application of Theorem 5.1 is not possible unless $\mu = 0$ and $\boldsymbol{m}$ is noiseless.

To evaluate the performance of a specific sampling pattern, the risk function [91], which is equal to the MSE except that it takes $\boldsymbol{S}$ as a parameter, will be used. Thus, the risk is

$$\begin{aligned} R_\mu(\boldsymbol{S}) &= \mathbb{E}_{\boldsymbol{e}}\{\|\boldsymbol{f} - \hat{\boldsymbol{f}}_K\|_2^2\} \\ &= \left\|(\boldsymbol{I} - \boldsymbol{K}_f\boldsymbol{S}^T(\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T + \mu\boldsymbol{I})^{-1}\boldsymbol{S})\boldsymbol{f}\right\|_2^2 + \mathbb{E}_{\boldsymbol{e}}\left\{\left\|\boldsymbol{K}_f\boldsymbol{S}^T(\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T + \mu\boldsymbol{I})^{-1}\bar{\boldsymbol{e}}\right\|_2^2\right\} \end{aligned}$$

and, assuming $\mu = 0$, it becomes

$$\begin{aligned} R_0(\boldsymbol{S}) &= \left\|\boldsymbol{f} - \boldsymbol{K}_f\boldsymbol{S}^T(\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T)^{-1}\boldsymbol{S}\boldsymbol{f}\right\|_2^2 + \mathbb{E}_{\boldsymbol{e}}\left\{\left\|\boldsymbol{K}_f\boldsymbol{S}^T(\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T)^{-1}\boldsymbol{S}\boldsymbol{e}\right\|_2^2\right\} \\ &= \|\boldsymbol{f} - \mathcal{P}\boldsymbol{f}\|_2^2 + \mathbb{E}_{\boldsymbol{e}}\left\{\|\mathcal{P}\boldsymbol{e}\|_2^2\right\}. \end{aligned} \tag{5.15}$$

Hence, the first term in (5.15), i.e., the bias, takes similar form as (5.11) in Theorem 5.1 whereas the second term or variance is the norm of the projection of the vectorized noise onto $\mathcal{S}$. If there were no noise, the variance term in (5.15) disappears and therefore the overall error depends uniquely on the accuracy of the Nyström approximation. With noise, there needs to be a balance between bias and variance in order to make the risk small, i.e., sampling the most important points in $\boldsymbol{f}$ while acquiring the least amount of noise. However, since the distribution of the noise over the sample is usually unknown, this strategy is not feasible. Under the assumption of Gaussian noise with variance ν^2, adapting the procedure in the proof of Theorem 5.1 to the 2-norm, we have that

$$R_0(\boldsymbol{S}) = \|\boldsymbol{f} - \mathcal{P}\boldsymbol{f}\|_2^2 + \nu^2\text{Tr}(\mathcal{P}) \leq \|\boldsymbol{f}\|_2^2\,\text{Tr}(\boldsymbol{K}_f - \boldsymbol{T}_f) + \nu^2 s$$

which shows that adding more samples can reduce the approximation error for $\boldsymbol{T}_f$ and hence the bias but it also increases the variance. Since $\mu > 0$ enables the regularization term in KKCMEX, which can reduce the impact of the noise and risk of overfitting, the passive sampling strategy will be derived by analyzing $R_\mu(\boldsymbol{S})$.

5.2.1 Design of the sampling matrix

Lemma 3.2 shows that the difference between the kernel matrix and its regularized Nyström approximation, which depends on μ, also regulates the MSE bound for KKMCEX. Therefore, it is well-grounded that one may pick the samples in $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$ by choosing the associated columns in $\boldsymbol{K}_f$ that best build the regularized Nyström approximation. This section frames the sampling of the entries in $\boldsymbol{m} = \text{vec}(\boldsymbol{M})$ as designing the sampling matrix $\boldsymbol{S}$ that minimizes the error $\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f$. Through analyzing the risk function from a probabilistic standpoint, $\boldsymbol{S}$ is cast as a weighted sampling matrix with weights set according to the leverage scores of the columns of $\boldsymbol{K}_f$. Since this process only involves the kernel matrix, it is a passive sampling approach. Once $\boldsymbol{S}$ is obtained, the corresponding samples are obtained as $\bar{\boldsymbol{m}} = \boldsymbol{S}\boldsymbol{m}$ and KKMCEX can then be applied to recover f.

Rewriting the risk function following Lemma 3.2 and (3.51), the risk is upper bounded as

$$\begin{aligned} R_\mu(\boldsymbol{S}) &= \left\|(\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f)\boldsymbol{\gamma}\right\|_2^2 + \mathbb{E}_{\boldsymbol{e}}\left\{\frac{1}{\mu^2}\left\|(\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f)\boldsymbol{S}^T\bar{\boldsymbol{e}}\right\|_2^2\right\} \\ &\leq \left\|\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f\right\|_2^2\left(\|\boldsymbol{\gamma}\|_2^2 + \frac{\nu^2 s}{\mu^2}\right) \end{aligned} \tag{5.16}$$

where the regularized Nyström approximation $\tilde{\boldsymbol{T}}_f = \boldsymbol{K}_f\boldsymbol{S}^T(\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}+\mu\boldsymbol{I})^{-1}\boldsymbol{S}\boldsymbol{K}_f$ is a function of $\boldsymbol{S}$. There exist many approaches to building Nyström approximations, all based around the idea of selecting the subset of columns of $\boldsymbol{K}_f$ yielding the best low-rank approximation. Such an approach can be derived by further examining the eigenvalues of $\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f$. Let us recall Eq. (B.16) from Appendix B.3,

$$\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f = \mu\boldsymbol{Q}_f\boldsymbol{\Sigma}_f^{\frac{1}{2}}(\boldsymbol{\Sigma}_f + \mu\boldsymbol{I})^{-\frac{1}{2}}(\boldsymbol{I} - \boldsymbol{P})^{-1}(\boldsymbol{\Sigma}_f + \mu\boldsymbol{I})^{-\frac{1}{2}}\boldsymbol{\Sigma}_f^{\frac{1}{2}}\boldsymbol{Q}_f^T \tag{5.17}$$

where $\boldsymbol{K}_f = \boldsymbol{Q}_f\boldsymbol{\Sigma}_f\boldsymbol{Q}_f^T$ and $\boldsymbol{P}$ is

$$\boldsymbol{P} = \boldsymbol{\Sigma}_f(\boldsymbol{\Sigma}_f + \mu\boldsymbol{I})^{-1} - (\boldsymbol{\Sigma}_f + \mu\boldsymbol{I})^{-\frac{1}{2}}\boldsymbol{\Sigma}_f^{\frac{1}{2}}\boldsymbol{Q}_f^T\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{Q}_f\boldsymbol{\Sigma}_f^{\frac{1}{2}}(\boldsymbol{\Sigma}_f + \mu\boldsymbol{I})^{-\frac{1}{2}}. \tag{5.18}$$

Defining $\boldsymbol{V} = (\boldsymbol{\Sigma}_f + \mu\boldsymbol{I})^{-\frac{1}{2}}\boldsymbol{\Sigma}_f^{\frac{1}{2}}\boldsymbol{Q}_f^T$ and knowing that the product of two diagonal matrices is

commutative, substituting into (5.18) yields

$$\boldsymbol{P} = \boldsymbol{V}\boldsymbol{V}^T - \boldsymbol{V}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{V}^T \tag{5.19}$$

Therefore, $\boldsymbol{P}$ is the difference between $\boldsymbol{V}\boldsymbol{V}^T$ and an approximation built from a subset of the columns in $\boldsymbol{V}$, and it is the only element in (5.17) depending on the sampling distribution through $\boldsymbol{S}$. Thus, if $\boldsymbol{S}$ is designed to minimize the norm of $\boldsymbol{P}$ so that in (5.17) $\|(\boldsymbol{I}-\boldsymbol{P})^{-1}\|_2$ is approximately minimized, a reduction on the difference $\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f$ in (5.17) and also on the bound on $R_\mu(\boldsymbol{S})$ in (5.16) can be expected.

Let ω be an ordered s-tuple containing indices drawn with replacement from $\{1,\dots,NL\}$ with the probability distribution $\Pr(i) = p_i$. Here, p_i denotes the probability of sampling the ith column in $\boldsymbol{K}_f$, and ω contains the indices of the sampled columns in non-descending order. Moreover, assume a weighted $\boldsymbol{S} \in \mathbb{R}^{s\times NL}$ with $\boldsymbol{S}_{i,j} = \frac{1}{\sqrt{sp_j}}$ if $\omega(i) = j$, $i = 1,\dots,s$, $j = 1,\dots,NL$, and 0 elsewhere. Note that, since the indices in ω are drawn with replacement, there might be repeated rows in $\boldsymbol{S}$ sampling the same item. Moreover, the change to a weighted $\boldsymbol{S}$ is necessary in order to satisfy certain theoretical guarantees. In [92] it is shown that the optimal probability distribution to minimize $\mathbb{E}\{\|\boldsymbol{P}\|_{\mathrm{F}}^2\}$ in (5.19) is $p_i = \frac{\|\boldsymbol{v}_i\|_2^2}{\|\boldsymbol{V}\|_{\mathrm{F}}^2}$, with $\boldsymbol{v}_i$ denoting the ith column of $\boldsymbol{V}$,

$$\|\boldsymbol{v}_i\|_2^2 = \sum_{j=1}^{NL} \frac{\sigma_j}{\sigma_j+\mu}\boldsymbol{q}_j(i)^2 \tag{5.20}$$

where $\sigma_i = (\boldsymbol{\Sigma}_f)_{ii}$, and $\boldsymbol{q}_i$ is the ith column in $\boldsymbol{Q}_f$. Ideally, making $\|\boldsymbol{P}\|_F$ smaller also minimizes $\|\boldsymbol{K}_f - \tilde{\boldsymbol{T}}\|_F^2$.

Note, however, that while a smaller $\|\boldsymbol{K}_f - \tilde{\boldsymbol{T}}\|_F^2$ makes the upper bound in (5.16) smaller, it does not actually ensure that $R_\mu(\boldsymbol{S})$ will also be reduced. Moreover, the optimal distribution is tied to the regularization parameter μ through (5.20). Hence, setting μ to minimize $\|\boldsymbol{K}_f - \tilde{\boldsymbol{T}}\|_F^2$ does not imply that the optimal regularization parameter for $R_\mu(\boldsymbol{S})$ is also chosen. For this reason, we decouple the problem of finding the optimal distribution from the regularization in the recovery problem and define

$$l_i = \sum_{j=1}^{NL} \frac{\sigma_j}{\sigma_j+\alpha}\boldsymbol{q}_j(i)^2 \tag{5.21}$$

where $\alpha > 0$ is a tunable parameter. Then, the chosen distribution is

$$p_i = \frac{l_i}{\sum_{j=1}^{NL} l_j}. \tag{5.22}$$

Thus, replacing μ in (5.20) with α to obtain (5.21) enables separate optimization of the regularization term in the regression problem and the probability distribution (5.22).

The quantity l_i is the so-called regularized leverage score[2] of the ith column of $\boldsymbol{K}_f$ [64], which is a measure of the importance of the column and can also be written as $l_i = (\boldsymbol{K}_f(\boldsymbol{K}_f + \alpha \boldsymbol{I})^{-1})_{i,i}$. Just so, the ith leverage score is a weighted average of the ith row of the eigenvector matrix $\boldsymbol{Q}$. Leverage scores are similar to the notion of coherence, in that if a row has high leverage score or coherence this indicates that it contains a point that "sticks out" with respect to the other elements. Leverage scores are typically used in ordinary least squares (OLS) problems in order to sample a subset of equations and solve a smaller problem [93]. While in OLS the regression matrix is usually tall and rank-deficient, this is not the case with kernel matrices in KRR. Therefore, having $\alpha > 0$ is necessary to avoid the trivial result of having all l_i being 1. Moreover, α regulates how much weight is given to the notable points, with a larger α inducing smaller scores with a more even distribution across all columns.

Calculating the leverage scores entails obtaining the eigendecomposition of $\boldsymbol{K}_f$, or inverting $(\boldsymbol{K}_f + \alpha \boldsymbol{I})$, which can be computationally costly. Since in KKMCEX $\boldsymbol{K}_f = \boldsymbol{K}_h \otimes \boldsymbol{K}_w$, the computation of l_i can be sped up by instead calculating the approximation

$$\tilde{l}_i = \left(\boldsymbol{K}_h(\boldsymbol{K}_h + \alpha_h \boldsymbol{I})^{-1} \otimes \boldsymbol{K}_w(\boldsymbol{K}_w + \alpha_w \boldsymbol{I})^{-1}\right)_{ii} = (\boldsymbol{l}_h \otimes \boldsymbol{l}_w)_i \tag{5.23}$$

where $\boldsymbol{l}_w = \operatorname{diag}\left(\boldsymbol{K}_w(\boldsymbol{K}_w + \alpha_w \boldsymbol{I})^{-1}\right)$, $\boldsymbol{l}_h = \operatorname{diag}\left(\boldsymbol{K}_h(\boldsymbol{K}_h + \alpha_h \boldsymbol{I})^{-1}\right)$ and $\alpha_w, \alpha_h \geq 0$. Then, the approximate probability is

$$\tilde{p}_i = \frac{\tilde{l}_i}{\sum_{i=1}^{NL} \tilde{l}_i}. \tag{5.24}$$

Since $\tilde{l}_i$ is an approximation to l_i, the resulting probability distribution $\tilde{p}_i$ will differ from p_i in (5.22). Moreover, it is generally not possible to have $\tilde{l}_i = l_i$ by tuning α_w and α_h since these two variables do not afford enough degrees of freedom to satisfy an equality. Nevertheless, the computation cost is greatly reduced, which can outweigh any potential loss in accuracy.

5.3 Numerical tests

This section presents numerical tests comparing uniform sampling with $p_i = \frac{1}{NL}$ to the approach based on leverage scores (5.22) and approximate leverage scores (5.24) for different values of s. The optimal hyperparamters μ and α are found via grid search for each s. The

[2]Hereafter the words leverage score refer to the regularized leverage score.

search is performed randomly: intervals $[\mu_{min}, \mu_{max}]$ and $[\alpha_{min}, \alpha_{max}]$ are set for μ and α respectively, and 100 tuples of values within the intervals are selected uniformly at random. For the approximate leverage scores, the parameter values are $\alpha_w = \alpha_h = \sqrt{\alpha}$. The s labelled samples form the training set, and the NMSE measures the estimation error as

$$\text{NMSE} = \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{\left\| \hat{\boldsymbol{F}}_i - \boldsymbol{F} \right\|_{\text{F}}^2}{\|\boldsymbol{F}\|_{\text{F}}^2} \tag{5.25}$$

where $\hat{\boldsymbol{F}} = \text{unvec}(\hat{\boldsymbol{f}}_K)$, $N_r = 20$, and a new training set is acquired at each realization. Since the proposed sampling scheme is applicable to any function in an RKHS that can be arranged as a vector with its associated kernel matrix, the simulations consider the recovery of missing entries in both vectors, where $\boldsymbol{K}_h = 1$, and matrices. Moreover, in the vector case only the results for uniform and leverage-based sampling (5.22) are shown since the exact and approximate leverage scores are the same for $\alpha_w = \alpha$.

The Boston housing dataset[3] is the first to be evaluated. This dataset contains 506 feature vectors detailing the characteristics of houses in Boston such as size or number of rooms, and the price of each house is the ground truth and variable to be predicted. Hence, $\boldsymbol{F}$ is cast as a 506×1 vector. The kernel used to define $\boldsymbol{K}_w$ is the Gaussian kernel applied on the feature vectors. Fig. 5.2 shows the NMSE for different values of s and the two sampling strategies: uniform and based on leverage scores. The figure shows that the passive sampling approach attains a smaller error than uniform sampling. While the gains are small, they are in line with those obtained by other methods [64, 87].

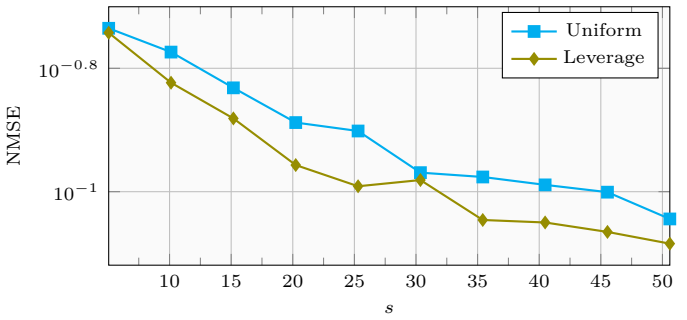


Figure 5.2: NMSE vs. s for the Boston housing data.

Fig. 5.3 shows the results for the mushroom dataset used in Section 3.6.3. The vector to be recovered is of length 5,000 and the kernel is the same as the one in Section 3.6.3. Here, the feature vectors contain details about each mushroom such as shape or size, and

[3] https://www.cs.toronto.edu/ delve/data/boston/bostonDetail.html

the objective is to determine whether a mushroom is edible or poisonous. It can be observed that passive sampling holds a slight edge over uniform sampling. Note that, since this is a binary classification problem with classes $[1, -1]$, the NMSE evaluates the difference between the regression result and the actual numerical value of the class before applying any decision rule.

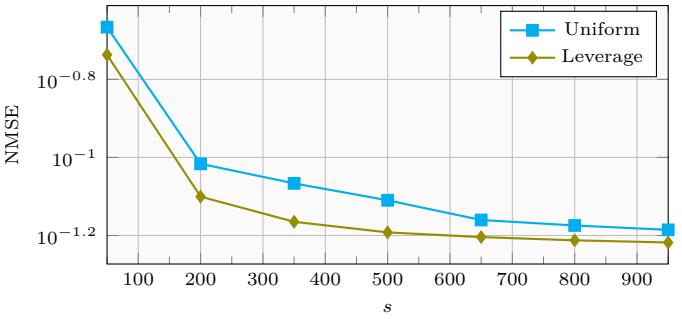


Figure 5.3: NMSE vs. s for the mushroom data.

In the first matrix case, the evaluation is done on a 500×500 adjacency matrix of the mushroom data as in Section 3.6.3. This is a clustering problem, where each mushroom is a node on a graph connected only to those nodes belonging to the same class, i.e., poisonous or edible. Then, the objective is to recover the whole graph adjacency matrix from a few observed entries. Fig. 5.4 shows that passive sampling lowers the NMSE by around 0.01 across the range of s. Moreover, the distribution obtained from the approximate scores delivers the same performance the exact ones at a smaller computational cost. As in the vector case in 5.3, the leverage scores of the chosen kernels reflect accurately the importance of each data point, hence reducing the MSE when passive sampling is adopted.

On the other hand, Fig. 5.5 shows the results for the temperature dataset, where the kernels are the row and covariance matrices of the data from the year previous to 2002. It can be observed that all sampling methods achieve almost the same NMSE. Due to the natural smoothness of the data, the chosen sampling distribution is not as important.

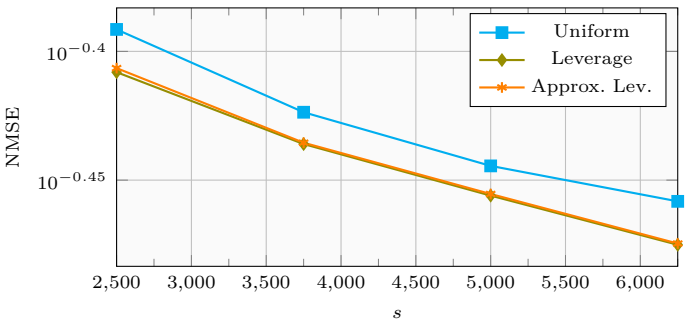


Figure 5.4: NMSE vs. s for the mushroom adjacency matrix.

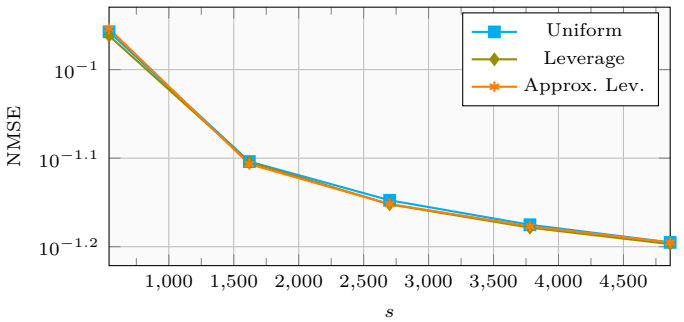


Figure 5.5: NMSE vs. s for the temperature data.

5.3.1 Suboptimal hyperparameters and grid search

The grid search process for μ and α ensures that the smallest possible error is achieved. Moreover, since in the simulations the whole matrix is known, training data can be drawn across the whole matrix in order to obtain a representative set. Nevertheless, it is unlikely that one is able to choose the best pair of hyperparameters; either because only a small percentage of entries can be known, or an exhaustive grid search is unfeasible due to computational cost. Fig. 5.6 depicts the results for the temperature matrix for a suboptimal $\mu = 10$ across all s. Compared to Fig. 5.5, where optimal hyperparameters are chosen, a suboptimal μ results in a smaller error for the passive sampling schemes.

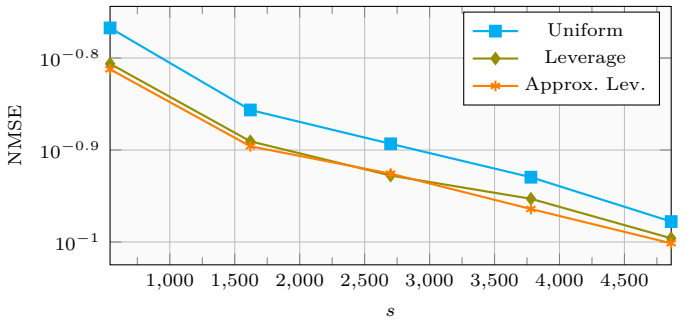


Figure 5.6: NMSE vs. s for the temperature data with $\mu = 10$.

Fig. 5.7 shows the result of the grid search for μ and α for several values of s. The two topmost subfigures depict the error obtained for 100 different pairs of values for each algorithm at $s = 1620$. For the search over μ, each tested value has several vertically aligned points which correspond to different values of α. The plot shows that the optimal μ is $10^{3.5}$ and that the vertical dispersion of the points diminishes as μ grows larger. A plausible explanation for this phenomenon is that a larger μ yields a smoother $\hat{\boldsymbol{F}}$, which reduces the influence of the outliers. On the other hand, the plot for α does not show any specific pattern, which in this case indicates that whether passive sampling brings any improvement is mostly determined by μ.

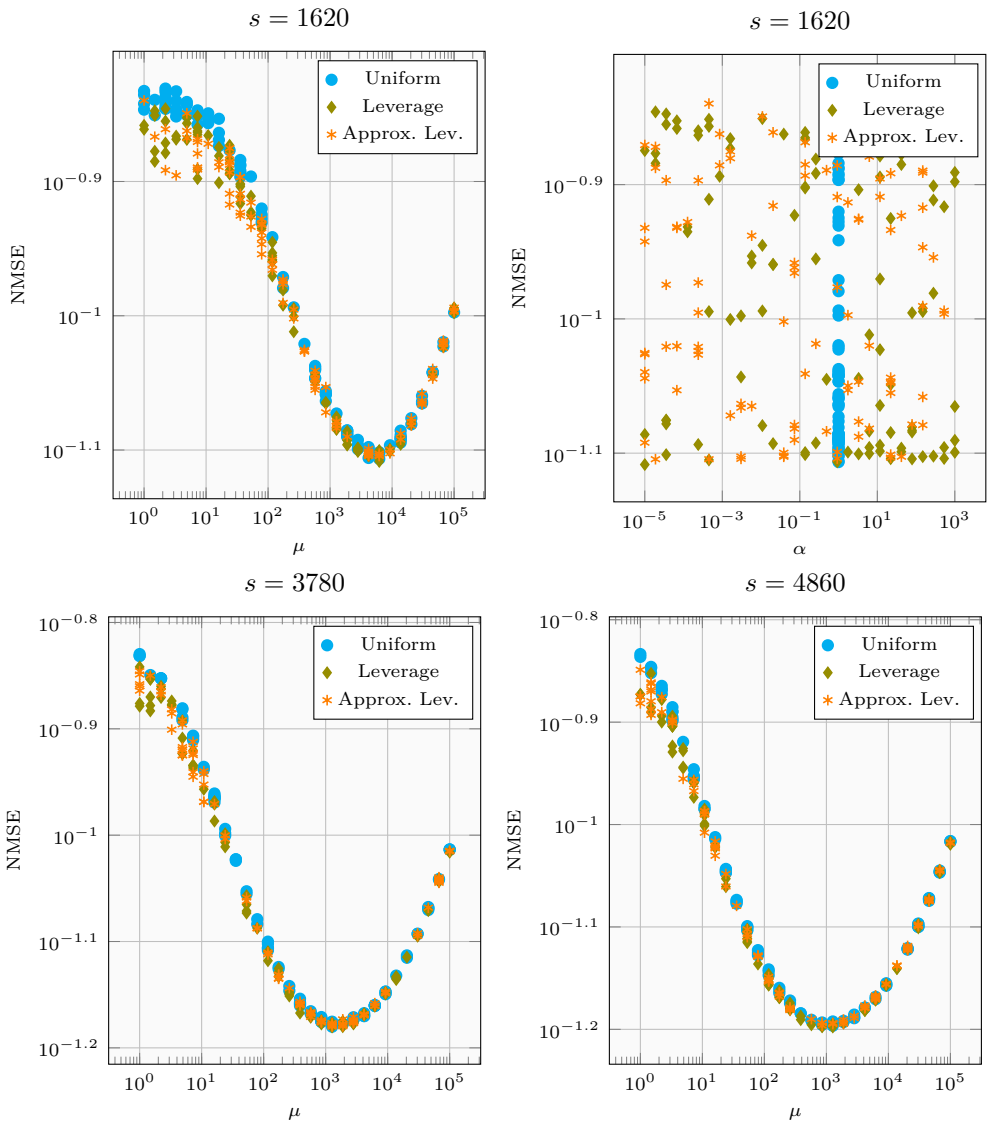


Figure 5.7: Hyperparameter grid search for the temperature data.

5.4 Conclusions

This chapter has peered into how the prior information embedded in the kernel functions spanning an RKHS can be used to determine which inputs are the most important to label in order to recover the complete function. First, through functional analysis, it has been shown that the recovery error of a sampled function in an RKHS is directly tied to the Nyström approximation to the kernel matrix. Hence, in the noiseless case, the error can be minimized by finding the best possible Nyström approximation and labelling the inputs corresponding to the chosen columns. This sampling approach is passive since it only involves knowledge of the input space and kernel matrix, which presents benefits over active sampling schemes which require online operation and are more vulnerable to noise. Given the theoretical background, the Nyström-based sampling approach has been applied to KKMCEX for the picking of the training set. In this, the weighted sampling matrix is designed according to the leverage scores of the kernel matrix, which measure the importance of each column in achieving a good Nyström approximation. Numerical tests have shown that the proposed approach works well for the recovery of sampled vectors and matrices, the latter case showing better results when the choice of the regression regularization parameter is suboptimal.

6

Conclusions

This thesis has addressed the problem of matrix completion (MC) from a variety of standpoints with the underlying theme of the availability of prior information. For one, it has delved into the algorithmic side by studying existing formulations and proposing enhancements to improve speed and accuracy. Furthermore, it has explored and proposed frameworks to encode and use said prior information to enhance the recovery accuracy of the unknown matrix. As it is, the work comprised in the thesis pushes forward the theoretical boundaries in MC and opens the door to improved application.

Chapter 2 has presented an introduction to MC which addresses core concepts and algorithms in an accessible manner. Proximal gradient algorithms have been studied in solving the minimization problem with nuclear norm regularization, and improved variants which increase the convergence speed have been proposed. Moreover, it has been shown that proximal gradient (PG) warrants the use of prior information about the unknown matrix in two ways: initialization and additional regularization. By modelling the columns of the matrix as graph signals, the structural information provided by the graph is used to obtain an initial point closer to the minimum and thus reducing the required PG iterations. Moreover, the availability of a graph enables more a accurate recovery through the incorporation of the graph Laplacian into the MC formulation as an additional regularization term. The PG algorithm is able to easily handle such terms, and through its analytical expressions theoretical measures on its performance have been derived.

Chapter 3 has introduced the implementation of MC as a kernel regression problem. The introduction to reproducing kernel Hilbert spaces demonstrates how kernel functions aid in the

recovery of missing data points. Since the kernel function measures similarity between points in the input space, predicting an unlabeled input amounts to finding a linear combination of the kernel evaluated at the sampled points. Moreover, through its regularization term kernel ridge regression enforces smoothness in the estimated function in order to reduce overfitting. When applied to MC, the rectangular matrix is cast as a function in a vector space to be recovered from a small number of observations. In this reformulation, the kernel matrix is formed as the Kronecker product of two smaller kernel matrices, corresponding to the column and row spaces of the matrix. The resulting Kronecker Kernel MC and extrapolation (KKMCEX) algorithm enables a straightforward MC which is also computationally efficient when the number of observations is small. Still, given the sparse nature of the observed matrix, it is not feasible to determine column and row similarities through application of the kernel function. Hence, this chapter has explored means of obtaining kernel matrices through side information available in the form of graphs or feature vectors associated to the rows and columns of the unknown matrix.

Besides KKMCEX, Chapter 3 also has introduced its dual variant, the ridge regression MC and extrapolation (RRMCEX) algorithm. This method aims to overcome one flaw in KKMCEX: the fact that its computational cost scales with the number of observations. By replacing the kernel matrix with a low-rank approximation product of two feature matrices, RRMCEX solves MC as a ridge regression problem offering accuracy close to that of KKMCEX at a potentially much smaller cost. Moreover, thanks to the separability of its cost function, the online variant online RRMCEX has been derived, which has been shown in numerical tests to outperform the widely used stochastic gradient descent. All in all, Chapter 3 has introduced a MC framework built atop the reproducing kernel Hilbert space toolset with proven usefulness and speed advantages when compared to existing kernel-based implementations.

In Chapter 4 the focus has been on comparing three approaches to MC, namely base MC, KMC and KKMCEX, through their generalization error. First, the concept of inductive GE has been introduced as a mean to measure the difference between the loss function applied to a random dataset and its expected value. However, practical implementations of MC operate in the transductive setting: there is no assumed sampling distribution and the objective is to transfer knowledge from a training set onto a testing set which evaluates algorithm performance. The chapter has presented a guide on how to obtain the transudctive generalization error (TGE), which can be approximated through the transductive Rademacher complexity (TRC). The TRC indicates how likely it is that a function within a hypothesis class will fit the training and testing points. Hence, having a high TRC can result in very accurate prediction of both known datasets but poor performance when predicting previously

unseen inputs. Base MC, KMC and KKMCEX rely on different regularization terms, which modulate the TRC and hence their generalization properties. Through the bounds derived in Chapter 4 it has been shown that KKMCEX has one major advantage over the other methods: its TGE is only dependent on the number of observations and independent of the matrix size. Hence, it is a more reliable method when dealing with a very small number of samples and large matrices.

Finally Chapter 5 has taken on on the issue of choosing the observations in the training set to minimize the recovery error. While active sampling is a commonly used scheme that offers good results it has two main drawbacks: the algorithm must be able to operate in online manner to sequentially request new observations, and observation noise can derail the sampling process. Hence, the chapter has taken a passive sampling approach. In it, the only information used to acquire the complete training set is contained within the input space and the chosen kernel. Essentially, the passive sampling method examines the kernel function to determine which points stick out the most and therefore should be sampled with higher probability. Then, a sampling probability is assigned to each input point according to its importance and the observations are drawn according to the distribution. Chapter 5 has shown that in KKMCEX finding the optimal sampling distribution amounts to finding the subset of columns in the kernel matrix that best form its Nyström approximation. Indeed, this is proven via functional analysis of the sampling process in the noiseless case. Moreover, is shown how separately sampling the column and row spaces, which results in a grid sampling pattern, leads to more efficient algorithms where the optimal distribution is much faster to calculate.

While this thesis has focused solely on MC, there are several avenues upon which the work can be continued. As mentioned in Chapter 4 the kernel-based approach to tensor completion is similar to kernel MC. Hence much of the analysis carried out here such as the TGE or the optimal sampling distribution can be extended to work on tensors. In fact, extending the grid sampling strategy onto tensors can lead to much faster algorithms while reducing the recovery error thanks to an adequate sample choice. The closely related field of Gaussian processes has proposed efficient algorithms [94] for when the input set is a tensor product of sets. Since this is the case for both kernel-based matrix and tensor completion, such algorithms can also be employed to accelerate the recovery of high dimension tensors. Hence, the Nyström-based sampling strategy from Chapter 5 can be coupled with this kind of algorithm in order to boost performance.

Another open question is the possibility of performing multi-kernel MC due to the difficulty of always being able to find an appropriate kernel. By allowing multiple kernel functions for the row and column spaces, the prediction error can be reduced [23] since the

optimal kernel is automatically chosen as a combination of other kernels. However, this comes at an increase in computational cost which for MC is due to be very high for large matrix sizes. A potential way to curb on cost is via an online multi-kernel implementation extended to work with kernel matrices that are obtained as the Kronecker product of two or more matrices.

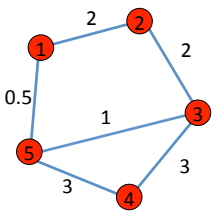
Finally, there is the always-present challenge of finding new applications for a method. With its current surge in popularity, deep learning is being used to solve all sorts of problems and also to improve existing machine learning techniques. Indeed, there are works [95, 96] tackling MC through deep learning in order to better exploit nonlinearities in the data matrix. Another hot topic for MC is in biomedical applications, especially in predicting drug-target interactions [52, 97]. Although not included in this manuscript, some work has been conducted within this thesis on using MC in array processing. Given the large amount of antennas implemented in newer communication standards, keeping power consumption low must be a priority. Here, MC is a powerful tool that can contribute to achieving this goal. Since in multiantenna arrays the received signal is the same at each antenna except for a small phase shift, the snapshot matrix is inherently low-rank. Hence, MC-based strategies can be designed in order to sample only a few antennas at each time instant and still be able to recover the full snapshot matrix. Moreover, this allows to downsize the receiver and implement a small number of radio frequency chains equal to the number of antennas to be sampled at each time instant. New collaborations [98, 99] have shown that MC outperforms other existing approaches, hence revealing a research path with innovation potential.

Appendices

A

Graph signals

A graph is defined as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{A}\}$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of vertices, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the ordered set of edges connecting the vertices, and $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix. Each vertex is associated with an entity, whereas an edge (v_i, v_j) represents that information flows from vertex j to vertex i. If the information flow between all connected pairs of vertices is bidirectional, then the graph is undirected such that $(v_i, v_j) \in \mathcal{E} \Leftrightarrow (v_j, v_i) \in \mathcal{E} \; \forall \; v_i, v_j \in \mathcal{V}$. The overall connectivity of the network is represented in $\boldsymbol{A}$, which can be unweighted or weighted. In the unweighted case, $\boldsymbol{A}_{i,j} = 1 \; \forall \; (v_i, v_j) \in \mathcal{E}$ and $\boldsymbol{A}_{i,j} = 0 \; \forall \; (v_i, v_j) \notin \mathcal{E}$. Hence, this adjacency matrix only represents whether there is a connection between any two vertices in the graph. On the other hand, the non-zero entries of a weighted adjacency matrix can take any value in $\mathbb{R}_+$, representing the degree of similarity between two vertices. In this thesis it is assumed that all graphs are undirected with a symmetrical weighted adjacency matrix, such as the pair illustrated in Fig. A.1.



$$\begin{bmatrix} 0 & 2 & 0 & 0 & 0.5 \\ 2 & 0 & 2 & 0 & 0 \\ 0 & 2 & 0 & 3 & 1 \\ 0 & 0 & 3 & 0 & 3 \\ 0.5 & 0 & 1 & 3 & 0 \end{bmatrix}$$

Figure A.1: Undirected graph and its associated weighted adjacency matrix.

Given $\mathcal{G}$, a graph signal is defined as a map from the set of $\mathcal{V}$ vertices into the set of real

numbers $x : \mathcal{V} \to \mathbb{R}$ that can be arranged in a column vector $\boldsymbol{x}$ defined as

$$\boldsymbol{x} = [x(1), \ldots, x(N)]^T, \tag{A.1}$$

where $x(i)$ is the value of the signal at vertex v_i. The values of the graph signal should reflect the structure of the underlying graph, with connected vertices having similar values. Fig. A.2 depicts a graph with 382 vertices and 654 edges, where most nodes are arranged into grid blocks and there are some edges connecting each block. Fig. A.3 shows a possible signal measured on this graph, where it is clear that vertices close in the grid have similar values, and vertices connected between different grids do as well. It should be noted that the entries in $\boldsymbol{x}$ do not have an inherent organization as is the case with, for instance, time signals. The entries of a graph signal can be shuffled and the signal will still be the same, since this shuffling amounts to a reordering of the domain of the function $x : \mathcal{V} \to \mathbb{R}$ and it does not alter the links between vertices. Thus, the graph signal in Fig. A.4 is equal to the one in Fig. A.3 since the edges remain unchanged even though the 2D positioning of the vertices in the figure has changed. Consider now the availability of different realizations of graph signals

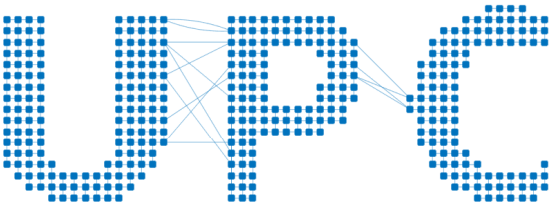


Figure A.2: Graph obtained from the pixels in an image of the UPC logo. Some edges connecting the letters have been added.

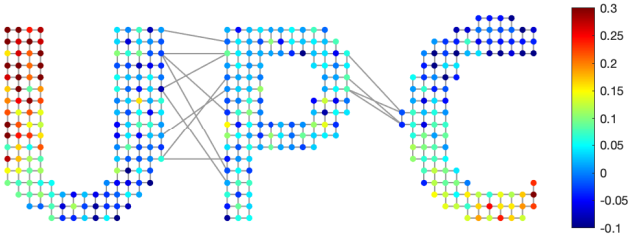


Figure A.3: Graph signal.

lying on the same graph. For instance, this situation arises in a wireless sensor network where each node is connected only to its close neighbors and taking periodic measurements. While each signal is described by the same graph, its values might be completely different and independent between signals. Thus, we have a set of graph signals $\{\boldsymbol{x}_l\}_{l=1}^{L}$ residing on the

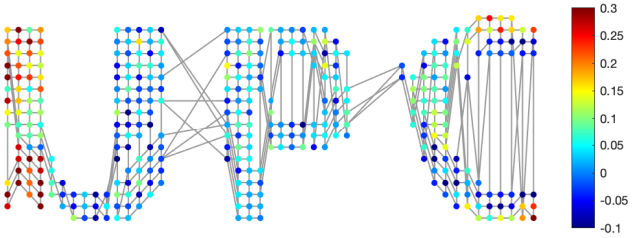


Figure A.4: Graph signal with some shuffled vertices.

graph $\mathcal{G}$, and we cast $\boldsymbol{X}$ into a $N \times L$ graph signal matrix

$$\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L]. \tag{A.2}$$

Besides establishing the graph connectivity through $\mathcal{E}$, the weights of the edges also need to be known. In graph learning applications, the entries of a weighted $\boldsymbol{A}$ are usually generated by applying pairwise similarity measures such as the Gaussian kernel or cosine distance to the samples of the signal [33]. Alternatively, when no samples are available and only the structural data $\mathcal{V}$ and $\mathcal{E}$ are known, $\boldsymbol{A}$ can be generated by applying pairwise structural similarity measures [100] to the vertices of the graph. For instance, given a graph $\mathcal{G}$, its adjacency matrix can be generated as

$$\boldsymbol{A} = \text{diag}(\boldsymbol{A}'\boldsymbol{1})^{-1}\boldsymbol{A}', \tag{A.3}$$

where

$$\boldsymbol{A}'_{i,j} = \exp(-\frac{N^2 d_{i,j}}{\sum_{i,j} d_{i,j}}), \tag{A.4}$$

and $d_{i,j}$ is the geodesic distance, i.e., the number of hops, between station j and station i on graph $\mathcal{G}$. The structural similarity measures are the most adequate for the matrix completion problem since the available graph signals are incomplete. Nevertheless, one can design $\boldsymbol{A}$ with any other method deemed appropriate for a specific application.

Besides the adjacency matrix, another important element for a graph is the Laplacian matrix since it gives further insight into the graph structure. For instance, it can be used to find the optimal partition of the graph into disjoint sets, or to obtain low dimensional embeddings to be used in, e.g., clustering applications. In graph signal processing, the Laplacian plays a central role as it allows to implement filters from classical signal processing into the graph domain, and develop new graph-specific operations.

The Laplacian of a graph is defined as

$$\boldsymbol{L} = \text{diag}^{-1}(\boldsymbol{A}\mathbf{1}) - \boldsymbol{A} \tag{A.5}$$

with eigendecomposition $\boldsymbol{L} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$. The eigendecomposition of the Laplacian elucidates the notion of frequency on a graph. Each eigenvalue and eigenvector pair represents a frequency, with the largest eigenvalue denoting the highest frequency, and the opposite for the smallest eigenvalue. Thus, each eigenvector represents the signal that varies with different speed across the graph according to the associated eigenvalue. For instance, the values at the vertices of a high frequency signal will change rapidly in magnitude between connected vertices, whereas for a low frequency signal the values will be similar. Fig. A.5 illustrates this concept for the second and sixth smallest frequencies of the graph introduced in Fig. A.2. Another important

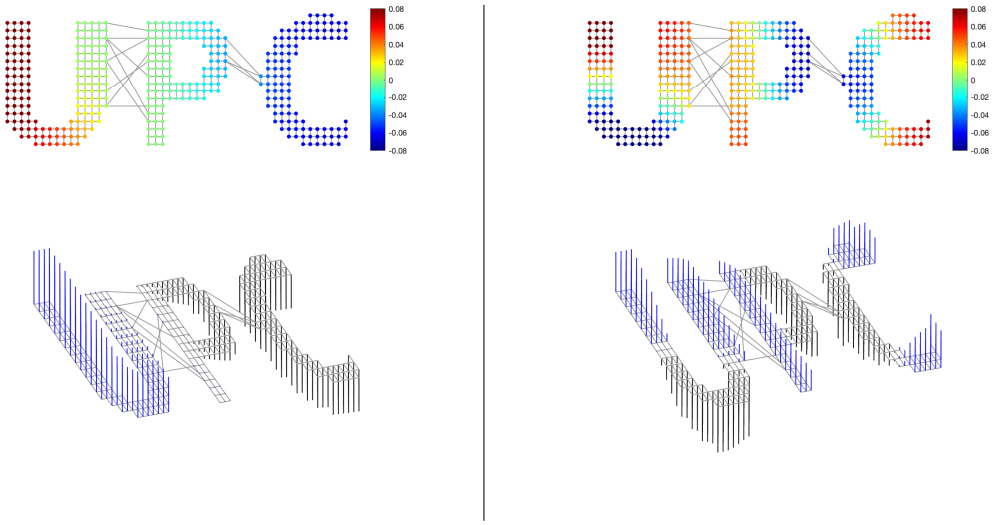


Figure A.5: Eigenvectors as graph signals for the second (l) and sixth (r) eigenvalues.

property of the Laplacian is that its smallest eigenvalue is 0 with associated eigenvector $\mathbf{1}$, and multiplicity equal to the number of connected components of the graph. Thus, a signal with constant value is the smoothest possible signal on a graph as shown in Fig. A.6.

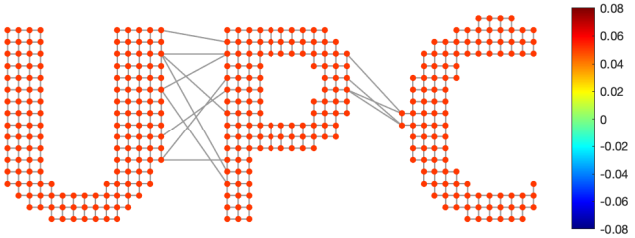


Figure A.6: Smoothest signal on the graph.

B

Proofs

B.1 Representer Theorem

Theorem 2.1. ***Representer Theorem.*** *Given the set* $\{\bar{x}_i, m_i\}_{i=1}^s$ *of input-observation points in* $\mathcal{X}\times\mathbb{R}\times\mathbb{R}$*, the unknown function* $f : \mathcal{X} \to \mathbb{R} \in \mathcal{H}_f$*, and a loss function* $l : \mathcal{X}\times\mathbb{R} \to \mathbb{R}$*, the solution to*

$$\arg\min_f \sum_{i=1}^{s} l(f(\bar{x}_i), m_{i,j}) + \mu||f||^2_{\mathcal{H}_f} \tag{B.1}$$

is an estimate $\hat{f}$ *that satisfies*

$$\hat{f}(x) = \sum_{i=1}^{s} \bar{\alpha}_i \kappa_f(x, \bar{x}_i) \tag{B.2}$$

for some $\bar{\alpha}_i \in \mathbb{R},\ i = 1, \ldots, s.$

Proof. In order to prove the Representer Theorem, we will show that the loss function l and the norm $||f||^2_{\mathcal{H}_f}$ only depend on the set of inputs $\bar{\mathcal{X}} = \{\bar{x}_1, \ldots, \bar{x}_s\}$. Let us define the subspace $\mathcal{S} \subseteq \mathcal{H}_f$ as

$$\mathcal{S} := \text{span}\{k_f(\cdot, \bar{x}_i)\ :\ i = 1, \ldots, s\}. \tag{B.3}$$

Now f can be decomposed as $f = f_{\mathcal{S}} + f_{\mathcal{S}^\perp}$, where $f_{\mathcal{S}} \in \mathcal{S}$ is the projection of f onto $\mathcal{S}$, and $f_{\mathcal{S}^\perp}$ is the projection on the subspace orthogonal to $\mathcal{S}$. From the reproducing property we have that for $\bar{x} \in \bar{\mathcal{X}}$

$$f(\bar{x}) = \langle f, k_f(\cdot, \bar{x})\rangle_{\mathcal{H}_f} = \langle f_{\mathcal{S}}, k_f(\cdot, \bar{x})\rangle_{\mathcal{H}_f} + \langle f_{\mathcal{S}^\perp}, k_f(\cdot, \bar{x})\rangle_{\mathcal{H}_f} = \langle f_{\mathcal{S}}, k_f(\cdot, \bar{x})\rangle_{\mathcal{H}_f}. \tag{B.4}$$

Using the reproducing property on the last equality, we have that $f(\bar{x}) = f_{\mathcal{S}}(\bar{x}) \; \forall \; \bar{x} \in \bar{\mathcal{X}}$. This implies

$$\sum_{i}^{s} l(f(\bar{x}_i), m_i) = \sum_{i}^{s} l(f_{\mathcal{S}}(\bar{x}_i), m_i), \tag{B.5}$$

that is, the loss function l only depends on the inputs $\bar{x} \in \bar{\mathcal{X}}$. On the other hand, the norm $\|f\|_2^2 \, \mathcal{H}_f$ satisfies

$$\|f\|_2^2 \, \mathcal{H}_f = \|f_{\mathcal{S}} + f_{\mathcal{S}^\perp}\|_2^2 \, \mathcal{H}_f = \|f_{\mathcal{S}}\|_2^2 \, \mathcal{H}_f + \|f_{\mathcal{S}^\perp}\|_2^2 \, \mathcal{H}_f \geq \|f_{\mathcal{S}}\|_2^2 \, \mathcal{H}_f. \tag{B.6}$$

This shows that the norm is minimized when f lies in the subspace $\mathcal{S}$ and therefore $\|f_{\mathcal{S}^\perp}\|_2^2 \, \mathcal{H}_f = 0$. Thus, the minimizer of (B.1) satisfies $\hat{f}(x) = \sum_{i=1}^{N} \alpha_i \kappa_f(x, x_i)$ where $\alpha_i = 0$ if $x_i \notin \bar{\mathcal{X}}$ for $i = 1, \ldots, N$. □

B.2 Proof of Lemma 3.2

For the KKMCEX estimator (3.46), the MSE is given as

$$MSE := \mathbb{E}_{\boldsymbol{e}} \left\{ \|\boldsymbol{f} - \hat{\boldsymbol{f}}_K\|_2^2 \right\} = \mathbb{E}_{\boldsymbol{e}} \left\{ \|\boldsymbol{f} - \boldsymbol{K}_f \hat{\boldsymbol{\gamma}}\|_2^2 \right\}. \tag{B.7}$$

Plugging the estimator from (3.45) into (B.7) yields

$$\begin{aligned} MSE &= \mathbb{E}_{\boldsymbol{e}} \left\{ \left\| \boldsymbol{f} - \boldsymbol{K}_f \boldsymbol{S}^T (\boldsymbol{S}\boldsymbol{K}_f \boldsymbol{S}^T + \mu \boldsymbol{I})^{-1} (\boldsymbol{S}\boldsymbol{f} + \bar{\boldsymbol{e}}) \right\|_2^2 \right\} \\ &= \left\| (\boldsymbol{I} - \boldsymbol{K}_f \boldsymbol{S}^T (\boldsymbol{S}\boldsymbol{K}_f \boldsymbol{S}^T + \mu \boldsymbol{I})^{-1} \boldsymbol{S}) \boldsymbol{f} \right\|_2^2 + \mathbb{E}_{\boldsymbol{e}} \left\{ \left\| \boldsymbol{K}_f \boldsymbol{S}^T (\boldsymbol{S}\boldsymbol{K}_f \boldsymbol{S}^T + \mu \boldsymbol{I})^{-1} \bar{\boldsymbol{e}} \right\|_2^2 \right\} \end{aligned} \tag{B.8}$$

where we have used that $\mathbb{E}\{\boldsymbol{e}\} = \boldsymbol{0}$. Further, the first and second terms in (B.8) are the bias and variance of the KKMCEX estimator, respectively. If we substitute $\boldsymbol{f} = \boldsymbol{K}_f \boldsymbol{\gamma}$ into the first term of (B.8), we obtain

$$\begin{aligned} bias &= \left\| (\boldsymbol{I} - \boldsymbol{K}_f \boldsymbol{S}^T (\boldsymbol{S}\boldsymbol{K}_f \boldsymbol{S}^T + \mu \boldsymbol{I})^{-1} \boldsymbol{S}) \boldsymbol{K}_f \boldsymbol{\gamma} \right\|_2^2 \\ &= \left\| (\boldsymbol{K}_f - \boldsymbol{K}_f \boldsymbol{S}^T (\boldsymbol{S}\boldsymbol{K}_f \boldsymbol{S}^T + \mu \boldsymbol{I})^{-1} \boldsymbol{S}\boldsymbol{K}_f) \boldsymbol{\gamma} \right\|_2^2 \\ &= \left\| (\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f) \boldsymbol{\gamma} \right\|_2^2 \end{aligned} \tag{B.9}$$

where $\tilde{\boldsymbol{T}}_f$ is the regularized Nyström approximation of $\boldsymbol{K}_f$ in (3.49). On the other hand, the variance term is

$$\begin{aligned} var &= \mathbb{E}_{\boldsymbol{e}} \left\{ \left\| \boldsymbol{K}_f \boldsymbol{S}^T (\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T + \mu \boldsymbol{I})^{-1} \bar{\boldsymbol{e}} \right\|_2^2 \right\} \\ &= \mathbb{E}_{\boldsymbol{e}} \left\{ \frac{1}{\mu^2} \left|\left| \boldsymbol{K}_f \boldsymbol{S}^T (\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T + \mu \boldsymbol{I})^{-1} (\mu \boldsymbol{I} + \boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T - \boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T) \bar{\boldsymbol{e}} \right|\right|_2^2 \right\} \\ &= \mathbb{E}_{\boldsymbol{e}} \left\{ \frac{1}{\mu^2} \left\| \boldsymbol{K}_f \boldsymbol{S}^T - \boldsymbol{K}_f \boldsymbol{S}^T (\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T + \mu \boldsymbol{I})^{-1} \boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T \bar{\boldsymbol{e}} \right\|_2^2 \right\} \\ &= \mathbb{E}_{\boldsymbol{e}} \left\{ \frac{1}{\mu^2} \left\| (\boldsymbol{K}_f - \boldsymbol{K}_f \boldsymbol{S}^T (\boldsymbol{S}\boldsymbol{K}_f\boldsymbol{S}^T + \mu \boldsymbol{I})^{-1} \boldsymbol{S}\boldsymbol{K}_f) \boldsymbol{S}^T \bar{\boldsymbol{e}} \right\|_2^2 \right\} \\ &= \mathbb{E}_{\boldsymbol{e}} \left\{ \frac{1}{\mu^2} \left\| (\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f) \boldsymbol{S}^T \bar{\boldsymbol{e}} \right\|_2^2 \right\}. \end{aligned} \tag{B.10}$$

Adding the two terms in (B.9) and (B.10), we obtain the MSE in (3.50).

B.3 Proof of Theorem 3.2

Since $\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f$ appears in the bias and variance terms in Lemma 1, we will first derive an upper bound on its eigenvalues that will eventually lead us to a bound on the MSE. To this end, we will need a couple of lemmas.

Lemma B.1. *Given a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ and a symmetric nonsingular matrix $\boldsymbol{B} \in \mathbb{R}^{N\times N}$, it holds that $\lambda_k(\boldsymbol{AB}) = \lambda_k(\boldsymbol{B}^{\frac{1}{2}}\boldsymbol{A}\boldsymbol{B}^{\frac{1}{2}})$; and also $\lambda_k(\boldsymbol{AB}) \leq \lambda_k(\boldsymbol{A})\lambda_N(\boldsymbol{B})$.*

Proof. Since $\boldsymbol{B}$ is invertible and symmetric, we can write $\boldsymbol{AB} = \boldsymbol{B}^{-\frac{1}{2}}(\boldsymbol{B}^{\frac{1}{2}}\boldsymbol{A}\boldsymbol{B}^{\frac{1}{2}})\boldsymbol{B}^{\frac{1}{2}}$. Therefore, $\boldsymbol{AB}$ is similar to $\boldsymbol{B}^{\frac{1}{2}}\boldsymbol{A}\boldsymbol{B}^{\frac{1}{2}}$, and they both share the same eigenvalues. Let $\mathcal{U} \subset \mathbb{R}^N \setminus \{\boldsymbol{0}\}$. From the min-max theorem [101], the k^{th} eigenvalue of $\boldsymbol{A}$ satisfies

$$\lambda_k(\boldsymbol{A}) = \inf_{\mathcal{U}} \left\{ \sup \boldsymbol{x} \in \mathcal{U} \frac{\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}} \mid \dim(\mathcal{U}) = k \right\}. \tag{B.11}$$

Therefore, we have

$$\begin{aligned}
\lambda_k(\boldsymbol{AB}) &= \lambda_k(\boldsymbol{B}^{\frac{1}{2}}\boldsymbol{AB}^{\frac{1}{2}}) \\
&= \inf_{\mathcal{U}} \left\{ \sup_{\boldsymbol{x}\in\mathcal{U}} \frac{\boldsymbol{x}^T\boldsymbol{B}^{\frac{1}{2}}\boldsymbol{AB}^{\frac{1}{2}}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} \mid \dim(\mathcal{U}) = k \right\} \\
&= \inf_{\mathcal{U}} \left\{ \sup_{\boldsymbol{x}\in\mathcal{U}} \frac{\boldsymbol{x}^T\boldsymbol{B}^{\frac{1}{2}}\boldsymbol{AB}^{\frac{1}{2}}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{B}^{\frac{1}{2}}\boldsymbol{B}^{\frac{1}{2}}\boldsymbol{x}} \frac{\boldsymbol{x}^T\boldsymbol{Bx}}{\boldsymbol{x}^T\boldsymbol{x}} \mid \dim(\mathcal{U}) = k \right\} \\
&\leq \inf_{\mathcal{U}} \left\{ \sup_{\boldsymbol{x}\in\mathcal{U}} \frac{\boldsymbol{x}^T\boldsymbol{Ax}}{\boldsymbol{x}^T\boldsymbol{x}} \mid \dim(\mathcal{U}) = k \right\} \lambda_N(\boldsymbol{B}) \\
&= \lambda_k(\boldsymbol{A})\lambda_N(\boldsymbol{B}).
\end{aligned} \tag{B.12}$$

□

The following lemma bounds the eigenvalues of $\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f$, and the regularized Nyström approximation $\tilde{\boldsymbol{T}}_f$ in (3.49).

Lemma B.2. *With $\boldsymbol{K}_f$ as in (3.35) and $\tilde{\boldsymbol{T}}_f$ as in (3.49), the eigenvalues of $\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f$ are bounded as*

$$\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f \preceq \frac{\mu\sigma_{NL}}{\sigma_{NL}+\mu}\boldsymbol{I}'_s + \sigma_{NL}\boldsymbol{I}_s \tag{B.13}$$

where σ_{NL} is the largest eigenvalue of $\boldsymbol{K}_f$, $\boldsymbol{I}_s := diag([0,0,\ldots,1,1])$ has s zeros on its diagonal, and $\boldsymbol{I}'_s := \boldsymbol{I} - \boldsymbol{I}_s$.

Proof. Using the eigendecomposition $\boldsymbol{K}_f = \boldsymbol{Q}_f\boldsymbol{\Sigma}_f\boldsymbol{Q}_f^T$, we can write

$$\begin{aligned}
\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f &= \boldsymbol{K}_f - \boldsymbol{K}_f\boldsymbol{S}^T(\boldsymbol{SK}_f\boldsymbol{S}^T + \mu\boldsymbol{I})^{-1}\boldsymbol{SK}_f \\
&= \boldsymbol{Q}_f\boldsymbol{\Sigma}_f^{\frac{1}{2}}\left[\boldsymbol{I} - \boldsymbol{\Sigma}_f^{\frac{1}{2}}\boldsymbol{Q}_f^T\boldsymbol{S}^T(\boldsymbol{SQ}_f\boldsymbol{\Sigma}_f^{\frac{1}{2}}\boldsymbol{\Sigma}_f^{\frac{1}{2}}\boldsymbol{Q}_f^T\boldsymbol{S}^T + \mu\boldsymbol{I})^{-1}\boldsymbol{SQ}_f\boldsymbol{\Sigma}_f^{\frac{1}{2}}\right]\boldsymbol{\Sigma}_f^{\frac{1}{2}}\boldsymbol{Q}_f^T.
\end{aligned} \tag{B.14}$$

Applying the MIL to the matrix inside the square brackets of (B.14), we arrive at

$$\begin{aligned}
&\boldsymbol{I} - \boldsymbol{\Sigma}_f^{\frac{1}{2}}\boldsymbol{Q}_f^T\boldsymbol{S}^T\left(\boldsymbol{SQ}_f\boldsymbol{\Sigma}_f^{\frac{1}{2}}\boldsymbol{\Sigma}_f^{\frac{1}{2}}\boldsymbol{Q}_f^T\boldsymbol{S}^T + \mu\boldsymbol{I}\right)^{-1}\boldsymbol{SQ}_f\boldsymbol{\Sigma}_f^{\frac{1}{2}} \\
&= \left(\boldsymbol{I} + \frac{1}{\mu}\boldsymbol{\Sigma}_f^{\frac{1}{2}}\boldsymbol{Q}_f^T\boldsymbol{S}^T\boldsymbol{SQ}_f\boldsymbol{\Sigma}_f^{\frac{1}{2}}\right)^{-1}.
\end{aligned} \tag{B.15}$$

That in turn implies

$$\begin{aligned}
\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f &= \mu \boldsymbol{Q}_f \boldsymbol{\Sigma}_f^{\frac{1}{2}} \left(\mu \boldsymbol{I} + \boldsymbol{\Sigma}_f^{\frac{1}{2}} \boldsymbol{Q}_f^T \boldsymbol{S}^T \boldsymbol{S} \boldsymbol{Q}_f \boldsymbol{\Sigma}_f^{\frac{1}{2}}\right)^{-1} \boldsymbol{\Sigma}_f^{\frac{1}{2}} \boldsymbol{Q}_f^T \\
&= \mu \boldsymbol{Q}_f \boldsymbol{\Sigma}_f^{\frac{1}{2}} \left(\boldsymbol{\Sigma}_f + \mu \boldsymbol{I} - \boldsymbol{\Sigma}_f + \boldsymbol{\Sigma}_f^{\frac{1}{2}} \boldsymbol{Q}_f^T \boldsymbol{S}^T \boldsymbol{S} \boldsymbol{Q}_f \boldsymbol{\Sigma}_f^{\frac{1}{2}}\right)^{-1} \boldsymbol{\Sigma}_f^{\frac{1}{2}} \boldsymbol{Q}_f^T \\
&= \mu \boldsymbol{Q}_f \boldsymbol{\Sigma}_f^{\frac{1}{2}} \Big[(\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{\frac{1}{2}} \Big(\boldsymbol{I} - (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}} \boldsymbol{\Sigma}_f (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}} \\
&\quad + (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}} \boldsymbol{\Sigma}_f^{\frac{1}{2}} \boldsymbol{Q}_f^T \boldsymbol{S}^T \boldsymbol{S} \boldsymbol{Q}_f \boldsymbol{\Sigma}_f^{\frac{1}{2}} (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}}) \Big) (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{\frac{1}{2}} \Big]^{-1} \boldsymbol{\Sigma}_f^{\frac{1}{2}} \boldsymbol{Q}_f^T \\
&= \mu \boldsymbol{Q}_f \boldsymbol{\Sigma}_f^{\frac{1}{2}} (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}} (\boldsymbol{I} - \boldsymbol{P})^{-1} (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}} \boldsymbol{\Sigma}_f^{\frac{1}{2}} \boldsymbol{Q}_f^T
\end{aligned} \tag{B.16}$$

where

$$\boldsymbol{P} := \boldsymbol{\Sigma}_f (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-1} - (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}} \boldsymbol{\Sigma}_f^{\frac{1}{2}} \boldsymbol{Q}_f^T \boldsymbol{S}^T \boldsymbol{S} \boldsymbol{Q}_f \boldsymbol{\Sigma}_f^{\frac{1}{2}} (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}}. \tag{B.17}$$

Regarding the eigenvalues of $\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f$ in (B.16), we can bound them for $k = 1, \dots, NL$ as

$$\begin{aligned}
\lambda_k(\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f) &= \mu \, \lambda_k(\boldsymbol{Q}_f \boldsymbol{\Sigma}_f^{\frac{1}{2}} (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}} (\boldsymbol{I} - \boldsymbol{P})^{-1} (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}} \boldsymbol{\Sigma}_f^{\frac{1}{2}} \boldsymbol{Q}_f^T) \\
&= \mu \, \lambda_k(\boldsymbol{\Sigma}_f^{\frac{1}{2}} (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}} (\boldsymbol{I} - \boldsymbol{P})^{-1} (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}} \boldsymbol{\Sigma}_f^{\frac{1}{2}}) \\
&= \mu \, \lambda_k((\boldsymbol{I} - \boldsymbol{P})^{-1} (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-1} \boldsymbol{\Sigma}_f) \\
&\leq \frac{\mu \sigma_{NL}}{\sigma_{NL} + \mu} \lambda_k((\boldsymbol{I} - \boldsymbol{P})^{-1})
\end{aligned} \tag{B.18}$$

where $\lambda_k(\cdot)$ denotes the kth eigenvalue of a matrix, and we have applied Lemma B.1 on the third equality and the last inequality. Knowing that $\lambda_k(\boldsymbol{I} - \boldsymbol{P}) = \boldsymbol{I} - \lambda_k(\boldsymbol{P})$ we can now bound the eigenvalues of $\boldsymbol{P}$ as

$$\begin{aligned}
\lambda_k(\boldsymbol{P}) &= \lambda_k(\boldsymbol{\Sigma}_f^{\frac{1}{2}} (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}} \boldsymbol{Q}_f^T (\boldsymbol{I} - \boldsymbol{S}^T \boldsymbol{S}) \boldsymbol{Q}_f (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-\frac{1}{2}} \boldsymbol{\Sigma}_f^{\frac{1}{2}}) \\
&= \lambda_k(\boldsymbol{Q}_f^T (\boldsymbol{I} - \boldsymbol{S}^T \boldsymbol{S}) \boldsymbol{Q}_f (\boldsymbol{\Sigma}_f + \mu \boldsymbol{I})^{-1} \boldsymbol{\Sigma}_f) \\
&\leq \frac{\sigma_{NL}}{\sigma_{NL} + \mu} \lambda_k(\boldsymbol{Q}_f^T (\boldsymbol{I} - \boldsymbol{S}^T \boldsymbol{S}) \boldsymbol{Q}_f) \\
&= \frac{\sigma_{NL}}{\sigma_{NL} + \mu} \lambda_k(\boldsymbol{I}_s)
\end{aligned} \tag{B.19}$$

where we have applied Lemma B.1 on the second and third inequalities, and $\boldsymbol{I}_s := \text{diag}[0, 0, \dots, 1, 1]$ has s zeros on its diagonal. Since $\lambda_k(\boldsymbol{I} - \boldsymbol{P}) = 1 - \lambda_k(\boldsymbol{P})$ $\forall k = 1, \dots, NL$, we have that $\boldsymbol{I} - \boldsymbol{P} \succeq \boldsymbol{I} - \frac{\sigma_{NL}}{\sigma_{NL} + \mu} \boldsymbol{I}_s$, and thus

$$(\boldsymbol{I} - \boldsymbol{P})^{-1} \preceq \boldsymbol{I}_s' + \frac{\sigma_{NL} + \mu}{\mu} \boldsymbol{I}_s \tag{B.20}$$

where $\boldsymbol{I}'_s := \boldsymbol{I} - \boldsymbol{I}_s$. Finally, combining (B.20) with (B.18) yields

$$\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f \preceq \frac{\mu\sigma_{NL}}{\sigma_{NL}+\mu}\boldsymbol{I}'_s + \sigma_{NL}\boldsymbol{I}_s \tag{B.21}$$

which concludes the proof. □

Using Lemmas B.1 and B.2, we can proceed to establish a bound on the bias and variance. Considering the eigendecomposition $\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f = \boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^T$, we can write the bias in (B.9) as

$$bias = \left\|\boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^T\boldsymbol{\gamma}\right\|_2^2. \tag{B.22}$$

With $\tilde{\boldsymbol{\gamma}} := \boldsymbol{L}^T\boldsymbol{\gamma}$, and using Lemma B.2 the bias is bounded as

$$\begin{aligned} bias &= \|\boldsymbol{L}\boldsymbol{\Lambda}\tilde{\boldsymbol{\gamma}}\|_2^2 = \tilde{\boldsymbol{\gamma}}^T\boldsymbol{\Lambda}^2\tilde{\boldsymbol{\gamma}} \\ &\leq \frac{\mu^2\sigma_{NL}^2}{(\sigma_{NL}+\mu)^2}\tilde{\boldsymbol{\gamma}}^T\boldsymbol{I}'_s\tilde{\boldsymbol{\gamma}} + \sigma_{NL}^2\tilde{\boldsymbol{\gamma}}^T\boldsymbol{I}_s\tilde{\boldsymbol{\gamma}} \\ &= \frac{\mu^2\sigma_{NL}^2}{(\sigma_{NL}+\mu)^2}\sum_{i=1}^{s}\tilde{\gamma_i}^2 + \sigma_{NL}^2\sum_{i=s+1}^{NL}\tilde{\gamma}_i^2. \end{aligned} \tag{B.23}$$

To bound the variance in (B.10), recall that $\boldsymbol{e}$ is a Gaussian random vector with covariance matrix $\nu^2\boldsymbol{I}$, while $\bar{\boldsymbol{e}}$ has covariance matrix $\nu^2\boldsymbol{S}\boldsymbol{S}^T$. Then (B.10) is a quadratic form in $\boldsymbol{e}$, whose variance becomes

$$\begin{aligned} var &= \mathbb{E}_{\boldsymbol{e}}\left\{\frac{1}{\mu^2}\left\|(\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f)\boldsymbol{S}^T\bar{\boldsymbol{e}}\right\|_2^2\right\} \\ &= \frac{\nu^2}{\mu^2}\mathrm{Tr}(\boldsymbol{S}(\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f)^2\boldsymbol{S}^T) \\ &= \frac{\nu^2}{\mu^2}\mathrm{Tr}((\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f)^2\boldsymbol{S}^T\boldsymbol{S}). \end{aligned} \tag{B.24}$$

The matrix inside the trace in (B.24) has $NL - s$ zero entries in its diagonal. Lemma B.2, on the other hand, implies that diagonal entries of $\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f$ are smaller than its largest eigenvalue; that is, $\left[\boldsymbol{K}_f - \tilde{\boldsymbol{T}}_f\right]_{i,i} \leq \sigma_{NL}$. Coupling this with (B.24) yields

$$var \leq \frac{s\nu^2\sigma_{NL}^2}{\mu^2}. \tag{B.25}$$

Finally, combining the bias bound in (B.23) with the variance bound in (B.25), yields the

bound for the MSE as

$$MSE \leq \frac{\mu^2 \sigma_{NL}^2}{(\sigma_{NL}+\mu)^2} \sum_{i=1}^{s} \tilde{\gamma_i}^2 + \sigma_{NL}^2 \sum_{i=s+1}^{NL} \tilde{\gamma}_i^2 + \frac{s\nu^2 \sigma_{NL}^2}{\mu^2}. \tag{B.26}$$

Bibliography

[1] P. Giménez-Febrer, A. Pages-Zamora, S. S. Pereira, and R. López-Valcarce, "Distributed AOA-based source positioning in NLOS with sensor networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2015, pp. 3197–3201.

[2] C. F. Higham, R. Murray-Smith, M. J. Padgett, and M. P. Edgar, "Deep learning for real-time single-pixel video," *Scientific reports*, vol. 8, no. 1, pp. 1–9, Feb. 2018.

[3] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[5] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, March 2008.

[6] A. Montanari and S. Oh, "On positioning via distributed matrix completion," in *2010 IEEE Sensor Array and Multichannel Signal Processing Workshop*. IEEE, Oct. 2010, pp. 197–200.

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

[8] K. Xie, L. Wang, X. Wang, G. Xie, G. Zhang, D. Xie, and J. Wen, "Sequential and adaptive sampling for matrix completion in network monitoring systems," in *2015*

IEEE Conference on Computer Communications (INFOCOM). IEEE, May 2015, pp. 2443–2451.

[9] Y. Hu, X. Liu, and M. Jacob, "A generalized structured low-rank matrix completion algorithm for MR image recovery," *IEEE transactions on medical imaging*, vol. 38, no. 8, pp. 1841–1851, 2018.

[10] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[11] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from noisy entries," *The Journal of Machine Learning Research*, vol. 11, pp. 2057–2078, July 2010.

[12] 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, July 2018.

[13] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, Dec. 2009.

[14] G. Liu and P. Li, "Advancing Matrix Completion by Modeling Extra Structures beyond Low-Rankness," *arXiv preprint arXiv:1404.4646*, 2014.

[15] Y. Chen, S. Bhojanapalli, S. Sanghavi, and R. Ward, "Coherent matrix completion," in *International Conference on Machine Learning*, June 2014, pp. 674–682.

[16] J. Fan and T. W. Chow, "Sparse subspace clustering for data with missing entries and high-rank matrix completion," *Neural Networks*, vol. 93, pp. 36–44, 2017.

[17] K. Yi, J. Wan, T. Bao, and L. Yao, "A DCT regularized matrix completion algorithm for energy efficient data gathering in wireless sensor networks," *Int. Journal of Distributed Sensor Networks*, vol. 11, no. 7, p. 272761, Jul. 2015.

[18] J. Cheng, Q. Ye, H. Jiang, D. Wang, and C. Wang, "STCDG: an efficient data gathering algorithm based on matrix completion for wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 2, pp. 850–861, Feb. 2013.

[19] A. Gogna and A. Majumdar, "Matrix completion incorporating auxiliary information for recommender system design," *Expert Systems with Applications*, vol. 42, no. 14, pp. 5789–5799, Aug. 2015.

[20] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovacević, "Signal Recovery on Graphs: Variation Minimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 17, pp. 4609–4624, Sep. 2015.

[21] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, "Matrix Completion on Graphs," in *Neural Information Processing Systems Workshop "Out of the Box: Robustness in High Dimension"*, Dec. 2014.

[22] J. A. Bazerque and G. B. Giannakis, "Nonparametric basis pursuit via sparse kernel-based learning: a unifying view with advances in blind methods," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 112–125, Jul. 2013.

[23] D. Romero, M. Ma, and G. B. Giannakis, "Kernel-based reconstruction of graph signals," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 764–778, Feb. 2017.

[24] T. Zhou, H. Shan, A. Banerjee, and G. Sapiro, "Kernelized probabilistic matrix factorization: Exploiting graphs and side information," in *Proc. of SIAM Int. Conf. on Data Mining*, Jul. 2012, pp. 403–414.

[25] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[26] 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, Jan. 2010.

[27] S. Ma, D. Goldfarb, and L. Chen, "Fixed point and Bregman iterative methods for matrix rank minimization," *Mathematical Programming*, vol. 128, no. 1-2, pp. 321–353, Jun. 2011.

[28] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 123–231, 2013.

[29] K.-C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems," *Pacific Journal of Optimization*, vol. 6, no. 615-640, p. 15, March 2010.

[30] K. Konishi, K. Uruma, T. Takahashi, and T. Furukawa, "Iterative partial matrix shrinkage algorithm for matrix rank minimization," *Signal Processing*, vol. 100, pp. 124–131, July 2014.

[31] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, May 2013.

[32] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE transactions on signal processing*, vol. 61, pp. 1644–1656, April 2013.

[33] T. Jebara, J. Wang, and S.-F. Chang, "Graph construction and b-matching for semi-supervised learning," in *Proc. of the 26th Annual Int. Conf. on Machine Learning*, June 2009, pp. 441–448.

[34] S. Bhojanapalli and P. Jain, "Universal Matrix Completion," ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 2, 2014, pp. 1881–1889.

[35] Z. Xu, "The minimal measurement number for low-rank matrix recovery," *Applied and Computational Harmonic Analysis*, vol. 44, no. 2, pp. 497–508, March 2018.

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

[37] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*. Springer Series in Statistics, 2001.

[38] Y. J. Liu, D. Sun, and K. C. Toh, "An implementable proximal point algorithmic framework for nuclear norm minimization," *Mathematical Programming*, vol. 133, no. 1-2, pp. 399–436, Jan. 2012.

[39] N. Srebro and A. Shraibman, "Rank, Trace-Norm and Max-Norm," Tech. Rep.

[40] T. Hastie, R. Mazumder, J. D. Lee, and R. Zadeh, "Matrix completion and low-rank SVD via fast alternating least squares," *Journal of Machine Learning Research*, vol. 16, pp. 3367–3402, Jan. 2015.

[41] P. Giménez-Febrer and A. Pagès-Zamora, "Matrix completion of noisy graph signals via proximal gradient minimization," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, March 2017, pp. 4441–4445.

[42] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-point algorithms for inverse problems in science and engineering*. Springer, May 2011, pp. 185–212.

[43] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[44] E. T. Hale, W. Yin, and Y. Zhang, "A Fixed-Point Continuation Method for l1-Regularized Minimization with Applications to Compressed Sensing," *CAAM Technical Report*, 2007.

[45] S. Chen, A. Sandryhaila, G. Lederman, Z. Wang, J. M. Moura, P. Rizzo, J. Bielak, J. H. Garrett, and J. Kovacevic, "Signal inpainting on graphs via total variation minimization," in *Acoustics, Speech and Signal Processing, 2014 IEEE Int. Conf. on*, 2014, pp. 8267–8271.

[46] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012, vol. 3.

[47] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Advances in Neural Information Processing Systems*, Dec. 2015, pp. 2107–2115.

[48] K. Hou, Z. Zhou, A. M.-C. So, and Z.-Q. Luo, "On the linear convergence of the proximal gradient method for trace norm regularization," in *Advances in Neural Information Processing Systems*, Dec. 2013, pp. 710–718.

[49] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, March 2009.

[50] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proc. of ACM Int. Conf. on Web Search and Data Mining*, Feb. 2011, pp. 287–296.

[51] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert, "Low-rank matrix factorization with attributes," Ecole des mines de Paris, Tech. Rep., Sept. 2006.

[52] M. Stock, T. Pahikkala, A. Airola, B. De Baets, and W. Waegeman, "A comparative study of pairwise learning methods based on kernel ridge regression," *Neural Computation*, vol. 30, no. 8, pp. 2245–2283, Aug. 2018.

[53] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proc. of ACM Symp. on Theory of Computing*, Jun. 2013, pp. 665–674.

[54] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proc. of ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Aug. 2011, pp. 69–77.

[55] C. Saunders, A. Gammerman, and V. Vovk, "Ridge Regression Learning Algorithm in Dual Variables," in *Proceedings of the Fifteenth International Conference on Machine Learning*, ser. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., July 1998, p. 515–521.

[56] A. Shashua, "Introduction to machine learning: Class notes 67577," *arXiv preprint arXiv:0904.3664*, 2009.

[57] N. Srebro, J. Rennie, and T. S. Jaakkola, "Maximum-margin matrix factorization," in *Advances in Neural Information Processing Systems*, Dec. 2005, pp. 1329–1336.

[58] C. Teflioudi, F. Makari, and R. Gemulla, "Distributed matrix completion," in *Proc. of Int. Conf. on Data Mining*, Dec. 2012, pp. 655–664.

[59] M. Mardani, G. Mateos, and G. B. Giannakis, "Subspace Learning and Imputation for Streaming Big Data Matrices and Tensors," *IEEE Transactions on Signal Processing*, vol. 63, no. 10, pp. 2663–2677, May 2015.

[60] H. V. Henderson and S. R. Searle, "On deriving the inverse of a sum of matrices," *Siam Review*, vol. 23, no. 1, pp. 53–60, Jan. 1981.

[61] D. Romero, V. N. Ioannidis, and G. B. Giannakis, "Kernel-Based Reconstruction of Space-Time Functions on Dynamic Graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 856–869, Sept. 2017.

[62] T. Pahikkala, M. Stock, A. Airola, T. Aittokallio, B. De Baets, and W. Waegeman, "A two-step learning approach for solving full and almost full cold start problems in dyadic prediction," in *Joint European Conf. on Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 517–532.

[63] P. Drineas and M. W. Mahoney, "On the Nyström method for approximating a Gram matrix for improved kernel-based learning," *Journal of Machine Learning Research*, vol. 6, pp. 2153–2175, Dec. 2005.

[64] A. Alaoui and M. W. Mahoney, "Fast randomized kernel ridge regression with statistical guarantees," in *Advances in Neural Information Processing Systems*, Dec. 2015, pp. 775–783.

[65] Y. Yang, M. Pilanci, and M. J. Wainwright, "Randomized sketches for kernels: Fast and optimal non-parametric regression," *The Annals of Statistics*, vol. 45, no. 3, pp. 991–1023, Jun. 2017.

[66] H. Avron, K. L. Clarkson, and D. P. Woodruff, "Faster kernel ridge regression using sketching and preconditioning," *SIAM Journal on Matrix Analysis and Applications*, vol. 38, no. 4, pp. 1116–1138, Jan. 2017.

[67] S. Van Vaerenbergh and I. Santamaría, "Online Regression with Kernels," in *Regularization, Optimization, Kernels, and Support Vector Machines*. Chapman and Hall/CRC, March 2014, ch. 21, pp. 477–501.

[68] J. Lu, S. C. Hoi, J. Wang, P. Zhao, and Z.-Y. Liu, "Large scale online kernel learning," *Journal of Machine Learning Research*, vol. 17, no. 47, pp. 1–43, Jan. 2016.

[69] F. Sheikholeslami, D. Berberidis, and G. B. Giannakis, "Large-Scale Kernel-Based Feature Extraction via Low-Rank Subspace Tracking on a Budget," *IEEE Transactions on Signal Processing*, vol. 66, no. 8, pp. 1967–1981, April 2018.

[70] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the Trade*. Springer, 2012, pp. 421–436.

[71] I. D. Schizas, G. Mateos, and G. B. Giannakis, "Distributed LMS for Consensus-Based In-Network Adaptive Processing," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2365–2382, June 2009.

[72] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning Theory and Kernel Machines*. Springer, 2003, pp. 144–158.

[73] A. Cotter, J. Keshet, and N. Srebro, "Explicit approximations of the Gaussian kernel," *arXiv preprint arXiv:1109.4603*, 2011.

[74] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," vol. 20, Jan. 2007.

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

[76] O. Shamir and S. Shalev-Shwartz, "Matrix Completion with the Trace Norm: Learning, Bounding, and Transducing," *Journal of Machine Learning Research*, vol. 15, pp. 3401–3423, Oct. 2014.

[77] R. Foygel and N. Srebro, "Concentration-based guarantees for low-rank matrix reconstruction," in *Proc. of the 24th Annual Conf. on Learning Theory*, July 2011, pp. 315–340.

[78] X. J. Zhu, "Semi-supervised learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2005.

[79] R. El-Yaniv and D. Pechyony, "Transductive Rademacher complexity and its applications," *Journal of Artificial Intelligence Research*, vol. 35, pp. 193–234, 2009.

[80] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems*, Dec. 2016, pp. 3981–3989.

[81] J. A. Bazerque, G. Mateos, and G. B. Giannakis, "Rank Regularization and Bayesian Inference for Tensor Completion and Extrapolation," *IEEE Transactions on Signal Processing*, vol. 61, no. 22, pp. 5689–5703, Nov. 2013.

[82] Q. Gu and J. Han, "Towards active learning on graphs: An error bound minimization approach," in *2012 IEEE 12th International Conference on Data Mining*. IEEE, Dec. 2012, pp. 882–887.

[83] S. Chakraborty, J. Zhou, V. Balasubramanian, S. Panchanathan, I. Davidson, and J. Ye, "Active matrix completion," in *2013 IEEE 13th International Conference on Data Mining*. IEEE, Dec. 2013, pp. 81–90.

[84] S. Mak and Y. Xie, "Active matrix completion with uncertainty quantification," *arXiv preprint arXiv:1706.08037*, 2017.

[85] D. J. Sutherland, B. Póczos, and J. Schneider, "Active learning and search on low-rank matrices," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, Aug. 2013, pp. 212–220.

[86] A. Bhargava, R. Ganti, and R. Nowak, "Active positive semidefinite matrix completion: Algorithms, theory and applications," in *Artificial Intelligence and Statistics*, 2017, pp. 1349–1357.

[87] H. Yu and S. Kim, "Passive Sampling for Regression," in *2010 IEEE International Conference on Data Mining*, Dec. 2010, pp. 1151–1156.

[88] D. Wu, "Pool-Based Sequential Active Learning for Regression," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 5, pp. 1348–1359, May 2019.

[89] A. Tanaka, H. Imai, and M. Miyakoshi, "Kernel-induced sampling theorem," *IEEE Transactions on Signal Processing*, vol. 58, no. 7, pp. 3569–3577, March 2010.

[90] R. Patel, T. Goldstein, E. Dyer, A. Mirhoseini, and R. Baraniuk, "Deterministic column sampling for low-rank matrix approximation: Nyström vs. incomplete Cholesky decomposition," in *Proceedings of the 2016 SIAM international conference on data mining*. SIAM, July 2016, pp. 594–602.

[91] C. Musco and C. Musco, "Recursive sampling for the Nyström method," in *Advances in Neural Information Processing Systems*, Dec. 2017, pp. 3833–3845.

[92] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication," *SIAM Journal on Computing*, vol. 36, no. 1, pp. 132–157, 2006.

[93] D. P. Woodruff, "Sketching as a Tool for Numerical Linear Algebra," *Theoretical Computer Science*, vol. 10, no. 1-2, pp. 1–157, 2014.

[94] A. G. Wilson, E. Gilboa, A. Nehorai, and J. P. Cunningham, "GPatt: Fast multidimensional pattern extrapolation with Gaussian processes," *arXiv preprint arXiv:1310.5288*, 2013.

[95] M. Zhang and Y. Chen, "Inductive matrix completion based on graph neural networks," in *International Conference on Learning Representations*, 2020.

[96] M. Yang and S. Xu, "A novel patch-based nonlinear matrix completion algorithm for image analysis through convolutional neural network," *Neurocomputing*, vol. 389, pp. 56 – 82, 2020.

[97] M. Bagherian, R. B. Kim, C. Jiang, M. A. Sartor, H. Derksen, and K. Najarian, "Coupled matrix–matrix and coupled tensor–matrix completion methods for predicting drug–target interactions," *Briefings in Bioinformatics*, 03 2020.

[98] V. Garg, P. Giménez-Febrer, A. Pagès-Zamora, and I. Santamaria, "Source Enumeration via Toeplitz Matrix Completion," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2020, pp. 6004–6008.

[99] V. Garg, P. Giménez-Febrer, A. Pagès-Zamora, and I. Santamaría, "Energy-Efficient DOA Estimation via Shift Invariance Matrix Completion," *Elsevier Journal on Advances in Signal Processing*, 2020 (submitted).

[100] E. Leicht, P. Holme, and M. E. Newman, "Vertex similarity in networks," *Physical Review E*, vol. 73, no. 2, p. 026120, Oct 2006.

[101] P. Lax, *Linear Algebra and its Applications*. Wiley, 2007.