2021

High-Dimensional Machine Learning Models In Fintech

Qiong Wu
William & Mary - Arts & Sciences, qiongwu9@gmail.com

High-dimensional Machine Learning Models in FinTech

Qiong Wu

Tongyu, Jilin, China

Bachelor of Engineering, Dalian University of Technology, 2014
Master of Science, The University of Hong Kong, 2016

A Dissertation presented to the Graduate Faculty of
The College of William & Mary in Candidacy for the Degree of
Doctor of Philosophy

Department of Computer Science

College of William & Mary
January 2022

APPROVAL PAGE

This Dissertation is submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

Qiong Wu

Approved by the Committee, January 2022

Committee Chair
Zhenming Liu, Assistant Professor, Computer Science
College of William & Mary

Andreas Stathopoulos, Professor, Computer Science
College of William & Mary

Gang Zhou, Professor, Computer Science
College of William & Mary

Bin Ren, Assistant Professor, Computer Science
College of William & Mary

Mihai Cucuringu, Associate Professor
University of Oxford

ABSTRACT

This thesis develops several forecasting models for simultaneously predicting the prices of d assets traded in financial markets, a most fundamental problem in the emerging area of "FinTech". The models are optimized to address three critical challenges, *C1. High-dimensional interactions between assets.* Assets could interact (e.g., Amazon's disclosure of its revenue change in cloud services could indicate that revenues also could change in other cloud providers). The number of possible interactions is quadratic in d, and is often much larger than the number of observations. *C2. Non-linearity of the hypothesis class.* Linear models are usually insufficient to characterize the relationship between the labels (responses) and the available information (features). *C3. Data scarcity for each asset.* The size of the data associated with an individual asset could be small. For example, a typical daily forecasting model based on technical factors uses three years (approx. 750 trading days) of data. We collect one data point for each day so only 750 observations are available for each asset. We develop the following works to address these challenges.
1. Adaptive reduced rank regression (addressing C1): We examine a linear regression model $\mathbf{y} = M\mathbf{x} + \epsilon$ that aims to directly capture the interactions between all features from all assets and all the responses, by estimating $d \times \omega(d)$ entries in M using $O(d)$ observations. In this setting, existing low-rank regularization techniques such as reduced rank regression or nuclear-norm based regularizations fail to work. Adaptive Reduced Rank Regression is a new provable algorithm for estimating M under a mild assumption on the spectrum of the covariance matrix of $\mathbf{x}$.
2. On embedding stocks (addressing C1 & C2). We next propose a semi-parametric model called the "additive influence model" that decomposes the inference problem into two orthogonal subroutines. One subroutine is used to learn the high-dim interactions between entities, and we solve the problem with techniques developed for Adaptive-RRR. The other subroutine is used to learn the non-linear signals, and we solve the problem with practical algorithms such as deep learning and ensemble learning.
3. Equity2Vec: Interaction beyond return correlations (addressing C2 & C3). We develop a specialized neural net model for each asset (e.g., train $g_i(\cdot)$ for asset i) but there is insufficient data to properly train g_i with data only from i (because of C3). Our idea is to shrink $g_i(\cdot)$'s toward one or more centroids to reduce model (sample) complexities. Specifically, we train a neural net model $g(\mathbf{x}_i, W, W_i)$, where W is shared across all entities, W_i is entity-specific and is learned through embedding, and $g_i(\mathbf{x}_i) = g(\mathbf{x}_i, W, W_i)$. When entities i and j are close, then W_i and W_j are close. Consequently, g_i and g_j will be similar when entity i and entity j are similar.
The proposed algorithms/models are verified via extensive experiments based on real-world equity datasets. Our forecasting models can also be applied to a wide range of applications, such as identifying biomarkers, understanding risks associated with various diseases, image recognition, and link prediction.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

This dissertation would not finish without the support of many people. First, I wish to express my deepest gratitude to my academic advisor, Dr. Zhenming Liu for his insightful and diligent mentoring along with the whole Ph.D. journal. He helped to identify interesting research topics, carry out deep analysis and build up my research skills.

During my Ph.D., I was fortunate to closely collaborate with Mihai Cucuringu, Andreas Stathopoulos, Wen-ling Hsu, Christopher G. Brinton, Yanhua Li, Lirong Xia, Varun Kande, Ao Liu, Tan Xu, and Guy Jacobson. I thank them for their enormous advice and countless hours. I extend my thank to my committee members: Dr. Andreas Stathopoulos, Dr. Gang Zhou, Dr. Bin Ren, and Dr. Mihai Cucuringu for their valuable questions and comments, which help a lot in improving the dissertation.

My Ph.D. research was performed using the computing facilities at William & Mary. I thank the entire technical staff for managing those facilities. In particular, I thank Eric Walter for always helping me with my requests. I also thank the entire administrative staff of the Computer Science department; Vanessa Godwin and Dale Hayes for being efficient, professional, and above all caring.

Finally, I have been gifted with a caring, patient, and supportive family and friends. I am thankful to my husband Hongyuan, my sister, my parents, and my friends for their unconditional love and endless support. In the end, I would like to thank my adorable full friend, Lume, for the joy she brought to me. By pursuing a Ph.D., I could become a much better researcher, engineer, and person. It was a truly valuable experience for me and thank everyone who helped me get through the Ph.D.

To my family.

LIST OF TABLES

LIST OF FIGURES

High-dimensional Machine Learning Models in FinTech

Chapter 1

Introduction

Predicting the assets prices or return is of fundamental importance to the financial technology community as the successful prediction of assets' future price could yield significant profit [71, 157, 103]. This thesis investigates using machine learning techniques to simultaneously forecast the future return for a large number of stocks traded in a region. For example, in the US market, we generally build models to predict the next 5-day returns for the S&P 500 or the Russell 3000.

Forecasting future prices of equities has been extensively studied [165, 22, 60, 12, 81, 69, 21, 86, 46, 70, 62, 26, 155, 49, 73, 53, 68, 38, 51, 28, 64, 26, 39, 167, 72, 80, 92]. Most have examined how the fundamental characteristics, such as the book/market ratio, capital, and momentum of the issuing firm could affect the price of the asset over the long term, or how various events or macroeconomic factors could cause price changes [36, 127, 81]. The studies have tended to use low-frequency models because the response horizon is in the order of months (e.g., how a stocks price changes after 3 months). Linear statistical models need to be used because the data sets are remarkably small, e.g., each asset is associated with only a few data points (e.g., each month corresponds to a data point). "Mid-frequency" models with one or multiple days of forecasting horizon had been studied more extensively in the private sectors. It has been observed that trading activities (e.g., price/volume changes) have a heavier impact on an equitys short term price movement than fundamental

factors so significant effort was devoted to understanding the prediction powers of technical factors such as volatility or trading volume changes [155, 26, 49] for empirical asset pricing. Because we can collect a more reasonable amount of data (e.g., one trading day results in one data point), using machine learning techniques to extract interactions becomes possible. Recent years have witnessed explosive development of mid-frequency ML models from both finance and computer science.

"High-frequency" models investigate price changes over an even shorter time horizon (e.g., next 10-second return). The price changes in this horizon are usually driven by the market microstructure (the dynamics of the bid-ask queues). Here, the modeling challenge is more on the computational side. It is often remarkably to fit an ML model properly with an abundant amount of data. So a significant fraction of effort focuses on speeding up the system, which resembles typical system and architecture research [160, 101, 75, 98, 94, 99].

Our works examine the mid-frequency models and focus on their statistical challenges, and do not examine market microstructure. Under this setting, we focus on learning how technical factors impact equities prices. We aim to tackle three key challenges that are not properly addressed in prior works [155, 72, 167, 92, 20, 25, 84, 118, 73]:

C1. High-dimensional interactions between assets. The current state of one asset could potentially impact the future state of another. For example, Amazon's disclosure of its revenue change in cloud services could indicate that revenues also could change in other cloud providers. The number of possible interactions is excessively large and can be even significantly larger than the number of observations. For example, in a portfolio of 3,000 stocks, the total number of potential links between pairs of stocks is $3,000 \times 3,000 \approx 10^7$, whereas we typically have 10 years of daily data with only 2,500 data points [124, 62, 81, 28]. This setting is also referred to as the high dimensional setting and is prone to have severe overfitting issues. It requires careful analysis of a models theoretical properties before fitting the data.

C2. Non-linearity of the hypothesis class. Linear models are usually insufficient

to characterize the relationship between the response/label and the available information (features), so techniques beyond simple linear regressions are heavily needed.

C3. Data scarcity for training individual asset model. While we usually have a large volume of data, the size of the data associated with an individual asset could be small and is insufficient for properly train the model for the individual asset. For example, a typical daily forecasting model based on technical factors uses 10 years (approx. 2500 trading days) of data. We collect one data point for each day so only 2500 observations are available for each asset.

1.1 Overview of our contribution

We develop three works that are optimized to address one or more of these critical challenges.

Chapter 2: Adaptive reduced rank regression (address C1). We propose adaptive reduced rank regression to address the high-dimensional interaction challenge. We assume a total number of d_2 stocks and we examine the regression model. The system proceeds in rounds for a total of T rounds and regression problem $\mathbf{y}_t = M\mathbf{x}_t + \epsilon$. At round t, asset i is associated with features $\mathbf{x}_{i,t} \in \mathbf{R}^k$ and a response $y_{i,t} \in \mathbf{R}$ that needs to be predicted. We use all $\mathbf{x}_t \in \mathbf{R}^{d_1} = \{\mathbf{x}_{i,t}\}_{i \in d_2}$ to predict all $\mathbf{y}_t = \{y_{i,t}\}_{i \in [d_2]}$. $M \in \mathbf{R}^{d_2 \times d_1}$ are the learnable parameters. For example, we aim to predict the returns of a collection of financial stocks in the S&P500. In round t, $y_{i,t}$ refers to the next period return of asset i, $x_{i,t}$ refers to the features associated with stock i such as the technical factors (e.g., recent trading volumes of i), $\mathbf{y}_t$ refers to the next period return of all the stocks and $\mathbf{x}_t$ refers to the features from all the stocks.

Recall that the regression is under high-dimension setting since the number of observations n is significantly less than d_1. Existing low-rank regularization techniques (e.g., [7, 76, 84, 118, 104] are not optimized for the large feature size setting. These results assume that either the features possess the so-called restricted isometry property [23], or their covariance matrix can be accurately estimated [118]. Therefore, their sample complexity n depends on either d_1 or the smallest eigenvalue value $\lambda_{\min}$ of $\mathbf{x}$'s covariance

matrix. For example, a mean-squared error (MSE) result that appeared in [118] is of the form $O\left(\frac{r(d_1+d_2)}{n\lambda_{\min}^2}\right)$. When $n \leq d_1/\lambda_{\min}^2$, this result becomes trivial because the forecast $\hat{\mathbf{y}} = \mathbf{0}$ produces a comparable MSE. We design a new provable algorithm for estimating M called Adaptive Reduced Rank Regression (Adaptive-RRR). Our algorithm is a simple two-stage algorithm. Let $\mathbf{X} \in \mathbf{R}^{n \times d_1}$ be a matrix that stacks together all features and $\mathbf{Y} \in \mathbf{R}^{n \times d_2}$ be the one that stacks the responses. In the first stage, we run a principal component analysis (PCA) on $\mathbf{X}$ to obtain a set of uncorrelated features $\hat{\mathbf{Z}}$. In the second stage, we run another PCA to obtain a low rank approximation of $\hat{\mathbf{Z}}^{\mathrm{T}}\mathbf{Y}$ and use it to construct an output.

While the algorithm is operationally simple, we show a powerful and generic result on using PCA to process features, a widely used practice for "dimensionality reduction" [24, 58, 53]. PCA is known to be effective to orthogonalize features by keeping only the subspace explaining large variations. But its performance can only be analyzed under the so-called factor model [143, 142]. We show the efficacy of PCA *without* the factor model assumption. Instead, PCA should be interpreted as a robust estimator of $\mathbf{x}$'s covariance matrix. The empirical estimator $C = \frac{1}{n}\mathbf{X}\mathbf{X}^{\mathrm{T}}$ in the high-dimensional setting cannot be directly used because $n \ll d_1 \times d_2$, but it exhibits an interesting regularity: the leading eigenvectors of C are closer to ground truth than the remaining ones. In addition, the number of reliable eigenvectors grows as the sample size grows, so our PCA procedure projects the features along reliable eigenvectors and *dynamically* adjusts $\hat{\mathbf{Z}}$'s rank to maximally utilize the raw features. Under mild conditions on the ground-truth covariance matrix C^* of $\mathbf{x}$, we show that it is always possible to decompose $\mathbf{x}$ into a set of near-independent features and a set of (discarded) features that have an inconsequential impact on a model's MSE.

When features $\mathbf{x}$ are transformed into uncorrelated ones $\mathbf{z}$, our original problem becomes $\mathbf{y} = N\mathbf{z} + \epsilon$, which can be reduced to a matrix denoising problem [41] and be solved by the second stage. Our algorithm guarantees that we can recover all singular vectors of N whose associated singular values are larger than a certain threshold τ. The performance guarantee can be translated into MSE bounds parametrized by commonly used variables

(though, these translations usually lead to looser bounds). For example, when N's rank is r, our result reduces the MSE from $O(racr(d_1+d_2)n\lambda_{\min}^2)$ to $O(\frac{rd_2}{n}+n^{-c})$ for a suitably small constant c. The improvement is most pronounced when $n \ll d_1$.

We also provide a new matching lower bound. Our lower bound asserts that *no algorithm* can recover a fraction of singular vectors of N whose associated singular values are smaller than $\rho\tau$, where ρ is a "gap parameter". Our lower bound contribution is twofold. First, we introduce a notion of "local minimax", which enables us to define a lower bound parametrized by the singular values of N. This is a stronger lower bound than those delivered by the standard minimax framework, which are often parametrized by the rank r of N [84]. Second, we develop a new probabilistic technique for establishing lower bounds under the new local minimax framework. Roughly speaking, our techniques assemble a large collection of matrices that share the same singular values of N but are far from each other, so no algorithm can successfully distinguish these matrices with identical spectra.

Adaptive reduced rank regression (Adaptive-RRR) is a new and provably optimal algorithm for estimating M under a mild average case assumption over the features. Our algorithm is a simple two-stage algorithm. Let $\mathbf{X} \in \mathbf{R}^{n\times d_1}$ be a matrix that stacks together all features and $\mathbf{Y} \in \mathbf{R}^{n\times d_2}$ be the one that stacks the responses. In the first stage, we run a principal component analysis (PCA) on $\mathbf{X}$ to obtain a set of uncorrelated features $\hat{\mathbf{Z}}$. In the second stage, we run another PCA to obtain a low-rank approximation of $\hat{\mathbf{Z}}^{\mathrm{T}}\mathbf{Y}$ and use it to construct an output. We also provide an upper bound and a new matching lower bound.

Chapter 3: On embedding stocks (addressing C1 & C2). This work describes how we can decouple the learning of high-dimensional stock interactions (C1) and non-linear learning of feature interactions (C2) so that the former problem is solvable by provable algorithms and the latter is solvable by a wide range of practical machine learning [155, 72, 100, 159] techniques such as deep learning, boosting tree, and non-

parametric methods.

We propose a latent position model for equity returns, dubbed as the additive influence model. Our model assumes that each stock i is associated with a vector representation $\mathbf{z}_i$ in a (latent) Euclidean space, and characterizes the interactions between stocks in the form as

$$\mathbf{y}_{t,i} = \sum_{j \in [d]} \kappa(\mathbf{z}_i, \mathbf{z}_j) g(\mathbf{x}_{j,t}) + \xi_{t,i}, \tag{1.1}$$

where $\mathbf{y}_{t,i} \in \mathbf{R}$ is the next period return at time t for stock i, $\mathbf{x}_{t,j} \in \mathbf{R}^k$ are the features associated with stock j at time t, $\xi_{t,i}$ is a noise term, $g : \mathbf{R}^k \to \mathbf{R}$ is an unknown function, and κ is a function that measures the interaction strength between stocks based on their vector representations. When $\mathbf{z}_i$ and $\mathbf{z}_j$ are close, $\kappa(\mathbf{z}_i, \mathbf{z}_j)$ will be large, and thus the variable $g(\mathbf{x}_{t,j})$ from stock j's has a stronger impact on i's return.

Our proposed model allows for feature interactions through $g(\cdot)$, and addresses the overfitting problem arising from stock interactions because the distances (interaction strength) between stocks are constrained by the latent Euclidean space: when $\mathbf{z}_i - \mathbf{z}_j$ and $\mathbf{z}_j - \mathbf{z}_k$ are small, $\mathbf{z}_i - \mathbf{z}_k$ is also small, and thus the degree of freedom for stock interactions becomes substantially smaller than $O(d^2)$.

Our goal is to learn both the $\mathbf{z}_i$'s and $g(\cdot)$. We note that these two learning tasks can be *decoupled*: high-dim methods can be developed to *provably* estimate the $\mathbf{z}_i$'s *without the knowledge of* $g(\cdot)$, and when estimates of $\mathbf{z}_i$'s are given, an experiment-driven process can be used to learn $g(\cdot)$ by examining prominent machine learning methods such as neural nets and boosting. In other words, when we learn stock interactions, we do not need to be troubled by the overfitting problem escalated by fine-tuning $g(\cdot)$, and when we learn feature interactions, the generalization error will not be jeopardized by the curse of dimensionality from stock interactions.

To learn the $\mathbf{z}_i$'s, we design a simple algorithm that uses low-rank approximation of $\mathbf{y}_t$'s covariance matrix to find the closeness of the stocks, and develop a novel theoretical analysis based on recent techniques from high-dim and kernel learning [15, 147, 161].

To learn $g(\cdot)$, we generalize major machine learning techniques, including neural nets, non-parametric, and boosting methods, to the additive influence model when estimates of $\mathbf{z}_i$'s are known. We specifically develop a moment-based algorithm for non-parametric learning of $g(\cdot)$, and a computationally efficient boosting algorithm based on linear learners by using the domain knowledge of equity data sets.

Chapter 4: Equity2Vec (addressing C2 & C3).

We propose Equity2Vec to learn the stock embedding reflects non-return interactions and address the data scarcity challenge (C3) using neural net model (C2).

First, we examine how to learn the stock embeddings. Most research analyzed the interactions between stocks by modeling their correlations over returns. However, the stock interactions are beyond return correlations. For example, in early 2021, the AMC theatres, GameStop, and BlackBerry suddenly show the co-movement. All of them show the soaring stock price due to the investors formed on social media are buying up these stocks. Thus, it is logical to ask how can we generalize the notion of correlation, or use non-return information to analyze the interactions. We make two key observations by analyzing news on stocks. When two stocks are frequently co-mentioned, 1) they are likely to share common characteristics such as sector and supply-chain relation, 2) their prices tend to have a similar trend. Based on our observations, we designed the graph-based component and learned the stock embeddings that effectively capture both long-term and evolving cross-sectional interactions using news co-mention.

To solve the data scarcity challenge, we develop a specialized neural net model for each asset (e.g., train $g_i(\cdot)$ for asset i) but there is insufficient data to properly train $g_i(\cdot)$ with data only from i (because of C3). Therefore, we use the data associated with other entities for training. We aim to train a neural net model $g(\mathbf{x}_i, W, W_i)$, where W is shared across all assets, W_i is entity-specific and is learned through embedding, and $g_i(\mathbf{x}_i) = g(\mathbf{x}_i, W, W_i)$. When assets i and j are close, then W_i and W_j are close. Consequently, g_i and g_j will be similar when asset i and asset j are similar.

All the proposed algorithms/models are verified via extensive experiments based on real-world equity datasets. Our forecasting models can also be applied to a wide range of applications, such as identifying biomarkers, understanding risks associated with various diseases, customer care [158], image recognition, and link prediction [96], topic modeling [].

1.2 Dissertation Organization

The rest of this dissertation is organized as follows. Chapter 2 discusses more details about Adaptive-RRR, where we demonstrate our upper bound, new matching lower bound, and the experiments. In Chapter 3, we demonstrate how we use high-dimensional kernel-based techniques to design a provably correct algorithm for revealing stock latent position, and how we integrate the algorithm with machine learning models including two new algorithmic techniques (a statistically sound non-parametric method and an ensemble learning algorithm optimized for vector regressions). In Chapter 4, we propose Equity2Vec, where we use both static and dynamic non-return interactions to price the assets. In Chapter 5, we summarize the contributions of our works and discuss the future research directions.

Chapter 2

Adaptive reduced rank regression

2.1 Introduction

We consider the regression problem $\mathbf{y} = M\mathbf{x} + \epsilon$ in the high dimensional setting, where $\mathbf{x} \in \mathbf{R}^{d_1}$ is the vector of features, $\mathbf{y} \in \mathbf{R}^{d_2}$ is a vector of responses, $M \in \mathbf{R}^{d_2 \times d_1}$ are the learnable parameters, and $\epsilon \sim N(0, \sigma_\epsilon^2 I_{d_2 \times d_2})$ is a noise term. High-dimensional setting refers to the case where the number of observations n is insufficient for recovery and hence regularization for estimation is necessary [84, 118, 27]. This high-dimensional model is widely used in practice, such as identifying biomarkers [170], understanding risks associated with various diseases [52, 14], image recognition [129, 47], forecasting equity returns in financial markets [124, 142, 107, 16], and analyzing social networks [163, 130].

We consider the "large feature size" setting, in which the number of features d_1 is excessively large and can be even larger than the number of observations n. This setting frequently arises in practice because it is often straightforward to perform feature-engineering and produce a large number of potentially useful features in many machine learning problems. For example, in a typical equity forecasting model, n is around 3,000 (i.e., using 10 years of market data), whereas the number of potentially relevant features can be in the order of thousands [124, 62, 81, 28]. In predicting the popularity of a user in an online social network, n is in the order of hundreds (each day is an observation and a typical dataset contains less than three years of data) whereas the feature size can easily

be more than 10k [133, 13, 140].

Existing low-rank regularization techniques (e.g., [84, 118, 104]) are not optimized for the large feature size setting. These results assume that either the features possess the so-called restricted isometry property [23], or their covariance matrix can be accurately estimated [118]. Therefore, their sample complexity n depends on either d_1 or the smallest eigenvalue value $\lambda_{\min}$ of $\mathbf{x}$'s covariance matrix. For example, a mean-squared error (MSE) result that appeared in [118]is of the form $O\left(\frac{r(d_1+d_2)}{n\lambda_{\min}^2}\right)$. When $n \leq d_1/\lambda_{\min}^2$, this result becomes trivial because the forecast $\hat{\mathbf{y}} = \mathbf{0}$ produces a comparable MSE. We design an efficient algorithm for the large feature size setting. Our algorithm is a simple two-stage algorithm. Let $\mathbf{X} \in \mathbf{R}^{n\times d_1}$ be a matrix that stacks together all features and $\mathbf{Y} \in \mathbf{R}^{n\times d_2}$ be the one that stacks the responses. In the first stage, we run a principal component analysis (PCA) on $\mathbf{X}$ to obtain a set of uncorrelated features $\hat{\mathbf{Z}}$. In the second stage, we run another PCA to obtain a low rank approximation of $\hat{\mathbf{Z}}^{\mathrm{T}}\mathbf{Y}$ and use it to construct an output.

While the algorithm is operationally simple, we show a powerful and generic result on using PCA to process features, a widely used practice for "dimensionality reduction" [24, 58, 53]. PCA is known to be effective to orthogonalize features by keeping only the subspace explaining large variations. But its performance can only be analyzed under the so-called factor model [143, 142]. We show the efficacy of PCA *without* the factor model assumption. Instead, PCA should be interpreted as a robust estimator of $\mathbf{x}$'s covariance matrix. The empirical estimator $C = \frac{1}{n}\mathbf{X}\mathbf{X}^{\mathrm{T}}$ in the high-dimensional setting cannot be directly used because $n \ll d_1 \times d_2$, but it exhibits an interesting regularity: the leading eigenvectors of C are closer to ground truth than the remaining ones. In addition, the number of reliable eigenvectors grows as the sample size grows, so our PCA procedure projects the features along reliable eigenvectors and *dynamically* adjusts $\hat{\mathbf{Z}}$'s rank to maximally utilize the raw features. Under mild conditions on the ground-truth covariance matrix C^* of $\mathbf{x}$, we show that it is always possible to decompose $\mathbf{x}$ into a set of near-independent features and a set of (discarded) features that have an inconsequential impact on a model's MSE.

When features $\mathbf{x}$ are transformed into uncorrelated ones $\mathbf{z}$, our original problem becomes

$\mathbf{y} = N\mathbf{z} + \epsilon$, which can be reduced to a matrix denoising problem [41] and be solved by the second stage. Our algorithm guarantees that we can recover all singular vectors of N whose associated singular values are larger than a certain threshold τ. The performance guarantee can be translated into MSE bounds parametrized by commonly used variables (though, these translations usually lead to looser bounds). For example, when N's rank is r, our result reduces the MSE from $O(\frac{r(d_1+d_2)}{n\lambda_{\min}^2})$ to $O(\frac{rd_2}{n} + n^{-c})$ for a suitably small constant c. The improvement is most pronounced when $n \ll d_1$.

We also provide a new matching lower bound. Our lower bound asserts that *no algorithm* can recover a fraction of singular vectors of N whose associated singular values are smaller than $\rho\tau$, where ρ is a "gap parameter". Our lower bound contribution is twofold. First, we introduce a notion of "local minimax", which enables us to define a lower bound parametrized by the singular values of N. This is a stronger lower bound than those delivered by the standard minimax framework, which are often parametrized by the rank r of N [84]. Second, we develop a new probabilistic technique for establishing lower bounds under the new local minimax framework. Roughly speaking, our techniques assemble a large collection of matrices that share the same singular values of N but are far from each other, so no algorithm can successfully distinguish these matrices with identical spectra.

2.2 Preliminaries

Notation. Let $\mathbf{X} \in \mathbf{R}^{n\times d_1}$ and $\mathbf{Y} \in \mathbf{R}^{n\times d_2}$ be data matrices with their i-th rows representing the i-th observation. For matrix A, we denote its singular value decomposition as $A = U^A\Sigma^A(V^A)^{\mathrm{T}}$ and $\mathbf{P}_r(A) \triangleq U_r^A\Sigma_r^A{V_r^A}^{\mathrm{T}}$ is the rank r approximation obtained by keeping the top r singular values and the corresponding singular vectors. When the context is clear, we drop the superscript A and use U, Σ, and V (U_r, Σ_r, and V_r) instead. Both $\sigma_i(A)$ and σ_i^A are used to refer to i-th singular value of A. We use MATLAB notation when we refer to a specific row or column, e.g., $V_{1,:}$ is the first row of V and $V_{:,1}$ is the first column. $\|A\|_F$, $\|A\|_2$, and $\|A\|_*$ are Frobenius, spectral, and nuclear norms of A. In general, we use

boldface upper case (e.g., $\mathbf{X}$) to denote data matrices and boldface lower case (e.g., $\mathbf{x}$) to denote one sample. Regular fonts denote other matrices. Let $C^* = \mathbb{E}[\mathbf{x}\mathbf{x}^{\mathrm{T}}]$ and $C = \frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X}$ be the empirical estimate of C^*. Let $C^* = V^*\Lambda^*(V^*)^{\mathrm{T}}$ be the eigen-decomposition of the matrix C^*, and $\lambda_1^* \geq \lambda_2^*, \ldots, \geq \lambda_{d_1}^* \geq 0$ be the diagonal entries of Λ^*. Let $\{\mathbf{u}_1, \mathbf{u}_2, \ldots \mathbf{u}_\ell\}$ be an arbitrary set of column vectors, and $\mathrm{Span}(\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_\ell\})$ be the subspace spanned by it. An event happens with high probability means that it happens with probability $\geq 1 - n^{-5}$, where 5 is an arbitrarily chosen large constant and is not optimized.

STEP-1-PCA-X($\mathbf{X}$)

$[U, \Sigma, V] = \mathrm{svd}(\mathbf{X})$
$\Lambda = \frac{1}{n}(\Sigma^2)$; $\lambda_i = \Lambda_{i,i}$.
▷ **Gap thresholding.**
▷ $\delta = n^{-O(1)}$ is a tunable parameter.
$k_1 = \max\{k_1 : \lambda_{k_1} - \lambda_{k_1+1} \geq \delta\}$,
Λ_{k_1}: diagonal matrix comprised of $\{\lambda_i\}_{i \leq k_1}$.
U_{k_1}, V_{k_1}: k_1 leading columns of U and V.
$\hat{\Pi} = (\Lambda_{k_1})^{-\frac{1}{2}} V_{k_1}^{\mathrm{T}}$
$\hat{\mathbf{Z}}_+ = \sqrt{n} U_{k_1} (= X\hat{\Pi}^{\mathrm{T}})$.
return $\{\hat{\mathbf{Z}}_+, \hat{\Pi}\}$.

STEP-2-PCA-DENOISE($\hat{\mathbf{Z}}_+, \mathbf{Y}$)

$\hat{N}_+^{\mathrm{T}} \leftarrow \frac{1}{n}\hat{\mathbf{Z}}_+^{\mathrm{T}}\mathbf{Y}$.
▷ **Absolute value thresholding.**
▷ θ is a suitable constant; σ_ϵ is std. of the noise.
$k_2 = \max\Big\{k_2 : \sigma_{k_2}(\hat{N}_+) \geq \theta\sigma_\epsilon\sqrt{\frac{d_2}{n}}\Big\}$.
return $\mathbf{P}_{k_2}(\hat{N}_+)$

ADAPTIVE-RRR($\mathbf{X}, \mathbf{Y}$)

$[\hat{Z}_+, \hat{\Pi}] =$ STEP-1-PCA-A($\mathbf{X}$).
$\mathbf{P}_{k_2}(\hat{N}_+) =$ STEP-2-PCA-DENOISE($\hat{Z}_+, \mathbf{Y}$).
return $\hat{M} = \mathbf{P}_{k_2}(\hat{N}_+)\hat{\Pi}$

Figure 2.1: Our algorithm (ADAPTIVE-RRR) for solving the regression $\mathbf{y} = M\mathbf{x} + \epsilon$.

Our model. We consider the model $\mathbf{y} = M\mathbf{x} + \epsilon$, where $\mathbf{x} \in \mathbf{R}^{d_1}$ is a multivariate

Gaussian, $\mathbf{y} \in \mathbf{R}^{d_2}$, $M \in \mathbf{R}^{d_2 \times d_1}$, and $\epsilon \sim N(0, \sigma_\epsilon^2 I_{d_2 \times d_2})$. We can relax the Gaussian assumptions on $\mathbf{x}$ and ϵ for most results we develop. We assume a PAC learning framework, i.e., we observe a sequence $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i \leq n}$ of independent samples and our goal is to find an $\hat{M}$ that minimizes the test error $\mathbb{E}_{\mathbf{x},\mathbf{y}}[\|\hat{M}\mathbf{x} - M\mathbf{x}\|_2^2]$. We are specifically interested in the setting in which $d_2 \approx n \leq d_1$.

The key assumption we make to circumvent the $d_1 \geq n$ issue is that the features are correlated. This assumption can be justified for the following reasons: *(i)* In practice, it is difficult, if not impossible, to construct completely uncorrelated features. *(ii)* When $n \ll d_1$, it is not even possible to *test* whether the features are uncorrelated [11]. *(iii)* When we indeed know that the features are independent, there are significantly simpler methods to design models. For example, we can build multiple models such that each model regresses on an individual feature of $\mathbf{x}$, and then use a boosting/bagging method [53, 136] to consolidate the predictions.

The correlatedness assumption implies that the eigenvalues of C^* *decays*. The only (full rank) positive semidefinite matrices that have non-decaying (uniform) eigenvalues are the identity matrix (up to some scaling). In other words, when C^* has uniform eigenvalues, $\mathbf{x}$ has to be uncorrelated.

We aim to design an algorithm that works *even when* the decay is slow, such as when $\lambda_i(C^*)$ has a heavy tail. Specifically, our algorithm assumes λ_i's are bounded by a heavy-tail power law series:

Assumption 2.2.1. *The $\lambda_i(C^*)$ series satisfies $\lambda_i(C^*) \leq c \cdot i^{-\omega}$ for a constant c and $\omega \geq 2$.*

We *do not* make functional form assumptions on λ_i's. This assumption also covers many benign cases, such as when C^* has low rank or its eigenvalues decay exponentially. Many empirical studies report power law distributions of data covariance matrices [4, 120, 151, 33]. Next, we make standard normalization assumptions. $\mathbb{E}\,\|\mathbf{x}\|_2^2 = 1$, $\|M\|_2 \leq \Upsilon = O(1)$, and $\sigma_\epsilon \geq 1$. Remark that we assume only the spectral norm of M is bounded, while its

Frobenius norm can be unbounded. Also, we assume the noise $\sigma_\epsilon \geq 1$ is sufficiently large, which is more important in practice. The case when σ_ϵ is small can be tackled in a similar fashion. Finally, our studies avoid examining excessively unrealistic cases, so we assume $d_1 \leq d_2^3$. We examine the setting where existing algorithms fail to deliver non-trivial MSE, so we assume that $n \leq rd_1 \leq d_2^4$.

2.3 Upper bound

Our algorithm (see Fig. 2.1) consists of two steps. **Step 1. Producing uncorrelated features.** We run a PCA to obtain a total number of k_1 orthogonalized features. See Step-1-PCA-X in Fig. 2.1. Let the SVD of $\mathbf{X}$ be $\mathbf{X} = U\Sigma(V)^{\mathrm{T}}$. Let k_1 be a suitable rank chosen by inspecting the gaps of $\mathbf{X}$'s singular values (Line 5 in Step-1-PCA-X). $\hat{\mathbf{Z}}_+ = \sqrt{n}U_{k_1}$ is the set of transformed features output by this step. The subscript + in $\hat{\mathbf{Z}}_+$ reflects that a dimension reduction happens so the number of columns in $\hat{\mathbf{Z}}_+$ is smaller than that in $\mathbf{X}$. Compared to standard PCA dimension reduction, there are two differences: *(i)* We use the left leading singular vectors of $\mathbf{X}$ (with a re-scaling factor $\sqrt{n}$) as the output, whereas the PCA reduction outputs $\mathbf{P}_{k_1}(\mathbf{X})$. *(ii)* We design a specialized rule to choose k_1 whereas PCA usually uses a hard thresholding or other ad-hoc rules. **Step 2. Matrix denoising.** We run a second PCA on the matrix $(\hat{N}_+)^{\mathrm{T}} \triangleq \frac{1}{n}\hat{\mathbf{Z}}_+^{\mathrm{T}}\mathbf{Y}$. The rank k_2 is chosen by a hard thresholding rule (Line 4 in Step-2-PCA-Denoise). Our final estimator is $\mathbf{P}_{k_2}(\hat{N}_+)\hat{\Pi}$, where $\hat{\Pi} = (\Lambda_{k_1})^{-\frac{1}{2}}V_{k_1}^{\mathrm{T}}$ is computed in Step-1-PCA-X($\mathbf{X}$).

2.3.1 Intuition of the design

While the algorithm is operationally simple, its design is motivated by carefully unfolding the statistical structure of the problem. We shall realize that applying PCA on the features *should not* be viewed as removing noise from a factor model, or finding subspaces that maximize variations explained by the subspaces as suggested in the standard literature [53, 143, 144]. Instead, it implicitly implements a robust estimator for $\mathbf{x}$'s precision matrix, and the design of the estimator needs to be coupled with our objective of forecasting $\mathbf{y}$, thus resulting in a new way of choosing the rank.

Design motivation: warm up. We first examine a simplified problem $\mathbf{y} = N\mathbf{z} + \epsilon$,

where variables in $\mathbf{z}$ are assumed to be uncorrelated. Assume $d = d_1 = d_2$ in this simplified setting. Observe that

$$\frac{1}{n}\mathbf{Z}^{\mathrm{T}}\mathbf{Y} = \frac{1}{n}\mathbf{Z}^{\mathrm{T}}(\mathbf{Z}N^{\mathrm{T}} + E) = (\frac{1}{n}\mathbf{Z}^{\mathrm{T}}\mathbf{Z})N^{\mathrm{T}} + \frac{1}{n}\mathbf{Z}^{\mathrm{T}}E \approx I_{d_1\times d_1}N^{\mathrm{T}} + \frac{1}{n}\mathbf{Z}^{\mathrm{T}}E = N^{\mathrm{T}} + \mathcal{E}, \tag{2.1}$$

where E is the noise term and $\mathcal{E}$ can be approximated by a matrix with independent zero-mean noises.

Solving the matrix denoising problem. Eq. 2.1 implies that when we compute $\mathbf{Z}^{\mathrm{T}}\mathbf{Y}$, the problem reduces to an extensively studied matrix denoising problem [41, 55]. We include the intuition for solving this problem for completeness. The signal N^{T} is overlaid with a noise matrix $\mathcal{E}$. $\mathcal{E}$ will elevate all the singular values of N^{T} by an order of $\sigma_\epsilon\sqrt{d/n}$. We run a PCA to extract reliable signals: when the singular value of a subspace is $\gg \sigma_\epsilon\sqrt{d/n}$, the subspace contains significantly more signal than noise and thus we keep the subspace. Similarly, a subspace associated a singular value $\lesssim \sigma_\epsilon\sqrt{d/n}$ mostly contains noise. This leads to a hard thresholding algorithm that sets $\hat{N}^{\mathrm{T}} = \mathbf{P}_r(N^{\mathrm{T}} + \mathcal{E})$, where r is the maximum index such that $\sigma_r(N^{\mathrm{T}} + \mathcal{E}) \geq c\sqrt{d/n}$ for some constant c. In the general setting $\mathbf{y} = M\mathbf{x} + \epsilon$, $\mathbf{x}$ may not be uncorrelated. But when we set $\mathbf{z} = (\Lambda^*)^{-\frac{1}{2}}(V^*)^{\mathrm{T}}\mathbf{x}$, we see that $\mathbb{E}[\mathbf{z}\mathbf{z}^{\mathrm{T}}] = I$. This means knowing C^* suffices to reduce the original problem to a simplified one. Therefore, our algorithm uses Step 1 to estimate C^* and $\mathbf{Z}$, and uses Step 2 to reduce the problem to a matrix denoising one and solve it by standard thresholding techniques.

Relationship between PCA and precision matrix estimation. In step 1, while we plan to estimate C^*, our algorithm runs a PCA on $\mathbf{X}$. We observe that empirical covariance matrix $C = \frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X} = \frac{1}{n}V(\Sigma)^2(V)^{\mathrm{T}}$, i.e., C's eigenvectors coincide with $\mathbf{X}$'s right singular vectors. When we use the empirical estimator to construct $\hat{\mathbf{z}}$, we obtain $\hat{\mathbf{z}} = \sqrt{n}(\Sigma)^{-1}(V)^{\mathrm{T}}\mathbf{x}$. When we apply this map to every training point and assemble the new feature matrix, we exactly get $\hat{\mathbf{Z}} = \sqrt{n}\mathbf{X}V(\Sigma)^{-1} = \sqrt{n}U$. It means that using C to construct $\hat{\mathbf{z}}$ is the same as running a PCA in Step-1-PCA-X with $k_1 = d_1$.

When $k_1 < d_1$, PCA uses a low rank approximation of C as an estimator for C^*. We

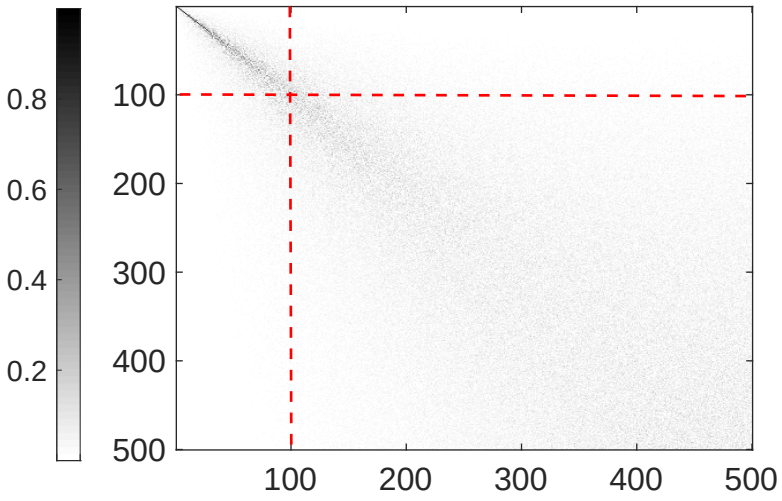


Figure 2.2: The angle matrix between C and C^*.

now explain why this is effective. First, note that C is *very far* from C^* when $n \ll d_1$, therefore it is dangerous to directly plug in C to find $\hat{\mathbf{z}}$. Second, an interesting regularity of C exists and can be best explained by a picture. In Fig. 2.2, we plot the pairwise angles between eigenvectors of C and those of C^* from a synthetic dataset. Columns are sorted by the C^*'s eigenvalues in decreasing order. When C^* and C coincide, this plot would look like an identity matrix. When C and C^* are unrelated, then the plot behaves like a block of white Gaussian noise. We observe a pronounced pattern: the angle matrix can be roughly divided into two sub-blocks (see the red lines in Fig. 2.2). The upper left sub-block behaves like an identity matrix, suggesting that the leading eigenvectors of C are close to those of C^*. The lower right block behaves like a white noise matrix, suggesting that the "small" eigenvectors of C are far from those of C^*. When n grows, one can observe the upper left block becomes larger and this the eigenvectors of C will sequentially get stabilized. Leading eigenvectors are first stabilized, followed by smaller ones. Our algorithm leverages this regularity by keeping only a suitable number of reliable eigenvectors from C while ensuring not much information is lost when we throw away those "small" eigenvectors.

Implementing the rank selection. We rely on three interacting building blocks:

1. Dimension-free matrix concentration. First, we need to find a concentration behavior of C for $n \leq d_1$ to decouple d_1 from the MSE bound. We utilize a dimension-free matrix

concentration inequality [121](also replicated as Lemma 2.3.5) . Roughly speaking, the concentration behaves as $\|C - C^*\|_2 \approx n^{-\frac{1}{2}}$. This guarantees that $|\lambda_i(C) - \lambda_i(C^*)| \leq n^{-\frac{1}{2}}$ by standard matrix perturbation results [79].

2. Davis-Kahan perturbation result. However, the pairwise closeness of the λ_i's does not imply the eigenvectors are also close. When $\lambda_i(C^*)$ and $\lambda_{i+1}(C^*)$ are close, the corresponding eigenvectors in C can be "jammed" together. Thus, we need to identify an index i, at which $\lambda_i(C^*) - \lambda_{i+1}(C^*)$ exhibits significant gap, and use a Davis-Kahan result to show that $\mathbf{P}_i(C)$ is close to $\mathbf{P}_i(C^*)$. On the other hand, the map $\Pi^*(\triangleq (\Lambda^*)^{-\frac{1}{2}}(V^*)^{\mathrm{T}})$ we aim to find depends on the square root of inverse $(\Lambda^*)^{-\frac{1}{2}}$, so we need additional manipulation to argue our estimate is close to $(\Lambda^*)^{-\frac{1}{2}}(V^*)^{\mathrm{T}}$.

3. The connection between gap and tail. Finally, the performance of our procedure is also characterized by the total volume of signals that are discarded, i.e., $\sum_{i>k_1} \lambda_i(C^*)$, where k_1 is the location that exhibits the gap. The question becomes whether it is possible to identify a k_1 that simultaneously exhibits a large gap and ensures the tail after it is well-controlled, e.g., the sum of the tail is $O(n^{-c})$ for a constant c. We develop a combinatorial analysis to show that it is *always possible* to find such a gap under the assumption that $\lambda_i(C^*)$ is bounded by a power law distribution with exponent $\omega \geq 2$. Combining all these three building blocks, we have:

Proposition 2.3.1. *Let ε and δ be two tunable parameters such that $\varepsilon = \omega(\log^3 n/\sqrt{n})$ and $\delta^3 = \omega(\varepsilon)$. Assume that $\lambda_i^* \leq c \cdot i^{-\omega}$. Consider running* Step-1-PCA-X *in Fig. 2.1, with high probability, we have (i) Leading eigenvectors/values are close: there exists a unitary matrix W and a constant c_1 such that $\|V_{k_1}(\Lambda_{k_1})^{-\frac{1}{2}} - V_{k_1}^*(\Lambda_{k_1}^*)^{-\frac{1}{2}}W\| \leq \frac{c_1\varepsilon}{\delta^3}$. (ii) Small tail: $\sum_{i\geq k_1} \lambda_i^* \leq c_2\delta^{\frac{\omega-1}{\omega+1}}$ for a constant c_2.*

Prop. 2.3.1 implies that our estimate $\hat{\mathbf{z}}_+ = \hat{\Pi}(\mathbf{x})$ is sufficiently close to $\mathbf{z} = \Pi^*(\mathbf{x})$, up to a unitary transform. We then execute Step-2-PCA-Denoise to reduce the problem to a matrix denoising one and solve it by hard-thresholding. Let us refer to $\mathbf{y} = N\mathbf{z} + \epsilon$, where $\mathbf{z}$ is a standard multivariate Gaussian and $N = MV^*(\Lambda^*)^{\frac{1}{2}}$ as the *orthogonalized form* of the problem. While we do not directly observe $\mathbf{z}$, our performance is characterized

by spectra structure of N.

Theorem 2.3.2. *Consider running* ADAPTIVE-RRR *in Fig. 2.1 on n independent samples* $(\mathbf{x}, \mathbf{y})$ *from the model* $\mathbf{y} = M\mathbf{x} + \epsilon$, *where* $\mathbf{x} \in \mathbf{R}^{d_1}$ *and* $\mathbf{y} \in \mathbf{R}^{d_2}$. *Let* $C^* = \mathbb{E}[\mathbf{x}\mathbf{x}^{\mathrm{T}}]$. *Assume that* (i) $\|M\|_2 \leq \Upsilon = O(1)$, *and* (ii) $\mathbf{x}$ *is a multivariate Gaussian with* $\|\mathbf{x}\|_2 = 1$. *In addition,* $\lambda_1(C^*) < 1$ *and for all* i, $\lambda_i(C^*) \leq c/i^\omega$ *for a constant* c, *and* (iii) $\epsilon \sim N(0, \sigma_\epsilon^2 I_{d_1})$, *where* $\sigma_\epsilon \geq \min\{\Upsilon, 1\}$.

Let $\varepsilon = \omega(\log^3 n/\sqrt{n})$, $\delta^3 = \omega(\varepsilon)$, *and* θ *be a suitably large constant. Let* $\mathbf{y} = N\mathbf{z} + \epsilon$ *be the orthogonalized form of the problem. Let* ℓ^* *be the largest index such that* $\sigma_{\ell^*}^N > \theta\sigma_\epsilon\sqrt{\frac{d_2}{n}}$. *Let* $\hat{\mathbf{y}}$ *be our testing forecast. With high probability over the training data:*

$$\mathbb{E}[\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2] \leq \sum_{i \geq \ell^*} (\sigma_i^N)^2 + O\left(\frac{\ell^* d_2 \theta^2 \sigma_\epsilon^2}{n}\right) + O\left(\sqrt{\frac{\varepsilon}{\delta^3}}\right) + O\left(\delta^{\frac{\omega-1}{4(\omega+1)}}\right) \tag{2.2}$$

The expectation is over the randomness of the test data.

Theorem 2.3.2 also implies that there exists a way to parametrize ε and δ such that $\mathbb{E}[\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2] \leq \sum_{i>\ell^*}(\sigma_i^N)^2 + O\left(\frac{\ell^* d_2 \theta^2 \sigma_\epsilon^2}{n}\right) + O(n^{-c_0})$ for some constant c_0. We next interpret each term in (2.2).

Terms $\sum_{i>\ell^*}(\sigma_i^N)^2 + O\left(\frac{\ell^* d_2 \theta^2 \sigma_\epsilon^2}{n}\right)$ are typical for solving a matrix denoising problem $\hat{N}_+^{\mathrm{T}} + \mathcal{E} (\approx N^{\mathrm{T}} + \mathcal{E})$: we can extract signals associated with ℓ^* leading singular vectors of N, so $\sum_{i>\ell^*}(\sigma_i^N)^2$ starts at $i > \ell^*$. For each direction we extract, we need to pay a noise term of order $\theta^2\sigma_\epsilon^2\frac{d_2}{n}$, leading to the term $O\left(\frac{\ell^* d_2 \theta^2 \sigma_\epsilon^2}{n}\right)$. *Terms* $O\left(\sqrt{\frac{\varepsilon}{\delta^3}}\right) + O\left(\delta^{\frac{\omega-1}{4(\omega+1)}}\right)$ come from the estimations error of $\hat{\mathbf{z}}_+$ produced from Prop. 2.3.1, consisting of both estimation errors of C^*'s leading eigenvectors and the error of cutting out a tail. We pay an exponent of $\frac{1}{4}$ on both terms (e.g., $\delta^{\frac{\omega-1}{\omega+1}}$ in Prop. 2.3.1 becomes $\delta^{\frac{\omega-1}{4(\omega+1)}}$) because we used Cauchy-Schwarz (CS) twice. One is used in running matrix denoising algorithm with inaccurate $\mathbf{z}_+$; the other one is used to bound the impact of cutting a tail. It remains open whether two CS is can be circumvented.

Sec. 2.4 explains how Thm 2.3.2 and the lower bound imply the algorithm is near-optimal. Sec. 2.5 compares our result with existing ones under other parametrizations, e.g.

Rank(M).

2.3.2 Analysis

We now analyze our algorithm. Our analysis consists of three steps. In step 1, we prove Proposition 2.3.1. In step 2, we relate $\mathbf{Z}^{\mathrm{T}}\mathbf{Y}$ with the product produced by our estimate $\hat{\mathbf{Z}}_{+}^{\mathrm{T}}\mathbf{Y}$. In step 3, we prove Theorem 2.3.2.

2.3.2.1 Step 1. PCA for the features (proof of Proposition 2.3.1)

This section proves Proposition 2.3.1. We shall first show that the algorithm always terminates. We have the following lemma.

Lemma 2.3.3. *Let $\{\lambda_i\}_{i\le d}$ be a sequence such that $\sum_{i\le n}\lambda_i = 1$, $\lambda_i \le ci^{-\omega}$ for some constant c, $\omega \ge 2$, and $\lambda_1 < 1$. Define $\delta_i = \lambda_i - \lambda_{i+1}$ for $i \ge 1$. Let ℓ_0 be a sufficiently large number, and c_1 and c_2 are two suitable constants. Let ℓ be any number such that $\ell \ge \ell_0$. Let τ be any parameter such that $\tau < \rho - 1$. There exists an i^* such that* (i) Gap is sufficiently large: $\delta_{i^*} \ge c_1 \cdot \ell^{-(\tau\omega/(\omega-1)+1)}$, *and (ii)* tail sum is small: $\sum_{i\ge i^*}\lambda_i \le c_2/\ell^{-\tau}$.

We remark that Lemma 2.3.3 will also be able to show part (ii) of Proposition 2.3.1 (this will be explained below). This lemma also corresponds to the "connection between gap and tail" building block referred in Section 2.3.1.

Proof of Lemma 2.3.3. Define the function $h(t) = \sum_{i\ge t} c/i^{\omega} = \frac{c_3+o(1)}{t^{\omega-1}}$ (by EulerMaclaurin formula), where $o(1)$ is a function of t

Next, let us define

$$i_1 = \min\left\{ i^* : \sum_{i\le i^*}\lambda_i \ge 1 - h(\ell^{\frac{\tau}{\omega-1}}) \right\} - 1$$
$$i_2 = \min\left\{ i^* : \sum_{i\le i^*}\lambda_i \ge 1 - \frac{1}{2}\times h(\ell^{\frac{\tau}{\omega-1}}) \right\} - 1.$$

Roughly speaking, we want to identify an i_1 such that $\sum_{i\le i_1}\lambda_i$ is smaller than $1 - h\left(\ell^{\frac{\tau}{\omega-1}}\right)$ but is as close to it as possible. We can interpret i_2 in a similar manner. $i_1, i_2 \ge 1$

because of the assumption $\lambda_1 < 1$.

We can verify that $i_1 < \ell^{\frac{\tau}{\omega-1}}$ because $\sum_{i \le \ell^{\frac{\tau}{\omega-1}}} \lambda_i \ge 1 - h\left(\ell^{\frac{\tau}{\omega-1}}\right)$. We can similarly verify that $i_2 < c_4 \ell^{\frac{\tau}{\omega-1}}$ for some constant c_4. Now using

$$\sum_{i \le i_2+1} \lambda_i \ge 1 - \frac{(c_3 + o(1))\ell^{-\tau}}{2}$$
$$\sum_{i \le i_1} \lambda_i \le 1 - (c_3 + o(1))\ell^{-\tau}.$$

We may use an averaging argement and show that there exists an $i_3 \in [i_1 + 1, i_2 + 1]$ such that

$$\lambda_{i_3} \ge \frac{(c_3 + o(1))\ell^{-\tau}}{i_2 - i_1} \ge \frac{(c_3 + o(1))\ell^{-\tau}}{c_4 \ell^{\frac{\tau}{\omega-1}}}$$
$$\ge c_5 \ell^{-\tau - \frac{\tau}{\omega-1}} = c_5 \ell^{-\frac{\tau\omega}{\omega-1}}.$$

Note that $c_5 \ell^{-\frac{\tau\omega}{\omega-1}} \ge 2c\ell^{-\omega}$ because $\tau < \omega - 1$. Next, using that $\lambda_\ell \le c/\ell^\omega$, we have

$$\lambda_\ell = \overbrace{\lambda_{i_3}}^{\ge c_5 \ell^{-\frac{\tau\omega}{\omega-1}}} + (\lambda_{i_3+1} - \lambda_{i_3}) + \cdots + (\lambda_\ell - \lambda_{\ell-1}) \le \overbrace{c/\ell^\omega}^{\le \frac{c_5}{2} \cdot \ell^{-\frac{\tau\omega}{\omega-1}}}. \tag{2.3}$$

This implies one of $(\lambda_{i_3} - \lambda_{i_3+1}), \ldots, (\lambda_{\ell-1} - \lambda_\ell)$ is at least $\frac{c_5}{2} \cdot \ell^{-\frac{\tau\omega}{\omega-1}}/\ell$. In other words, there exists an $i^* \in [i_3 + 1, \ell]$ such that $\lambda_{i^*} - \lambda_{i^*+1} \ge \frac{c_5}{2}\ell^{-\left(\frac{\tau\omega}{\omega-1}+1\right)}$. Finally, we can check that

$$\sum_{i \ge i^*} \lambda_{i^*} \le \sum_{i \ge i_1} \lambda_{i^*} \le h\left(\ell^{\frac{\tau}{\omega-1}}\right) \le \frac{c_2}{\ell^\tau}. \tag{2.4}$$

□

We apply Lemma 2.3.3 by setting $\tau \to \omega - 1$. There is a parameter ℓ that we can tune, such that it is always possible to find an i^* where $\delta_{i^*} \ge c_1 \ell^{-(\omega+1)}$ and $\sum_{i \ge i^*} \lambda_i \le 1 - c_2 \ell^{-(\omega-1)}$. For any $\delta = o(1)$ (a function of n), we can set $\ell = \Theta\left(\left(\frac{1}{\delta}\right)^{\frac{1}{\omega+1}}\right)$. In addition, $\sum_{i \ge k_1} \lambda_i = O\left(\delta^{\frac{\omega-1}{\omega+1}}\right)$. This also proves the second part of the Proposition.

It remains to prove part (i) of Proposition 2.3.1. It consists of three steps.

Step 1. Dimension-free Chernoff bound for matrices. We first give a bound on $\|C^* - C\|_2$, which characterizes the tail probability by using the first and second moments of random vectors. This is the key device enabling us to meaningfully recover signals even when $n \ll d_1$.

Lemma 2.3.4. *Recall that* $C^* = \mathbb{E}[\mathbf{x}\mathbf{x}^{\mathrm{T}}]$ *and* $C = \frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X}$. *For any* $\varepsilon > 0$,

$$\Pr[\|C^* - C\|_2 \geq \varepsilon] \leq (2n^2)\exp(-n\varepsilon^2/(\log^4 n)) + n^{-10}. \tag{2.5}$$

The exponent 10 is chosen arbitrarily and is not optimized.

Proof of Lemma 2.3.4. We use the following specific form of Chernoff bound ([121])

Lemma 2.3.5. *Let* $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n$ *be i.i.d. random vectors such that* $\|\mathbf{z}_i\| \leq \alpha$ *a.s. and* $\|\,\mathbb{E}[\mathbf{z}_i\mathbf{z}_i^{\mathrm{T}}]\| \leq \beta$. *Then for any* $\epsilon > 0$,

$$\Pr\left[\left\|\frac{1}{n}\sum_{i\leq n}\mathbf{z}_i\mathbf{z}_i^{\mathrm{T}} - \mathbb{E}[\mathbf{z}_i\mathbf{z}_i^{\mathrm{T}}]\right\|_2 \geq \varepsilon\right] \leq (2n^2)\exp\left(-\frac{n\varepsilon^2}{16\beta\alpha^2 + 8\alpha^2\varepsilon}\right) \tag{2.6}$$

We aim to use Lemma 2.3.5 to show Lemma 2.3.4 and we set $\mathbf{z}_i = \mathbf{x}_i$. But the ℓ_2-norm of $\mathbf{z}_i$'s are unbounded so we need to use a simple coupling technique to circumvent the problem. Specifically, let c_0 be a suitable constant and define

$$\tilde{\mathbf{z}}_i = \begin{cases} \mathbf{z}_i & \text{if } |\mathbf{z}_i| \leq c_0 \log^2 n \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

By using a standard Chernoff bound, we have

$$\Pr[\exists i : \tilde{\mathbf{z}}_i \neq \mathbf{z}_i] \leq \frac{1}{n^{10}}. \tag{2.8}$$

Let us write $\tilde{C} = \frac{1}{n}\sum_{i\leq n}\tilde{\mathbf{z}}_i\tilde{\mathbf{z}}_i^{\mathrm{T}}$. We set $\alpha = c_0 \log^2 n$ and $\beta = \Theta(1)$ in Lemma 2.3.5. One

can see that

$$\Pr[\|C^* - C\|_2 \geq \varepsilon] \leq \Pr\left[(\|\tilde{C} - C\|_2 \geq \varepsilon) \vee (\tilde{C} \neq C)\right] \leq 2n^2 \exp\left(-\frac{n\varepsilon^2}{\log^4 n}\right) + \frac{1}{n^{10}}. \quad (2.9)$$

□

Step 2. Davis-Kahan bound. The above analysis gives us that $\|C^* - C\|_2 \leq \varepsilon$. We next show that the first a few eigenvectors of C are close to those of C^*.

Lemma 2.3.6. *Let $\varepsilon = \omega\left(\frac{\log^3 n}{\sqrt{n}}\right)$ and $\delta^3 = \omega(\varepsilon)$. Considering running* Step-1-PCA-X *in Fig. 2.1. Let $\mathcal{P}^* = V_{k_1}^*(V_{k_1}^*)^{\mathrm{T}}$ and $\mathcal{P} = V_{k_1}V_{k_1}^{\mathrm{T}}$. When $\|C^* - C\|_2 \leq \varepsilon$, $\|\mathcal{P}^* - \mathcal{P}\|_2 \leq \frac{2\varepsilon}{\delta}$.*

Proof. Recall that $\lambda_1^*, \lambda_2^*, \ldots, \lambda_{d_1}^*$ are the eigenvalues of C^*. Let also $\lambda_1, \lambda_2, \ldots, \lambda_{d_1}$ be the eigenvalues of C. Define

$$S_1 = [\lambda_{k_1} - \delta/10, \infty] \quad \text{and} \quad S_2 = [0, \lambda_{k_1+1} + \delta/10]. \quad (2.10)$$

The constant 10 is chosen in an arbitrary manner. Because $\|C^* - C\|_2 \leq \varepsilon$, we know that S_1 contains $\lambda_1^*, \ldots, \lambda_k^*$ and that S_2 contains $\lambda_{k_1+1}^*, \ldots, \lambda_{d_1}^*$ [79]. Using the Davis-Kahan Theorem [37], we get

$$\|\mathcal{P}^* - \mathcal{P}\|_2 \leq \frac{\|C^* - C\|_2}{0.8\delta} \leq \frac{2\epsilon}{\delta} \quad (2.11)$$

□

We also need the following building block.

Lemma 2.3.7. *[147] Let A and B be $n \times n$ positive semidefinite matrices with the same rank of d. Let X and Y be of full column rank such that $XX^{\mathrm{T}} = A$ and $YY^{\mathrm{T}} = B$. Let δ be the smallest non-zero eigenvalue of B. Then there exists a unitary matrix $W \in \mathbf{R}^{d\times d}$ such that*

$$\|XW - Y\|_2 \leq \frac{\|A - B\|_2(\sqrt{\|A\|_2} + \sqrt{\|B\|_2})}{\delta}.$$

Step 3. Commuting the unitary matrix. Roughly speaking, Lemma 2.3.6 and Lemma 2.3.7 show that there exists a unitary matrix W such that $\|V_{k_1}W - V_{k_1}^*\|_2$ is close to 0. Standard matrix perturbation result also shows that Λ_{k_1} and $\Lambda_{k_1}^*$ are close. This gives us that $V_{k_1}W\Lambda_{k_1}^{-\frac{1}{2}}$ and $V_{k_1}^*(\Lambda_{k_1}^*)^{-\frac{1}{2}}$ are close, whereas we need that $V_{k_1}\Lambda_{k_1}^{-\frac{1}{2}}W$ and $V_{k_1}^*(\Lambda_{k_1}^*)^{-\frac{1}{2}}$ are close. The unitary matrix W is not in the right place. This is a standard technical obstacle for analyzing PCA based techniques [147, 48, 90]. We develop the following lemma to address the issue.

Lemma 2.3.8. *Let U_1, U_2 be $n \times d$ matrices such that $U_1^\top U_1 = U_2^\top U_2 = I$. Let S_1, S_2 be diagonal matrices with strictly positive entries, and let $W \in \mathbf{R}^{d\times d}$ be a unitary matrix. Then,*

$$\|U_1S_1^{-1}W - U_2S_2^{-1}\| \leq \frac{\|U_1S_1W - U_2S_2\|}{\min\{(S_1)_{ii}\}\cdot\min\{(S_2)_{ii}\}} + \frac{\|U_1U_1^\top - U_2U_2^\top\|}{\min\{(S_2)_{ii}\}}$$

Proof. Observe that,

$$U_1S_1^{-1}W - U_2S_2^{-1} = U_1S_1^{-1}W(S_2U_2^\top - W^\top S_1U_1^\top)U_2S_2^{-1} + U_1U_1^\top U_2S_2^{-1} - U_2U_2^\top U_2S_2^{-1}.$$

The result then follows by taking spectral norms of both sides, the triangle inequality and the sub-multiplicativity of the spectral norm. □

Results from Step 1 to Step 3 suffice to prove the first part of Proposition 2.3.1. First, we use Lemma 2.3.6 and Lemma 2.3.7 (adopted from [147]) to get that

$$\|V_{k_1}^*(\Lambda_{k_1}^*)^{\frac{1}{2}}W - V_{k_1}(\Lambda_{k_1})^{\frac{1}{2}}\|_2 \leq \frac{c_0\varepsilon}{\delta^2}. \tag{2.12}$$

Next, observe that $\lambda_{k_1}, \lambda_{k_1}^* = \Omega(\delta)$. By applying Lemma 2.3.8, with $U_1 = V_{k_1}^*$ and $S_1 = (\Lambda_{k_1}^*)^{\frac{1}{2}}$, $U_2 = V_{k_1}$ and $S_2 = (\Lambda_{k_1})^{\frac{1}{2}}$, we obtain

$$\|V_{k_1}\Lambda_{k_1}^{-\frac{1}{2}}W - V_{k_1}^*\Lambda_{k_1}^{*-\frac{1}{2}}\|_2 \leq \frac{\|V_{k_1}^*\Lambda_{k_1}^{*\frac{1}{2}}W - V_{k_1}\Lambda_{k_1}^{\frac{1}{2}}\|_2}{\delta} + \frac{\|\mathcal{P}^* - \mathcal{P}\|_2}{\delta} \leq \frac{c_1\varepsilon}{\delta^3}$$

This completes the proof of Proposition 2.3.1.

2.3.2.2 Step 2. Analysis of $\mathbf{Z}^{\mathrm{T}}\mathbf{Y}$

Proposition 2.3.9. *Consider running* Adaptive-RRR *in Fig. 2.1 to solve the regression problem* $\mathbf{y} = M\mathbf{x} + \epsilon$. *Let* $\hat{\mathbf{Z}}_+$ *be the output of the first stage* Step-1-PCA-X. *Let* W *be the unitary matrix specified in Proposition 2.3.1. Let* $\hat{N}_+ = \hat{\mathbf{Z}}_+^{\mathrm{T}}\mathbf{Y}$. *We have with high probability (over the training data),*

$$\hat{N}_+^{\mathrm{T}} = W^{\mathrm{T}}N^{\mathrm{T}} + \mathcal{E}_L + \mathcal{E}_T,$$

where

$$\|\mathcal{E}_L\|_2 \leq 2.2\sigma_\epsilon\sqrt{\frac{d_2}{n}} \quad \textit{and} \quad \|\mathcal{E}_T\|_F = O(\epsilon/\delta^3).$$

The rest of this Section proves Proposition 2.3.9. Recall that $\mathbf{Y} = \mathbf{Z}N^{\mathrm{T}} + E$ is the orthogonalized form of our problem. Let us split $N = [N_+, N_-]$, where $N_+ \in \mathbf{R}^{d_2\times k_1}$ consists of the k_1 leading columns of N and $N_- \in \mathbf{R}^{d_2\times(d_1-k_1)}$ consists of the remaining columns. Similarly, let $\mathbf{z} = [\mathbf{z}_+, \mathbf{z}_-]$, where $\mathbf{z}_+ \in \mathbf{R}^{n\times k_1}$ and $\mathbf{z}_- \in \mathbf{R}^{n\times(d_1-k_1)}$. Let $\mathbf{Z} = [\mathbf{Z}_+, \mathbf{Z}_-]$, where $\mathbf{Z}_+ \in \mathbf{R}^{n\times k_1}$ and $\mathbf{Z}_- \in \mathbf{R}^{n\times(d_1-k_1)}$. Finally, when we refer to estimated features of an individual instance produced from Step 1, we use $\hat{\mathbf{z}}_+$.

We have $\mathbf{Y} = \mathbf{Z}_+N_+^{\mathrm{T}} + \mathbf{Z}_-N_-^{\mathrm{T}} + E$. We let

$$\delta_+ = \hat{\mathbf{z}}_+ - W^{\mathrm{T}}\mathbf{z}_+$$

$$\Delta_+ = \hat{\mathbf{Z}}_+ - \mathbf{Z}_+W,$$

where $\|\delta_+\|_2 = O(\epsilon/\delta^3)$ and $\|\Delta_+\|_2 = O(\sqrt{n}\epsilon/\delta^3)$. We have

$$\begin{aligned}\frac{1}{n}\hat{\mathbf{Z}}_+^{\mathrm{T}}\mathbf{Y} &= \frac{1}{n}(\Delta_+ + \mathbf{Z}_+W)^{\mathrm{T}}(\mathbf{Z}_+N_+^{\mathrm{T}} + \mathbf{Z}_-N_-^{\mathrm{T}} + E) \\ &= W^{\mathrm{T}}N_+^{\mathrm{T}} + W^{\mathrm{T}}\left(\frac{1}{n}\mathbf{Z}_+^{\mathrm{T}}\mathbf{Z}_+ - I_{k_1\times k_1}\right)N_+^{\mathrm{T}} + \frac{1}{n}W^{\mathrm{T}}\mathbf{Z}_+^{\mathrm{T}}\mathbf{Z}_-N_-^{\mathrm{T}} + \frac{1}{n}W^{\mathrm{T}}\mathbf{Z}_+^{\mathrm{T}}E \\ &\quad + \frac{1}{n}\Delta_+^{\mathrm{T}}(\mathbf{Z}_+N_+^{\mathrm{T}} + \mathbf{Z}_-N_-^{\mathrm{T}} + E).\end{aligned}$$

We shall let

$$\hat{N}_+^{\mathrm{T}} = W^{\mathrm{T}}N^{\mathrm{T}} + \mathcal{E}, \tag{2.13}$$

where

$$\begin{aligned}\mathcal{E} &= \mathcal{E}_1 + \mathcal{E}_2 + \mathcal{E}_3 + \mathcal{E}_4 + \mathcal{E}_5 \\ \mathcal{E}_1 &= W^{\mathrm{T}}\left(\frac{1}{n}\mathbf{Z}_+^{\mathrm{T}}\mathbf{Z}_+ - I_{k_1\times k_1}\right)N_+^{\mathrm{T}} \\ \mathcal{E}_2 &= \frac{1}{n}W^{\mathrm{T}}\mathbf{Z}_+^{\mathrm{T}}\mathbf{Z}_-N^{\mathrm{T}} \\ \mathcal{E}_3 &= \frac{1}{n}W^{\mathrm{T}}\mathbf{Z}_+^{\mathrm{T}}E \\ \mathcal{E}_4 &= \frac{1}{n}\Delta_+^{\mathrm{T}}E \\ \mathcal{E}_5 &= \frac{1}{n}\Delta_+^{\mathrm{T}}(\mathbf{Z}_+N_+^{\mathrm{T}} + \mathbf{Z}_-N_-^{\mathrm{T}}).\end{aligned}$$

We next analyze each term. We aim to find bounds in either spectral norm or Frobenius norm. In some cases, it suffices to use $\|\mathcal{E}_i\|_2 \cdot \mathrm{Rank}(\mathcal{E}_i)$ to upper bound $\|\mathcal{E}_i\|_F$. So we bound only $\mathcal{E}_i$'s spectral norm. On the other hand, in the case of analyzing $\mathcal{E}_5$, we can get a tight Frobenius norm bound but we cannot get a non-trivial spectral bound.

From time to time, we will label the dimension of matrices in complex multiplication operations to enable readers to do sanity checks.

Bounding $\mathcal{E}_1$. We use the following Lemmas.

Lemma 2.3.10. *Let $\mathbf{Z} \in \mathbf{R}^{n\times k_1}$, where $k_1 < n$. Let each entry of $\mathbf{Z}$ be an independent*

standard Gaussian. We have

$$\left\| \frac{1}{n} \mathbf{Z}^{\mathrm{T}} \mathbf{Z} - I \right\| \leq \max \left\{ \frac{10 \log^2 n}{\sqrt{n}}, 4 \sqrt{\frac{k_1}{n}} \right\} \tag{2.14}$$

Proof of Lemma 2.3.10. We rely on the Lemma [134]:

Lemma 2.3.11. *Let $S \in \mathbf{R}^{n \times k}$ $(n > k)$ be a random matrix so that each $S_{i,j}$ is an independent standard Gaussian random variable. Let $\sigma_{\max}(S)$ be the maximum singular value of S and $\sigma_{\min}(S)$ be the minimum singular value of it. We have*

$$\Pr[\sqrt{n} - \sqrt{k} - t \leq \sigma_{\min}(S) \leq \sigma_{\max}(S) \leq \sqrt{n} + \sqrt{k} + t] \geq 1 - 2 \times \exp(-t^2/2). \tag{2.15}$$

We set $t = \max \left\{ \frac{\sqrt{k_1}}{10}, \log^2 n \right\}$. Let us start with considering the case $\frac{\sqrt{k_1}}{10} > \log^2 n$. We have

$$\sigma_{\min}(\mathbf{Z}^{\mathrm{T}} \mathbf{Z}) \geq n - 2.2 \sqrt{n k_1} + 1.21 k_1 \geq n - 2.2 \sqrt{n k_1}. \tag{2.16}$$

and

$$\sigma_{\max}(\mathbf{Z}^{\mathrm{T}} \mathbf{Z}) \leq n + 2.2 \sqrt{n k_1} + 1.21 k_1 \leq n + 4 \sqrt{n k_1}. \tag{2.17}$$

The case $\frac{\sqrt{k_1}}{10} \leq \log^2 n$ can be analyzed in a similar fashion so that we can get

$$\left\| \frac{1}{n} \mathbf{Z}^{\mathrm{T}} \mathbf{Z} - I \right\| \leq \max \left\{ \frac{10 \log^2 n}{\sqrt{n}}, 4 \sqrt{\frac{k_1}{n}} \right\}. \tag{2.18}$$

□

Therefore, we have

$$\|\mathcal{E}_1\|_2 \leq \max \left\{ \frac{10 \log^2 n}{\sqrt{n}}, 4 \sqrt{\frac{k_1}{n}} \right\} \|N_+^{\mathrm{T}}\|_2 = \Upsilon \max \left\{ \frac{10 \log^2 n}{\sqrt{n}}, 4 \sqrt{\frac{k_1}{n}} \right\}.$$

Bounding $\mathcal{E}_2$. Observe that $\mathbb{E}[\|\mathbf{Z}_- N_-^{\mathrm{T}}\|_F^2] = n\|N_-^{\mathrm{T}}\|_F^2$. Also,

$$\mathbb{E}[\| \underbrace{\mathbf{Z}_+^{\mathrm{T}}}_{k_1\times n} \underbrace{\mathbf{Z}_-}_{n\times(d_1-k_1)} \underbrace{N_-^{\mathrm{T}}}_{(d_1-k_1)\times d_2} \|_F^2 \mid \mathbf{Z}_- N_-^{\mathrm{T}}] = k_1\|\mathbf{Z}_- N_-^{\mathrm{T}}\|_F^2.$$

Therefore,

$$\mathbb{E}[\mathbf{Z}_+^{\mathrm{T}}\mathbf{Z}_- N_-^{\mathrm{T}}] = k_1 n\|N_-^{\mathrm{T}}\|_F. \tag{2.19}$$

We next bound $\|N_-\|_F$.

Lemma 2.3.12. *Let N be the learnable parameter in normalized form $N = [N_+, N_-]$, where $N_+ \in \mathbf{R}^{d_2\times k_1}$ and $N_- \in \mathbf{R}^{d_2\times(d_1-k_1)}$, and k_1 is determined by* Step-1-PCA-X. *We have $\|N_-\|_F = O\left(\delta^{\frac{\omega-1}{\omega+1}}\right) = o(1)$.*

Proof of Lemma 2.3.12. Recall that

$$N = \underbrace{M}_{d_2\times d_1} \underbrace{V^*}_{d_1\times d_1} \underbrace{(\Lambda^*)^{\frac{1}{2}}}_{d_1\times d_1}.$$

We let $\Lambda^* = [\Lambda_+^*, \Lambda_-^*]$, where $\Lambda_+^* \in \mathbf{R}^{d_1\times k_1}$ and $\Lambda_-^* \in \mathbf{R}^{d_1\times(d_1-k_1)}$. We have $N_- = MV^*(\Lambda_-^*)^{\frac{1}{2}}$. Therefore,

$$\|N_-\|_F^2 \le \|M\|_2^2\|V^*\|_2^2 \left\|(\Lambda^*)^{\frac{1}{2}}\right\|_F^2 = O\left(\Upsilon\delta^{\frac{\omega-1}{\omega+1}}\right) = o(1). \tag{2.20}$$

Here, we used the assumption $\|M\|_2 = O(1)$ and the last equation holds because of Proposition 2.3.1. □

By (2.19), (2.20), and a standard Chernoff bound, we have whp

$$\|\mathcal{E}_2\|_2 \le \|\mathcal{E}_2\|_F \le 2\sqrt{\frac{k_1}{n}}\|N_-\|_F = o\left(\sqrt{\frac{k_1}{n}}\right).$$

Bounding $\mathcal{E}_3$. We have the following Lemma.

Lemma 2.3.13. *Let $\mathbf{Z} \in \mathbf{R}^{n \times k_1}$ so that each entry in $\mathbf{Z}$ is an independent standard Gaussian and $E \in \mathbf{R}^{n \times d_2}$ so that each entry in E is an independent Gaussian $N(0, \sigma_\epsilon^2)$. For sufficiently large n, k_1, and d_2, where $k_1 \leq d_2$, we have*

$$\left\| \frac{1}{n} \mathbf{Z}^{\mathrm{T}} E \right\| \leq \frac{1.1\sigma_\epsilon}{\sqrt{n}} (\sqrt{k_1} + \sqrt{d_2}).$$

Proof of Lemma 2.3.13. Let $t = \max\left\{ \frac{10 \log^2 n}{\sqrt{n}}, 4\sqrt{\frac{k_1}{n}} \right\}$. By Lemma 2.3.10, with high probability $\| \frac{1}{n} \mathbf{Z}^{\mathrm{T}} \mathbf{Z} - I \| \leq t$. This implies that the eigenvalues of $\mathbf{Z}^{\mathrm{T}} \mathbf{Z}$ are all within the range $n(1 \pm t)$. Note that for $0 < \eta < 1/3$, if $\xi \in [1 - \eta, 1 + \eta]$, then $\sqrt{\xi} \in [1 - 2\eta, 1 + 2\eta]$. This implies that the singular values of $\mathbf{Z}$ are within the range $\sqrt{n}(1 \pm 2t)$.

Let $\Sigma^Z / \sqrt{n} = I + \Delta^Z$, where $\|\Delta^Z\| \leq 2t$. We have

$$\begin{aligned} \frac{1}{n} \mathbf{Z}^{\mathrm{T}} E &= V^Z \left(\frac{\Sigma^Z}{\sqrt{n}} \right) (U^Z)^{\mathrm{T}} \frac{E}{\sqrt{n}} = V^Z (I + \Delta^Z)(U^Z)^{\mathrm{T}} \frac{E}{\sqrt{n}} \\ &= \underbrace{V^Z}_{k_1 \times k_1} \underbrace{(U^Z)^{\mathrm{T}}}_{k_1 \times n} \underbrace{\frac{E}{\sqrt{n}}}_{n \times d_2} + V^Z \Delta^Z (U^Z)^{\mathrm{T}} \frac{E}{\sqrt{n}}. \end{aligned} \tag{2.21}$$

Using the fact that the columns of U^Z are orthonormal vectors, V^Z is a unitary matrix, and $k_1 \leq d_2$, we see that $V^Z (U^Z)^{\mathrm{T}} E / \sqrt{n}$ is a matrix with i.i.d. Gaussian entries with standard deviation $\sigma_\epsilon / \sqrt{n}$.

Let $B = V^Z (U^Z)^{\mathrm{T}} E / \sigma_\epsilon$ and $\tilde{B} = (U^Z)^{\mathrm{T}} E / \sigma_\epsilon$. Then, from (2.21), we have

$$\frac{1}{n} \mathbf{Z}^{\mathrm{T}} E = \frac{\sigma_\epsilon}{\sqrt{n}} \left(B + V^Z \Delta^Z \tilde{B} \right). \tag{2.22}$$

The entries in B ($\tilde{B}$) are all i.i.d Gaussian. By Marchenko-Pastar's law (and the finite sample bound of it [134]), we have with high probability $\|\tilde{B}\|, \|B\| = \sqrt{k_1} + \sqrt{d_2} + o(\sqrt{k_1} + \sqrt{d_2})$. Therefore, with high probability:

$$\left\| \frac{1}{n} \mathbf{Z}^{\mathrm{T}} E \right\|_2 \leq \frac{1.1\sigma_\epsilon}{\sqrt{n}} (\sqrt{k_1} + \sqrt{d_2}).$$

□

Lemma 2.3.13 implies that

$$\|\mathcal{E}_3\|_2 \leq \frac{1.1\sigma_\epsilon}{\sqrt{n}}(\sqrt{k_1} + \sqrt{d_2}).$$

Bounding $\mathcal{E}_4$. We have

$$\mathbb{E}[\|\mathcal{E}_4\|_F^2] = \frac{1}{n^2}\,\mathbb{E}[\|\Delta_+^{\mathrm{T}} E\|_F^2] = \frac{d_2}{n}\|\Delta_+\|_F^2 = O\left(\frac{d_2\epsilon}{n\delta^3}\right) = o\left(\frac{d_2}{n}\right).$$

Using a Chernoff bound, we have whp $\|\mathcal{E}_4\|_F = o\left(\frac{d_2}{n}\right)$.

Bounding $\mathcal{E}_5$. Because $\mathcal{E}_5 = \frac{1}{n}\Delta_+^{\mathrm{T}}(\mathbf{Z}N^{\mathrm{T}})$, we have

$$\|\mathcal{E}_5\|_F^2 \leq \frac{1}{n^2}\|\Delta_+^{\mathrm{T}}\|_2^2\|\mathbf{Z}N^{\mathrm{T}}\|_F^2.$$

Using a simple Chernoff bound, we have whp,

$$\|\mathbf{Z}N^{\mathrm{T}}\|_F^2 \leq 2n\|M\mathbf{x}\|_2^2 \leq 2n\|M\|_2^2\|\mathbf{x}\|_2^2 \leq 2\Upsilon n.$$

This implies $\|\mathcal{E}_5\|_F^2 \leq O\left(\frac{\epsilon^2}{n^2\delta^6}n^2\Upsilon^2\right) = O\left(\frac{\epsilon^2}{\delta^6}\right)$.

We may let

$$\mathcal{E}_L = \mathcal{E}_1 + \mathcal{E}_2 + \mathcal{E}_3 + \mathcal{E}_4$$
$$\mathcal{E}_T = \mathcal{E}_5.$$

We can check that

$$\begin{aligned}\|\mathcal{E}_L\|_2 &\le \|\mathcal{E}_1\|_2 + \|\mathcal{E}_2\|_2 + \|\mathcal{E}_3\|_2 + \|\mathcal{E}_4\|_2 \\ &\le \Upsilon \max\left\{\frac{10\log^2 n}{\sqrt{n}}, 4\frac{k_1}{n}\right\} + o\left(\sqrt{\frac{k_1}{n}}\right) + \frac{1.1\sigma_\epsilon}{\sqrt{n}}(\sqrt{k_1}+\sqrt{d_2}) + o\left(\frac{d_2}{n}\right) \\ &\le 2.2\sigma_{\sigma_\epsilon}\sqrt{\frac{d_2}{n}}\end{aligned}$$

Also, we can see that $\|\mathcal{E}_T\|_F = \|\mathcal{E}_5\|_F = O(\epsilon/\delta^3)$. This completes the proof for Proposition 2.3.9.

2.3.2.3 Step 3. Analysis of our algorithm's MSE

Let us recall our notation:

1. $\mathbf{z} = (\Lambda^*)^{-\frac{1}{2}}(V^*)^{\mathrm{T}}\mathbf{x}$ and $\delta_+ = \hat{\mathbf{z}}_+ - W^{\mathrm{T}}\mathbf{z}_+$.
2. We let $\hat{N}_+^{\mathrm{T}} = \hat{\mathbf{Z}}_+^{\mathrm{T}}\mathbf{Y}$ be the output of Step-1-PCA-X in Fig. 2.1.
3. All singular vectors in $\hat{N}_+$ whose associated singular values $\ge \theta\sigma_\epsilon\sqrt{\frac{d_2}{n}}$ are kept.

Let ℓ be the largest index such that $\sigma_\ell^{N_+} \ge \theta\sigma_\epsilon\sqrt{\frac{d_2}{n}}$. One can see that our testing forecast is $\mathbf{P}_{k_2}(\hat{N}_+)\hat{\mathbf{z}}_+$. Therefore, we need to bound $\mathbb{E}_{\mathbf{z}}[\|\mathbf{P}_{k_2}(\hat{N}_+)\hat{\mathbf{z}}_+ - N\mathbf{z}\|^2]$.

By Proposition 2.3.9, we have $\hat{N}_+ = (W^{\mathrm{T}}N_+^{\mathrm{T}} + \mathcal{E}_L + \mathcal{E}_T)^{\mathrm{T}}$, where $\|\mathcal{E}_L\|_2 \le 2.2\sigma_\varepsilon\sqrt{\frac{d_2}{n}}$ and $\|\mathcal{E}_T\|_F = O(\varepsilon/\delta^3)$ whp. Let $\mathcal{E} \triangleq \mathcal{E}_L + \mathcal{E}_T$. We have

$$\begin{aligned}\mathbf{P}_{k_2}(N_+W + \mathcal{E}^{\mathrm{T}})\hat{\mathbf{z}}_+ &= \mathbf{P}_{k_2}(N_+W + \mathcal{E}^{\mathrm{T}})(W^{\mathrm{T}}\mathbf{z}_+ + \delta_+) \\ &= \mathbf{P}_{k_2}(N_+W + \mathcal{E}^{\mathrm{T}})W^{\mathrm{T}}W(W^{\mathrm{T}}\mathbf{z}_+ + \delta_+) \\ &= \mathbf{P}_{k_2}((\underbrace{N_+}_{d_2\times k_1}\underbrace{W}_{k_1\times k_1} + \underbrace{\mathcal{E}^{\mathrm{T}}}_{d_2\times k_1})\underbrace{W^{\mathrm{T}}}_{k_1\times k_1})(\underbrace{WW^{\mathrm{T}}}_{k_1\times k_1}\underbrace{\mathbf{z}_+}_{k_1\times 1} + \underbrace{W}_{k_1\times k_1}\underbrace{\delta_+}_{k_1\times 1}) \\ &= \mathbf{P}_{k_2}\left(N_+ + (W\mathcal{E})^{\mathrm{T}}\right)(\mathbf{z}_+ + W\delta_+).\end{aligned}$$

Let $\mathcal{E}' = (W\mathcal{E})^{\mathrm{T}}$, $\mathcal{E}'_L = (W\mathcal{E}_L)^{\mathrm{T}}$, $\mathcal{E}'_T = (W\mathcal{E}_T)^{\mathrm{T}}$, and $\delta'_+ = W\delta_+$. We still have $\|\mathcal{E}'_L\|_2 \le 2.2\sigma_\epsilon\sqrt{\frac{d_2}{n}}$, and $\|\mathcal{E}'_T\|_F = O(\epsilon/\delta^3)$.

We next have

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\mathbf{z}}\left[\|\mathbf{P}_{k_2}(\hat{N}_+)\hat{\mathbf{z}}_+ - N\mathbf{z}\|_2^2\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{z}}\left[\left\|(\mathbf{P}_{k_2}(N_+ + \mathcal{E}')\mathbf{z}_+ - N_+\mathbf{z}_+) + \mathbf{P}_{k_2}(N_+ + \mathcal{E}')\delta'_+ - N_-\mathbf{z}_-\right\|_2^2\right] \\
&\le \underbrace{\mathop{\mathbb{E}}_{\mathbf{z}}\left[\left\|(\mathbf{P}_{k_2}(N_+ + \mathcal{E}')\mathbf{z}_+ - N_+\mathbf{z}_+)\right\|_2^2\right]}_{\triangleq\Phi_1} + \underbrace{\mathop{\mathbb{E}}_{\mathbf{z}}\left[\left\|\mathbf{P}_{k_2}(N_+ + \mathcal{E}')\delta'_+ - N_-\mathbf{z}_-\right\|_2^2\right]}_{\triangleq\Phi_2} \\
&\quad + 2\sqrt{\mathop{\mathbb{E}}_{\mathbf{z}}\left[\left\|(\mathbf{P}_{k_2}(N_+ + \mathcal{E}')\mathbf{z}_+ - N_+\mathbf{z}_+)\right\|_2^2\right] \cdot \mathop{\mathbb{E}}_{\mathbf{z}}\left[\left\|\mathbf{P}_{k_2}(N_+ + \mathcal{E}')\delta'_+ - N_-\mathbf{z}_-\right\|_2^2\right]} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Cauchy Schwarz for random variables)} \\
&= \Phi_1 + \Phi_2 + 2\sqrt{\Phi_1\Phi_2}.
\end{aligned}
$$

We first bound Φ_2 (the easier term). We have

$$
\begin{aligned}
\Phi_2 &= \mathop{\mathbb{E}}_{\mathbf{z}}\left[\left\|\mathbf{P}_{k_2}(N_+ + \mathcal{E}')\delta'_+ - N_-\mathbf{z}_-\right\|_2^2\right] \\
&\le 2\mathop{\mathbb{E}}_{\mathbf{z}}\left[\left\|\mathbf{P}_{k_2}(N_+ + \mathcal{E}')\delta'_+\right\|_2^2\right] + 2\,\mathbb{E}\left[\left\|N_-\mathbf{z}_-\right\|_2^2\right]
\end{aligned}
$$

We first bound $\mathbb{E}_{\mathbf{z}}\left[\left\|\mathbf{P}_{k_2}(N_+ + \mathcal{E}')\delta'_+\right\|_2^2\right]$. We consider two cases.

Case 1. $\sigma_{\max}(N_+) > \frac{\theta}{2}\sigma_\epsilon\sqrt{\frac{d_2}{n}}$. In this case, we observe that $\|\mathcal{E}\|_2 \le 2.2\sigma_\epsilon\sqrt{\frac{d_2}{n}} + o(1)$. This implies that $\|N_+ + \mathcal{E}'\|_2 = O(\|N_+\|_2) = O(1)$. Therefore, $\mathbb{E}_{\mathbf{z}}\left[\left\|\mathbf{P}_{k_2}(N_+ + \mathcal{E}')\delta'_+\right\|_2^2\right] \le \|(N_+ + \mathcal{E}')\delta'_+\|_2^2 = O(\|\delta'_+\|_2^2)$.

Case 2. $\sigma_{\max}(N_+) \le \frac{\theta}{2}\sigma_\epsilon\sqrt{\frac{d_2}{n}}$. In this case, $\|N_+ + \mathcal{E}'\|_2 \le \theta\sigma\sqrt{\frac{d_2}{n}}$. This implies $\mathbf{P}_{k_2}(N_+ + \mathcal{E}')\delta'_+ = 0$ (i.e., the projection $\mathbf{P}_{k_2}(\cdot)$ will not keep any subspace).

This case also implies $\mathbb{E}_{\mathbf{z}}\left[\left\|\mathbf{P}_{k_2}(N_+ + \mathcal{E}')\delta'_+\right\|_2^2\right] = 0 = O(\|\delta'_+\|_2^2)$

Next, we have $\mathbb{E}[\|N_-\mathbf{z}_-\|_2^2] = \|N_-\|_F^2 = O\left(\delta^{\frac{\omega-1}{\omega+1}}\right)$.

Therefore,

$$
\Phi_2 = O\left(\frac{\epsilon^2}{\delta^6} + \delta^{\frac{\omega-1}{\omega+1}}\right).
$$

Next, we move to bound

$$\mathop{\mathbb{E}}_{\mathbf{z}}\left[\left\|(\mathbf{P}_{k_2}(N_+ + \mathcal{E}')\mathbf{z}_+ - N_+\mathbf{z}_+)\right\|_2^2\right].$$

We shall construct an orthonormal basis on $\mathbf{R}^{d_2}$ and use the basis to "measure the mass". Let us describe this simple idea at high-level first. Let $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_{d_2}$ be a basis for $\mathbf{R}^{d_2}$ and let $A \in \mathbf{R}^{d_2 \times k_1}$ be an arbitrary matrix. We have $\|A\|_F^2 = \sum_{i \le d_2} \|\mathbf{v}_i^{\mathrm{T}} A\|_2^2$. The meaning of this equality is that we may apply a change of basis on the columns of A and the "total mass" of A should remain unchanged after the basis change. Our orthonormal basis consists of three groups of vectors.

Group 1. $\{U_{:,i}^{N_+}\}$ for $i \le \ell$, where ℓ is the number of $\sigma_i(N^+)$ such that $\sigma_i(N^+) \ge \theta\sigma_\epsilon\sqrt{\frac{d_2}{n}}$.

Group 2. The subspace in $\mathrm{Span}(\{U_{:,i}^{\hat{N}}\}_{i \le k_2})$ that is orthogonal to $\{U_{:,i}^{N_+}\}_{i \le \ell}$. Let us refer to these vectors as $\hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_s$ and $\hat{U}_{[s]} = [\hat{\mathbf{u}}_1, \ldots \hat{\mathbf{u}}_s]$.

Group 3. An arbitrary basis that is orthogonal to vectors in group 1 and group 2. Let us refer to them as $\mathbf{r}_1, \ldots, \mathbf{r}_t$.

We have

$$\begin{aligned}
&\|\mathbf{P}_{k_2}(N_+ + \mathcal{E}') - N_+\|_F^2 \\
&= \sum_{i \le \ell} \left\| \left(U_{:,i}^{N_+}\right)^{\mathrm{T}} \left(\mathbf{P}_{k_2}(N_+ + \mathcal{E}') - N_+\right) \right\|_2^2 && \text{Term 1} \\
&\quad + \sum_{i \le s} \|\hat{\mathbf{u}}_i^{\mathrm{T}} \left(\mathbf{P}_{k_2}(N_+ + \mathcal{E}') - N_+\right)\|_2^2 && \text{Term 2} \\
&\quad + \sum_{i \le r} \|\mathbf{r}_i^{\mathrm{T}} \left(\mathbf{P}_{k_2}(N_+ + \mathcal{E}') - N_+\right)\|_2^2 && \text{Term 3}
\end{aligned}$$

To understand the reason we perform such grouping, we can imagine making a decision for an (overly) simplified problem for each direction in the basis: consider a univariate estimation problem $y = \mu + \epsilon$ with μ being the signal, $\epsilon \sim N(0, \sigma^2)$ being the noise, and y being the observation. Let us consider the case we observe only one sample. Now when

$y \gg \sigma$, we can use y as the estimator and $\mathbb{E}[(y-\mu)^2] = \sigma^2$. This high signal-to-noise setting corresponds to the vectors in Group 1.

When $y \approx 3\sigma$, we have $\mu^2 = \mathbb{E}[(y-\epsilon)^2] \approx (3-1)^2\sigma^2 = 4\sigma^2$. On the other hand, $\mathbb{E}[(y-\mu)^2] = \sigma^2$. This means if we use y as the estimator, the forecast is at least better than trivial. The median signal-to-noise setting corresponds to the vectors in group 2.

When $y \ll \sigma$, we can simply use $\hat{y} = 0$ as the estimator. This low signal-to-noise setting corresponds to vectors in group 3.

In other words, we expect: *(i)* In term 1, signals along each direction of vectors in group 1 can be extracted. Each direction also pays a σ^2 term, which in our setting corresponds to $\theta\sigma_\epsilon\sqrt{\frac{d_2}{n}}$. Therefore, the MSE can be bounded by $O(\ell\theta^2\sigma_\epsilon^2 d_2/n)$. *(ii)* In terms 2 and 3, we do at least (almost) as well as the "trivial forecast" ($\hat{\mathbf{y}} = 0$). There is also an error produced by the estimator error from $\hat{\mathbf{z}}_+$, and the tail error produced from cutting out features in Step-1-PCA-X in Fig. 2.1.

Now we proceed to execute this idea.

Term 1. $\sum_{i\le\ell}\left\|\left(U_{:,i}^{N_+}\right)^{\mathrm{T}}(\mathbf{P}_{k_2}(N_+ + \mathcal{E}') - N_+)\right\|_2^2$**.** Let $\hat{U} \in \mathbf{R}^{d_2\times d_2}$ be the left singular vector of $N_+ + \mathcal{E}'$. We let $\hat{U}$ have d_2 columns to include those vectors whose corresponding

singular values are 0 for the ease of calculation. We have

$$
\begin{aligned}
&\sum_{i\le\ell}\left\|\left(U^{N_+}_{:,i}\right)^{\mathrm{T}}\left(\mathbf{P}_{k_2}(N_+ + \mathcal{E}') - N_+\right)\right\|_2^2 \\
&= \sum_{i\le\ell}\left\|\left(U^{N_+}_{:,i}\right)^{\mathrm{T}}\left(\hat{U}_{:,1:k_2}\hat{U}^{\mathrm{T}}_{:,1:k_2}(N_+ + \mathcal{E}') - N_+\right)\right\|_2^2 \\
&= \sum_{i\le\ell}\left\|\left(U^{N_+}_{:,i}\right)^{\mathrm{T}}\left(\left(\hat{U}\hat{U}^{\mathrm{T}} - \hat{U}_{:,k_2+1:d_2}\hat{U}^{\mathrm{T}}_{:,k_2+1:d_2}\right)(N_+ + \mathcal{E}') - N_+\right)\right\|_2^2 \\
&= \sum_{i\le\ell}\left\|\left(U^{N_+}_{:,i}\right)^{\mathrm{T}}\left(\mathcal{E}' - \hat{U}_{:,k_2+1:d_2}\hat{U}^{\mathrm{T}}_{:,k_2+1:d_2}(N_+ + \mathcal{E}')\right)\right\|_2^2 \\
&\le 2\left\{\sum_{i\le t}\left\|\left(U^{N_+}_{:,i}\right)^{\mathrm{T}}\mathcal{E}'\right\|_2^2 + \sum_{i\le\ell}\left\|\left(U^{N_+}_{:,i}\right)^{\mathrm{T}}\hat{U}_{:,k_2+1:d_2}\hat{U}^{\mathrm{T}}_{:,k_2+1:d_2}(N_+ + \mathcal{E}')\right\|_2^2\right\} \\
&\le O\left(\ell\|\mathcal{E}'_L\|_2^2 + \|\mathcal{E}'_T\|_F^2 + \sum_{i\le\ell}\left\|\left(U^{N_+}_{:,i}\right)^{\mathrm{T}}\right\|_2^2\left\|\hat{U}_{:,k_2+1:d_2}\right\|_2^2\underbrace{\left\|\hat{U}^{\mathrm{T}}_{:,k_2+1:d_2}(N_+ + \mathcal{E}')\right\|_2^2}_{\le\frac{\theta^2\sigma_\epsilon^2 d_2}{n}\text{ by the definition of }k_2.}\right) \\
&= O\left(\ell\|\mathcal{E}'_L\|_2^2 + \|\mathcal{E}'_T\|_F^2 + \frac{\ell d_2\theta^2\sigma_\epsilon^2}{n}\right) \\
&= O\left(\frac{\ell d_2\theta^2\sigma_\epsilon^2}{n} + +\|\mathcal{E}'_T\|_F^2\right)
\end{aligned}
$$

Term 2. $\sum_{i\le s}\|\hat{\mathbf{u}}_i^{\mathrm{T}}\left(\mathbf{P}_{k_2}(N_+ + \mathcal{E}') - N_+\right)\|_2^2$**.** We have

$$
\begin{aligned}
&\sum_{i\le s}\|\hat{\mathbf{u}}_i^{\mathrm{T}}\left(\mathbf{P}_{k_2}(N_+ + \mathcal{E}') - N_+\right)\|_2^2 = \sum_{i\le s}\|\hat{\mathbf{u}}_i^{\mathrm{T}}\left(\mathbf{P}_{k_2}(N_+ + \mathcal{E}') - (N_+ + \mathcal{E}') + \mathcal{E}'\right)\|_2^2 \\
&= \sum_{i\le s}\|\hat{\mathbf{u}}_i^{\mathrm{T}}\mathcal{E}'\|_2^2
\end{aligned}
$$

On the other hand, note that

$$
\begin{aligned}
&\sum_{i\le s}\|\hat{\mathbf{u}}_i^{\mathrm{T}}N_+\|_2^2\\
&=\sum_{i\le s}\big\|\hat{\mathbf{u}}_i^{\mathrm{T}}(N_++\mathcal{E}')-\hat{\mathbf{u}}_i^{\mathrm{T}}\mathcal{E}'\big\|_2^2\\
&=\sum_{i\le s}\Big(\big\|\hat{\mathbf{u}}_i^{\mathrm{T}}(N_++\mathcal{E}')\big\|_2^2+\|\hat{\mathbf{u}}_i^{\mathrm{T}}\mathcal{E}'\|_2^2-2\left\langle\hat{\mathbf{u}}_i^{\mathrm{T}}(N_++\mathcal{E}'),\hat{\mathbf{u}}_i^{\mathrm{T}}(\mathcal{E}'_L+\mathcal{E}'_T)\right\rangle\Big)\\
&=\sum_{i\le s}\Big(\big\|\hat{\mathbf{u}}_i^{\mathrm{T}}(N_++\mathcal{E}')\big\|_2^2-2\langle\hat{\mathbf{u}}_i^{\mathrm{T}}(N_++\mathcal{E}'),\hat{\mathbf{u}}_i^{\mathrm{T}}\mathcal{E}'_L\rangle\Big)+\underbrace{\sum_{i\le s}\|\hat{\mathbf{u}}_i^{\mathrm{T}}\mathcal{E}'\|_2^2}_{=\text{ Term 2.}}-2\sum_{i\le s}\langle\hat{\mathbf{u}}_i^{\mathrm{T}}(N_++\mathcal{E}'),\hat{\mathbf{u}}_i^{\mathrm{T}}\mathcal{E}'_T\rangle
\end{aligned}
\tag{2.23}
$$

Note that

$$
\begin{aligned}
&\sum_{i\le s}\Big(\big\|\hat{\mathbf{u}}_i^{\mathrm{T}}(N_++\mathcal{E}')\big\|_2^2-2\langle\hat{\mathbf{u}}_i^{\mathrm{T}}(N_++\mathcal{E}'),\hat{\mathbf{u}}_i^{\mathrm{T}}\mathcal{E}'_L\rangle\Big)\\
&\ge\sum_{i\le s}\|\hat{\mathbf{u}}_i^{\mathrm{T}}(N_++\mathcal{E}')\|_2\big(\underbrace{\|\hat{\mathbf{u}}_i^{\mathrm{T}}(N_++\mathcal{E}')\|_2}_{\ge\theta\sigma_\epsilon\sqrt{\frac{d_2}{n}}}-2\underbrace{\|\hat{\mathbf{u}}_i^{\mathrm{T}}\mathcal{E}'_L\|_2}_{\le 2.2\sigma_\epsilon\sqrt{\frac{d_2}{n}}}\big)\\
&\ge 0\text{ (using the fact that }\theta\text{ is sufficiently large).}
\end{aligned}
\tag{2.24}
$$

Next, we examine the term $-2\sum_{i\le s}\langle\hat{\mathbf{u}}_i^{\mathrm{T}}(N_++\mathcal{E}'),\hat{\mathbf{u}}_i^{\mathrm{T}}\mathcal{E}'_T\rangle$.

$$
\begin{aligned}
&- 2\sum_{i\le s}\langle \hat{\mathbf{u}}_i^{\mathrm{T}}(N_+ + \mathcal{E}'), \hat{\mathbf{u}}_i^{\mathrm{T}}\mathcal{E}'_T\rangle \\
&= -2\langle \hat{U}_{[s]}^{\mathrm{T}}(N_+ + \mathcal{E}'), \hat{U}_{[s]}^{\mathrm{T}}\mathcal{E}'_T\rangle \\
&= -2\mathrm{trace}\left(\hat{U}_{[s]}^{\mathrm{T}}\mathcal{E}'_T(N_+ + \mathcal{E}')^{\mathrm{T}}\hat{U}_{[s]}\right) \\
&\ge -2\left\|\mathrm{trace}(\mathcal{E}'_T(N_+ + \mathcal{E}')^{\mathrm{T}}\right\|_2 \left\|\hat{U}_{[s]}^{\mathrm{T}}\hat{U}_{[s]}\right\|_2 \\
&= -2\left|\langle (\mathcal{E}'_T)^{\mathrm{T}}, N_+ + \mathcal{E}'\rangle\right| \\
&\ge -2\|\mathcal{E}'_T\|_F\|N_+ + \mathcal{E}'\|_F && \text{(Cauchy Schwarz)} \\
&\ge -2\|\mathcal{E}'_T\|_F(\|N_+\|_F + \|\mathcal{E}'\|_F)) \\
&\ge -O(\|\mathcal{E}'_T\|_F) && (\|N_+\|_F^2 \le \|N\|_F^2 = \mathbb{E}[\|N\mathbf{z}\|^2] = \mathbb{E}[\|M\mathbf{x}\|^2] = O(1))
\end{aligned}
\tag{2.25}
$$

(2.23), (2.24), and (2.25) imply that

$$
\sum_{i\le s}\left\|\mathbf{P}_{k_2}(N_+ + \mathcal{E}') - N_+\right\|_2^2 \le \sum_{i\le s}\left\|\hat{U}^{\mathrm{T}}N_+\right\|_2^2 + O(\|\mathcal{E}'_T\|_F).
$$

Term 3. We have

$$
\sum_{i\le r}\|\mathbf{r}_i^{\mathrm{T}}\left(\mathbf{P}_{k_2}(N_+ + \mathcal{E}') - N_+\right)\|_2^2 = \sum_{i\le r}\|\mathbf{r}_i^{\mathrm{T}}N_+\|_2^2.
$$

This is because $\mathbf{r}_i$'s are orthogonal to the first k_2 left singular vectors of $N_+ + \mathcal{E}'$.

We sum together all the terms:

$$
\begin{aligned}
\left\|\mathbf{P}_{k_2}(N_+ + \mathcal{E}') - N_+\right\|_F^2 &\le O\left(\frac{\ell d_2\theta^2\sigma_\epsilon^2}{n}\right) + \underbrace{\sum_{i\le s}\left\|\hat{U}_i^{\mathrm{T}}N_+\right\|_2^2}_{(*)} + \underbrace{\sum_{i\le t}\|\mathbf{r}_i^{\mathrm{T}}N_+\|_2^2}_{(**)} + O\left(\|\mathcal{E}'_T\|_F\right). \\
&= O\left(\frac{\ell d_2\theta^2\sigma_\epsilon^2}{n}\right) + \|N_+\|_F^2 - \sum_{i\le \ell}\left(\sigma_i^{N_+}\right)^2 + O\left(\|\mathcal{E}'_T\|_F\right)
\end{aligned}
\tag{2.26}
$$

In the above analysis, we used $\text{Span}(\{\hat{\mathbf{u}}_i\}_{i\le s}, \{\mathbf{r}_i\}_{i\le t})$ is orthogonal to $\text{Span}(\{U^{N_+}_{:,i}\}_{i\le \ell})$. Therefore, we can collect (*) and (**) and obtain

$$\sum_{i\le s}\left\|\hat{U}_i^{\mathrm{T}} N_+\right\|_2^2 + \sum_{i\le t}\|\mathbf{r}_i^{\mathrm{T}} N_+\|_2^2 = \|N_+\|_F^2 - \sum_{i\le \ell}(\sigma_i^{N_+})^2.$$

Now the MSE is in terms of $\sigma_i^{N_+}$. We aim to bound the MSE in σ_i^N. So we next relate $\sum_{i\le\ell}(\sigma_i^{N_+})^2$ with $\sum_{i\le\ell}(\sigma_i^N)^2$. Recall that $\tilde{N}_+ = [N_+, \mathbf{0}]$, where $\mathbf{0} \in \mathbf{R}^{d_2\times(d_1-k_1)}$. The singular values of $\tilde{N}_+$ are the same as those of N_+. By using a standard matrix perturbation result, we have

$$\sum_{i\le\ell}\left(\sigma_i^{N_+} - \sigma_i^N\right)^2 = \sum_{i\le\ell}\left(\sigma_i^{\tilde{N}_+} - \sigma_i^N\right)^2 \le \|N_-\|_F^2 = c\delta^{\frac{\omega-1}{\omega+1}} \tag{2.27}$$

for some constant c. We may think (2.27) as a constraint and maximize the difference $\sum_{i\le\ell}(\sigma_i^N)^2 - \sum_{i\le\ell}(\sigma_i^{N_+})^2$. This is maximized when $\sigma_1^N = \sigma_1^{N_+} + \sqrt{c\delta^{\frac{\omega-1}{\omega+1}}}$ and $\sigma_i^N = \sigma_i^{N_+}$ for $i > 1$.

Therefore,

$$\begin{aligned}\sum_{i\le\ell}\left(\sigma_i^N\right)^2 &\le \sum_{1\le i\le\ell}\left(\sigma_i^{N_+}\right)^2 + \left(\sigma_1^{N_+} + \sqrt{c\delta^{\frac{\omega-1}{\omega+1}}}\right)^2 \\ &= \sum_{1\le i\le\ell}\left(\sigma_i^{N_+}\right)^2 + O\left(\sqrt{\delta^{\frac{\omega-1}{\omega+1}}}\right).\end{aligned} \tag{2.28}$$

Now (2.26) becomes

$$(2.26) \le \|N_+\|_F^2 - \sum_{i\le\ell}(\sigma_i^N)^2 + O\left(\frac{\ell d_2\theta^2\sigma_\epsilon^2}{n}\right) + O(\varepsilon/\delta^3) + O\left(\sqrt{\delta^{\frac{\omega-1}{\omega+1}}}\right). \tag{2.29}$$

Next, we assert that $\ell \ge \ell^*$. Recall that $\|\sigma_i^{N_+} - \sigma_i^N\|_2^2 = \|N_-\|_F^2 = o(1)$. This implies

$\sigma_i^{N_+} > \theta\sigma_\epsilon\sqrt{\frac{d_2}{n}}$ for $i \le \ell^*$, i.e., $\ell \ge \ell^*$. So we have

$$\Phi_1 = \|\mathbf{P}_{k_2}(N_+ + \mathcal{E}') - N_+\|_F^2 \le \|N\|_F^2 - \sum_{i\le\ell^*}(\sigma_i^N)^2 + O\left(\frac{\ell^* d_2\theta^2\sigma_\epsilon^2}{n}\right) + O(\varepsilon/\delta^3) + O\left(\sqrt{\delta^{\frac{\omega-1}{\omega+1}}}\right). \tag{2.30}$$

Finally, we obtain the bound for $\Phi_1 + \Phi_2 + 2\sqrt{\Phi_1\Phi_2}$. Note that

$$\frac{\Phi_2}{\Phi_1} \le \frac{O\left(\frac{\varepsilon^2}{\delta^6} + \delta^{\frac{\omega-1}{\omega+1}}\right)}{O\left(\frac{\varepsilon}{\delta^3} + \sqrt{\delta^{\frac{\omega-1}{\omega+1}}}\right)} \le \min\left\{\frac{\varepsilon}{\delta^3}, \sqrt{\delta^{\frac{\omega-1}{\omega+1}}}\right\} (= o(1)).$$

We have

$$\begin{aligned}
&\Phi_1 + \Phi_2 + 2\sqrt{\Phi_1\Phi_2} \\
&= \Phi_1(1 + 2\sqrt{\frac{\Phi_2}{\Phi_1}}) + \Phi_2 \\
&= \left[\|N\|_F^2 - \sum_{i\le\ell^*}(\sigma_i^N)^2 + O\left(\frac{\ell^* d_2\theta^2\sigma_\epsilon^2}{n}\right) + O(\varepsilon/\delta^3) + O\left(\sqrt{\delta^{\frac{\omega-1}{\omega+1}}}\right)\right] \\
&\qquad \times \left\{1 + \min\left\{\sqrt{\frac{\varepsilon}{\delta^3}} + \left(\delta^{\frac{\omega-1}{\omega+1}}\right)^{\frac{1}{4}}\right\}\right\} + O\left(\frac{\varepsilon^2}{\delta^6}\right) + O\left(\delta^{\frac{\omega-1}{\omega+1}}\right) \\
&\le \|N\|_F^2 - \sum_{i\le\ell^*}(\sigma_i^N)^2 + O\left(\frac{\ell^* d_2\theta^2\sigma_\epsilon^2}{n}\right) + O\left(\sqrt{\frac{\varepsilon}{\delta^3}}\right) + O\left(\delta^{\frac{\omega-1}{4(\omega+1)}}\right).
\end{aligned} \tag{2.31}$$

This completes the proof of Theorem 2.3.2.

2.4 Lower bound

Our algorithm accurately estimates the singular vectors of N that correspond to singular values above the threshold $\tau = \theta\sigma_\epsilon\sqrt{\frac{d_2}{n}}$. However, it may well happen that most of the spectral 'mass' of N lies only slightly below this threshold τ. In this section, we establish that *no algorithm* can do better than us, in a bi-criteria sense, i.e. we show that any algorithm that has a slightly smaller sample than ours can only minimally outperform ours in terms of MSE.

We establish 'instance dependent' lower bounds: When there is more 'spectral mass'

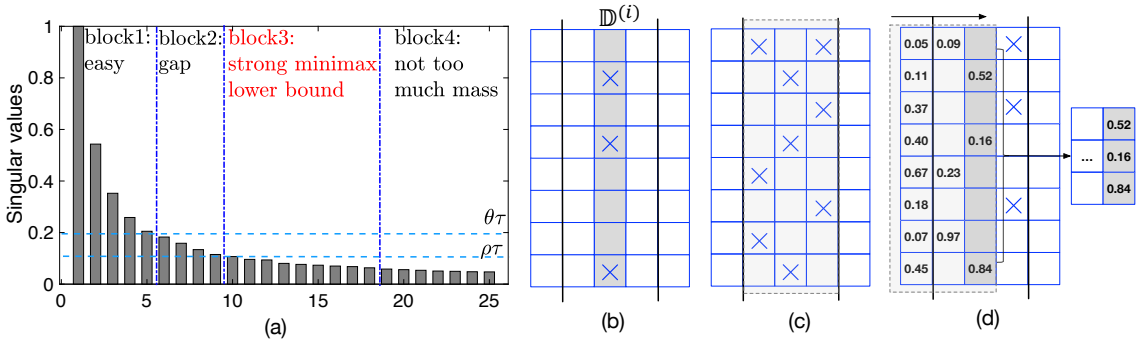


Figure 2.3: (a) **Major result:** signals in N are partitioned into four blocks. All signals in block 1 can be estimated (Thm 2.3.2). All signals in block 3 cannot be estimated (Prop 2.4.2). Our lower bound techniques does not handle a small tail in Block 4. A gap in block 2 exists between upper and lower bounds. (b)-(d) **Constructing $\mathbb{N}$:** Step 1 and 2 belong to the first stage; step 3 belongs to the second stage. (b) Step 1. Generate a random subset $\mathbb{D}^{(i)}$ for each row i, representing its non-zero positions. (c) Step 2. Randomly sample from $\mathbb{D}$, where $\mathbb{D}$ is the Cartesian product of $\mathbb{D}^{(i)}$. (d) Step 3. Fill in non-zero entries sequentially from left to right.

below the threshold, the performance of our algorithm will be worse, and we will need to establish that no algorithm can do much better. This departs from the standard minimax framework, in which one examines the entire parameter space of N, e.g. all rank r matrices, and produces a large set of statistically indistinguishable 'bad' instances [150]. These lower bounds are not sensitive to instance-specific quantities such as the spectrum of N, and in particular, if prior knowledge suggests that the unknown parameter N is far from these bad instances, the minimax lower bound cannot be applied.

We introduce the notion of *local minimax*. We partition the space into parts so that *similar* matrices are together. Similar matrices are those N that have the same singular values and right singular vectors; we establish strong lower bounds even against algorithms that know the singular values and right singular vectors of N. An equivalent view is to assume that the algorithm has oracle access to C^*, M's singular values, and M's right singular vectors. This algorithm can solve the orthogonalized form as N's singular values and right singular vectors can easily be deduced. Thus, the only reason why the algorithm needs data is to *learn* the left singular vectors of N. The lower bound we establish is the minimax bound for this 'unfair' comparison, where the competing algorithm is given more information. In fact, this can be reduced further, i.e., even if the algorithm 'knows' that

the left singular vectors of N are sparse, identifying the locations of the non-zero entries is the key difficulty that leads to the lower bound.

Definition 2.4.1 (Local minimax bound)**.** *Consider a model* $\mathbf{y} = M\mathbf{x} + \epsilon$*, where* $\mathbf{x}$ *is a random vector, so* $C^*(\mathbf{x}) = \mathbb{E}[\mathbf{x}\mathbf{x}^{\mathrm{T}}]$ *represents the co-variance matrix of the data distribution, and* $M = U^M \Sigma^M (V^M)^{\mathrm{T}}$*. The relation* $(M, \mathbf{x}) \sim (M', \mathbf{x}') \Leftrightarrow (\Sigma^M = \Sigma^{M'} \wedge V^M = V^{M'} \wedge C^*(\mathbf{x}) = C^*(\mathbf{x}'))$ *is an equivalence relation and let the equivalence class of* $(M, \mathbf{x})$ *be*

$$\mathcal{R}(M, \mathbf{x}) = \{(M', \mathbf{x}') : \Sigma^{M'} = \Sigma^M, V^{M'} = V^M, \text{ and } C^*(\mathbf{x}') = C^*(\mathbf{x})\}. \tag{2.32}$$

The local minimax bound for $\mathbf{y} = M\mathbf{x} + \epsilon$ *with* n *independent samples and* $\epsilon \sim N(0, \sigma_\epsilon^2 I_{d_2 \times d_2})$ *is*

$$\mathbf{r}(\mathbf{x}, M, n, \sigma_\epsilon) = \min_{\hat{M}} \max_{(M', \mathbf{x}') \in \mathcal{R}(M, \mathbf{x})} \mathop{\mathbb{E}}_{\substack{\mathbf{X}, \mathbf{Y} \text{ from} \\ \mathbf{y} \sim M'\mathbf{x}' + \epsilon}} [\mathop{\mathbb{E}}_{\mathbf{x}'} [\|\hat{M}(\mathbf{X}, \mathbf{Y})\mathbf{x}' - M'\mathbf{x}'\|_2^2 \mid \mathbf{X}, \mathbf{Y}]]. \tag{2.33}$$

It is worth interpreting (2.33) in some detail. For any two $(M, \mathbf{x})$, $(M', \mathbf{x}')$ in $\mathcal{R}(M, \mathbf{x})$, the algorithm has the same 'prior knowledge', so it can only distinguish between the two instances by using the *observed data*, in particular $\hat{M}$ is a function only of $\mathbf{X}$ and $\mathbf{Y}$, and we denote it as $\hat{M}(\mathbf{X}, \mathbf{Y})$ to emphasize this. Thus, we can evaluate the performance of $\hat{M}$ by looking at the worst possible $(M', \mathbf{x}')$ and considering the MSE $\mathbb{E}\,\|\hat{M}(\mathbf{X}, \mathbf{Y})\mathbf{x}' - M'\mathbf{x}'\|^2$.

Proposition 2.4.2. *Consider the problem* $\mathbf{y} = M\mathbf{x} + \epsilon$ *with normalized form* $\mathbf{y} = N\mathbf{z} + \epsilon$*. Let* ε *be a sufficient small constant. There exists a sufficiently small constant* ρ_0 *(that depends on* ε*) and a constant* c *such that for any* $\rho \le \rho_0$*,* $\mathbf{r}(\mathbf{x}, M, n, \sigma_\epsilon) \ge (1 - c\rho^{\frac{1}{2}-\varepsilon}) \sum_{i \ge \underline{t}} (\sigma_i^N)^2 - O\Big(\frac{\rho^{\frac{1}{2}-\varepsilon}}{d_2^{\omega-1}}\Big)$*, where* $\underline{t}$ *is the smallest index such that* $\sigma_{\underline{t}}^N \le \rho\sigma_\epsilon\sqrt{\frac{d_2}{n}}$*.*

Proposition 2.4.2 gives the lower bound on the MSE in expectation; it can be turned into a high probability result with suitable modifications. The proof of the lower bound uses a similar 'trick' to the one used in the analysis of the upper bound analysis to cut the tail. This results in an additional term $O\Big(\frac{\rho^{\frac{1}{2}-\varepsilon}}{d_2^{\omega-1}}\Big)$ which is generally smaller than the n^{-c_0} tail term in Theorem 2.3.2 and does not dominate the gap.

Gap requirement and bi-criteria approximation algorithms. Let $\tau = \sigma_\epsilon \sqrt{\frac{d_2}{n}}$. Theorem 2.3.2 asserts that any signal above the threshold $\theta\tau$ can be detected, i.e., the MSE is at most $\sum_{\sigma_i^N > \theta\tau} \sigma_i^2(N)$ (plus inevitable noise), whereas Proposition 2.4.2 asserts that any signal below the threshold $\rho\tau$ cannot be detected, i.e., the MSE is approximately at least $\sum_{\sigma_i^N \geq \rho\tau}(1 - \text{poly}(\rho))\sigma_i^2(N)$. There is a 'gap' between $\theta\tau$ and $\rho\tau$, as $\theta > 1$ and $\rho < 1$. See Fig. 2.3(a). This kind of gap is inevitable because both bounds are 'high probability' statements. This gap phenomenon appears naturally when the sample size is small as can be illustrated by this simple example. Consider the problem of estimating μ when we see one sample from $N(\mu, \sigma^2)$. Roughly speaking, when $\mu \gg \sigma$, the estimation is feasible, and whereas $\mu \ll \sigma$, the estimation is impossible. For the region $\mu \approx \sigma$, algorithms fail with constant probability and we cannot prove a high probability lower bound either.

While many of the signals can 'hide' in the gap, the inability to detect signals in the gap is a transient phenomenon. When the number of samples n is modestly increased, our detection threshold $\tau = \theta\sigma_\epsilon\sqrt{\frac{d_2}{n}}$ shrinks, and this hidden signal can be fully recovered. This observation naturally leads to a notion of bi-criteria optimization that frequently arises in approximation algorithms.

Definition 2.4.3. *An algorithm for solving the $\mathbf{y} = M\mathbf{x} + \epsilon$ problem is (α, β)-optimal if, when given an i.i.d. sample of size αn as input, it outputs an estimator whose MSE is at most β worse than the local minimax bound, i.e., $\mathbb{E}[\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2] \leq \mathbf{r}(\mathbf{x}, M, n, \sigma_\epsilon) + \beta$.*

Corollary 2.4.4. *Let ε and c_0 be small constants and ρ be a tunable parameter. Our algorithm is (α, β)-optimal for*

$$\alpha = \frac{\theta^2}{\rho^{\frac{5}{2}}}$$
$$\beta = O(\rho^{\frac{1}{2}-\varepsilon})\|M\mathbf{x}\|_2^2 + O(n^{-c_0})$$

The error term β consists of $\rho^{\frac{1}{2}-\epsilon}\|M\mathbf{x}\|_2^2$ that is directly characterized by the signal strength and an additive term $O(n^{-c_0}) = o(1)$. Assuming that $\|M\mathbf{x}\| = \Omega(1)$, i.e., the signal is not too weak, the term β becomes a single multiplicative bound $O(\rho^{\frac{1}{2}-\varepsilon} + n^{-c_0})\|M\mathbf{x}\|_2^2$.

This gives an easily interpretable result. For example, when our data size is $n \log n$, the performance gap between our algorithm and *any* algorithm that uses n samples is at most $o(\|M\mathbf{x}\|_2^2)$. The improvement is significant when other baselines deliver MSE in the additive form that could be larger than $\|M\mathbf{x}\|_2^2$ in the regime $n \leq d_1$.

Preview of techniques. Let $N = U^N \Sigma^N (V^N)^{\mathrm{T}}$ be the instance (in orthogonalized form). Our goal is to construct a collection $\mathcal{N} = \{N_1, \ldots, N_K\}$ of K matrices so that *(i)* For any $N_i \in \mathcal{N}$, $\Sigma^{N_i} = \Sigma^N$ and $V^{N_i} = V^N$. *(ii)* For any two $N_i, N_j \in \mathcal{N}$, $\|N - N'\|_F$ is large, and *(iii)* $K = \exp(\Omega(\mathrm{poly}(\rho) d^2))$ (cf. [150, Chap. 2]).

Condition (i) ensures that it suffices to construct unitary matrices U^{N_i}'s for $\mathcal{N}$, and that the resulting instances will be in the same equivalence class. Conditions (ii) and (iii) resemble standard construction of codes in information theory: we need a large 'code rate', corresponding to requiring a large K as well as large distances between codewords, corresponding to requiring that $\|U_i - U_j\|_F$ be large. Standard approaches for constructing such collections run into difficulties. Getting a sufficiently tight concentration bound on the distance between two random unitary matrices is difficult as the matrix entries, by necessity, are correlated. On the other hand, starting with a large collection of random unit vectors and using its Cartesian product to build matrices does not necessarily yield unitary matrices.

We design a two-stage approach to decouple condition (iii) from (i) and (ii) by only generating sparse matrices U^{N_i}. See Fig. 2.3(b)-(d). In the first stage (Steps 1 & 2 in Fig. 2.3(b)-(c)), we only specify the non-zero positions (sparsity pattern) in each U^{N_i}. It suffices to guarantee that the sparsity patterns of the matrices U^{N_i} and U^{N_j} have little overlap. The existence of such objects can easily be proved using the probabilistic method. Thus, in the first stage, we can build up a large number of sparsity patterns. In the second stage (Step 3 in Fig. 2.3(d)), we carefully fill in values in the non-zero positions for each U^{N_i}. When the number of non-zero entries is not too small, satisfying the unitary constraint is feasible. As the overlap of sparsity patterns of any two matrices is small, we can argue the distance between them is large. By carefully trading off the number

of non-zero positions and the portion of overlap, we can simultaneously satisfy all three conditions.

2.4.1 Roadmap

This section describes the roadmap for executing the above idea.

Normalized form. Recall that $d_2 \le d_1$, we have

$$M\mathbf{x} = \underbrace{U^M}_{d_2\times d_2}\ \underbrace{\Sigma^M}_{d_2\times d_2}\ \underbrace{(V^M)^{\mathrm{T}}}_{d_2\times d_1}\ \underbrace{V^*}_{d_1\times d_1}\ \underbrace{(\Lambda^*)^{\frac{1}{2}}}_{d_1\times d_1}\ \underbrace{\mathbf{z}}_{d_1\times 1} \tag{2.34}$$

We may perform an SVD on $\Sigma^M(V^M)^{\mathrm{T}}V^*(\Lambda^*)^{\frac{1}{2}} = \underbrace{A}_{d_2\times d_2}\ \underbrace{L^\dagger}_{d_2\times d_2}\ \underbrace{B^{\mathrm{T}}}_{d_2\times d_1}$. We may also set $\mathbf{z}^\dagger = B^{\mathrm{T}}\mathbf{z}$, which is a standard multi-variate Gaussian in $\mathbf{R}^{d_2}$. Then we have

$$M\mathbf{x} = U^M A L^\dagger B^{\mathrm{T}}\mathbf{z} = (U^M A)L^\dagger \mathbf{z}^\dagger. \tag{2.35}$$

Let $N^\dagger = (U^M A)L^\dagger$. The SVD of $N^\dagger$ is exactly $(U^M)AL^\dagger I_{d_2\times d_2}$ because $U^M A$ is unitary. The *normalized form* of our problem is

$$\mathbf{y} = N^\dagger \mathbf{z}^\dagger + \epsilon. \tag{2.36}$$

Recall our local minimax has an oracle access interpretation. An algorithm with the oracle can reduce a problem into the normalized form on its own, and the algorithm knows $L^\dagger$. But the oracle still does not have any information on $N^\dagger$'s left singular vectors because being able to manipulate U^M is the same as being able to manipulate $U^M A$. Therefore, we can analyze the lower bound in normalized form, with the assumption that the SVD of $N^\dagger = U^\dagger L^\dagger I_{d_2\times d_2}$, in which $L^\dagger$ is known to the algorithm. We shall also let $\sigma_i^\dagger = L_{i,i}^\dagger = \sigma_i^{N^\dagger}$. Because $N^\dagger$ is square, we let $d = d_2$ in the analysis below.

We make two remarks in comparison to the orthogonalized form $\mathbf{y} = N\mathbf{z} + \epsilon$. (i) $\mathbf{z}^\dagger \in \mathbf{R}^{d_2}$, whereas $\mathbf{z} \in \mathbf{R}^{d_1}$. $\mathbf{z}^\dagger$'s dimension is smaller because the knowledge of Σ^M and

V^M enable us to remove the directions from $\mathbf{x}$ that are orthogonal to M's row space. *(ii)* $\sigma_i^N = \sigma_i^{N^\dagger}$ for $i \leq d_2$.

We rely on the following theorem (Chapter 2 in [150]) to construct the lower bound.

Theorem 2.4.5. *Let $\mathbb{N}^\dagger = \{N_1^\dagger, N_2^\dagger, \ldots, N_K^\dagger\}$, where $K \geq 2$. Let P_i be distribution of the training data produced from the model $\mathbf{y} = N_i^\dagger \mathbf{z}^\dagger + \epsilon$. Assume that*

- *$\|N_i^\dagger - N_j^\dagger\|_F^2 \geq 2s > 0$ for any $0 \leq j \leq k \leq K$.*
- *For any $j = 1, \ldots K$ and*

$$\frac{1}{K}\sum_{j=1}^{K} \mathrm{KL}(P_j, P_1) \leq \alpha \log K \tag{2.37}$$

with $0 \leq \alpha \leq \frac{1}{8}$.

Then

$$\inf_{\hat{N}^\dagger} \sup_{N^\dagger \in \mathbb{N}^\dagger} \Pr_{N^\dagger}(\|\hat{N}^\dagger, N^\dagger\| \geq s) \geq \frac{\sqrt{K}}{1+\sqrt{K}}(1 - 2\alpha - \sqrt{\frac{2\alpha}{\log K}}). \tag{2.38}$$

Because $L^\dagger$ is fixed, this problem boils down to finding a collection $\mathbb{U}^\dagger$ of unitary matrices in $\mathbf{R}^{d\times d}$ such that any two elements in $\mathbb{U}^\dagger$ are sufficiently far. Then we can construct $\mathbb{N}^\dagger = \{U^\dagger L^\dagger : U^\dagger \in \mathbb{U}\}$.

We next reiterate (with elaboration) the challenge we face when using existing techniques. Then we describe our approach. We shall first examine a simple case, in which we need only design vectors for one column. Then we explain the difficulties of using an existing technique to generalize the design. Finally, we explain our solution.

Recall that $(\mathbf{Z}^\dagger)^{\mathrm{T}}\mathbf{Y} = (N^\dagger)^{\mathrm{T}} + \mathcal{E}$, where we can roughly view $\mathcal{E}$ as a matrix that consists of independent Gaussian $(0, \sigma_\epsilon/\sqrt{n})$. For the sake of discussion, we assume $\sigma_\epsilon = 1$ in our discussion below.

Warmup: one colume case. The problem of packing one column roughly corresponds to establishing a lower bound on the estimation problem $\mathbf{y} = \mathbf{u} + \epsilon$, where $\mathbf{y}, \mathbf{u}, \epsilon \in \mathbf{R}^d$. ϵ corresponds to a column in $\mathcal{E}$ and consists of d independent Gaussian $N(0, 1/\sqrt{n})$. $\mathbf{u}$ correpsonds to a column in $U^\dagger$ and we require $\|\mathbf{u}\|_2 \approx \rho\sqrt{d/n}$. To apply Theorem 2.4.5,

we shall construct a $\mathbb{D} = \{\mathbf{u}_1, \ldots, \mathbf{u}_K\}$ such that $\|\mathbf{u}_i - \mathbf{u}_j\|$ is large for each $\{i, j\}$ pair and K is also large. Specifically, we require $\|\mathbf{u}_i - \mathbf{u}_j\|_2^2 \approx 2\rho^2 \frac{d}{n}$ (large distance requirement) and $K = \exp(\Theta(\sqrt{\rho}d))$ (large set requirement). $\sqrt{\rho}$ is carefully optimized and we will defer the reasoning to the full analysis below. A standard tool to construct $\mathbb{D}$ is to use a probabilistic method. We sample $\mathbf{u}_i$ independently from the same distribution and argue that with high probability $\|\mathbf{u}_i - \mathbf{u}_j\|_2^2$ is sufficiently large. Then a union bound can be used to derive K. For example, we may set $\mathbf{u}_i \sim N(0, \rho\sqrt{\frac{d}{n}} I_{d\times d})$ for all i, and the concentration quality suffices for us to sample K vectors.

Multiple column case. We now move to the problem of packing multiple columns in U together. K is required to be much larger, e.g., $K = \exp(\Theta(\sqrt{\rho}d^2))$ for certain problem instances. A natural generalization of one-column case is to build our parameter set by taking the Cartesian product of multiple copies of $\mathbb{D}$. This gives us a large K for free but the key issue is that vectors in $\mathbb{D}$ are generated independently. So there is no way to guarantee they are independent to each other. In fact, it is straightforward to show that many elements in the Cartesian product are far from unitary. One may also directly sample random unitary matrices and argue that they are far from each other. But there seems to exist no tool that enables us to build up a concentration of $\exp(-\Theta(\sqrt{\rho}d^2))$ between two random unitary matrices.

Therefore, a fundamental problem is that we need independence to build up a large set but the unitary constraint limits the independence. So we either cannot satisfy the unitary requirement (Cartesian product approach) or cannot properly use the independence to build concentrations (random unitary matrix approach).

Our approach. We develop a technique that decouples the three requirements (unitary matrices, large distance, and large K). Let us re-examine the Cartesian product approach. When the vectors for each column are completely determined, then it is remarkably difficult to build a Cartesian product that guarantees orthogonality between columns. To address this issue, our approach only "partially" specify vectors in each column. Then we take

a Cartesian product of these partial specifications. So the size of the Cartesian product is sufficiently large; meanwhile an element in the product does not fully specify $U^\dagger$ so we still have room to make them unitary. Thus, our final step is to transform each partial specification in the Cartesian product into a unitary matrix. Specifically, it consists of three steps (See Fig. 2.3)

Step 1. Partial specification of each column. For each column i of interest, we build up a collection $\mathbb{D}^{(i)} = \{R^{(i,1)}, \ldots, R^{(i,K)}\}$. Each $R^{(i,j)} \subset [d]$ specifies only the positions of non-zero entire for a vector prepared for filling in $U_{:,i}$.

Step 2. Cartesian product. Then we randomly sample elements from the Cartesian product $\mathbb{D} \triangleq \bigotimes_i \mathbb{D}^{(i)}$. Each element in the product specifies the non-zero entries of $U^\dagger$. We need to do another random sampling instead of using the full Cartesian product because we need to guarantee that any two matrices share a small number of non-zero entries. For example, $(R^{(1,1)}, R^{(2,1)}, R^{(3,1)})$ and $(R^{(1,1)}, R^{(2,1)}, R^{(3,2)})$ are two elements in $\bigotimes_i \mathbb{D}^{(i)}$ but they specify two matrices with the same locations of non-zero entries for the first two columns.

Step 3. Building up unitary matrices. Finally, for each element $\vec{R} \in \mathbb{D}$ (that specify positions of non-zero entries), we carefully fill in the values of the non-zero entries so that all our matrices are unitary and far from each other. We shall show that it is *always* possible to construct unitary matrices that "comply with" $\vec{R}$. In addition, our unitary matrices have few entries with large magnitude so when two matrices share few positions of non-zero entries, they are far.

2.4.2 Analysis

We now execute the plan outlined in the roadmap. Let K and λ be tunable parameters. For the purpose of getting intuition, K is related to the size of $\mathbb{U}^\dagger$ so it can be thought as being exponential in d, whereas λ is a constant and we use ρ^λ to control the density of non-zero entries in each $U^\dagger \in \mathbb{U}^\dagger$.

Let $\mathbb{D}^{(i)} = \{R^{(i,1)}, R^{(i,2)}, \cdots, R^{(i,K)}\}$ be a collection of random subsets in $[d]$, in which

each $R^{(i,j)}$ is of size $\rho^\lambda d$. We sample the subsets without replacement. Recall that $\underline{t}$ is the smallest index such that $\sigma^\dagger_{\underline{t}} \le \rho\sigma_\epsilon\sqrt{\frac{d}{n}}$ and let us also define $\overline{t} = \lfloor \frac{\rho^\lambda d}{2} \rfloor$. We let $\gamma = \overline{t} - \underline{t} + 1$. We assume that $\overline{t} \ge \underline{t}$; otherwise Proposition 2.4.2 becomes trivial. Let $\mathbb{D}$ be the Cartesian product of $\mathbb{D}^{(i)}$ for integers $i \in [\underline{t}, \overline{t}]$. We use $\vec{R}$ to denote an element in $\mathbb{D}$. $\vec{R} = (\vec{R}_{\underline{t}}, \vec{R}_{\underline{t}+1}, \cdots, \vec{R}_{\overline{t}})$ is a γ-tuple so that each element $\vec{R}_i$ corresponds to an element in $\mathbb{D}^{(i)}$. There are two ways to represent $\vec{R}$. Both are found useful in our analysis.

1. *Set representation.* We treat $\vec{R}_i$ as a set in $\mathbb{D}^{(i)}$.
2. *Index representation.* We treat $\vec{R}_i$ as an index from $[K]$ that specifies the index of the set that $\vec{R}_i$ refers to.

Note that the subscript i of $\vec{R}_i$ *starts at* $\underline{t}$ (instead of 1 or 0) for convenience.

Example 2.4.1. The index of $\mathbb{D}_i$ starts at $\underline{t}$. Assume that $\overline{t} = \underline{t} + 1$. Let $\mathbb{D}^{(\underline{t})} = \big(\{2,3\},\{1,4\},\{1,2\}\big)$ and $\mathbb{D}^{(\underline{t}+1)} = \big(\{1,3\},\{2,4\},\{3,4\}\big)$. The element $\big(\{1,2\},\{2,4\}\big) \in \mathbb{D}^{(\underline{t})} \otimes \mathbb{D}^{(\underline{t}+1)}$. There are two ways to represent this element. *(i) Set representation.* $\vec{R} = \big(\{1,2\},\{2,4\}\big)$, in which $\vec{R}_{\underline{t}} = \{1,2\}$ and $\vec{R}_{\underline{t}+1} = \{2,4\}$. *(ii) Index representation.* $\vec{R} = (3,2)$. $\vec{R}_{\underline{t}} = 3$ refers to that the third element $\{1,2\}$ in $\mathbb{D}^{(\underline{t})}$ is selected.

We now describe our proof in detail. We remark that throughout our analysis, constants are re-used in different proofs.

2.4.2.1 Step 1. Partial specification for each column

This step needs only characterize the behavior of an individual $\mathbb{D}^{(i)}$.

Lemma 2.4.6. *Let $\rho < 1$ be a sufficiently small variable, λ be a tunable parameter and let $\mathbb{D}^{(i)} = \{R^{(i,1)}, R^{(i,2)}, \cdots, R^{(i,K)}\}$ be a collection of random subsets in $[d]$ (sampled without replacement) such that $|R^{(i,j)}| = \rho^\lambda d$ for all j. There exist constants c_0, c_1, and c_2 such that when $K = \exp(c_0\rho^{2\lambda}d)$, with probability $1 - \exp(-c_1\rho^{2\lambda}d)$, for any two distinct $R^{(i)}$ and $R^{(j)}$, $|R^{(i,j)} \cap R^{(i,k)}| \le c_2\rho^{2\lambda}d$.*

Proof of Lemma 2.4.6. This can be proved by a standard probabilistic argument. Let $R^{(i,j)}$ be an arbitrary subset such that $|R^{(i,j)}| = \rho^\lambda d$. Let $R^{(i,k)}$ be a random subset of size

$\rho^\lambda d$. We compute the probability that $|R^{(i,j)} \cap R^{(i,k)}| \geq c_2 \rho^{2\lambda} d$ for a fixed $R^{(i,j)}$.

Let us sequentially sample elements from $R^{(i,k)}$ (without replacement). Let I_t be an indicator random variable that sets to 1 if and only if the t-th random element in $R^{(i,k)}$ hits an element in $R^{(i,j)}$. We have

$$\Pr[I_t = 1] \leq \frac{\rho^\lambda d}{(1-\rho^\lambda)d} \leq \frac{\rho^\lambda}{2}. \tag{2.39}$$

By using a Chernoff bound, we have

$$\Pr\left[|\sum_{i=1}^{\rho^\lambda d} I_t| \geq \frac{c_2 \rho^{2\lambda} d}{2} \right] \leq \exp(-\Omega(\rho^{2\lambda} d)). \tag{2.40}$$

By using a union bound, we have

$$\Pr[\exists i,j : |R^{(i,j)} \cap R^{(i,k)}| \geq \frac{c_2 \rho^{2\lambda} d}{2}] \leq \binom{K}{2} \exp(-\Omega(\rho^{2\lambda} d)) \leq \exp(-\Omega(\rho^{2\lambda} d) + 2\log K). \tag{2.41}$$

Therefore, when we set $K = \exp(c_0 \rho^{2\lambda} d)$, the failure probability is $1 - \exp(-\Theta(\rho^{2\lambda} d))$. □

2.4.2.2 Step 2. Random samples from the Cartesian product

We let $\mathbb{D} = \bigotimes_{i \in [\underline{t}, \bar{t}]} \mathbb{D}^{(i)}$. Note that each $\mathbb{D}^{(i)}$ is sampled independently. We define $\mathbb{S}$ be a random subset of $\mathbb{D}$. We next explain the reason we need to sample random subsets. Recall that for each $\vec{R} \in \mathbb{S}$, we aim to construct a unitary matrix $U^\dagger$ (to be discussed in Step 3) such that the positions of non-zero entries in $U^\dagger_{:,i}$ are specified by $\vec{R}_i$ (i.e., $U^\dagger_{j,i} \neq 0$ only when $j \in \vec{R}_i$).

Let $\vec{R}$ and $\vec{R}'$ be two distinct elements in $\mathbb{D}$. Let $U^\dagger$ and $\tilde{U}^\dagger$ be two unitary matrices generated by $\vec{R}$ and $\vec{R}'$. We ultimately aim to have that $\|U^\dagger L^\dagger - \tilde{U}^\dagger L^\dagger\|_F^2$ being large. We shall make sure *(i)* $U^\dagger$ and $\tilde{U}^\dagger$ share few non-zero positions (Step 2; this step), and *(ii)* few entries in $U^\dagger$ and $\tilde{U}^\dagger$ have excessive magnitude (Step 3). These two conditions will imply that $U^\dagger$ and $\tilde{U}^\dagger$ are far, which then implies a lower bound on $U^\dagger L^\dagger$ and $\tilde{U}^\dagger L^\dagger$.

Because we do not want $U^\dagger$ and $\tilde{U}^\dagger$ share non-zero positions, we want to maximize the

Hamming distance (in index representation) between $\vec{R}$ and $\vec{R}'$ (i.e., $\vec{R}_i = \vec{R}'_i$ implies $U^\dagger_{:,i}$ and $\tilde{U}^\dagger_{:,i}$ share all non-zero positions, which is a bad event). We sample $\mathbb{S}$ randomly from $\mathbb{D}$ because random sample is a known procedure that generates "code" with large Hamming distance [106].

Before proceeding, we note one catch in the above explanation, i.e., different columns are of different importance. Specifically, $\|U^\dagger L^\dagger - \tilde{U}^\dagger L^\dagger\|_F^2 = \sum_{i\in[d]} (\sigma_i^\dagger)^2 \|U^\dagger_{:,i} - \tilde{U}^\dagger_{:,i}\|^2$. When $\vec{R}_i$ and $\vec{R}'_i$ collide for a large $(\sigma_i^\dagger)^2$, it makes more impact to the Frobenius norm. Thus, we define a weighted cost function that resembles the structure of Hamming distance.

$$\mathbf{c}(\vec{R}, \vec{R}') = \sum_{i\in[\underline{t},\bar{t}]} (\sigma_i^\dagger)^2 I(\vec{R}_i = \vec{R}'_i). \tag{2.42}$$

Note that the direction we need for $\mathbf{c}(\vec{R}, \vec{R}')$ is opposite to Hamming distance. One usually maximizes Hamming distance whereas we need to minimize the weighted cost.

We need to develop a specialized technique to produce concentration behavior for $\mathbb{S}$ because $\mathbf{c}(\vec{R}, \vec{R}')$ is weighted.

Lemma 2.4.7. *Let ρ and λ be the parameters for producting $\mathbb{D}^{(i)}$. Let $\zeta < 1$ be a tunable parameter. Let $\mathbb{S}$ be a random subset of $\mathbb{D}$ of $\mathbb{D} \triangleq \bigotimes_{i\in[\underline{t},\bar{t}]} \mathbb{D}^{(i)}$ such that*

$$|\mathbb{S}| = \exp\left(c_3 \frac{n\rho^{2\lambda+\zeta}}{\rho^2\sigma_\epsilon^2} \left(\sum_{i\in[\underline{t},\bar{t}]} (\sigma_i^\dagger)^2\right)\right) \tag{2.43}$$

for some constant c_3. With high probability at least $1 - \exp(-c_4\rho^{2\lambda}d)$ (c_4 a constant), for any $\vec{R}$ and $\vec{R}'$ in $\mathbb{S}$, $\mathbf{c}(\vec{R}, \vec{R}') \le \rho^\zeta(\sum_{i\in[\underline{t},\bar{t}]} (\sigma_i^\dagger)^2)$.

Proof. Let $\Psi = \sum_{i\in[\underline{t},\bar{t}]} (\sigma_i^\dagger)^2$. Let $\vec{R}$ and $\vec{R}'$ be two different random elements in $\mathbb{D}$. We shall first compute that $\Pr\left[\mathbf{c}(\vec{R}, \vec{R}') \ge \rho^\zeta \Psi\right]$. Here, we assume that $\vec{R}$ is an arbitrary fixed element and $\vec{R}'$ is random.

Recall that $\sigma_i^\dagger \in [0, \rho\sigma_\epsilon\sqrt{\frac{n}{d}}]$ for $i \in [\underline{t}, \bar{t}]$. We shall partition $[0, \rho\sigma_\epsilon\sqrt{\frac{n}{d}}]$ into subintervals and group $\sigma_i^\dagger$ by these intervals. Let $\mathcal{I}_t$ be the set of $\sigma_i^\dagger$ that are in $[2^{-t-1}\rho\sigma_\epsilon\sqrt{\frac{n}{d}}, 2^{-t}\rho\sigma_\epsilon\sqrt{\frac{n}{d}}]$

($t \geq 0$). Let T be the largest integer such that $\mathcal{I}_T$ is not empty. Let $L_t = |\mathcal{I}_t|$ and $\ell_t = \sum_{i \in \mathcal{I}_t} I(\vec{R}_i = \vec{R}'_i)$. We call $\{\ell_t\}_{t \leq T}$ the *overlapping coefficients* between $\vec{R}$ and $\vec{R}'$.

Note that $\mathbf{c}(\vec{R}, \vec{R}') \leq \sum_{t \leq T} \ell_t 2^{-2t} \rho^2 \sigma_\epsilon^2 d/n$. Therefore, a necessary condition for $\mathbf{c}(\vec{R}, \vec{R}') \geq \rho^\zeta \Psi$ is $\sum_{t \leq T} \frac{\ell_t 2^{-2t} \rho^2 \sigma_\epsilon^2 d}{n} \geq \rho^\eta \Psi$. Together with the requirement that $\sum_{t \leq T} \ell_t \geq 1$, we need

$$\sum_{t \leq T} \ell_t \geq \max \left\{ \frac{n \rho^\zeta \Psi}{d \rho^2 \sigma_\epsilon^2}, 1 \right\} \tag{2.44}$$

Recall that we assume that $\vec{R}$ is fixed and $\vec{R}'$ is random. When $\mathbf{c}(\vec{R}, \vec{R}') \geq \rho^\zeta \Psi$, we say $\vec{R}'$ is bad. We next partition all bad $\vec{R}'$ into sets indexed by $\{\ell_t\}_{t \leq T}$. Let $\mathbb{C}(\{\ell_t\}_{t \leq T})$ be all the bad $\vec{R}'$ such that the overlapping coefficients between $\vec{R}$ and $\vec{R}'$ are $\{\ell_t\}_{t \leq T}$. We have

$$\Pr[\mathbf{c}(\vec{R}, \vec{R}') \geq \rho^\zeta \Psi] = \Pr[\vec{R}' \text{ is bad }] = \Pr\left[\vec{R}' \in \bigcup_{\{\ell_t\}_{t \leq T}} \mathbb{C}(\{\ell_t\}_{t \leq T}) \right] = \sum_{k \geq 1} \sum_{\substack{\text{all } \mathbb{C}(\{\ell_i\}) \\ \text{s.t. } \sum_t \ell_t = k}} \Pr[\vec{R}' \in \mathbb{C}(\{\ell_t\}_{t \leq T})]$$

Next also note that

$$\Pr[\vec{R}' \in \mathbb{C}(\{\ell_t\}_{t \leq T})] \leq \prod_{t \leq T} \binom{L_t}{\ell_t} \left(\frac{1}{K} \right)^{\sum_{t \leq T} \ell_t},$$

where K is the size of each $\mathbb{D}^{(i)}$.

The number of possible $\{\ell_i\}_{t\le T}$ such that $\sum_{t\le T}\ell_i = k$ is at most $\binom{d+k}{k}$. Therefore,

$$
\begin{aligned}
&\sum_{k\ge 1}\sum_{\substack{\text{all } \mathbb{C}(\{\ell_i\})\\ \text{s.t. } \sum_t \ell_t = k}} \Pr\left[\vec{R}' \in \mathbb{C}(\{\ell_i\}_{i\le T})\right]\\
&\le \sum_{k\ge 1}\sum_{\substack{\text{all } \mathbb{C}(\{\ell_i\})\\ \text{s.t. } \sum_t \ell_t = k}} \prod_{t\le T}\binom{L_t}{\ell_t}\left(\frac{1}{K}\right)^{\ell_t}\\
&\le \sum_{k\ge 1}\sum_{\substack{\text{all } \mathbb{C}(\{\ell_i\})\\ \text{s.t. } \sum_t \ell_t = k}} \prod_{t\le T}\left(\frac{eL_t}{K}\right)^{\ell_t} \quad (\text{using } \tbinom{L_t}{\ell_t} \le \left(\tfrac{eL_t}{\ell_t}\right)^{\ell_t} \le (eL_i)^{\ell_i})\\
&\le \sum_{k\ge 1}\binom{d+k}{k}\prod_{t\le T}\left(\frac{eL_t}{K}\right)^{\ell_t}\\
&\le d\max_{\substack{k \text{ s.t.}\\ \sum_t \ell_t = k}}\binom{d+k}{k}\prod_{i\le T}\left(\frac{eL_i}{K}\right)^{\ell_t}\\
&\le d\left(\frac{e(d+k)}{k}\right)^{k}\prod_{t\le T}\left(\frac{eL_t}{K}\right)^{\ell_t} \quad (\text{where } k = \textstyle\sum_{t\le T}\ell_t \text{ from the previous line})\\
&\le d(2ed)^k\prod_{t\le T}\left(\frac{eL_t}{K}\right)^{\ell_t}\\
&\le d\prod_{t\le T}\left(\frac{2e^2d^2}{K}\right)^{\ell_t} \le \exp\left(-c\rho^{2\lambda}d(\sum_{t\le T}\ell_t)\right) \quad (c \text{ is a suitable constant;})\\
&\le \exp\left(-c\rho^{2\lambda}d\max\left\{\frac{n\rho^{\zeta}\Psi}{d\rho^2\sigma_\epsilon^2}, 1\right\}\right)
\end{aligned}
$$

By using a union bound on all pairs of $\vec{R}$ and $\vec{R}'$ in $\mathbb{S}$, we have for sufficiently small c_3, there exists a c_4 such that

$$\Pr\left[\exists \vec{R}, \vec{R}' \in \mathbb{S} : \mathbf{c}(\vec{R}, \vec{R}') \ge \rho^{\zeta}\Psi\right] \le \exp\left(-c_4\rho^{2\lambda}d\max\left\{\frac{n\rho^{\zeta}\Psi}{d\rho^2\sigma_\epsilon^2}, 1\right\}\right) \le \exp(-c_4\rho^{2\lambda}d).$$

□

2.4.2.3 Step 3. Building up unitary matrices

We next construct a set of unitary matrices in $\mathbf{R}^{d\times d}$ based on elements in $\mathbb{S}$. We shall refer to the procedure to generate a unitary U from an element $\vec{R} \in \mathbb{S}$ as $q(\vec{R})$.

Procedure $q(\vec{R})$: Let $U \in \mathbf{R}^{d\times d}$ be the matrix $q(\vec{R})$ aims to fill in.

Let $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(\underline{t}-1)} \in \mathbf{R}^d$ be an arbitrary set of orthonormal vectors. $q(\vec{R})$ partitions the matrix U into three regions of columns and it fills different regions with different strategies. See also three regions illustrated by matrices in Fig. 2.3.

Region 1: $U_{:,i}$ for $i < \underline{t}$. We set $U_{:,i} = \mathbf{v}^{(i)}$ when $i < \underline{t}$. This means all the unitary matrices we construct share the first $\underline{t} - 1$ columns.

Region 2: $U_{:,i}$ for $i \in [\underline{t}, \bar{t}]$. We next fill in non-zero entries of each $U_{:,i}$ by $\vec{R}_i$ for $i \in [\underline{t}, \bar{t}]$. We fill in each $U_{:,i}$ sequentially from left to right (from small i to large i). We need to make sure that *(i)* it is feasible to construct $U_{:,i}$ that is orthogonal to all $U_{:,j}$ $(j < i)$ by using only entries specified by R_i. *(ii)* there is a way to fill in $U_{:,i}$ so that not too many entries are excessively large. (ii) is needed because for any $\vec{R}$ and $\vec{R}'$, $\vec{R}_i$ and $\vec{R}'_i$ still share a small number of non-zero positions (whp $|\vec{R}_i \cap \vec{R}'_i| = O(\rho^{2\lambda} d)$, according to Lemma 2.4.6). When the mass in $|\vec{R}_i \cap \vec{R}'_i|$ is large, the distance between U and U' is harder to control.

Region 3: $U_{:,i}$ for $i > \bar{t}$. $q(\vec{R})$ fills in the rest of the vectors arbitrarily so long as U is unitary. Unlike the first $\underline{t} - 1$ columns, these columns depend on $\vec{R}$ so each $U \in \mathbb{U}$ has a different set of ending column vectors.

Analysis for region 2. Our analysis focuses on two asserted properties for Region 2 are true. We first show (i) is true and describe a procedure to make sure (ii) happens.

For $j \le i - 1$, let $\mathbf{w}^{(j)} \in \mathbf{R}^{\rho^\lambda d}$ be the projection of $U_{:,j}$ onto the coordinates specified by $\vec{R}_i$. See Fig. 2.3(d) for an illustration. Note that $\bar{t} = \frac{\rho^\lambda d}{2}$, the dimension of the subspace spanned by $\mathbf{w}^{(1)}, \ldots \mathbf{w}^{(i-1)}$ is at most $\rho^\lambda d/2$. Therefore, we can find a set of orthonormal vectors $\{\mathbf{u}^{(1)}, \ldots \mathbf{u}^{(\kappa)}\} \subseteq \mathbf{R}^{\rho^\lambda d}$ $(\kappa \ge \frac{\rho^\lambda d}{2})$ that are orthogonal to $\mathbf{w}^{(j)}$ $(j \le i - 1)$. To build a $U_{:,i}$ that's orthogonal to all $U_{:,j}$ $(j \le i - 1)$, we can first find a $\mathbf{u} \in \mathbf{R}^{\rho^\lambda d}$ that is a linear combination of $\{\mathbf{u}^{(j)}\}_{j\le\kappa}$, and then "inflate" $\mathbf{u}$ back to $\mathbf{R}^d$, i.e., the k-th non-zero

coordinate of $U_{:,i}$ is $\mathbf{u}_k$. One can see that

$$\langle U_{:,j}, U_{:,i} \rangle = \langle \mathbf{w}^{(j)}, \mathbf{u} \rangle = 0$$

for any $j < i$.

We now make sure (ii) happens. We have the following Lemma.

Lemma 2.4.8. *Let $\{\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(\kappa)}\}$ ($\kappa \geq \rho^\lambda d/2$) be a collection of orthonormal vectors in $\mathbf{R}^{\rho^\lambda d}$. Let η be a small tunable parameter. There exists a set of coefficients $\beta_1, \ldots, \beta_\kappa$ such that $\mathbf{u} = \sum_{i=1}^{\kappa} \beta_i \mathbf{u}^{(i)}$ is a unit vector and there exist constant c_5, c_6, and c_7 such that*

$$\sum_{i \leq \rho^\lambda d} \mathbf{u}_i^2 I\left(\mathbf{u}_i \geq \frac{c_5}{\sqrt{\rho^{\lambda+\eta} d}}\right) \leq c_7 \varepsilon, \tag{2.45}$$

where $\varepsilon = \exp(-\frac{c_6}{\rho^\eta})$.

Proof of Lemma 2.4.8. We use a probabilistic method to find $\mathbf{u}$. Let $z_i \sim N(0, 1/\sqrt{\rho^\lambda d})$ for $i \in [\rho^\lambda d]$. Let $S = \sqrt{\sum_{i \leq \rho^\lambda d} z_i^2}$. We shall set $\beta_i = z_i/S$. One can check that $\mathbf{u} = \sum_{i \in [\rho^\lambda d]} \beta_i \mathbf{u}^{(i)}$ is a unit vector. We then examine whether (2.45) is satisfied for these β_i's we created. If not, we re-generate a new set of z_i's and β_i's. We repeat this process until (2.45) is satisfied.

Because β_i's are normalized, setting the standard deviation of z_i is unnecessary. We nevertheless do so because S will be approximately a constant, which helps us simplify the calculation.

We claim that there exists a constant c such that for any ℓ,

$$\mathbb{E}\left[\mathbf{u}_\ell^2 I\left(\mathbf{u}_\ell \geq \frac{c_5}{\sqrt{\rho^{\lambda+\eta} d}}\right)\right] \leq \frac{c\varepsilon}{\rho^\lambda d}. \tag{2.46}$$

We first show that (2.46) implies Lemma 2.4.8. Then we will show (2.46).

By linearity of expectation, (2.46) implies

$$\mathbb{E}\left[\sum_{\ell\le\rho^{\lambda}d}\mathbf{u}_{\ell}^{2}I\left(\mathbf{u}_{\ell}\ge\frac{c_{5}}{\sqrt{\rho^{\lambda+\eta}d}}\right)\right]\le c\varepsilon.$$

Then we use a Markov inequality and obtain

$$\Pr\left[\left(\sum_{\ell\le\rho^{\lambda}d}\mathbf{u}_{\ell}^{2}I\left(\mathbf{u}_{\ell}\ge\frac{c_{5}}{\sqrt{\rho^{\lambda+\eta}d}}\right)\right)\ge 2c\varepsilon\right]\le\frac{1}{2}$$

So our probabilistic method described above is guaranteed to find a $\mathbf{u}$ that satisfies (2.45)

We next move to showing (2.46). Recall that

$$\mathbf{u}_{\ell}=\frac{1}{S}(\sum_{i\le\kappa}z_{i}\mathbf{u}_{\ell}^{(i)}).$$

We also let $Z_{\ell}=\sum_{i\le\kappa}z_{i}\mathbf{u}_{\ell}^{(i)}$. We can see that Z_{ℓ} is a Gaussian random variable with a standard deviation $\sqrt{\frac{1}{\rho^{\lambda}d}\sum_{i\le\kappa}(\mathbf{u}_{\ell}^{(i)})^{2}}\le\sqrt{\frac{1}{\rho^{\lambda}d}}$. The inequality uses the fact that $\mathbf{u}^{(i)}$'s are orthonormal to each other and therefore $\sum_{i\le\kappa}(\mathbf{u}_{\ell}^{(i)})^{2}\le 1$.

On the other hand, one can see that

$$\mathbb{E}[S]=\mathbb{E}[\sum_{i\le\kappa}z_{i}^{2}]=\left(\frac{1}{\sqrt{\rho^{\lambda}d}}\right)^{2}\cdot\kappa\ge\frac{1}{2}.$$

Therefore, by a standard Chernoff bound, $\Pr\left[S\le\frac{1}{4}\right]\le\exp(-\Theta(\kappa))$.

Next, because $\{\mathbf{z}_{i}\}_{i\le\kappa}$ collectively form a spherical distribution, we have $\left\{\frac{z_{i}}{S}\right\}_{i\le\kappa}$ is independent to S. Therefore,

$$\mathbb{E}\left[\mathbf{u}_{\ell}^{2}I\left(\mathbf{u}_{\ell}\ge\frac{c_{5}}{\sqrt{\rho^{\lambda+\eta}d}}\right)\right]=\mathbb{E}\left[\mathbf{u}_{\ell}^{2}I\left(\mathbf{u}_{\ell}\ge\frac{c_{5}}{\sqrt{\rho^{\lambda+\eta}d}}\right)\mid S\ge\frac{1}{4}\right]\tag{2.47}$$

Conditioned on $S \geq \frac{1}{4}$, we use the fact that $\mathbf{u}_\ell = Z_\ell / S$ to get $\mathbf{u}_\ell \leq 4Z_\ell$ and

$$I\left(\mathbf{u}_\ell \geq \frac{c_5}{\sqrt{\rho^{\lambda+\eta}d}}\right) = I\left(Z_\ell \geq \frac{c_5}{\sqrt{\rho^{\lambda+\eta}d}}S\right) \leq I\left(Z_\ell \geq \frac{c_5}{4\sqrt{\rho^{\lambda+\eta}d}}\right). \tag{2.48}$$

Therefore,

(2.47)

$$\begin{aligned}
&\leq 16\,\mathbb{E}\left[Z_\ell^2 I\left(Z_\ell \geq \frac{c_5}{4\sqrt{\rho^{\lambda+\eta}d}}\right) \mid S \geq \frac{1}{4}\right] \\
&= \frac{16}{\Pr\left[S \geq \frac{1}{4}\right]}\left(\mathbb{E}\left[Z_\ell^2 I\left(Z_\ell \geq \frac{c_5}{4\sqrt{\rho^{\lambda+\eta}d}}\right)\right] - \mathbb{E}\left[Z_\ell^2 I\left(Z_\ell \geq \frac{c_5}{4\sqrt{\rho^{\lambda+\eta}d}}\right) \mid S < \frac{1}{4}\right]\Pr\left[S \leq \frac{1}{4}\right]\right) \\
&\leq 16(1+\exp(-\Theta(\kappa)))\left(\mathbb{E}\left[Z_\ell^2 I\left(Z_\ell \geq \frac{c_5}{4\sqrt{\rho^{\lambda+\eta}d}}\right)\right]\right) \\
&\leq 16(1+\exp(-\Theta(\kappa)))\,\mathbb{E}\left[Z_\ell^2 I\left(Z_\ell \geq \frac{2c_5}{\sqrt{\rho^{\lambda+\eta}d}}\right)\right] \\
&\leq \frac{16(1+\exp(-\Theta(\kappa)))}{\sqrt{2\pi}\sigma_{Z_\ell}}\int_{\frac{2c_5}{\sqrt{\rho^{\lambda+\eta}d}}}^{\infty} z^2 \exp\left(\frac{z^2}{\sigma_{Z_\ell}^2}\right) dz \\
&\leq \frac{c}{\rho^{\lambda+\eta}d}\exp\left(-\Theta(\rho^{-\eta})\right) \quad (\text{using } \sigma_{Z_\ell} \leq \sqrt{\tfrac{1}{\rho^\lambda d}}) \\
&\leq O(\frac{\varepsilon}{\rho^\lambda d})
\end{aligned}$$

□

2.4.2.4 Proof of Proposition 2.4.2

Now we are ready to use Theorem 2.4.5 to prove Proposition 2.4.2. Let $\mathbb{S}$ be the set constructed from Step 1 and $\mathbb{U}^\dagger = \{U^\dagger : U^\dagger = q(\vec{R}), \vec{R} \in \mathbb{S}\}$. Let $\mathbb{N}^\dagger = \{U^\dagger L^\dagger : U^\dagger \in \mathbb{U}^\dagger\}$. Let also $\mathbf{P}_{N^\dagger}$ be the distribution of $(\mathbf{y}, \mathbf{z}^\dagger)$ generated from the model $\mathbf{y} = N^\dagger \mathbf{z}^\dagger + \epsilon$. Let $\mathbf{P}_{N^\dagger|\mathbf{z}^\dagger}$ be the distribution of $\mathbf{y}$ from the normalized model when $\mathbf{z}^\dagger$ is given. Let $\mathbf{P}_{N^\dagger,n}$ be the product distribution when we observe n samples from the model $\mathbf{y} = N^\dagger \mathbf{z}^\dagger + \epsilon$. Let $f_{N^\dagger}(\mathbf{y}, \mathbf{z}^\dagger)$ be the pdf of for $\mathbf{P}_{N^\dagger}$, $f_{N^\dagger}(\mathbf{y} \mid \mathbf{z}^\dagger)$ be the pdf of $\mathbf{y}$ given $\mathbf{z}^\dagger$, and $f(\mathbf{z}^\dagger)$ be the

pdf of $\mathbf{z}^\dagger$.

We need to show that for any $N^\dagger, \tilde{N}^\dagger \in \mathbb{N}$ *(i)* $\|N^\dagger - \tilde{N}^\dagger\|_F$ is large, and *(ii)* the KL-divergence between $\mathbf{P}_{N^\dagger,n}$ and $\mathbf{P}_{N^\dagger,n}$ is bounded.

Lemma 2.4.9. *Let $\mathbb{S}$, $\mathbb{U}^\dagger$, and $\mathbb{N}^\dagger$ be generated by the parameters λ, η, and ζ (see Steps 1 to 3). For any $N^\dagger$ and $\tilde{N}^\dagger$ in $\mathbb{N}$, we have*

$$\|N^\dagger - \tilde{N}^\dagger\|_F^2 \geq \sum_{i\in[\underline{t},\overline{t}]} \sigma_i^\dagger (2 - c_8\rho^{\lambda-\eta} - c_9\rho^\zeta) \tag{2.49}$$

for some constant c_8 and c_9.

Proof. Let $U^\dagger, \tilde{U}^\dagger \in \mathbb{U}^\dagger$ such that $N^\dagger = U^\dagger L^\dagger$ and $\tilde{N}^\dagger = \tilde{U}^\dagger L^\dagger$; let $\vec{R}, \vec{R}' \in \mathbb{S}$ be that $U^\dagger = q(\vec{R})$ and $\tilde{U}^\dagger = q(\vec{R}')$. Also, recall that we let $\Psi = \sum_{i\in[\underline{t},\overline{t}]}(\sigma_i^\dagger)^2$.

We have

$$\|N^\dagger - \tilde{N}^\dagger\|_F^2 = \sum_{i\in[d]} \|(U_{:,i}^\dagger - \tilde{U}_{:,i}^\dagger)\sigma_i^\dagger\|_2^2 \geq \sum_{i\in[\underline{t},\overline{t}]} \|(U_{:,i}^\dagger - \tilde{U}_{:,i}^\dagger)\sigma_i^\dagger\|_2^2$$

Let also $H = \{\vec{R}_i = \vec{R}'_i, i \in [\underline{t},\overline{t}]\}$, i.e., the set of coordinates that $\vec{R}$ and $\vec{R}'$ agree. Whp, we have

$$\Psi = \sum_{i\in H}(\sigma_i^\dagger)^2 + \sum_{\substack{i\in[\underline{t},\overline{t}]\\ i\notin H}}(\sigma_i^\dagger)^2 = \mathbf{c}(\vec{R},\vec{R}') + \sum_{\substack{i\in[\underline{t},\overline{t}]\\ i\notin H}}(\sigma_i^\dagger)^2 \leq \rho^\zeta\Psi + \sum_{\substack{i\in[\underline{t},\overline{t}]\\ i\notin H}}(\sigma_i^\dagger)^2 \tag{2.50}$$

The last inequality holds because of Lemma 2.4.7. Therefore, we have $\sum_{\substack{i\in[\underline{t},\overline{t}]\\ i\notin H}}(\sigma_i^\dagger)^2 \geq (1-\rho^\zeta)\Psi$.

Now we have

$$\sum_{i\in[\underline{t},\overline{t}]} \|(U_{:,i}^\dagger - \tilde{U}_{:,i}^\dagger)\sigma_i^\dagger\|_2^2 \geq \sum_{\substack{i\in[\underline{t},\overline{t}]\\ i\notin H}} \|(U_{:,i}^\dagger - \tilde{U}_{:,i}^\dagger\|_2^2(\sigma_i^\dagger)^2.$$

Next we bound $\|U^\dagger_{:,i} - \tilde{U}^\dagger_{:,i}\|_2^2$ when $\vec{R}_i \neq \vec{R}'_i$. We have

$$\|U^\dagger_{:,i} - \tilde{U}^\dagger_{:,i}\|_2^2 \geq 2 - \sum_{j \in \vec{R}_i \cap \vec{R}'_i} \left((U^\dagger_{j,i})^2 + (\tilde{U}^\dagger_{j,i})^2 \right). \tag{2.51}$$

By Lemma 2.4.6, whp $|\vec{R}_i \cap \vec{R}'_i| \leq c_2 \rho^{2\lambda} d$. Next, we give a bound for $\sum_{j \in \vec{R}_i \cap \vec{R}'_i} (U^\dagger_{j,i})^2$. The bound for $\sum_{j \in \vec{R}_i \cap \vec{R}'_i} (\tilde{U}^\dagger_{j,i})^2$ can be derived in a similar manner.

For each $j \in |\vec{R}_i \cap \vec{R}'_i|$, we check whether $U^\dagger_{j,i} \geq \frac{c_5}{\sqrt{\rho^{\lambda+\eta} d}}$:

$$\begin{aligned}
\sum_{j \in \vec{R}_i \cap \vec{R}'_i} (U^\dagger_{j,i})^2 &= \sum_{j \in \vec{R}_i \cap \vec{R}'_i} \left[(U^\dagger_{j,i})^2 I(U^\dagger_{j,i} \leq \frac{c_5}{\sqrt{\rho^{\lambda+\eta} d}}) + (U^\dagger_{j,i})^2 I(U^\dagger_{j,i} > \frac{c_5}{\sqrt{\rho^{\lambda+\eta} d}}) \right] \\
&\leq O\left(\frac{\rho^{2\lambda} d}{\rho^{\lambda+\eta} d} + \exp(-\Theta(1/\rho^\eta)) \right) = O(\rho^{\lambda-\eta}).
\end{aligned}$$

Therefore,

$$\sum_{i \in [\underline{t}, \bar{t}]} \|(U^\dagger_{:,i} - \tilde{U}^\dagger_{:,i})(\sigma^\dagger_i)^2\|_2^2 \geq \sum_{i \in [\underline{t}, \bar{t}]} (\sigma^\dagger_i)^2 (2 - O(\rho^{\lambda-\eta})(1 - \rho^\zeta)) \geq \sum_{i \in [\underline{t}, \bar{t}]} (\sigma^\dagger_i)^2 (2 - c_8 \rho^{\lambda-\eta} - c_9 \rho^\zeta).$$

□

We need two additional building blocks.

Lemma 2.4.10. *Consider the regression problem $\mathbf{y} = M\mathbf{x} + \epsilon$, where $\|M\| \leq \Upsilon = O(1)$ and the eigenvalues $\sigma_i(\mathbb{E}[\mathbf{x}\mathbf{x}^{\mathrm{T}}])$ of the features follow a power law distribution with exponent ω. Consider the problem $\mathbf{y} = N\mathbf{z} + \epsilon$ in orthogonalized form. Let σ^N_i be the i-th singular value of N. Let t be an arbitrary value in $[0, \min\{d_1, d_2\}]$. There exists a constant c_{10} such that*

$$\sum_{i \geq t} \sigma_i^2(N) \leq \frac{c_{10}}{t^{\omega-1}}.$$

Proof of Lemma 2.4.10. Without loss of generality, assume that $d_1 \geq d_2$. We first split the columns of N into two parts, $N = [N_+, N_-]$, in which $N_+ \in \mathbf{R}^{d_2 \times t}$ consists of the first t columns of N and $N_- \in \mathbf{R}^{d_2 \times (d_1 - t)}$ consists of the remaining columns. Let $\mathbf{0}$ be a zero

matrix in $\mathbf{R}^{d_2\times(d_1-t)}$. $[N_+, \mathbf{0}]$ is a matrix of rank at most t.

Let us split other matrices in a similar manner.

- $M = [M_+, M_-]$, where $M_+ \in \mathbf{R}^{d_2\times t}$ and $M_- \in \mathbf{R}^{d_2\times(d_1-t)}$,
- $V^* = [V^*_+, V^*_-]$, where $V^*_+ \in \mathbf{R}^{d_1\times t}$, and $V^*_- \in \mathbf{R}^{d_1\times(d_1-t)}$, and
- $\Lambda^* = [\Lambda^*_+, \Lambda^*_-]$, where $\Lambda^*_+ \in \mathbf{R}^{d_1\times t}$ and $\Lambda^*_- \in \mathbf{R}^{d_1\times(d_1-t)}$.

We have

$$\begin{aligned}\sum_{i\geq t}(\sigma_i^N)^2 &= \|N - \mathbf{P}_t(N)\|_F^2 \\ &\leq \|N - [N_+, \mathbf{0}]\|_F^2 \quad (\mathbf{P}_t(N) \text{ gives an optimal rank-}t \text{ approximation of } N). \\ &= \|N_-\|_F^2 \leq \|M_- V^*_-\|^2 \|(\Lambda^*_-)^{\frac{1}{2}}\|_F^2 \leq \frac{c_{10}}{t^{\omega-1}}\end{aligned}$$

□

We next move to our second building block.

Fact 2.4.1.

$$\mathrm{KL}(\mathbf{P}_{N^\dagger,n}, \mathbf{P}_{\tilde{N}^\dagger,n}) = \frac{n\|N^\dagger - \tilde{N}^\dagger\|_F^2}{2\sigma_\epsilon^2}. \tag{2.52}$$

Proof of Fact 2.4.1.

$$\begin{aligned}&\mathrm{KL}(\mathbf{P}_{N^\dagger,n}, \mathbf{P}_{\tilde{N}^\dagger,n}) \\ =\ & n\mathrm{KL}(\mathbf{P}_{N^\dagger}, \mathbf{P}_{\tilde{N}^\dagger}) \\ =\ & n \mathop{\mathbb{E}}_{\mathbf{y}=N^\dagger\mathbf{z}^\dagger+\epsilon}\left[\log\left(\frac{f_{N^\dagger}(\mathbf{y},\mathbf{z}^\dagger)}{f_{\tilde{N}^\dagger}(\mathbf{y},\mathbf{z}^\dagger)}\right)\right] \\ =\ & n \mathop{\mathbb{E}}_{\mathbf{z}^\dagger}\left[\mathop{\mathbb{E}}_{\mathbf{y}=N^\dagger\mathbf{z}^\dagger+\epsilon}\left[\log\left(\frac{f_{N^\dagger}(\mathbf{y},\mathbf{z}^\dagger)}{f_{\tilde{N}^\dagger}(\mathbf{y},\mathbf{z}^\dagger)}\right) \mid \mathbf{z}^\dagger\right]\right] \\ =\ & n \mathop{\mathbb{E}}_{\mathbf{z}^\dagger}\left[\mathop{\mathbb{E}}_{\mathbf{y}=N^\dagger\mathbf{z}^\dagger+\epsilon}\left[\log\left(\frac{f_{N^\dagger}(\mathbf{y}\mid\mathbf{z}^\dagger)f(\mathbf{z}^\dagger)}{f_{\tilde{N}^\dagger}(\mathbf{y}\mid\mathbf{z}^\dagger)f(\mathbf{z}^\dagger)}\right) \mid \mathbf{z}^\dagger\right]\right] \\ =\ & n \mathop{\mathbb{E}}_{\mathbf{z}}\mathop{\mathbb{E}}_{\mathbf{y}=N^\dagger\mathbf{z}^\dagger+\epsilon}\left[\left[\log\left(\frac{f_{N^\dagger}(\mathbf{y}\mid\mathbf{z}^\dagger)}{f_{\tilde{N}^\dagger}(\mathbf{y}\mid\mathbf{z}^\dagger)}\right) \mid \mathbf{z}^\dagger\right]\right] \\ =\ & n \mathop{\mathbb{E}}_{\mathbf{z}}\left[\mathrm{KL}(\mathbf{P}_{N^\dagger|\mathbf{z}^\dagger}, \mathbf{P}_{\tilde{N}^\dagger|\mathbf{z}^\dagger})\right] = \frac{n\|N^\dagger - \tilde{N}^\dagger\|_F^2}{2\sigma_\epsilon^2}\end{aligned}$$

□

We now complete the proof for Proposition 2.4.2. First, define $\psi \triangleq \frac{\sum_{i\geq \bar{t}+1}(\sigma_i^\dagger)^2}{\sum_{i\in[\underline{t},\bar{t}]}(\sigma_i^\dagger)^2}$. For any $N^\dagger$ and $\tilde{N}^\dagger$ in $\mathbb{N}$, we have

$$\|N^\dagger - \tilde{N}^\dagger\|_F^2 = \sum_{i\geq \underline{t}}(\sigma_i^\dagger)^2 = (1+\psi)\Psi,$$

where recall that $\Psi = \sum_{i\in[\underline{t},\bar{t}]}(\sigma_i^\dagger)^2$. Using Lemma 2.4.9, we have $\mathrm{KL}(\mathbf{P}_{N^\dagger,n}, \mathbf{P}_{\tilde{N}^\dagger,n}) = n(1+\psi)\Psi$.

Next, we find a smallest α such that

$$\max_{N^\dagger,\tilde{N}^\dagger} \mathrm{KL}(\mathbf{P}_{N^\dagger,n}, \mathbf{P}_{\tilde{N}^\dagger,n}) \leq \alpha \log|\mathbb{N}|. \tag{2.53}$$

By Lemma 2.4.7, we have $|\mathbb{N}| = \exp(c_3 \frac{n\rho^{2\lambda+\zeta-2}}{\sigma_\epsilon^2}\Psi)$. (2.53) is equivalent to requiring

$$\frac{n(1+\psi)\Psi}{2\sigma_\epsilon^2} \leq \frac{\alpha c_3 n\rho^{2\lambda+\zeta-2}\Psi}{\sigma_\epsilon^2}.$$

We may thus set $\alpha = O(\rho^{2-2\lambda+\zeta}(1+\psi))$. Now we may invoke Theorem 2.4.5 and get

$$\begin{aligned}
\mathbf{r}(\mathbf{x}, M, n, \sigma_\epsilon) &\geq \Psi\left(1 - \frac{1}{\sqrt{|\mathbb{N}|}}\right) \underbrace{(2 - c_8\rho^{\lambda-\eta} - c_9\rho^\zeta)}_{\text{Lemma 2.4.9}} \Big(1 - O(\rho^{2-2\lambda-\zeta})(1+\psi)\Big) \\
&\geq \Psi(1 - O(\rho^{\lambda-\eta} + \rho^\zeta + \rho^{2-2\lambda-\zeta})) - \underbrace{\Psi\psi}_{=\sum_{i>\bar{t}}(\sigma_i^\dagger)^2} \rho^{2-2\lambda-\zeta} \\
&\geq \Psi(1 - O(\rho^{\lambda-\eta} + \rho^\zeta + \rho^{2-2\lambda-\zeta})) - O\left(\frac{1}{(\rho^\lambda d)^{\omega-1}}\right)\rho^{2-2\lambda-\zeta} \\
&\qquad \text{(Use Lemma 2.4.10 to bound } \textstyle\sum_{i>\bar{t}}(\sigma_i^\dagger)^2 \text{)}
\end{aligned}$$

We shall set $\varepsilon = \eta$ be a small constant (say 0.001), $\lambda = \frac{1}{2} + \varepsilon$, and $\zeta = \frac{1}{2}$. This gives us

$$\mathbf{r}(\mathbf{x}, M, n, \sigma_\epsilon) \geq \Psi(1 - O(\rho^{\frac{1}{2}} + \rho^{\frac{1}{2}-2\varepsilon})) - \frac{\rho^{\frac{1}{2}-2\varepsilon-(\omega-1)(\frac{1}{2}+\varepsilon)}}{d^{\omega-1}}$$

Together with the fact that $\sum_{i>\bar{t}}(\sigma_i^\dagger)^2 = O\left(\frac{1}{(\rho^\lambda\omega)^{\omega-1}}\right)$, we complete the proof of Proposition 2.4.2.

2.5 Related work and comparison

In this section, we compare our results to other regression algorithms that make low rank constraints on M. Most existing MSE results are parametrized by the rank or spectral properties of M, e.g. [118]defined a generalized notion of rank

$$\mathbb{B}_q(R_q^A) \in \left\{A \in \mathbf{R}^{d_2\times d_1} : \sum_{i=1}^{\min\{d_1,d_2\}} |\sigma_i^A|^q \le R_q\right\}, \quad q \in [0,1], A \in \{N, M\}, \tag{2.54}$$

i.e. R_q^N characterizes the generalized rank of N whereas R_q^M characterizes that of M. When $q = 0$, $R_q^N = R_q^M$ is the rank of the N because $\mathrm{Rank}(N) = \mathrm{Rank}(M)$ in our setting. In their setting, the MSE is parametrized by R^M and is shown to be $O\Big(R_q^M\Big(\frac{\sigma_\epsilon^2\lambda_1^*(d_1+d_2)}{(\lambda_{\min}^*)^2 n}\Big)^{1-q/2}\Big)$. In the special case when $q = 0$, this reduces to $O\left(\frac{\sigma_\epsilon^2\lambda_1^*\mathrm{Rank}(M)(d_1+d_2)}{(\lambda_{\min}^*)^2\cdot n}\right)$. On the other hand, the MSE in our case is bounded by (cf. Thm. 2.3.2). We have $\mathbb{E}[\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2] = O\big(R_q^N(\frac{\sigma_\epsilon^2 d_2}{n})^{1-q/2} + n^{-c_0}\big)$. When $q = 0$, this becomes $O\big(\frac{\sigma_\epsilon^2\mathrm{Rank}(M)d_2}{n} + n^{-c_0}\big)$.

The improvement here is twofold. First, our bound is directly characterized by N in orthogonalized form, whereas result of [118]needs to examine the interaction between M and C^*, so their MSE depends on both R_q^M and $\lambda_{\min}^*$. Second, our bound no longer depends on d_1 and pays only an additive factor n^{-c_0}, thus, when $n < d_1$, our result is significantly better. Other works have different parameters in the upper bounds, but all of these existing results require that $n > d_1$ to obtain non-trivial upper bounds [84, 20, 27, 84]. Unlike these prior work, we require a stochastic assumption on $\mathbf{X}$ (the rows are i.i.d.) to ensure that the model is identifiable when $n < d_1$, e.g. there could be two sets of disjoint features that fit the training data equally well. Our algorithm produces an adaptive model whose complexity is controlled by k_1 and k_2, which are adjusted dynamically depending on the sample size and noise level. [20] and [27] also point out the need for adaptivity; however they still require $n > d_1$ and make some strong assumptions. For instance, [20]assumes that there is a

gap between $\sigma_i(\mathbf{X}M^{\mathrm{T}})$ and $\sigma_{i+1}(\mathbf{X}M^{\mathrm{T}})$ for some i. In comparison, our sufficient condition, the decay of λ_i^*, is more natural. Our work is not directly comparable to standard variable selection techniques such as LASSO [148] because they handle univariate $\mathbf{y}$. Column selection algorithms [35] generalize variable selection methods for vector responses, but they cannot address the identifiability concern.

2.5.1 Missing proof in comparison

This section proves the following corollary.

Corollary 2.5.1. *Use the notation appeared in Theorem 2.3.2. Let the ground-truth matrix $N \in \mathbb{B}_q(R_q^N)$ for $q \in [0,1]$. We have whp*

$$\mathbb{E}[\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2] = O\left(R_q^N \left(\frac{\sigma_\epsilon^2 d_2}{n}\right)^{1-q/2} + n^{-c_0}\right). \tag{2.55}$$

Proof. We first prove the case $q = 0$ as a warmup. Observe that

$$\|N\|_F^2 - \sum_{i \le \ell^*} (\sigma_i^N)^2 = \sum_{\ell^* < i \le r} (\sigma_i^N)^2 \le \frac{\theta^2 \sigma_\epsilon^2 d_2 (r - \ell)}{n}.$$

The last inequality uses $\sigma_i^N \le \theta\sigma_\epsilon\sqrt{\frac{d_2}{n}}$ for $i > \ell^*$. Therefore, we have

$$\mathbb{E}[\|\hat{\mathbf{y}} - \mathbf{y}\|^2)2] \le O\left(\frac{(r-\ell^*)\theta^2\sigma_\epsilon^2 d_2}{n} + \frac{\ell^* d_2 \theta^2 \sigma_\epsilon^2}{n} + n^{-c_0}\right) = O\left(\frac{r\theta^2\sigma_\epsilon^2 d_2}{n} + n^{-c_0}\right).$$

Next, we prove the general case $q \in (0,1]$. We can again use an optimization view to give an upper bound of the MSE. We view $\sum_{i \le d_2} (\sigma_i^N)^q \le R_q$ as a constraint. We aim to maximize the uncaptured signals, i.e., solve the following optimization problem

$$
\begin{aligned}
\text{maximize:} \quad & \sum_{i>\ell^*} (\sigma_i^N)^2 \\
\text{subject to:} \quad & \sum_{i>\ell} (\sigma_i^N)^q \le R_-, \quad \text{where } R_- = R_q - \textstyle\sum_{i\le\ell^*} (\sigma_i^N)^q \\
& \sigma_i^N \le \frac{\theta^2 \sigma_\epsilon^2 d_2}{n}, \quad \text{for } i \ge \ell^*.
\end{aligned}
$$

The optimal solution is achieved when $(\sigma_i^N)^2 = \frac{\theta^2\sigma_\epsilon^2 d_2}{n}$ for $\ell^* < i \le \ell^* + k$, where $k = \frac{R_-}{\left(\frac{\theta^2\sigma_\epsilon^2 d_2}{n}\right)^{\frac{q}{2}}}$, and $\sigma_i^N = 0$ for $i > \ell^* + k$. We have

$$
\begin{aligned}
& \mathbb{E}[\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2] \\
& \le \sum_{i>\ell^*} (\sigma_i^N)^2 + O\left(\frac{\ell^* d_2 \theta^2 \sigma_\epsilon^2}{n} + n^{-c_0}\right) \\
& = \left(R_q - \sum_{i\le\ell^*} (\sigma_i^N)^q\right) \left(\frac{\theta^2\sigma_\epsilon^2 d_2}{n}\right)^{1-q/2} + O\left(\frac{\ell^* d_2 \theta^2 \sigma_\epsilon^2}{n} + n^{-c_0}\right) \qquad (2.56)
\end{aligned}
$$

We can also see that

$$
\sum_{i\le\ell^*} (\sigma_i^N)^q \left(\frac{\theta^2\sigma_\epsilon^2 d_2}{n}\right)^{1-q/2} \ge \sum_{i\le\ell^*} \left(\frac{\theta^2\sigma_\epsilon^2 d_2}{n}\right)^{q/2} \left(\frac{\theta^2\sigma_\epsilon^2 d_2}{n}\right)^{1-q/2} = \ell^* \left(\frac{\theta^2\sigma_\epsilon^2 d_2}{n}\right).
$$

Therefore,

$$
(2.56) \le O\left(R_q \left(\frac{\theta^2\sigma_\epsilon^2 d_2}{n}\right)^{1-q/2} + n^{-c_0}\right).
$$

□

2.6 Experiments

We apply our algorithm on an equity market and a social network dataset to predict equity returns and user popularity respectively. Our baselines include ridge regression ("Ridge"), reduced rank ridge regression [114] ("Reduced ridge"), LASSO ("Lasso"), nuclear norm

Model	MSE_{out}	MSE_{in}	MSE_{out-in}	R^2_{out}(bps)	R^2_{in} (bps)	Sharpe	t-statistic
ARRR, N= 983	**0.9935**	1.0140	**-0.0205**	**46.3761**	158.2564	**2.4350**	**8.3268**
Lasso	1.1158	0.3953	0.7205	6.6049	7147.0116	2.1462	0.0601
Ridge	1.2158	0.1667	1.0491	9.8596	8511.9076	0.6603	-0.0497
Reduced ridge	1.0900	0.8687	0.2213	13.0321	1555.5136	0.3065	-0.3275
RRR	1.2200	0.5867	0.6332	7.0830	4121.2548	0.3647	-0.6626
Nuclear norm	1.2995	0.12078	1.1787	4.7297	8789.0625	0.6710	0.2340
PCR	1.0259	0.8456	0.1802	1.1278	1544.7258	1.8070	0.3947
ARRR, N= 2838	**1.0056**	0.9050	**0.1006**	**18.5761**	689.0625	**1.6239**	**15.4134**
Lasso	1.0625	0.5286	0.5339	1.1236	6029.5225	0.5954	0.0179
Ridge	1.0289	0.6741	0.3548	0.2116	5342.1481	0.5739	0.0670
Reduced ridge	1.9722	0.7373	1.2349	1.0816	2416.7056	1.5482	0.0619
RRR	1.0873	0.61376	0.4735	4.5795	3844.124	-0.477	0.6399
Nuclear norm	1.1086	0.15346	0.9551	2.2097	8461.2402	-0.3698	-0.8986
PCR	1.0263	0.5336	0.4927	5.233	4653.9684	1.2799	0.6990

Table 2.1: Summary of results for equity return forecasts. R^2 are measured by basis points (bps). 1bps = 10^{-4}. Bold font denotes the best *out-of-sample* results and smallest gap.

Model	MSE_{in}	MSE_{out}	MSE_{out-in}	Corr_{in}	Corr_{out}
ARRR	5.0104 ± 0.38	**9.4276** ± 2.31	**4.4172**	0.7425 ± 0.07	**0.6730** ± 0.13
Lasso	2.3755 ± 1.95	14.8279 ± 4.81	12.4524	0.9171 ± 0.09	0.4754 ± 0.15
Ridge	1.3974 ± 0.53	13.6244 ± 4.39	12.2270	0.9555 ± 0.04	0.4742 ± 0.17
Reduced ridge	4.5260 ± 1.93	12.2339 ± 2.70	7.7079	0.7905 ± 0.09	0.4972 ± 0.18
RRR	4.3456 ± 0.47	13.0768 ± 2.63	8.7313	0.7725 ± 0.12	0.3820 ± 0.22
Nuclear norm	4.9190 ± 2.04	13.0532 ± 4.38	8.6677	0.7872 ± 0.10	0.4869 ± 0.16
PCR	6.4037 ± 1.99	13.0847± 4.19	8.8892	0.7199 ± 0.05	0.4861 ± 0.15

Table 2.2: Average results for Twitter dataset from 10 random samples. Bold font denotes the best *out-of-sample* results and smallest gap

regularized regression ("Nuclear norm"), and reduced rank regression [153] ("RRR"), and principal component regression [3] ("PCR").

Predicting equity returns. We use a stock market dataset from an emerging market that consists of approximately 3600 stocks between 2011 and 2018. We focus on predicting the *next 5-day returns*. For each asset in the universe, we compute its past 1-day, past 5-day and past 10-day returns as features. We use a standard approach to translate forecasts into positions [8, 169]. We examine two universes in this market: (i) *Universe 1* is equivalent to S&P 500 and consists of 983 stocks, and (ii) *Full universe* consists of all stocks except for illiquid ones.

Results. Table 2.1 reports the forecasting power and portfolio return for *out-of-sample* periods in two universes. We observe that *(i)* The data has a low signal-to-noise ratio.

The out-of-sample R^2 values of all the methods are close to 0. *(ii)* ADAPTIVE-RRR has the highest forecasting power. *(iii)* ADAPTIVE-RRR has the smallest in-sample and out-of-sample gap (see column MSE_{out-in}), suggesting that our model is better at avoiding spurious signals.

Predicting user popularity in social networks. We collected tweet data on political topics from October 2016 to December 2017. Our goal is to predict a user's *next 1-day* popularity, which is defined as the sum of retweets, quotes, and replies received by the user. There are a total of 19 million distinct users, and due to the huge size, we extract the subset of 2000 users with the most interactions for evaluation. For each user in the 2000-user set, we use its past 5 days' popularity as features. We further randomly sample 200 users and make predictions for them, i.e., setting $d_2 = 200$ to make d_2 of the same magnitude as n.

Results. We randomly sample users for 10 times and report the average MSE and correlation (with standard deviations) for both *in-sample* and *out-of-sample* data. In Table 2.2 we can see results consistent with the equity returns experiment: *(i)* ADAPTIVE-RRR yields the best performance in out-of-sample MSE and correlation. *(ii)* ADAPTIVE-RRR achieves the best generalization error by having a much smaller gap between training and test metrics.

2.6.1 Setup of experiments

2.6.1.1 Equity returns

We use daily historical stock prices and volumes from an emerging market to build our model. Our license agreement prohibits us to redistribute the data so we include only a subset of samples. Full datasets can be purchased by standard vendors such as quandl or algoseek. Our dataset consists of approximately 3,600 stocks between 2011 and 2018.

Universes. We examine two different universes in this market *(i) Universe 1* is equivalent to S&P 500. It consists of 800 stocks at any moment. Similar to S&P 500, the list of

stocks appeared in universe 1 is updated every 6 months. A total number of 983 stocks have appeared in universe 1 at least once. *(ii) Universe 2* consists of all stocks except for illiquid ones. This excludes the smallest 5% stocks in capital and the smallest 5% stocks in trading volume. Standard procedure is used to avoid universe look-ahead and survival bias [169].

Returns. We use return information to produce both features and responses. Our returns are the "log-transform" of all open-to-open returns [169]. For example, the next 5-day return of stock i is $\log(p_{i,t+5}/p_{i,t})$, where $p_{i,t}$ is the open price for stock i on trading day t. Note that all non-trading days need to be removed from the time series $p_{i,t}$. Similarly, the past 1-day return is $\log(p_{i,t}/p_{i,t-1})$.

Model. We focus on predicting the *next 5-day returns* for different universes. Let $\mathbf{r}_t = (r_{1,t}, r_{2,t}, \ldots, r_{d_2,t})$, where $r_{i,t}$ is the next 5-day return of stock i on day t. Our regression model is

$$\mathbf{r}_{t+1} = M\mathbf{x}_t + \epsilon. \tag{2.57}$$

The features consist of the past 1-day, past 5-day, and past 10-day returns of all the stocks in the same universe. For example, in *Universe 1*, the number of responses is $d_2 = 800$. The number of features is $d_1 = 800 \times 3 = 2,400$. We further optimize the hyperparameters k_1 and k_2 by using a validation set because our theoretical results are asymptotic ones (with unoptimized constants). Baseline models use the same set of features and the same hyper-parameter tuning procedure.

We use three years of data for training, one year for validation, and one year for testing. The model is re-trained every test year. For example, the first training period is May 1, 2011 to May 1, 2014. The corresponding validation period is from June 1, 2014 to June 1, 2015. We use the validation set to determine the hyperparameters and build the model, and then we use the trained model to forecast returns of equity in the same universe from July 1, 2015 to July 1, 2016. Then the model is retrained by using data in

the second training period (May 1, 2012 to May 1, 2015). This workflow repeats. To avoid looking-ahead, there is a gap of one month between training and validation periods, and between validation and test periods.

We use standard approach to translate forecasts into positions [60, 126, 8, 169]. Roughly speaking, the position is proportional to the product of forecasts and a function of average dollar volume. We allow short-selling. We do not consider transaction cost and market impact. We use Newey-West estimator to produce t-statistics of our forecasts.

2.6.1.2 User popularity

We use Twitter dataset to build models for predicting a user's *next 1-day* popularity, which is defined as the sum of retweets, quotes, and replies received by the user.

Data collection. We collected 15 months Twitter data from October 01, 2016 to December 31, 2017, using the Twitter streaming API. We tracked the tweets with topics related to politics with keywords "trump, "clinton, "kaine, "pence, and "election2016. There are a total of 804 million tweets and 19 million distinct users. User u has one interaction if and only if he or she is retweeted/replied/quoted by another user v. Due to the huge size, we extract the subset of 2000 users with the most interactions for evaluation.

Model. Our goal is to forecast the popularity of a random subset of 200 users. Let $\mathbf{y}_t = (y_{1,t}, \ldots, y_{d_2,t})$, where $d_2 = 200$ and $y_{i,t}$ is the popularity of user i at time t. Our regression model is

$$\mathbf{y}_{t+1} = M\mathbf{x}_t + \epsilon. \tag{2.58}$$

Features. For each user, we compute his/her daily popularity for 5 days prior to day t. Therefore, the total number of features is $d_1 = 2000 \times 5 = 10,000$.

We remark that there are $n = 240$ observations. This setup follows our assumption $d_1 \gg d_2 \approx n$.

Training and hyper-parameters. We use the period from October 01, 2016 to June 30, 2017 as the training dataset, the period from July 01, 2017 to October 30 as validation

dataset to optimize the hyper-parameters, and the rest of the period, from September 10, 2017 to December 31, 2017, is used for the performance evaluation.

2.7 Conclusion

This paper examines the low-rank regression problem under the high-dimensional setting. We design the first learning algorithm with provable statistical guarantees under a mild condition on the features' covariance matrix. Our algorithm is simple and computationally more efficient than low rank methods based on optimizing nuclear norms. Our theoretical analysis of the upper bound and lower bound can be of independent interest. Our preliminary experimental results demonstrate the efficacy of our algorithm. The full version explains why our (algorithm) result is unlikely to be known or trivial.

Chapter 3

On Embedding Stocks: Orchestrating High-dimensional Techniques for Financial Machine Learning Models

3.1 Introduction

We study the problem of forecasting equity (stock) returns as there is a direct link between a portfolio's profitability and its forecasting quality. Our proposed forecasting model uses information available up to time t to predict $\mathbf{p}_{t+1,i}$, the price of stock i at time $t+1$, for a total number of d stocks in the financial market. While there has been extensive research concerning the predictability of stock prices [43, 113, 139], many anomalies have been discovered and recent studies suggest that machine learning (ML) techniques can also be effective in forecasting returns [155, 26, 72, 166, 161]. The relevant learning models can be categorized into two groups:

Univariate models (UM) to learn feature interactions. Univariate models fit a function $\mathbf{p}_{t+1,i} = f(\mathbf{x}_{t,i}) + \xi_{t,i}$ to forecast one stock's future price (return) by using features

constructed from that stock's historical data. Univariate models primarily learn interactions between features by using off-the-shelf ML techniques. See e.g., [51, 62, 28].

Cross-asset models (CAM) to learn stock interactions. Cross-asset models solve a vector regression problem $\mathbf{p}_{t+1} = f(\mathbf{x}_t) + \xi_t$, to forecast the future prices of all stocks in a universe, where $\mathbf{p}_{t+1} \triangleq (\mathbf{p}_{t+1,1}, \ldots, \mathbf{p}_{t+1,d})$, and $\mathbf{x}_t$ denotes the features of all stocks, constructed from their historical data. Since the features of one stock can be used to predict the future prices of another (such as co-movement, lead-lag relation etc.), CAMs have stronger expressive and predictive power. CAMs are both computationally and statistically challenging because we need to solve the "high-dimensional" (overparametrized) problem, i.e., the total number of learnable parameters far exceeds the number of observations. Consider, for example, a vector auto-regressive model $\mathbf{y}_t = M\mathbf{y}_{t-1} + \xi_t$ for predicting next-day returns of $d = 3,000$ stocks in Russell 3000 using their past 1-day returns. There are $d^2 \approx 10$ million learnable parameters in M, but there are usually less than $n = 3,000$ observations (using 10 years of approximately 3,000 trading days of training data). The model can exactly fit the training data, but may not deliver predictive power. Extensive research has been undertaken to design regularization techniques to address the issue, most of which focus on linear models [118, 161].

In this paper, we design models that are expressive enough to capture interactions between features and between stocks, and develop new learning methodologies that can easily leverage modern ML algorithms and are less prone to overfitting issues. Our major challenge is to integrate two seemingly incompatible modeling processes (i.e., UM and CAM) with different design philosophies, which we briefly discuss below.

In high-dim models for stock interactions, because fitting is straightforward and producing meaningful generalization errors is challenging, theoretical tools are used to develop algorithms with *provable guarantee*, to alleviate the data scarcity problems. In univariate models for feature interactions, fitting is typically non-trivial, but since the generalization errors are usually manageable, an *experiment-driven* process is used to fit the model.

Here, the theoretical analysis is more difficult, but also less important, because the model performance can be assessed by using standard (cross)-validation methods. It remains unclear how we can reconcile a design process that promotes theoretical analysis and down-weighs the role of fitting, with an experiment-driven one that relies heavily on fitting and has a lower demand on theoretical guarantees.

Our approach & contribution. We propose a latent position model for equity returns, dubbed as the *additive influence model*, that enables us to orchestrate mathematically rigorous high-dim techniques with practically effective machine learning algorithms. Our model assumes that each stock i is associated with a vector representation $\mathbf{z}_i$ in a (latent) Euclidean space, and characterizes the interactions between stocks in the form of $\mathbf{y}_{t,i} = \sum_{j\in[d]} \kappa(\mathbf{z}_i, \mathbf{z}_j) g(\mathbf{x}_{j,t}) + \xi_{t,i}$, where $\mathbf{y}_{t,i} \in \mathbf{R}$ is the next period return at time t for stock i, $\mathbf{x}_{t,j} \in \mathbf{R}^k$ are the features associated with stock j at time t, $\xi_{t,i}$ is a noise term, $g : \mathbf{R}^k \to \mathbf{R}$ is an unknown function, and κ is a function that measures the interaction strength between stocks based on their vector representations. When $\mathbf{z}_i$ and $\mathbf{z}_j$ are close, $\kappa(\mathbf{z}_i, \mathbf{z}_j)$ will be large, and thus the variable $g(\mathbf{x}_{t,j})$ from stock j's has a stronger impact on i's return.

Our proposed model allows for feature interactions through $g(\cdot)$, and addresses the overfitting problem arising from stock interactions because the distances (interaction strength) between stocks are constrained by the latent Euclidean space: when $\mathbf{z}_i - \mathbf{z}_j$ and $\mathbf{z}_j - \mathbf{z}_k$ are small, $\mathbf{z}_i - \mathbf{z}_k$ is also small, and thus the degree of freedom for stock interactions becomes substantially smaller than $O(d^2)$.

Our goal is to learn both the $\mathbf{z}_i$'s and $g(\cdot)$. We note that these two learning tasks can be *decoupled*: high-dim methods can be developed to *provably* estimate the $\mathbf{z}_i$'s *without the knowledge of* $g(\cdot)$, and when estimates of $\mathbf{z}_i$'s are given, an experiment-driven process can be used to learn $g(\cdot)$ by examining prominent machine learning methods such as neural nets and boosting. In other words, when we learn stock interactions, we do not need to be troubled by the overfitting problem escalated by fine-tuning $g(\cdot)$, and when we learn feature interactions, the generalization error will not be jeopardized by the curse of dimensionality

from stock interactions.

To learn the $\mathbf{z}_i$'s, we design a simple algorithm that uses low-rank approximation of $\mathbf{y}_t$'s covariance matrix to find the closeness of the stocks, and develop a novel theoretical analysis based on recent techniques from high-dim and kernel learning [15, 147, 161].

To learn $g(\cdot)$, we generalize major machine learning techniques, including neural nets, non-parametric, and boosting methods, to the additive influence model when estimates of $\mathbf{z}_i$'s are known. We specifically develop a moment-based algorithm for non-parametric learning of $g(\cdot)$, and a computationally efficient boosting algorithm based on linear learners by using the domain knowledge of equity data sets.

Finally, we perform extensive experiments on data sets from a major equity market to confirm the efficacy of our modeling approaches and analysis.

3.2 Problem definition

Notations. For a matrix A, $\mathcal{P}_r(A)$ denotes its rank-r approximation obtained by keeping the top r singular values and the corresponding singular vectors. $\sigma_i(A)$ (resp. $\lambda_i(A)$) is the i-th singular value (resp. eigenvalue) of A. We use Python/MATLAB notation when we refer to a specific row or column. For example, $A_{1,:}$ is the first row of A, and $A_{:,1}$ is the first column. $\|A\|_F$ and $\|A\|_2$ denote the Frobenius and spectral norms, respectively, of A. In general, we use boldface upper case (e.g., $\mathbf{X}$) to denote data matrices and boldface lower case (e.g., $\mathbf{x}$) to denote one sample. $\mathbf{x}_{t,i}$, which refers to the features associated with stock i at time t, can be one or multi-dimensional. Let $(\mathbf{x}_{t,i})_j$ be the j-th coordinate (feature) of $\mathbf{x}_{t,i}$. An event occurring with high probability (whp) means that it happens with probability $\geq 1 - n^{10}$, where 10 is an arbitrarily chosen large constant and is not optimized. A bivariate function is a Gaussian kernel if $\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2)$, an inverse multi-quadratic (IMQ) kernel if $\kappa(\mathbf{x}, \mathbf{x}') = (c^2 + \|\mathbf{x} - \mathbf{x}'\|^2)^{-\alpha}$ ($\alpha > 0$), and an inner product kernel if $\kappa(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$.

A function $g(\cdot)$ is Lipschitz-continuous if $|g(\mathbf{x}_1) - g(\mathbf{x}_2)| \leq c\|\mathbf{x}_1 - \mathbf{x}_2\|$ for a constant c. A distribution $\mathcal{D}$ with bounded domain and probability density function $f_{\mathcal{D}}$ is near-uniform

if $\frac{\sup f_{\mathcal{D}}(\mathbf{x})}{\inf f_{\mathcal{D}}(\mathbf{x})} = O(1)$.

The forecasting problem. Let d denote the total number of stocks in our universe of interest. Our model assumes that the equity market proceeds in rounds. Let $\mathbf{y}_{t,i} \in \mathbf{R}$ be the next-period return of stock i at the t-th round, and $\mathbf{y}_t = (\mathbf{y}_{t,1}, \ldots, \mathbf{y}_{t,d}) \in \mathbf{R}^d$. Our goal is to forecast $\mathbf{y}_t$ based on all information available up to round t. We are primarily interested in forecasting the next 5-day return (i.e., each round consists of 5 days) and using only technical factors as features because our forecasting horizon is short and unsuitable for fundamental factors [38]. In practice, we use standard overlapping techniques to train the model [169]. See also App. 3.13.1.

Model Assumptions. Under the additive influence model

$$\mathbf{y}_{t,i} = \sum_{j \leq d} \kappa(\mathbf{z}_i, \mathbf{z}_j) g(\mathbf{x}_{t,j}) + \xi_{t,i}, \tag{3.1}$$

our goal is to learn $g(\cdot)$ and $\mathbf{z}_i$'s with a total number of n observations. Let also $K \in \mathbf{R}^{d \times d}$ such that $K_{i,j} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$. Here, we assume that • *(A.1)* vector representations $\mathbf{z}_i$'s of the stocks and features $\mathbf{x}_{t,i}$ are i.i.d. samples from (two different) near-uniform distributions on bounded supports, • *(A.2)* $\mathbf{x}_{t,i} \in [-1, 1]$ and $\mathbb{E}[g(\mathbf{x}_{t,i})] = 0$, • *(A.3)* $g(\cdot)$ is Lipschitz-continous, and • *(A.4)* $\xi_{t,i}$'s are zero-mean i.i.d. Gaussian random variables with standard deviation σ_ξ.

(A.1) is standard in the literature [2, 146, 147, 90, 131]. Assuming (A.2) simplifies the calculation and is without loss of generality, and (A.4) can also be relaxed to settings in which $\xi_{t,i}$ are sub-Gaussian. See App. 3.7 for more discussion of the assumptions.

3.3 Our algorithms

Sec. 3.3.1 describes an algorithm for learning the embedding without knowing $g(\cdot)$, and Sec. 3.3.2 explains estimation of $g(\cdot)$ using machine learning techniques. Due to the space limit, detailed proofs of all the Props can be found in App. 3.8.

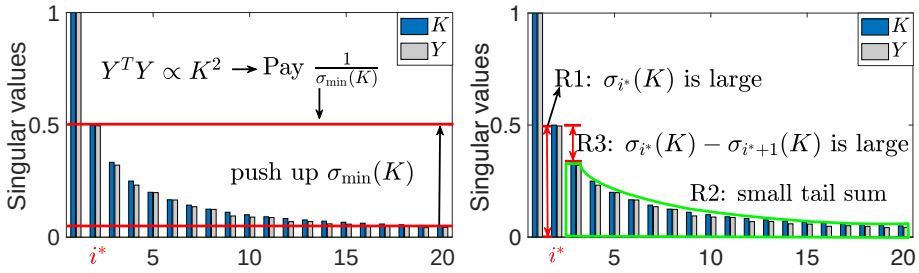


Figure 3.1: (a) We use the square root of $\mathcal{P}_{i^*}(\mathbf{Y}^{\mathrm{T}}\mathbf{Y})$ to approximate K so that we pay a factor of $1/\sigma_{i^*}(K)$, instead of $1/\sigma_{\min}(K)$. (b) Three key requirements for i^*: $\sigma_{i^*}(K)$ is large (R1), $\mathcal{P}_{i^*}(K^2)$ is close to K^2 (R2), and $\sigma_{i^*}(K) - \sigma_{i^*+1}(K)$ is large (R3).

3.3.1 Learning vector representation provably

This section presents a provable algorithm to estimate the kernel matrix K and the embedding $\mathbf{z}_i$'s. Our algorithm does not need to know $g(\cdot)$, thus it provides a conceptually new approach to construct CAMs: high-dim learning of stock interactions can be decoupled from using ML techniques to fit the features. Because learning stock-interactions could be a major source for causing overfitting, disentangling it from the downstream task of learning $g(\cdot)$ enables us to leverage the function-fitting power of ML techniques without the cost of amplifying generalization errors.

We next walk through our design intuition. We first need to introduce additional notations. Let $\mathbf{Y} \in \mathbf{R}^{n\times d}$ be such that $\mathbf{Y}_{t,i} = \mathbf{y}_{t,i}$ ($\mathbf{Y}$ is a matrix and $\mathbf{y}$ a random variable), $\mathbf{S} \in \mathbf{R}^{n\times d}$ with $\mathbf{S}_{t,i} = \mathbf{s}_{t,i} \triangleq g(\mathbf{x}_{t,i})$, and $E \in \mathbf{R}^{n\times d}$ with $E_{t,i} = \xi_{t,i}$. Recall that $K \in \mathbf{R}^{d\times d}$ s.t. $K_{i,j} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$, and $\mathcal{P}_r(A)$ denotes A's rank-r approximation obtained by keeping the top r singular values and vectors. Finally, for any PSD matrix A with SVD $A = U\Sigma U^{\mathrm{T}}$, let $\sqrt{A} \triangleq U\Sigma^{\frac{1}{2}}U^{\mathrm{T}}$.

Eq. (3.1) can be re-written as $\mathbf{Y} = \mathbf{S}K + E$, in which we need to infer K using only $\mathbf{Y}$. We first observe that while none of the entries in $\mathbf{S}$ are known, $\mathbf{S}_{t,i}$'s are i.i.d. random variables (because $\mathbf{x}_{t,i}$'s are i.i.d.); therefore, our problem resembles a dictionary learning problem, in which K can be viewed as the dictionary to be learned, and $\mathbf{S}$ is the measurement matrix (see e.g., [9]). However, in our case, K is neither low-rank nor sparse, we cannot use standard dictionary learning techniques.

First, we observe that if we have infinitely many samples, $\mathbf{Y}^{\mathrm{T}}\mathbf{Y}/n$ approaches to K^2. Hence, intuitively we could use $\sqrt{\mathbf{Y}^{\mathrm{T}}\mathbf{Y}/n}$ to approximate $\sqrt{K^2} = K$. However, existing matrix square root result has the notorious "$1/\sigma_{\min}$-blowup" problem, i.e., it gives us only $\|\sqrt{\mathbf{Y}^{\mathrm{T}}\mathbf{Y}/n} - KW\|_F \propto 1/\sigma_{\min}(K)$ (W a unitary matrix), where typically $\sigma_{\min}(K)$ is extremely small, so the bound is too loose to be useful [18].

To tackle the problem, our algorithm uses $\sqrt{\mathcal{P}_{i^*}(\mathbf{Y}^{\mathrm{T}}\mathbf{Y})/n}$ to approximate K for a carefully chosen i^* so that we pay a factor of $\sigma_{i^*}(K)$, instead of $\sigma_{\min}(K)$, to substantially tighten the error. See Alg. 2 in App. 3.8.2 and Fig. 3.1(a). To implement this idea, we need to show that there always exists an i^* such that • *(R1):* $\sigma_{i^*}(K)$ is sufficiently large, • *(R2):* $\mathcal{P}_{i^*}(K^2)$ is close to K^2, and • *(R3):* the spectral gap $\sigma_{i^*}(K) - \sigma_{i^*+1}(K)$ is sufficiently large so that we can use the Davis-Kahan theorem to prove that $\mathcal{P}_{i^*}(K^2) \propto \mathcal{P}_{i^*}(Y^{\mathrm{T}}Y)$ [141]. See also Fig. 3.1(b).

These three requirements may not always be met simultaneously. For example, when $\sigma_i(K^2) \propto \frac{1}{i}$, the gap is insufficient and the tail diverges (R2 and R3 are violated). Therefore, we integrate the following two results. • *(i)* The eigenvalues decay fast. This stems from two classical results from the *kernel learning* literature. First, when $\kappa(\cdot,\cdot)$ is sufficiently smooth (such as the Gaussian, IMQ, or inner product kernels), the eigenvalues of the kernel operator $\mathcal{K}$ associated with $\kappa(\cdot,\cdot)$ decay exponentially (e.g., $\lambda_i(\mathcal{K}) \leq \exp(-Ci^{\frac{1}{r}})$ for Gaussian kernels [15]). Second, it holds true that $\sum_{i\geq 1} |\lambda_i(\mathcal{K}) - \lambda_i(K/d)|_F^2 \propto \frac{1}{n}$, a convergence result under the PAC setting [147]. Therefore, $\lambda_i(K)$ also approximately decays exponentially. • *(ii)* Combinatorial analysis between gaps and tails. We then leverage a recent analysis [161] showing that when $\lambda_i(K)$ decays fast, it is always possible to find an i^* such that $\lambda_{i^*}(\mathcal{K}) - \lambda_{i^*+1}(\mathcal{K})$ is sufficiently large (R1 & R3 are satisfied) and $\sum_{j\geq i^*} \lambda_j^2(\mathcal{K}) = o(1)$ (R2 is satisfied). We have

Proposition 3.3.1. *Consider the additive influence model. Let $\kappa(\mathbf{z}_i, \mathbf{z}_j)$ be a Gaussian, inverse multi-quadratic (IMQ) or inner product kernel. Let $n \geq d$ be the number of observations and $\epsilon = \frac{c_0 \log^3 d}{\sqrt{d}}$. Assume that the noise level $\sigma_\xi = O(\sqrt{d})$. Let δ be a tunable*

parameter (also appeared in Alg. 2 in App. 3.8.2) such that $\delta^3 = \omega(\epsilon^2)$*. There exists an efficient algorithm that outputs* $\hat{K}$ *such that* $\frac{1}{d^2}\|\hat{K} - K\|_F^2 = O(\frac{\epsilon^2}{\delta^3} + \delta^{\frac{4}{5}})(= \tilde{O}(d^{-\Theta(1)}))$.

We remark that *(i)* the algorithm does not need to know the exact form of κ, so long as it is one of Gaussian, IMQ, or inner product kernels, *(ii)* once K is estimated, an Isomap-flavored algorithm may be used to estimate $\mathbf{z}_i$'s [90], and *(iii)* knowing $\hat{K}$ (without reconstructing $\mathbf{z}_i$'s) is sufficient for the downstream $g(\cdot)$-learners we consider in this work.

Generalization. Our algorithm needs only the covariance matrix of returns (aka the "risk" of the assets [45]). We polish the risk model by exploiting the equity market specific structure (e.g., the returns' volatility possesses a momentum property [19]) and use an enhanced risk model to improve the estimation of K. We may also use news data to estimate the risk matrix under other suitable assumptions [162]. See App. 3.8.3.

3.3.2 Learning $g(\cdot)$

Here, we explain how prominent machine learning techniques, including neural nets (deep learning), non-parametric methods, and boosting, can be used to learn $g(\cdot)$. These techniques make different functional form assumptions of $g(\cdot)$, and possess different "iconic" properties: deep learning assumes that $g(\cdot)$ can be represented by a possibly sophisticated neural net and uses stochastic gradient descent to train the model; non-parametric methods learn a Lipschitz-continuous $g(\cdot)$ with statistical guarantees; boosting consolidates forecasts produced from computationally efficient weak learners.

The cost structure in our setting is different: in univariate models, $g(\mathbf{x}_{t,j})$ controls only one response $\hat{\mathbf{y}}_{t,j}$, whereas here, $g(\mathbf{x}_{t,j})$ impacts all responses $\hat{\mathbf{y}}_{t,i}$ for $i \in [d]$ because $\hat{\mathbf{y}}_{t,i} = \sum_j K_{i,j} g(\mathbf{x}_{t,j})$. We aim to generalize ML techniques under the new cost functions, while retaining the iconic properties of each technique.

Technique 1. Learn $g(\cdot)$ using neural nets. When an estimate $\hat{K}$ is given, the training cost is $\sum_{t,i}(\mathbf{y}_{t,i} - \sum_{j\in[d]} \hat{K}_{i,j} g(\mathbf{x}_{t,j}))^2$ and we can use stochastic gradient descent when $g(\cdot)$ is a neural net.

Technique 2. Learn $g(\cdot)$ using non-parametric methods. When the response is

univariate, e.g., $\mathbf{y}_{t,i} = g(\mathbf{x}_{t,i}) + \xi_{t,i}$, we can use a neighbor-based approach to estimate $g(\mathbf{x})$ for a new $\mathbf{x}$: we find one or multiple $\mathbf{x}_{t,i}$'s in the training set that are close to $\mathbf{x}$, and output $\mathbf{y}_{t,i}$ or their averages when multiple $\mathbf{x}_{t,i}$ are chosen, using $g(\mathbf{x}) \approx g(\mathbf{x}_{t,i})$ when $\mathbf{x}$ is close to $\mathbf{x}_{t,i}$.

Algorithm 1 nparam-gEST:

Input $\mathbf{X}$, $\mathbf{Y}$, $\hat{K}$; **Output** μ_1 (estimating other μ_i's is similar)

procedure NPARAM-GEST($\hat{K}, \mathbf{X}, \mathbf{Y}$)
  for all $t \leftarrow 1$ **to** n **do**
    $q_t = \text{Rand}(d)$
    $L_{(t,q_t),j} = \text{MAP-REGRESS}(q_t, \hat{K}, \mathbf{X}_{t,:})$
  end for
  return $\mu_1 \leftarrow \text{FLIPSIGN}\ (q_t, \{\mathbf{y}_t, L_{(t,q_t),j}\}_{t \le n})$
end procedure
procedure MAP-REGRESS($q_t, \hat{K}, \mathbf{x}_t$)
  Let $L_{(t,q_t),j} = 0$
  for all $k \leftarrow 1$ **to** d **do**
    $L_{(t,q_t),j} += \hat{K}_{q_t,k}$ with j s.t. $\mathbf{x}_{t,k} \in \Omega_j$.
  end for
  return $L_{(t,q_t),j}$
end procedure
procedure FLIPSIGN($q_t, \{\mathbf{y}_t, L_{(t,q_t),j}\}_{t \le n}$)
  for all $t \leftarrow 1$ **to** n **do**
    $\hat{\Pi}_1^{(q_t)}(t) \triangleq L_{(t,q_t),1} - \frac{1}{\ell - 1}\left(\sum_{j \neq 1} L_{(t,q_t),j}\right)$
    $$\tilde{b}_{t,q_t} = \begin{cases} 1 & \text{if } \hat{\Pi}_1^{(q_t)}(t) \ge \frac{c}{\log d}\sqrt{\frac{d}{\ell}} \\ -1 & \text{if } \hat{\Pi}_1^{(q_t)}(t) < -\frac{c}{\log d}\sqrt{\frac{d}{\ell}} \\ 0 & \text{otherwise} \end{cases}$$
  end for
  return $\mu_1 = \frac{\sum_{t \le n} \tilde{b}_{t,q_t} \mathbf{y}_{t,q_t}}{\sum_{t \le n} \tilde{b}_{t,q_t} \hat{\Pi}_1^{(q_t)}(t)}$
end procedure

Here, we do not directly observe the values of individual $g(\mathbf{x}_{t,i})$'s. Instead, each response is a linear combination of multiple $g(\cdot)$'s evaluated at different points, e.g., $\mathbf{y}_{t,1} = K_{i,1} \cdot g(\mathbf{x}_{t,1}) + \cdots + K_{i,d} \cdot g(\mathbf{x}_{t,d}) + \xi_{t,i}$. We show that finding neighbors reduces to solving a linear system. Furthermore, we design a moment-based algorithm, namely

"nparam-gEST", which estimates $g(\cdot)$ with provable guarantees.

Proposition 3.3.2. *Consider the problem of learning additive influence model with the same setup/parameters as in Prop. 3.3.1. Assume that $\mathbf{x}_{t,i} \in \mathbf{R}^{O(1)}$. Let ℓ be a tunable parameter. There exists an efficient algorithm to compute $\hat{g}(\cdot)$, based on $\hat{K}$ such that $\sup_{\mathbf{x}} |\hat{g}(\mathbf{x}) - g(\mathbf{x})| \leq (\log^6 n)\big(\sqrt{\gamma} + \sqrt{\frac{\ell}{n}} + \frac{1}{\ell}\big) = \tilde{O}(d^{-c})$ for suitable parameters, where $\gamma \triangleq \frac{\epsilon^2}{\delta^3} + \delta^{\frac{4}{5}}$.*

Our algorithm (Alg. 1) consists of the following 3 steps:

Step 1. Approximation of $g(\cdot)$. Partition $\Omega = [-1, 1]^k$ into subsets $\{\Omega_j\}_{j \leq \ell}$, and use piece-wise constant function to approximate $g(\cdot)$, i.e., $\tilde{g}(\mathbf{x}_{t,i})$ take the same value for all $\mathbf{x}_{t,i}$ in the same Ω_j. We partition $\{\Omega_j\}_{j \leq \ell}$ in a way such that $\Pr[\mathbf{x}_{t,i} \in \Omega_j]$ are the same for all j.

Step 2. Reduction to linear regression. Each observation can be construed as a linear combination of μ_j's ($j \in [\ell]$), where $\mu_j = \mathbb{E}[g(\mathbf{x}_{t,i}) \mid \mathbf{x}_{t,i} \in \Omega_j]$. For example, $\mathbf{y}_{t,1} = \sum_{i \leq d} K_{1,i}\mu_{j_i} + \xi_{t,1} + o(1)$, where $\mathbf{x}_{t,i} \in \Omega_{j_i}$, and in general, we have

$$\mathbf{y}_{t,i} = \sum_{j \leq \ell} L_{(t,i),j}\mu_j + \xi_{t,i} + o(1), \tag{3.2}$$

$$\text{where } L_{(t,i),j} = \sum_{m \in \mathcal{L}_{t,j}} K_{i,m} \text{ and } \mathcal{L}_{t,j} = \{m : \mathbf{x}_{t,m} \in \Omega_j\}.$$

Therefore, our learning problem reduces to a linear regression problem, in which the $L_{(t,i),j}$'s are features and the $\{\mu_j\}_{j \leq \ell}$ are coefficients to be learned.

Step 3. Moment-based estimation. An MSE-based estimator is consistent but finding its confidence interval (error bound) requires knowing the spectrum of the features' covariance matrix, which is remarkably difficult in our setting. Therefore, we propose a moment-based algorithm with provable performance (FlipSign in Alg. 1).

We illustrate each steps above through a toy example, in which we assume $K_{i,j} = 1$ for all i and j so the model simplifies to $\mathbf{y}_{t,1} = \sum_{j \leq d} g(\mathbf{x}_{t,j}) + \xi_{t,1}$. See Fig. 3.2.

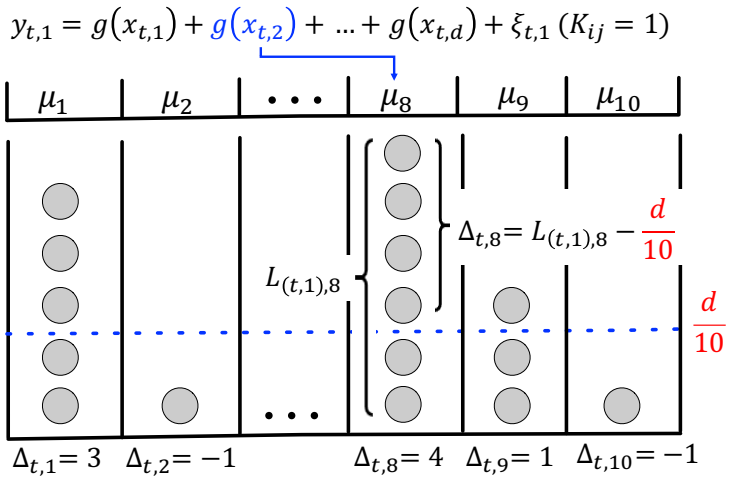


Figure 3.2: A toy example of nparam-gEST when $K_{i,j} = 1$ for all i and j and $\Omega = [-1, 1]$ and is uniformly partitioned into 10 pieces. Sampling a $g(\mathbf{x}_{t,i})$ corresponds to randomly placing a ball into a total number of 10 bins. For example, $\mathbf{x}_{t,2}$ falls into the 8-th interval so μ_8 is used to approximate $g(\mathbf{x}_{t,2})$, which may be viewed as a new ball of type μ_8 (or in 8-th bin) is created. The mean load for each bin is $d/\ell = d/10$. We calculate $\sum_{i \le d} g(\mathbf{x}_{t,i})$ by counting the balls in each bin: $\mathbf{y}_{t,1} = 5 \times \mu_1 + 1 \times \mu_2 + \ldots + 6 \times \mu_8 + 3 \times \mu_9 + 1 \times \mu_{10} + \xi_{t,1}$.

First, we view the generation of samples as a balls-and-bins process so that the $g(\cdot)$-estimation problem reduces to a regression problem (Steps 1 & 2). Specifically, we generate $(\mathbf{y}_{t,1}, \{\mathbf{x}_{t,i}\}_{i \le d})$ as first sequentially sampling $\{\mathbf{x}_{t,i}\}_{i \le d}$ and computing the corresponding $g(\mathbf{x}_{t,i})$, then summing each term up together with $\xi_{t,1}$ to produce $\mathbf{y}_{t,1}$. When an $\mathbf{x}_{t,i}$ is sampled, it falls into one of Ω_i's with uniform probability. Let j_i be the bin that $\mathbf{x}_{t,i}$ falls into. Then $g(\mathbf{x}_{t,i})$ is approximated by μ_{j_i} according to Step 1. Thus, we may view a ball of "type μ_{j_i}" (or in j_i-th bin) is created. For example, in Fig. 3.2, $\mathbf{x}_{t,2}$ falls into the 8-th interval so a ball is added in the 8-th bin. After all $\mathbf{x}_{t,i}$'s are sampled, compute $\mathbf{y}_{t,1}$ by counting the numbers of balls in different bins. Recalling that the load of j-th bin is $L_{(t,1),j}$, we have $\mathbf{y}_{t,1} \approx \sum_{j \le d} L_{(t,1),j} \cdot \mu_j + \xi_{t,1}$. Let $\Delta_{t,j} = L_{(t,1),j} - d/\ell$ and using that $\mathbb{E}[L_{(t,1),j}] = d/\ell$ and $\sum_{j \le d} \mu_j = 0$, we have

$$\mathbf{y}_{t,1} = \Delta_{t,1}\mu_1 + \cdots + \Delta_{t,\ell}\mu_\ell + \xi_{t,1}. \tag{3.3}$$

Eq. (3.3) is a standard (univariate) regression: for each t, we know $\mathbf{y}_{t,1}$, and know all $\Delta_{t,j}$'s because all $\mathbf{x}_{t,j}$'s are observed so the number of balls in each bin can be calculated. We need to estimate the unknown μ_j's. Note that $\mathbb{E}[\Delta_{t,j}] = 0$.

Next, we solve the regression (Step 3). Our algorithm "tweaks" the observations so that the features associated with μ_1 are always positive: let $b_{t,1} = 1$ if $\Delta_{t,1} > 0$ and -1 otherwise. Multiply $b_{t,1}$ to both sides of Eq. (3.3) for each t:

$$b_{t,1}\mathbf{y}_{t,1} = |\Delta_{t,1}|\mu_1 + \cdots + b_{t,1} \cdot \Delta_{t,\ell} \cdot \mu_\ell + b_{t,1}\xi_{t,1} \tag{3.4}$$

We sum up lhs and rhs of (3.4) and obtain

$$\sum_{t\le n} b_{t,1}\mathbf{y}_{t,1} = (\sum_{t\le n} |\Delta_{t,1}|)\mu_1 + \cdots + (\sum_{t\le n} b_{t,1} \cdot \Delta_{t,\ell})\mu_\ell + \sum_{t\le n} b_{t,1}\xi_{t,1} \tag{3.5}$$

Next, we have $\sum_{t\le n} |\Delta_{t,1}| = \Theta(n)$ whp. Also, we can see that $b_{t,1}$ and $\Delta_{t,j}$ are "roughly" independent for $j \neq 1$ (careful analysis will make it rigorous). Therefore, for any $j \neq 1$, $\mathbb{E}[b_{t,1}\cdot\Delta_{t,j}] = 0$, and thus $\sum_{t\le n} b_{t,1}\cdot\Delta_{t,j} = O(\sqrt{n})$ whp. Now (3.5) becomes $\sum_{t\le n} b_{t,1}\cdot\mathbf{y}_{t,1} = \big(\sum_t |\Delta_{t,1}|\big)\mu_1 + O(\ell\cdot\sqrt{n})$. So our estimator is $\hat{\mu}_1 \triangleq \frac{\sum_t b_{t,1}\cdot\mathbf{y}_{t,1}}{\big(\sum_t |\Delta_{t,1}|\big)} = \mu_1 + \frac{O(\ell\cdot\sqrt{n})}{\Theta(n)} = \mu_1 + O\big(\frac{\ell}{\sqrt{n}}\big)$. Here analysis of covariance for $\Delta_{t,j}$'s is circumvented because $\Delta_{t,j}$'s interactions are compressed into the term $O\big(\frac{\ell}{\sqrt{n}}\big)$. We remark that the analysis contains some crude steps and can be tightened up (App. 3.9.2).

Technique 3. Learn $g(\cdot)$ using boosting. In the univariate setting, we have $\mathbf{y}_{t,i} = \sum_{m\le b} g_m(\mathbf{x}_{t,i}) + \xi_{t,i}$, in which each $g_m(\mathbf{x}_{t,i})$ is a weak learner. Standard boosting algorithms [128, 29] assume that each $g_m(\cdot)$ is represented by a regression tree and constructed sequentially. A greedy strategy is used to build a new tree e.g., iteratively splitting a node in a tree by choosing a variable that optimizes prediction improvement. In our setting, $\mathbf{y}_{t,i}$ depends on evaluating $g_m(\cdot)$ at d different locations $\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,d}$, so the splitting procedure either is d (3000) times slower in a standard implementation, or requires excessive engineering tweak of existing systems.

Here, we propose a simple and effective weak learner based on intuition of the tree structure and equity data. Let

$$(\mathbf{x}_t)_i = \big((\mathbf{x}_{t,1})_i, (\mathbf{x}_{t,2})_i, \ldots, (\mathbf{x}_{t,d})_i\big) \in \mathbf{R}^d,$$
$$(\mathbf{x}_t)_{i,j} = \big((\mathbf{x}_{t,1})_i \cdot (\mathbf{x}_{t,1})_j, \ldots, (\mathbf{x}_{t,d})_i \cdot (\mathbf{x}_{t,d})_j\big) \in \mathbf{R}^d,$$

and $(\mathbf{x}_t)_{i,j,k}$ can be defined in a similar manner. We observe that regression trees used in GBRT models for equity return are usually shallow and can be *linearized* (See Fig. 3.3): we may unfold a tree into disjunctive normal form (DNF) [1], and approximate the DNF by a sum of multiple interaction terms, e.g., $I((\mathbf{x}_{t,i})_1 > 0) \cdot I((\mathbf{x}_{t,i})_2 > 0)$ can be approximated by $(\mathbf{x}_{t,i})_1 \cdot (\mathbf{x}_{t,i})_2$.

Our algorithm, namely LIN-PVEL (linear projected vector ensemble learner), consists of weak learners in linear forms. Each linear learner consists of a subset of features and their interactions. The number of features included and the depth of their interactions are hyper-parameters corresponding to the depth of the decision tree. For example, if the first three features are included in the learner, we then need to fit $\mathbf{y}_{t,i} \sim$

$$\sum_{j \in [d]} \underbrace{\hat{K}_{i,j}}_{\text{given}} \cdot \Big[\underbrace{\beta_1 (\mathbf{x}_{t,j})_1 + \ldots}_{\text{linear terms}} \underbrace{+ \beta_4 (\mathbf{x}_{t,j})_{1,2} + \cdots + \beta_7 (\mathbf{x}_{t,j})_{1,2,3}}_{\text{interaction terms}} \Big], \tag{3.6}$$

by MSE. Conceptually, although we use linearized models to approximate the trees, the "target" trees are unavailable (for the computational efficiency reasons above). We need a new procedure to select features for each learner. Our intuition is that, in equity data sets, if an interaction term could have predictive power, each feature involved in the interaction should also have predictive power. Our procedure is simply to select a fixed number of i's with the largest $\text{corr}((\mathbf{y}_{\text{Res}})_t, \hat{K}(\mathbf{x}_t)_i)$, where $(\mathbf{y}_{\text{Res}})_t$ is the residual error. See Table. 3.1.

Domain knowledge of the equity data sets is used in our design. First, using feature interactions to approximate DNF ($I((\mathbf{x}_{t,i})_1 > 0) \cdot I((\mathbf{x}_{t,i})_2 > 0) \approx (\mathbf{x}_{t,i})_1 \cdot (\mathbf{x}_{t,i})_2$ may not always be accurate). In our setting, however, linear interaction models often outperform decision trees or DNFs. We believe this occurs because interaction terms are continuous (whereas DNFs are discrete functions), and thus they are more suitable to model smooth price changes. Second, we use the predictive power of linear terms to select the variables, which is effective for equity return models because, in part, the features are constructed from trading activity data (i.e., technical factors) and have economics interpretation, and

may not remain valid for generic machine learning problems.

Steps	GBRT	LIN-PVEL
1. Approximation	Multiple trees	Linear model with interaction terms
2. Choose variables	Split algorithm to determine variable selections and trees.	Choose three i's with largest $\text{corr}((\mathbf{y}_{\text{Res}})_t, K(\mathbf{x}_t)_i)$ Assume $K(\mathbf{x}_t)_{i,j}$ is predictive, then either $K(\mathbf{x}_t)_i$ or $K(\mathbf{x}_t)_j$ is predictive.
3. Determine the model		Fit a linear model with interaction terms (Eq. 3.6)
4. Find residual errors	$\mathbf{Y}_{\text{Res}} \leftarrow \mathbf{Y}_{\text{Res}} - \eta(\hat{\mathbf{Y}})_k$	Same as GBRT.

Table 3.1: Comparison between GBRT and LIN-PVEL.

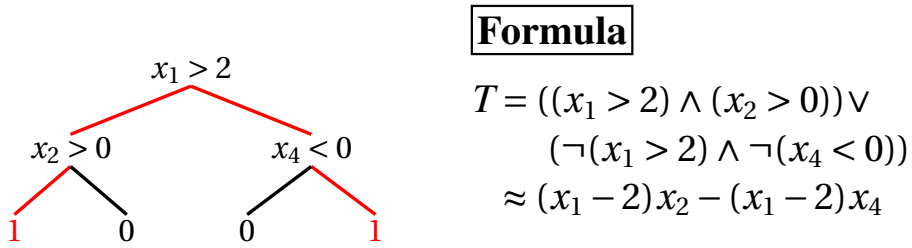


Figure 3.3: An example of representing trees as a DNF formula.

		Universe 800				*Full universe*				Backtesting	
Models	Our CAMs	corr	w_corr	t-stat	w_t-stat	corr	w_corr	t-stat	w_t-stat	PnL	Sharpe
LIN-PVEL	Opt.	**0.0764**	**0.0936**	**6.7939**	**6.3362**	**0.0944**	**0.1009**	**8.2607**	**6.4435**	**0.5261**	**10.97**
	DD	0.0692	0.0874	6.3141	5.8140	0.0934	0.1000	9.0551	6.9076	0.5181	11.80
nparam-gEST	Opt.	**0.0446**	**0.0320**	**3.2961**	**1.5753**	**0.0618**	**0.0553**	**5.7327**	**3.5212**	**0.3386**	**7.59**
	DD	0.0445	0.0307	3.2781	1.5399	0.0603	0.0507	5.0648	3.0551	0.3330	7.04
MLP	Opt.	**0.0550**	**0.0567**	**6.4782**	**5.0172**	**0.0738**	**0.0692**	**9.2034**	**6.4151**	**0.4202**	**9.43**
	DD	0.0540	0.0562	6.4396	4.9953	0.0671	0.0640	8.9526	6.1564	0.3493	8.16
LSTM	Opt.	**0.0286**	**0.0347**	**3.4517**	**3.0261**	**0.0473**	**0.0491**	**6.3615**	**4.2385**	**0.2487**	**7.10**
	DD	0.0231	0.0352	2.6108	2.9779	0.0415	0.0429	6.0979	4.0582	0.2012	5.62
Linear	Opt.	**0.0449**	**0.0517**	**4.802**	**4.5514**	**0.0548**	**0.056**	**6.3589**	**4.9856**	**0.3218**	**6.47**
UM: poor man Lin-PVEL		0.0674	0.0866	6.0947	5.7312	0.0827	0.0884	7.4297	5.6659	0.4565	9.76
UM: poor man nparam-gEST		0.0432	0.0309	3.1505	1.4912	0.0584	0.0509	5.0098	3.0844	0.3070	6.59
UM: MLP		0.0507	0.5050	6.0234	4.4966	0.0606	0.0467	8.2857	4.4555	0.2782	6.38
UM: LSTM		0.0178	0.0200	2.2136	1.8077	0.0352	0.0297	4.0602	2.3619	0.175	4.33
UM: Lasso		0.0106	0.0192	1.6471	2.3030	0.0290	0.0251	4.4711	2.6010	0.1888	4.79
UM: Ridge		0.0247	0.0246	2.3553	1.9628	0.0358	0.0406	4.5172	3.6941	0.1839	3.98
UM: GBRT		0.0516	0.0591	7.5739	5.6310	0.0673	0.0747	9.3379	7.8931	0.3858	4.45
UM: SFM		0.0027	0.0032	0.4688	0.4050	0.0147	0.0051	1.2683	0.3892	0.0169	0.54
Existing CAM: VR		0.0156	0.0159	2.4997	1.7046	0.0041	-0.0025	0.8847	-0.3021	0.0430	1.20
Existing CAM: ARRR		0.0314	0.0382	2.5336	2.4213	0.0222	0.0273	1.8557	1.8968	0.1674	3.24
Existing CAM: AlphaStock		0.0085	0.0063	2.1045	1.2516	0.0027	0.0032	0.4688	0.4050	0.0045	0.10
Existing CAM: HAN		0.0105	0.0081	1.7992	1.0017	0.0080	0.0050	1.5716	0.7340	0.0570	2.02
Consolidated: all		**0.0775**	**0.0950**	**6.8687**	**6.4108**	**0.0958**	**0.1025**	**8.5703**	**6.6487**	**0.5346**	**11.30**

Table 3.2: Summary of results for equity raw return forecasts. LIN-PVEL is the gradient boosting method with linear learner. Bold face denotes the best performance in each group. DD denotes the method using Alg. 2. Opt. denotes the optimal results from different estimators of K (App. 3.8.3). Backtesting results pertain to the *Full universe*. See App. 4.6.

3.4 Related work and comparison

Recent ML works. Univariate machine learning models for handling feature-interactions include [165, 39, 167, 51, 68, 62, 28, 81, 80, 26, 92]. The models mostly rely on deep learning, and some use transfer learning techniques (i.e., use one stock's data to build a model for another stock) [26, 92] but they are not CAMs (cannot use one stock's features to predict another's return). Recent cross-asset models are mostly linear models [20, 84, 118, 161, 73] that have theoretical guarantees, but they cannot automatically extract signals from feature interactions. Efforts for building non-linear CAMs include [155, 49, 72] but [72, 155] have reproducibility issues (see Sec. 3.5) whereas [49] uses non-market information.

Serial correlations and risk. We do not consider serial correlation in modeling, even though it is considered in some *linear* CAMs [67, 97]. We remark that • *(i)* the standard tools used to "de-serialize" data (e.g., using residuals of AR as response) do not help in our settings, and • *(ii)* it remains an open problem to design a model that simultaneously leverages stock-interactions, feature-interactions, and serial correlations.

Our algorithm relies on $\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$ (i.e., the risk matrix [88]) for learning the embedding. While the risk matrix was extensively studied and sometimes used to produce forecasting models [19, 88]), our model is conceptually new: a *provable* embedding can be learned from the risk matrix, and it can be supplied to downstream ML $g(\cdot)$-learners to build CAMs.

3.5 Evaluation

Experimental setup. We use 10 years of equity data from the Chinese market to evaluate our algorithms and focus on predicting the next 5-day returns, in which the last three years are out-of-sample. The test period is substantially longer than those in recent works [167, 72, 92]. We constructed 337 standard technical factors to serve as a feature database for all models (App. 3.15). We consider two universes: (i) *Universe 800* can be construed as an equivalence to the S&P 500 in the US, and consists of 800 stocks, and (ii) *Full universe* consists of all stocks except for the very illiquid ones. Visualizations are

shown in App. 4.6.

We next explain our metrics and argue why they are different from those for standard ML problems (see App. 3.13.2) • *(i) Correlation vs MSE.* While the MSE is a standard metric for regression problems, correlations are better suited metrics for our setting [169]. • *(ii) Significance testing.* The use of t-statistics estimators [119] can account for the serial and cross-sectional correlations (App. 3.13.2) • *(iii) Stock capacity/liquidity considerations.* Predicting illiquid stocks is less valuable compared to predicting liquid ones because they cannot be used to build large portfolios. We use a standard approach to weight correlations (w_corr) and t-statistics by a function of historical *notional (dollar) traded volume* to reflect the capacity of the signals.

New CAMs from our model. We estimate K and $g(\cdot)$ separately. To estimate K, we use both the algorithm discussed in Sec. 3.3.1 and other refinements discussed in App. 3.8.3. To estimate $g(\cdot)$, we use SGD-based algorithms (MLP, LSTM, and linear), nparam-gEST, and Lin-PVEL.

Baselines. *(i) The UMs* include linear, MLP, LSTM, GBRT, and SFM [167]. We also implement a "poor man's version" of both Lin-PVEL and nparam-gEST for UM, which assumes that influences from other stocks are 0; *(ii) The CAMs* include a standard linear VAR [118], ARRR [161], AlphaStock [155], and HAN [72].

Results. See Table 3.2 for the results and Fig. 3.4 for the simulated Profit & Loss (PnL). App. 3.13 describes additional details and experiments. The experiments confirm that • *(i)* generic ML techniques are effective for UMs but ineffective for CAMs; • *(ii)* CAMs produced out of our model consistently outperform prior works. In addition, our Lin-PVEL model has the best performance; • *(iii)* By using a simple consolidation algorithm, the aggregated signal outperforms all individual ones. Our new models pick up signals that are orthogonal to existing ones because we rely on a new mechanism to use stock and feature interactions.

Discussion about prior evaluations. SFM ([167]) is open-sourced, and other studies reported similar performance of SFM. SFM, HAN, and AlphaStock did not compute

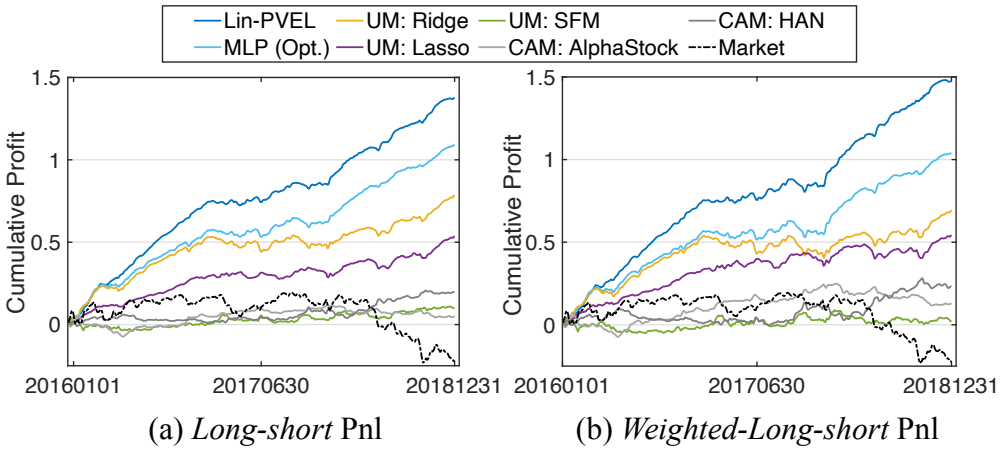


Figure 3.4: Cumulative PnL (Profit and Loss) curves of the top quintile portfolio from *Full universe* (i.e., on any given day, we consider a portfolio with only the top 20% strongest predictions in magnitude, against future *market excess returns*). See App. 4.6

t-statistics. HAN's PnL is positive, but its signal is not significant. AlphaStock cannot beat the market and generates negative PnL returns since 2010.

3.6 Conclusion

This paper proposes an additive influence model for equity returns that enables us to decouple the learning of stock-interactions from the learning of feature-interactions. Our upstream stock-interaction learner has provable performance guarantees, thanks to the deployment of high-dim and kernel learning techniques, whereas our downstream $g(\cdot)$-learners can leverage a wide set of effective ML techniques. Our algorithms are proven to be superior to the existing baselines, especially those CAMs developed recently. App. 3.14 gives the answers to commonly asked questions (e.g., "why don't we trade ourselves").

3.7 Additional notes on problem definition

Independence of $\mathbf{x}_{t,i}$. Our analysis assumes that $\mathbf{x}_{t,i}$ are independent across t's and i's. Our discussion assumes that $\mathbf{x}_{t,i} \in \mathbf{R}$. The arguments can easily generalize to multi-dimensional $\mathbf{x}_{t,i}$. When $\mathbf{x}_{t,i}$ are correlated across stocks, we can apply a factor model to obtain

$$\mathbf{x}_t = L\mathbf{f}_t + \tilde{\mathbf{x}}_t, \tag{3.7}$$

where $\mathbf{x}_t = (\mathbf{x}_{t,1}, \mathbf{x}_{t,2}, \ldots, \mathbf{x}_{t,d}) \in \mathbf{R}^d$, $\mathbf{f}_t$ is a low-dimensional vector that explains the co-moving (correlated) components, L is the factor loading matrix, and $\tilde{\mathbf{x}}_t = (\tilde{\mathbf{x}}_{t,1}, \ldots, \tilde{\mathbf{x}}_{t,d}) \in \mathbf{R}^d$ is the idiosyncratic component. There exists a rich literature on algorithms that identify latent factors [34, 45, 74, 78]. The shared factors driving the co-movements of the features can be utilized in other ways to forecast equity returns [111]. We can use the idiosyncratic component $\tilde{\mathbf{x}}_t$ as input features in our model, since the coordinates in $\tilde{\mathbf{x}}_t$ are independent. In the setting where serial correlation is presented in $\tilde{\mathbf{x}}_t$, one can use the standard differencing operator for decorrelating purposes [66].

3.8 Estimation of K

We prove Proposition 3.3.1 and explain other variations of estimating K. For exposition purposes, our analysis focuses on the case where κ is Gaussian kernel or IMQ. The case for κ being an inner product function can be analyzed in a similar manner. See also Remark at the end of this section.

In Sec. 3.8.1, we first describe the background (e.g., notation and building blocks) needed. In Sec. 3.8.1, we present our proof for Prop 3.3.1. Our analysis assumes that $n \leq d^2$ to simplify calculations and ease the exposition. The case $n \geq d^2$ corresponds to the scenario when abundant samples are available, and is easier to analyze. In Sec. 3.8, we explain additional algorithms for estimating K.

3.8.1 Background

Notation. Let $A = \frac{1}{d^2}K^{\mathrm{T}}K$ and $B = \frac{1}{d^2 n}Y^{\mathrm{T}}Y$. Let V_k^A be the first k eigenvectors associated with A and V_k^B be the first k eigenvectors associated with B. Note that A and B are symmetric. Let $\mathcal{P}_A = V_{i^*}^A (V_{i^*}^A)^{\mathrm{T}}$ and $\mathcal{P}_B = V_{i^*}^B (V_{i^*}^B)^{\mathrm{T}}$, where i^* is defined in Alg. 2.

Distance between matrices. For any positive-definite matrix A, there could be multiple square roots of A (the square root is defined as any matrix B such that $BB^{\mathrm{T}} = A$). Any pair of square roots of the same matrix differ only by a unitary matrix and should be considered as "the same" in most of our analysis. We adopt the following (standard) definition to measure the difference between two matrices.

Definition 3.8.1. *(Distance between two matrices) Let $X, Y \in \mathbf{R}^{d_1 \times d_2}$. The distance between X and Y is defined as*

$$\mathrm{Dist}^2(X, Y) = \min_{W \text{ unitary}} \|XW - Y\|_F^2. \tag{3.8}$$

Building blocks related to distances.

Lemma 3.8.2. *(From [18]) For any two rank-r matrices U and X, we have*

$$\mathrm{Dist}^2(U, X) \leq \frac{1}{2(\sqrt{2}-1)\sigma_r^2(X)} \|UU^{\mathrm{T}} - XX^{\mathrm{T}}\|_F^2.$$

Lemma 3.8.3. *(From [56]) Let M_1 and M_2 be two matrices such that*

$$M_1 = U_1 D_1 V_1^{\mathrm{T}} \quad \text{and} \quad M_2 = U_2 D_2 V_2^{\mathrm{T}}. \tag{3.9}$$

It holds true that

$$\|U_1 D_1 U_1^{\mathrm{T}} - U_2 D_2 U_2^{\mathrm{T}}\|_F^2 + \|V_1 D_1 V_1^{\mathrm{T}} - V_2 D_2 V_2^{\mathrm{T}}\|_F^2 \leq 2\|M_1 - M_2\|_F^2. \tag{3.10}$$

Building block related to gap vs. tail.

Lemma 3.8.4. *Let $\{\lambda_i\}_{i \geq 1}$ be a sequence such that $\sum_{i \geq 1} \lambda_i = 1$, $\lambda_i \leq c i^{-\omega}$ for some constant c and $\omega \geq 2$. Assume also that $\lambda_1 < 1$. Define $\delta_i = \lambda_i - \lambda_{i+1}$, for $i \geq 1$. Let δ_0 be a sufficiently small number, and c_1 and c_2 be two suitable constants. For any $\delta < \delta_0$, there exists an i^* such that $\delta_{i^*} \geq \delta$ and $\sum_{j \geq i^*} \lambda_j = O\left(\delta^{\frac{4}{5}}\right)$.*

Kernel learning. Let $\kappa(\mathbf{x}, \mathbf{x}')$ be a smooth radial basis function, i.e., $\kappa(\mathbf{x}, \mathbf{x}') = \kappa(\|\mathbf{x} - \mathbf{x}'\|)$, and use the notation $f(\cdot) = \kappa(\sqrt{\cdot})$. We assume that $|f^{(\ell)}(r)| \leq \ell! M^{\ell}$, for all ℓ sufficiently large and $r > 0$. Note that both Gaussian kernels and inverse multi-quadratic kernels satisfy this property.

Define an integral operator $\mathcal{K}$ as

$$\mathcal{K}f(\mathbf{x}) = \int \kappa(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') dF(\mathbf{x}'), \tag{3.11}$$

where $F(\cdot)$ is the cumulative probability function over the support of $\mathbf{x}$. Let $\mathcal{H}$ be the rank of $\mathcal{K}$, which can be either finite or countably infinite. Let $\psi_1, \psi_2, \ldots, \psi_{\mathcal{H}}$ be the eigenfunctions of $\mathcal{K}$, and $\lambda_1, \lambda_2, \ldots, \lambda_{\mathcal{H}}$ be the corresponding eigenvalues such that $\lambda_1 \geq \lambda_2 \geq \ldots$. Let $K \in \mathbf{R}^{d \times d}$ be the Gram matrix such that $K_{i,j} = \kappa(\|\mathbf{z}_i - \mathbf{z}_j\|)$.

Our analysis relies on the following two key building blocks.

Lemma 3.8.5. *([15]) Let λ_i^* be the i-th eigenvalue of $\mathcal{K}$. There exist constants C and C' such that*

$$\lambda_i^* \leq C' \exp(-C i^{\frac{1}{r}}). \tag{3.12}$$

Lemma 3.8.6. *Let λ_i^* be the i-th eigenvalue of $\mathcal{K}$. Let $\lambda_i(K)$ be the i-th eigenvalue of K. Let $\hat{\lambda}_j = \lambda_j(K)/d$. Let $\tau > 0$ be a tunable parameter. With probability at least $1 - \exp(-c_0\tau)$ for some constant c_0, it holds true that*

$$\left(\sum_{j \geq 1} (\lambda_j^* - \hat{\lambda}_j)^2 \right)^{\frac{1}{2}} \leq 2\sqrt{\frac{\tau}{d}}. \tag{3.13}$$

In addition, with probability at least $1 - \exp(-c_0\tau)$,

$$\left(\sum_{j \geq 1} \left((\lambda_j^*)^2 - (\hat{\lambda}_j)^2 \right)^2 \right)^{\frac{1}{2}} \leq c\sqrt{\frac{\tau}{d}} \tag{3.14}$$

for some constant c.

Proof of Lemma 3.8.6. Eq. 3.13 is from Theorem B.2 from [147]. Now to prove Eq. 3.14, we have

$$\left(\sum_{j \geq 1} \left((\lambda_j^*)^2 - (\hat{\lambda}_j)^2 \right)^2 \right)^{\frac{1}{2}} = \left(\sum_{j \geq 1} (\lambda_j^* - \hat{\lambda}_j)^2 (\lambda_j^* + \hat{\lambda}_j)^2 \right)^{\frac{1}{2}} \leq \; c' \left(\sum_{j \geq 1} (\lambda_j^* - \lambda_j)^2 \right)^{\frac{1}{2}} \leq c\sqrt{\frac{\tau}{d}}.$$

□

3.8.2 Proof for Prop 3.3.1

Algorithm 2 Data-driven (DD) estimation of K

Input $\mathbf{X}, \mathbf{Y}$; **Output** $\hat{K}$

1: $[V, \Sigma, V^{\mathrm{T}}] = \mathrm{svd}\left(\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\right)$

2: Let $\sigma_i = \Sigma_{i,i}$

3: $i^* = \max\{i : \sigma_{i^*} - \sigma_{i^*+1} \geq \delta d^2\}$ ▷ δ is tunable

4: **return** $\hat{K} = \mathcal{P}_{i^*}(V\Sigma^{\frac{1}{2}}V^{\mathrm{T}})$

Our analysis consists of four steps:

- **Step 1.** Show that $\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y} - K^{\mathrm{T}}K$ is sufficiently small.
- **Step 2.** Show that a low rank approximation of $\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$ is sufficiently close to $\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$.
- **Step 3.** Show that $\mathcal{P}_{i^*}(\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y})$ is close to $\mathcal{P}_{i^*}(\mathbf{K}^{\mathrm{T}}\mathbf{K})$.
- **Step 4.** Use results from the first three steps, together with Lemma 3.8.4, to prove the first part of Theorem 3.3.1.

Step 1. $\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$ **and** $K^{\mathrm{T}}K$ **are close.** To formally prove this step, we rely on the following proposition.

Proposition 3.8.7. *Consider the problem of learning the stock latent embedding model. Let n be the number of observations. Let $\mathbf{Y} \in \mathbf{R}^{n\times d}$ be such that $\mathbf{Y}_{i,:}$ contains the i-th observation. Assume that $n \leq d^2$ and $\sigma_\xi = O(\sqrt{d})$. With overwhelming probability, it holds true that*

$$\left\|\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y} - K^{\mathrm{T}}K\right\|_F = O\left(\frac{d^2\log^3 n}{\sqrt{n}}\right). \tag{3.15}$$

Proving Proposition 3.8.7 requires a standard manipulation of concentration inequalities for matrices. See the proof in App. 3.12.1.

Step 2. $\mathcal{P}_{i^*}(K^{\mathrm{T}}K)$ **is close to** $K^{\mathrm{T}}K$**.**

Lemma 3.8.8. *There exists a sufficiently large d_0 so that when $d \geq d_0$, Algorithm 2 always terminates. In addition, it holds true that*

$$\sum_{i\geq i^*}\lambda_i^2\left(\frac{K}{d}\right) = O(\delta^{\frac{4}{5}}).$$

Proof. Let $\tilde{\delta} = 10\delta \geq \frac{c \log^3 d}{\sqrt{d}}$ for a suitably large c. By Lemma 3.8.4, we have that there exists an $\tilde{i}$ such that

1. $(\lambda^*_{\tilde{i}})^2 - (\lambda^*_{\tilde{i}+1})^2 \geq \tilde{\delta}$.
2. $\sum_{i \geq \tilde{i}} (\lambda^*_i)^2 \leq \left(\tilde{\delta}\right)^{\frac{4}{5}}$.

We first show that the algorithm terminates. We have that

$$\left|(\lambda^*_i)^2 - \lambda_i^2(K/d)\right| = O\left(|\lambda^*_i - \lambda_i(K/d)|\right) = O\left(\sqrt{\frac{\log d}{d}}\right)$$

The last equality uses Proposition 3.8.7. Next, by using Lemma 3.8.4, we have

$$\left|\lambda_i\left(\frac{1}{nd^2}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\right) - \lambda_i\left(\frac{K^2}{d^2}\right)\right| = O\left(\frac{\log^3 n}{\sqrt{n}}\right).$$

Therefore, we can also see that

$$\lambda_{\tilde{i}}\left(\frac{1}{nd^2}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\right) - \lambda_{\tilde{i}+1}\left(\frac{1}{nd^2}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\right) \geq \delta.$$

Our algorithm always terminates. In addition, we have $i^* \geq \tilde{i}$. Finally, we have

$$\begin{aligned}\sum_{i \geq i^*} \lambda_i^2\left(\frac{K}{d}\right) &\leq \sum_{i \geq \tilde{i}} \lambda_i^2\left(\frac{K}{d}\right) \\ &\leq 2\left(\sum_{i \geq \tilde{i}} (\lambda^*_{\tilde{i}})^2 + \left(\lambda^*_{\tilde{i}} - \lambda_i\left(K/d\right)\right)^2\right) \\ &= O\left(\tilde{\delta}^{\frac{4}{5}}\right) = O\left(\delta^{\frac{4}{5}}\right).\end{aligned}$$

□

Step 3. Analysis of the projection. To show that $\mathcal{P}_{i^*}\left(\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\right)$ and $\mathcal{P}_{i^*}(K^{\mathrm{T}}K)$ are close. We have the following lemma.

Lemma 3.8.9. *Consider running Algorithm* ESTIMATE-K *in Alg. 2 for estimating K. Let $A = \frac{1}{d^2}K^{\mathrm{T}}K$ and $B = \frac{1}{d^2 n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$. Let $\mathcal{P}_A$ and $\mathcal{P}_B$ be defined as above. Let $\epsilon \leq \frac{c_0 \log^3 d}{\sqrt{d}}$ for some constant c_0, and δ be the gap parameter in Alg. 2 such that $\delta^3 = \omega(\epsilon^2)$. With high probability, we have*

$$\|\mathcal{P}_A - \mathcal{P}_B\|_2 = O\left(\frac{\|A - B\|_2}{\delta}\right). \tag{3.16}$$

Proof of Lemma 3.8.9. Define $S_1 = [\lambda_{i^*}(A) - \delta/10, \infty)$ and $S_2 = [0, \lambda_{i^*}(A) + \delta/10]$. By Lemma 3.8.7, we have $\left\|\frac{1}{nd^2}\mathbf{Y}^{\mathrm{T}}\mathbf{Y} - \frac{1}{d^2}K^{\mathrm{T}}K\right\|_F = O\left(\frac{\log^2 n}{\sqrt{n}}\right)$. Also using that $\delta \geq \frac{c \log^3 n}{\sqrt{n}}$,

we have that S_1 contains the first i^* eigenvalues of A and B, whereas S_2 contains the rest of eigenvalues. We may then use a variant of the Davis-Kahan [141] theorem to show that

$$\|\mathcal{P}_A - \mathcal{P}_B\|_2 \leq \frac{\|A-B\|_2}{0.8\delta} \leq \frac{2\epsilon}{\delta}.$$

□

Step 4. Gluing everything. Recall that $A = \frac{1}{d^2}K^{\mathrm{T}}K$ and $B = \frac{1}{d^2 n}Y^{\mathrm{T}}Y$. Let $A_{i^*} = \mathcal{P}_A(A)(= \mathcal{P}_{i^*}(A))$ and $B_{i^*} = \mathcal{P}_{i^*}(B)$. By Lemma 3.8.9, we have

$$\begin{aligned}
\|A_{i^*} - B_{i^*}\|_F =& \|\mathcal{P}_A(A) - \mathcal{P}_B(B)\|_F, \\
=& \|\mathcal{P}_A(A) - \mathcal{P}_B(A) + \mathcal{P}_B(A) - \mathcal{P}_B(B)\|_F, \\
\leq& \|\mathcal{P}_A - \mathcal{P}_B\|_2 \|A\|_F + \|A-B\|_F, \\
\leq& \Theta\left(\frac{\epsilon}{\delta} + \epsilon\right) = \Theta(\epsilon/\delta).
\end{aligned}$$

Next, we define the following matrix notation

$$A_{i^*}^{\frac{1}{2}} = U_{i^*}^A(\Sigma_{i^*}^A)^{\frac{1}{2}} \quad \text{and} \quad B_{i^*}^{\frac{1}{2}} = U_{i^*}^B(\Sigma_{i^*}^B)^{\frac{1}{2}}.$$

By Lemma 3.8.2, there exists a unitary matrix W such that

$$\left\| U_{i^*}^A(\Sigma_{i^*}^A)^{\frac{1}{2}}W - U_{i^*}^B(\Sigma_{i^*}^B)^{\frac{1}{2}} \right\|_F^2 = O\left(\frac{\epsilon^2}{\delta^3}\right). \tag{3.17}$$

By Lemma 3.8.3, we obtain

$$\|U_{i^*}^A(\Sigma_{i^*}^A)^{\frac{1}{2}}U_{i^*}^A - U_{i^*}^B(\Sigma_{i^*}^B)^{\frac{1}{2}}U_{i^*}^B\|_F^2 = \left\| U_{i^*}^A(\Sigma_{i^*}^A)^{\frac{1}{2}}W - \mathcal{P}_{i^*}\left(\frac{K}{d}\right) \right\|_F^2 = O\left(\frac{\epsilon^2}{\delta^3}\right).$$

Together with $\|\mathcal{P}_{i^*}(K/d) - K/d\|_F^2 = O\left(\delta^{\frac{4}{5}}\right)$, we have

$$\begin{aligned}
\left\| \frac{1}{d^2}\hat{K} - \frac{1}{d^2}K \right\|_F = \left\| U_{i^*}^A(\Sigma_{i^*}^A)^{\frac{1}{2}}W - \frac{K}{d} \right\|_F &= \left\| U_{i^*}^A(\Sigma_{i^*}^A)^{\frac{1}{2}}W - \mathcal{P}_{i^*}\left(\frac{K}{d}\right) \right\|_F + \left\| \mathcal{P}_{i^*}\left(\frac{K}{d}\right) + \frac{K}{d} \right\| \\
&= O\left(\sqrt{\frac{\epsilon^2}{\delta^3}} + \sqrt{\delta^{\frac{4}{5}}}\right).
\end{aligned}$$

Remark. Our analysis relies only on the eigenvalues of $\mathcal{K}$ decaying sufficiently fast. Many other kernels, such as inner product kernels with points on the surface of a unit ball [65, 10], also exhibit this property. In conclusion, our algorithms for estimating K can

be generalized to these $\kappa(\cdot,\cdot)$ functions.

3.8.3 Additional estimators for K

This section explains additional possible ways to estimate K. Our intuition is that estimations of K effectively rely only on $\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$, which is the empirical covariance (aka risk) of the equities' returns. There are multiple ways to enhance the estimation of returns' covariance matrix such as using third-party risk models.

Estimation based on dynamically evolving hints. Here, we focus on describing the estimation algorithm that is most effective in practice. Specifically, we assume that the latent positions evolve. Let K_t be the Gram matrix at round t. Because K_t is evolving, we may not have sufficient data to track $\hat{K}$. Therefore, we derive a new algorithm to estimate $\hat{K}$ using the so-called "hint" matrices, based on two observations: *(i)* $\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$ is effectively the covariance matrix of the returns. Third-party risk models such as Barra provide a more accurate estimation of the covariance matrix in practice. Thus, we may directly use the risk matrix produced by Barra as our estimation for K. *(ii)* The movements of two stocks are related because they are economically linked. It is possible to estimate these links by using fundamental and news data. Specifically, we assume that $K_t = \exp(\beta_1 K_t^{(1)} + \cdots + \beta_c K_t^{(c)})$, where $K_t^{(i)}$ $(i \le c)$ can be observed. We then need only tune β_i's to determine $\hat{K}$. Each of $K_t^{(i)}$ is considered as our "hint". We use a hint matrix $K_t^{(1)}$ constructed from Barra factor loading and a hint matrix $K^{(2)}$ constructed from news so that $\hat{K}_t = \exp(\beta_1 K_t^{(1)} + \beta_2 K^{(2)})$. The hint matrices are constructed as follows.

$K_t^{(1)}$ from Barra loading. Let $F_{t,i} \in \mathbf{R}^{10}$ be the factor exposure of the i-th stock on day t. Construct $\hat{K}_t^{(1)}$ using two standard methods.

- *Inner product.* $(\hat{K}_t^{(1)})_{i,j} = \langle F_{t,i}, F_{t,j} \rangle$.
- *Distance.* $(\hat{K}_t^{(1)})_{i,j} = \exp(-\lambda |F_{t,i} - F_{t,j}|^2)$, where λ is a hyperparameter

$K_t^{(2)}$ from News Data. We next build $K_t^{(2)}$ from the news using two steps. *Step 1.* Construct $\tilde{K}_t^{(2)} \in \mathbf{R}^{d \times d}$ such that $(\tilde{K}_t^{(2)})_{i,j}$ represents the number of news articles that mention both stock i and stock j between day $t-k$ and day t (i.e., we maintain a sliding

window of k days and k is a hyper parameter). *Step 2.* Then construct $\hat{K}_t^{(2)}$ by taking a moving average of $\tilde{K}_t^{(2)}$.

Construction of $\hat{K}_t$. $\hat{K}_t$ be constructed from $\hat{K}_t^{(1)}$ (produced from Barra data) or $\hat{K}_t^{(2)}$ (constructed from news data set), or a consolidation of $\hat{K}_t^{(1)}$ and $\hat{K}_t^{(2)}$. We shall examine the following consolidation algorithm. Specifically, we let $\hat{K}_t = \exp(\beta \hat{K}_t^{(1)} + (1-\beta)\hat{K}_t^{(2)})$, where $\beta \in [0,1]$ and β is a hyperparameter.

3.9 Estimating $g(\cdot)$ with non-parametric methods

This section proves Proposition 3.3.2, i.e., we describe our non-parametric algorithm (nparam-gEST) for $\mathbf{x}_{t,i} \in \mathbf{R}^{O(i)}$ and analyze its performance. Assume that the probability cumulative density function $F_x(\cdot)$ of $\mathbf{x}_{t,i}$ is known. In practice, this can be substituted by standard non-parametric density estimation methods [150].

We first describe a high-level roadmap of our algorithm analysis and then proceed to present the full analysis.

3.9.1 Overview of our algorithms

As shown in Alg. 1, our algorithm consists of three steps.

- *Step 1.* Partition the feature space $[-1,1]^k$ into $\{\Omega_j\}_{j \le \ell}$ so that $\Pr[\mathbf{x}_{t,i} \in \Omega_j]$ are equal for all j.
- *Step 2.* Reduce the original problem to a linear regression problem.
- *Step 3.* Implement the *FlipSign* algorithm for the scenario when only an estimated $\hat{K}$ available.

We first comment on Steps 1 and 2. Then we explain the challenges in implementing the FlipSign idea, as well as our solution.

Step 1. Construction of $\{\Omega_j\}_{j \le \ell}$. We use a simple algorithm to find axis-parallel Ω_j's so that $\Pr[\mathbf{x}_{t,i} \in \Omega_j]$ is uniform for all j. Recall that we assume that the cumulative probability function of $\mathbf{x}_{t,i}$ is known (denoted as $F_{\mathbf{x}}(\cdot)$).

We describe the method for the case $k = 1, 2$ (recall that k is the dimension of the feature $\mathbf{x}_{t,i}$). Extensions to the case where $k \geq 3$ can be easily generalized. When $k = 1$, each Ω_j is simply an interval, and thus we only need to find $\{\mathbf{x}_1 = -1, \mathbf{x}_2, \ldots, \mathbf{x}_{\ell+1} = 1\}$ such that $F_{\mathbf{x}}(\mathbf{x}_{t+1}) - F_{\mathbf{x}}(\mathbf{x}_t) = 1/\ell$ for all $1 \leq t \leq \ell$. For example, note that in the $k = 1$ case, for $l = 4$, the recovered values $\{\mathbf{x}_1, \ldots, \mathbf{x}_5\}$ are simply identified with the usual quantiles of the distribution.

When $k = 2$, we can find Ω_i's recursively. Specifically, we first "chop" along the x-axis into $\sqrt{\ell}$ pieces so that each piece has uniform probability mass. Then we chop each of the $\sqrt{\ell}$ "bars" into smaller rectangles so that each rectangle has probability mass $1/\ell$. See Fig. 3.5. This procedure can be generalized for any $k = O(1)$.

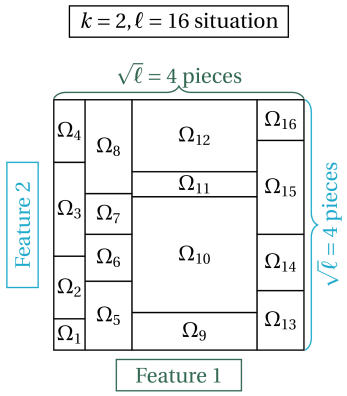


Figure 3.5: Example of a two-dimensional scenario for the construction of $\{\Omega_j\}_{j \leq \ell}$. Each rectangle in the graph has the same probability mass.

Step 2. We next explain how the original problem can be reduced to a set of regression problems. Using MAP-REGRESS (line 8 in Alg. 1). Recalling that for any $\mathbf{y}_{t,i} = \sum_{j \leq d} K_{i,j} g(\mathbf{x}_{t,j}) + \xi_{t,i}$ (with fixed i and t), we can approximate it as $\mathbf{y}_{t,i} = \sum_{j \leq d} K_{i,j} \tilde{g}(\mathbf{x}_{t,j}) + \xi_{t,i}$. We may then re-arrange the terms and obtain

$$\mathbf{y}_{t,i} \approx \sum_{j \leq \ell} L_{(t,i),j} \mu_j + \xi_{t,i}, \text{ where } L_{(t,i),j} = \sum_{m \in \mathcal{L}_{t,j}} K_{i,m} \text{ and } \mathcal{L}_{t,j} = \{m : \mathbf{x}_{t,m} \in \Omega_j\}. \quad (3.18)$$

Here, $\{\mu_j\}_{j \leq \ell}$ are unknown coefficients whereas $\mathbf{y}_{t,i}$ and $L_{(t,i),j}$ are observable.

Note that for any fixed t, there is a total number of d observations (i.e., $\{(\mathbf{y}_{t,i}, \{L_{(t,i),j}\}_{i \leq d})\}_{i \leq d}$

and these observations are all correlated: $L_{(t,i),j}$ and $L_{(t,i'),j}$ depend on the same set $\mathcal{L}_{t,j}$. So our algorithm chooses only one i for each fixed t.

Sec 3.3.2 asserts that we can use the same i for different t when we have accurate information on K. In practice, we have only an estimate $\hat{K}$ of K. In addition, the estimation quality for any fixed i depends on $\|K_{i,:} - \hat{K}_{i,:}\|_F^2$. We do not know a priori which row of $\hat{K}$ is more accurate, although we know that on average, $\hat{K}$ is sufficiently close to K (i.e., $\frac{1}{d^2}\|K - \hat{K}\|_F^2 = o(1)$ from Proposition 3.3.1). To avoid the same "bad" i being picked up repeatedly, we run a randomized procedure: let q_t be a random number from $[d]$. We use the observations $\{(\mathbf{y}_{t,q_t}, \{\mathcal{L}_{(t,q_t),1}\}\}_{t\leq n}$ to learn the variables μ_1.

3.9.2 Implementing the FlipSign algorithm

Building a robust estimator. We focus on estimating μ_1 (See Line 15 in Algorithm 1). The estimations for other μ_i's are the same. Let b_{t,q_t} be the sign of $L_{(t,q_t),1} - d/\ell$ but we only observe an estimate of $L_{(t,q_t),1}$ (referred to as $\hat{L}_{(t,q_t),1}$ in the forthcoming discussion). A major error source is that when $L_{(t,q_t),1}$ gets too close to d/ℓ, the sign of $\hat{L}_{(t,q_t),1}$ can be different from $L_{(t,q_t),1}$ (i.e., b_{t,q_t} is calculated incorrectly). We slove this problem by keeping only the observations when $|\hat{L}_{(t,q_t),1} - d/\ell|$ is large. Specifically, let

$$\Pi_1^{(q_t)}(t) \triangleq \sum_{k\in\mathcal{L}_{t,1}} K_{q_t,k} - \left(\sum_{k\notin\mathcal{L}_{t,1}} K_{q_t,k}\right)\frac{1}{\ell-1}, \tag{3.19}$$

and let $\hat{\Pi}_1^{(q_t)}(t)$ be computed using the estimate $\hat{K}$. We now define a robust variable $\tilde{b}_{t,q_t}$ to control the estimator

$$\tilde{b}_{t,q_t} = \begin{cases} 1 & \text{if } \Pi_1^{(q_t)}(t) \geq \frac{c}{\log d}\sqrt{\frac{d}{\ell}} \\ -1 & \text{if } \Pi_1^{(q_t)}(t) < -\frac{c}{\log d}\sqrt{\frac{d}{\ell}} \\ 0 & \text{otherwise.} \end{cases} \tag{3.20}$$

In this case, the chance of obtaining an incorrect $\tilde{b}_{t,q_t}$ (i.e., $\hat{\Pi}_1^{q_t}(t) > \frac{c}{\log d}\sqrt{\frac{d}{\ell}}$ but $\Pi_1^{(q_t)} \leq -\frac{c}{\log d}\sqrt{\frac{d}{\ell}}$ or vice verse) is significantly reduced (see line 19 in Alg. 1).

Analysis of the estimator. Recall that $\{\Omega_j\}_{j\leq\ell}$ is a partition such that $\Pr[\mathbf{x}_{t,i} \in \Omega_j]$ is

uniform for all j. We first formalize the "ideal" μ_j that we want to track. Specifically, let $\mu_j = \mathbb{E}[g(\mathbf{x}_{t,i}) \mid \mathbf{x}_{t,i} \in \Omega_j]$. Our error analysis aims to track $\{\mu_j\}_{j \le \ell}$ (i.e., we aim to find $(\hat{\mu}_j - \mu_j)^2$). We then articulate $\tilde{g}(\mathbf{x})$ as

$$\tilde{g}(\mathbf{x}) = \mu_j, \text{ where } \mathbf{x} \in \Omega_j.$$

Our analysis consists of two parts.

Part 1. Analysis of a stylized model. We analyze a model in which the observations are assumed to be generated from

$$\mathbf{y}_{t,i} = \sum_{j \le d} K_{i,j} \tilde{g}(\mathbf{x}_{t,j}) + \xi_{t,i}, \tag{3.21}$$

where $K_{i,j}$ is assumed to be known.

Next, we analyze Alg. 1 when it is executed over this stylized model with the assumption that K is given.

Part 2. Analysis of the original problem with $g(\cdot)$ and unknown K. When we run Alg. 1 over the original process, we need to analyze two perturbations (deviations):

1. $\mathbf{y}_{t,i}$ is generated through $g(\cdot)$, instead of $\tilde{g}(\cdot)$.
2. Our algorithm uses only an estimate of K.

Warm-up and notation. Before proceeding, let us introduce additional notation. Recall that $\mathcal{L}_{t,j} = \{k : \mathbf{x}_{t,k} \in \Omega_j\}$, i.e., the set of $\mathbf{x}_{t,k}$ that falls into the j-th bin Ω_j on time t. Also, recall that

$$L_{(t,i),j} = \sum_{k \in \mathcal{L}_{t,j}} K_{i,k}.$$

We have

$$\mathbf{y}_{t,i} = \sum_{j \le d} L_{(t,i),j} \mu_j + \xi_{t,i} = \left(\sum_{k \in \mathcal{L}_{t,1}} K_{i,k} \right) \mu_1 + \sum_{k \notin \mathcal{L}_{t,1}} K_{i,k} \tilde{g}(\mathbf{x}_{t,k} \mid \mathbf{x}_{t,k} \notin \Omega_1) + \xi_{t,i}.$$

We interpret the meaning of the above equation. We treat $\mathbf{x}_{t,k}$ and $\mathcal{L}_{t,j}$ as random variables and the $\mathcal{L}_{t,j}$'s are measurable by $\mathbf{x}_{t,i}$. We imagine that an observation is generated by using the following procedure:

- Step 1. Generate $\mathcal{L}_{t,1}$. That is, we determine the subset of "balls" (those $\mathbf{x}_{t,i}$ for a fixed t) that fall into Ω_1.

- Step 2. Generate the rest of $\mathbf{x}_{t,k}$ for $k \notin \mathcal{L}_{t,1}$ sequentially. This corresponds to the terms $\sum_{k \notin \mathcal{L}_{t,1}} K_{i,k} \tilde{g}(\mathbf{x}_{t,k} \mid \mathbf{x}_{t,k} \notin \Omega_1)$. $\mathbf{x}_{t,k}$ is sampled from the conditional distribution $\mathbf{x}_{t,k} \mid \mathbf{x}_{t,k} \notin \Omega_1$. This explains why we write $\tilde{g}(\mathbf{x}_{t,k} \mid \mathbf{x}_{t,k} \notin \Omega_1)$.
- Step 3. After $\mathcal{L}_{t,1}$ and $\mathbf{x}_{t,k} \mid \mathbf{x}_{t,k} \notin \Omega_1$ are fixed, we generate $\mathbf{y}_{t,i}$ using the stylized model.

Let

$$\tilde{g}_j(\mathbf{x}_{t,i}) = \tilde{g}(\mathbf{x}_{t,i}) - \mathbb{E}[\tilde{g}(\mathbf{x}_{t,i}) \mid \mathbf{x}_{t,i} \notin \Omega_j] = \tilde{g}(\mathbf{x}_{t,i}) + \frac{\mu_j}{\ell - 1}.$$

We have

$$\mathbf{y}_{t,i} = \underbrace{\left(\sum_{k \in \mathcal{L}_{t,1}} K_{i,k} - \left(\sum_{k \notin \mathcal{L}_{t,1}} K_{i,k} \right) \frac{1}{\ell - 1} \right) \mu_1}_{\Pi_1^{(i)}(t)} + \underbrace{\sum_{k \in \mathcal{L}_{t,1}} K_{i,k} \tilde{g}_j(\mathbf{x}_{t,i} \mid \mathbf{x}_{t,i} \notin \Omega_1)}_{\Pi_2^{(i)}(t)} + \xi_{t,i} \quad (3.22)$$

$$= \Pi_1^{(i)}(t)\mu_1 + \Pi_2^{(i)}(t) + \xi_{t,i}.$$

We use the following abbreviation.

- $\hat{b}_t$ is an abbreviation for $\hat{b}_{t,q_t}$.
- $\Pi_1(t)$ is an abbreviation for $\Pi_1^{(q_t)}(t)$.
- $\Pi_2(t)$ is an abbreviation for $\Pi_2^{(q_t)}(t)$.

Let $\Delta = \hat{K} - K \in \mathbf{R}^{d \times d}$ and $\Delta^2(q_t) = \sum_{i \le d} \Delta_{q_t,i}^2$. Let $\hat{\mathcal{B}} = \{t \in [n] : \hat{b}_{t,q_t} = 1\}$. Also, let

$$s_t = \begin{cases} 1 & \text{if } \Pi_1(t) > 0 \\ -1 & \text{otherwise.} \end{cases} \quad (3.23)$$

3.9.2.1 Part 1. Analysis of the stylized model

Our main lemma in this section is an anti-concentration result on $\Pi_1^{(i)}(t)$ for any i and t.

Lemma 3.9.1. *Let $\ell = O(d/\log^2 d)$. There exist constants c_0 and c_1 such that*

$$\Pr\left[|\Pi_1^{(i)}(t)| \ge \frac{c_0}{\log d} \sqrt{\frac{d}{\ell}} \right] \ge c_1$$

The probability is over the random tosses of $\{\mathbf{x}_{t,i}\}_{i \le d}$.

Proof. We use a random-walk interpretation of $\Pi_1^{(i)}$. For each $k \in [d]$, with probability $1/\ell$, it (i.e., $\mathbf{x}_{t,k}$) falls into $\mathcal{L}_{t,j}$. When this happens, $\Pi_1^{(i)}(t)$ is incremented by $K_{i,k}$. With

probability $1 - 1/\ell$, it does not fall into $\mathcal{L}_{t,j}$. In this case, $\Pi_1^{(i)}(t)$ is decremented by $K_{i,t}/(\ell - 1)$.

We define a sequence $\{Z_k\}_{k \le d}$ to clarify the random-walk interpretation.

$$Z_k = \begin{cases} K_{i,k} & \text{with probability } \frac{1}{\ell}. \\ -\frac{K_{i,k}}{\ell - 1} & \text{with probability } 1 - \frac{1}{\ell}. \end{cases}$$

We couple $\Pi_1^{(i)}(t)$ with $\{Z_i\}_{i \le d}$ such that $\Pi_1^{(i)}(t) = \sum_{k \le d} Z_k$. Apply Lemma 3.12.4 (a folklore that generalizes Littlewood-Offord-Erdős) to prove our Lemma. □

3.9.2.2 Part 2. Analysis of the original problem with $g(\cdot)$ and unknown K

Our analysis consists of three components.

Part 2.1. Building blocks. We develop the essential building blocks needed in our analysis.

Part 2.2. Using $\hat{K}$. We show that when K is substituted by $\hat{K}$, the error of the estimator is well-managed.

Part 2.3. Using $g(\cdot)$. We show that when $\tilde{g}(\cdot)$ is substituted by $g(\cdot)$, not much additional error is introduced.

Part 2.1. Building blocks. We start with a variance-based Chernoff bound [32].

Theorem 3.9.2. *Suppose that X_i are independent random variables satisfying $X_i \le M$ for $1 \le i \le n$. Let $X = \sum_{i=1}^n X_i$ and $\|X\| = \sqrt{\sum_{i=1}^n \mathbb{E}[X_i^2]}$. Then we have*

$$\Pr[X \ge \mathbb{E}[X] + \lambda] \le \exp\left(-\frac{\lambda^2}{2(\|X\|^2 + M\lambda/3}\right). \tag{3.24}$$

Lemma 3.9.3. *Consider running Alg. 1 to learn the stylized model. Let $\hat{K}$ be such that $|\hat{K} - K|_F^2 \le \gamma d^2$, for $\gamma = o(1)$. Let $\Pi_1^{(i)}(t)$ and $\hat{\Pi}_1^{(i)}(t)$ be those defined around Eq. 3.19. Let $\Delta = \hat{K} - K \in \mathbf{R}^{d \times d}$ and $\Delta^2(q_t) = \sum_{i \le d} \Delta_{q_t,t}^2$. Let λ_t be any random variable that is measurable by q_t. With high probability we have*

$$\Pr\left[\left|\hat{\Pi}_1^{(q_t)}(t) - \Pi_1^{(q_t)}(t)\right| \ge \lambda_t \mid q_t\right] \le \exp\left(-\frac{\lambda_t^2}{\Delta^2(q_t)/\ell + \lambda_t/3}\right),$$

and

$$\sum_{t \le n} \left|\hat{\Pi}_1^{(q_t)}(t) - \Pi_1^{(q_t)}(t)\right| = O\left(n \log n \sqrt{\frac{\gamma d}{\ell}}\right).$$

Proof. We shall again use random-walk techniques to analyze $\left|\hat{\Pi}_i^{(q_t)}(t) - \Pi_i^{(q_t)}(t)\right|$. Let

$$Z_i = \begin{cases} \Delta_{q_t,i} & \text{with probability } \frac{1}{\ell} \\ -\frac{\Delta_{q_t,i}}{\ell-1} & \text{with probability } 1 - \frac{1}{\ell}. \end{cases}$$

We have $\mathbb{E}[Z_i^2] = O\left(\frac{\Delta_{q_t,i}^2}{\ell}\right)$, which implies that $\sqrt{\sum_{i\le d}\mathbb{E}[Z_i^2]} = \sqrt{\frac{\sum_{i\le d}\Delta_{q_t,i}^2}{\ell}}$. Also, we can use a standard way to couple Z_i's with $\hat{\Pi}_1^{(q_t)}(t)$ and $\Pi_1^{(q_t)}(t)$ such that

$$\left|\sum_{i\le d} Z_i\right| = \left|\hat{\Pi}_i^{(q_t)}(t) - \Pi_i^{(q_t)}(t)\right|.$$

By using a Chernoff bound from Theorem 3.9.2, we have

$$\Pr\left[\left|\sum_{i\le d} Z_i\right| \ge \lambda_t \mid q_t\right] \le \exp\left(-\frac{\lambda_t^2}{\frac{1}{\ell}\left(\sum_{i\le d}\Delta_{q_t,i}^2\right) + \frac{\lambda}{3}}\right) = \exp\left(-\frac{\lambda_t^2}{\Delta^2(q_t)/\ell + \lambda_t/3}\right).$$

This proves the first part of the Lemma. Next, we set $\lambda_t = c_0(\log d)\sqrt{\frac{\Delta^2(q_t)}{\ell}}$. Then we obtain $\Pr\left[\left|\sum_{i\le d} Z_i\right| \ge \lambda_t \mid q_t\right] = \exp(-\Theta(\log^2 d))$. Now, conditioned on knowing $\{q_t\}_{t\le n}$, with high probability, we have

$$\sum_{t\le n}\left|\hat{\Pi}_1^{(i)}(t) - \Pi_1^{(i)}(t)\right| \le \sum_{t\le n}\lambda_t = \frac{\log d}{\sqrt{\ell}}\sum_{t\le n}\sqrt{\Delta^2(q_t)}.$$

Next, we give a concentration bound for $\sum_{t\le n}\sqrt{\Delta^2(q_t)}$. Let $v_i^2 = \|K_{i,:} - \hat{K}_{i,:}\|^2$. We know that $\Delta^2(q_t)$ can only take values from $v_1^2, \ldots, v_d^2$ with $\sum_{i\le d} v_i^2 = \|\hat{K} - K\|_F^2 \le \gamma d^2$. We have $\sqrt{\Delta^2(q_t)} \le \sqrt{\gamma} d$.

Again using the condition that $\|\hat{K} - K\|_2^2 \le \gamma d^2$ and Jensen's inequality, we have $\mathbb{E}[\sqrt{\Delta^2(q_t)}] \le \sqrt{\gamma d}$.

Use the Chernoff bound, we have

$$\Pr\left[\sum_{t\le n}\sqrt{\Delta^2(q_t)} \ge \mathbb{E}\left[\sum_{t\le n}\sqrt{\Delta^2(q_t)}\right] + \lambda\right] \le \exp\left(-\frac{\lambda^2}{\gamma dn + \sqrt{\gamma d}\lambda/3}\right).$$

We set $\lambda = \frac{\sqrt{\gamma dn}}{\log^2 d}$ such that the right hand side is negligible. Now with high probability we have

$$\sum_{t\le n}\left|\hat{\Pi}_1^{(i)}(t) - \hat{\Pi}_1^{(i)}(t)\right| \le \frac{\log d}{\sqrt{\ell}}\sum_{t\le n}\sqrt{\Delta^2(q_t)} = O\left(n\log n\sqrt{\frac{\gamma d}{\ell}}\right).$$

□

Fact 3.9.1. *For any j,*

$$\mathbb{E}[g(\mathbf{x}_{t,k}) \mid \mathbf{x}_{t,k} \notin \Omega_j] = \mathbb{E}[\tilde{g}(\mathbf{x}_{t,k}) \mid \mathbf{x}_{t,k} \notin \Omega_j] = -\frac{\mu_j}{\ell - 1}$$

Proof. By our model assumption,

$$\mathbb{E}[\mathbf{s}_{t,k}] = \mathbb{E}[g(\mathbf{x}_{t,k})] = \mu_1 + \cdots + \mu_\ell = 0. \tag{3.25}$$

On the other hand,

$$\mathbb{E}[g(\mathbf{x}_{t,k}) \mid \mathbf{x}_{t,k} \notin \Omega_j] = \frac{\mu_1 + \cdots + \mu_{j-1} + \mu_{j+1} + \cdots + \mu_\ell}{\ell - 1} = -\frac{\mu_j}{\ell - 1}.$$

Similarly, we prove that $\mathbb{E}[g(\mathbf{x}_{t,k}) \mid \mathbf{x}_{t,k} \notin \Omega_j] = \mathbb{E}[\tilde{g}(\mathbf{x}_{t,k}) \mid \mathbf{x}_{t,k} \notin \Omega_j]$. □

Lemma 3.9.4. *Let $\hat{\mathcal{B}} = \{t \in [n] : \hat{b}_{t,q_t} = 0\}$, where $\hat{b}_{t,q_t}$ is defined in Sec 3.9.2. With high probability we have $|\hat{\mathcal{B}}| = \Omega(n)$.*

This can be shown by Lemma 3.9.1 and a Chernoff bound.

Lemma 3.9.5. *Let s_t be defined in Eq. 3.23. Recall that $\hat{\mathcal{B}} = \{t \in [n] : \hat{b}_{t,q_t} = 1\}$. We have*

$$\sum_{t \in \hat{\mathcal{B}}} \Pi_2(t) s_t = \sum_{t \in \hat{\mathcal{B}}} s_t \left(\sum_{k \notin \mathcal{L}_{t,1}} K_{q_t,k} \tilde{g}_1(\mathbf{x}_{t,k} \mid \mathbf{x}_{t,k} \notin \Omega_1) \right) = O(\sqrt{nd}).$$

Proof. Our key observation is that conditioned on $t \in \hat{\mathcal{B}}$ and $\mathbf{x}_{t,k} \notin \Omega_1$, $\tilde{g}_1(\mathbf{x}_{t,k})$'s are bounded independent zero-mean random variables. Also $|\hat{\mathcal{B}}| = \Omega(n)$ (Lemma 3.9.1). Therefore, a standard Chernoff bound gives $\sum_{t \in \hat{\mathcal{B}}} s_t \Pi_2(t) = O(\sqrt{nd})$. □

Lemma 3.9.6. *Recall that $\hat{\mathcal{B}} = \{t \in [n] : \hat{b}_{t,q_t} = 1\}$, we have*

$$\left| \sum_{t \in \hat{\mathcal{B}}} \hat{b}_t \Pi_1(t) - \sum_{t \in \hat{\mathcal{B}}} |\Pi_1(t)| \right| \leq 2n \log^5 d \sqrt{\frac{d}{\ell}} \gamma$$

Proof. Recall that

$$s_t = \begin{cases} 1 & \text{if } \Pi_1(t) > 0 \\ -1 & \text{otherwise.} \end{cases}$$

We have

$$\left\| \sum_{t \in \hat{\mathcal{B}}} \hat{b}_t \Pi_1(t) - \sum_{t \in \hat{\mathcal{B}}} |\Pi_1(t)| \right\| \leq 2 \sum_{t \in \hat{\mathcal{B}}} |\Pi_1(t)| I(\hat{b}_t \neq s_t),$$

where $I(\cdot)$ is an indicator function that sets to 1 if and only if its argument evaluates to true. Note that $I(\hat{b}_t \neq s_t)$'s are i.i.d. random variables for different t's. We compute

$$
\begin{aligned}
&\Pr[I(\hat{b}_t \neq s_t)] \\
\leq &\Pr[s_t = -1 \wedge \hat{b}_t = 1] + \Pr[s_t = 1 \wedge \hat{b}_t = -1] \\
= &\Pr\left[\Pi_1(t) < 0 \wedge \hat{\Pi}_1(t) > \frac{c_0}{\log d}\sqrt{\frac{d}{\ell}}\right] + \Pr\left[\Pi_1(t) > 0 \wedge \hat{\Pi}_1(t) < -\frac{c_0}{\log d}\sqrt{\frac{d}{\ell}}\right] \\
\leq &\Pr\left[\left|\Pi_1(t) - \hat{\Pi}_1(t)\right| > \frac{c_0}{\log d}\sqrt{\frac{d}{\ell}}\right] \\
\leq &\Pr\left[\left(|\Pi_1(t) - \hat{\Pi}_1(t)| > \log d\sqrt{\frac{\Delta^2(q_t)}{\ell}}\right) \vee \left(\log d \cdot \sqrt{\frac{\Delta^2(q_t)}{\ell}} \geq \sqrt{\frac{d}{\ell}}\frac{c_0}{\log d}\right)\right] \\
\leq &\frac{1}{n^{10}} + \Pr\left[\log d \cdot \sqrt{\frac{\Delta^2(q_t)}{\ell}} \geq \sqrt{\frac{d}{\ell}}\frac{c_0}{\log d}\right] \qquad \text{(by Lemma 3.9.3)} \\
= &\Pr\left[(\log^4 d)\Delta^2(q_t) \geq d\right] + n^{-10}.
\end{aligned}
$$

Note that $\mathbb{E}[\Delta^2(q_t)] = \gamma d$. Using a Markov inequality, we have

$$\Pr\left[\log^4 d\Delta^2(q_t) > d\right] \leq \gamma \log^4 d. \tag{3.26}$$

Using the fact that $I(\hat{b}_t \neq s_t)$ are independent across t, with high probability we have

$$\left|\sum_{t\in\hat{\mathcal{B}}} \hat{b}_t\Pi_1(t) - \sum_{t\in\hat{\mathcal{B}}} |\Pi_1(t)|\right| \leq 2n\log^5 d\sqrt{\frac{d}{\ell}}\gamma$$

□

Part 2.2. When K is substituted by $\hat{K}$. When K is substituted by $\hat{K}$, our estimator becomes

$$
\begin{aligned}
\hat{\mu}_1 &= \frac{\left(\sum_{t\in\hat{\mathcal{B}}} \hat{b}_t\Pi_1(t)\right) + \sum_{t\in\hat{\mathcal{B}}}\left(\hat{b}_t\Pi_2(t) + \hat{b}_t\xi_{t,q_t}\right)}{\sum_{t\in\hat{\mathcal{B}}} \hat{b}_t\hat{\Pi}_1(t).} \\
&= \frac{\sum_{t\in\hat{\mathcal{B}}} \hat{b}_t\Pi_1(t)}{\sum_{t\in\hat{\mathcal{B}}} \hat{\Pi}_1(t)}\mu_1 + \frac{\sum_{t\in\hat{\mathcal{B}}}\left(\hat{b}_t\Pi_2(t) + \hat{b}_t\xi_{t,q_t}\right)}{\sum_{t\in\hat{\mathcal{B}}} \hat{b}_t\hat{\Pi}_1(t)}.
\end{aligned}
$$

We note that

$$\left|\sum_{t\in\hat{\mathcal{B}}} \hat{b}_t\hat{\Pi}_1(t) - \sum_{t\in\hat{\mathcal{B}}} \hat{b}_t\Pi_1(t)\right| \leq \sum_{t\in\hat{\mathcal{B}}}\left|\hat{\Pi}_1(t) - \Pi_1(t)\right| \leq \sum_{t\leq n}\left|\hat{\Pi}_1(t) - \Pi_1(t)\right| \leq n\sqrt{\frac{\gamma d}{\ell}}\log d \qquad \text{(Lemma 3.9.3)}$$

Also, we can see that (Lemma 3.9.6)

$$\left|\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\Pi_1(t)-\sum_{t\in\hat{\mathcal{B}}}|\Pi_1(t)|\right|\le c_0 n\log^5 d\sqrt{\frac{d}{\ell}}\gamma.$$

Both of the inequalities above imply that with high probability, the following holds true

$$\left|\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\hat{\Pi}_1(t)-\sum_{t\in\hat{\mathcal{B}}}|\Pi_1(t)|\right|=O\left(n\log^5 d\sqrt{\frac{d}{\ell}}\sqrt{\gamma}\right).$$

Next, by Lemma 3.9.4 and Lemma 3.9.1, we have

$$\sum_{t\in\hat{\mathcal{B}}}|\Pi_1(t)|=\Omega\left(\frac{n}{\log n}\sqrt{\frac{\ell}{d}}\right).$$

Therefore,

$$\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\Pi_1(t)=(1+\tau_1)\sum_{t\in\hat{\mathcal{B}}}|\Pi_1(t)|$$

$$\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\hat{\Pi}_1(t)=(1+\tau_2)\sum_{t\in\hat{\mathcal{B}}}|\Pi_1(t)|,$$

where $|\tau_1|,|\tau_2|=O\left(\log^6 n\sqrt{\gamma}\right)$. This implies that

$$\left|\frac{\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\Pi_1(t)}{\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\hat{\Pi}_1(t)}-1\right|=O(\tau_1).$$

Now we analyze the second term. By Lemma 3.9.4 and Lemma 3.9.1, we have $\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\hat{\Pi}_1(t)=\Omega\left(\frac{n}{\log d}\sqrt{\frac{d}{\ell}}\right)$. By Lemma 3.9.5, we have $\sum_{t\in\hat{\mathcal{B}}}\left(\hat{b}_t\Pi_2(t)+\hat{b}_t\xi_{t,q_t}\right)=O(\sqrt{nd})$, which implies

$$\frac{\sum_{t\in\hat{\mathcal{B}}}\left(\hat{b}_t\Pi_2(t)+\hat{b}_t\xi_{t,q_t}\right)}{\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\hat{\Pi}_1(t)}=O\left(\log d\sqrt{\frac{\ell}{n}}\right).$$

Therefore,

$$\hat{\mu}_1=(1+O(\tau))\mu_1+O\left(\log d\sqrt{\frac{\ell}{n}}\right),$$

where $\tau=\log^6 n\sqrt{\gamma}$.

Part 2.3. Analysis when observations are from $g(\cdot)$. We assume that the process is generated by $g(\cdot)$ instead of $\tilde{g}(\cdot)$. We aim to understand how the estimator changes. To distinguish the observations produced from two "worlds", we let

$$\mathbf{y}_{t,i}^{(1)} = \sum_{j \le d} K_{i,j} \tilde{g}(\mathbf{x}_{t,j}) + \xi_{t,i},$$
$$\mathbf{y}_{t,i}^{(2)} = \sum_{j \le d} K_{i,j} g(\mathbf{x}_{t,j}) + \xi_{t,i}.$$

Let their corresponding estimators be $\mu_1^{(1)}$ and $\mu_1^{(2)}$. We next bound the difference between these two estimators. Our crucial observation is that each $\tilde{g}(\mathbf{x}_{t,i}) - g(\mathbf{x}_{t,i})$ are bounded zero mean independent random variables. Seeing that $|\hat{\mu}_1^{(1)} - \hat{\mu}_1^{(2)}| = O(\sqrt{nd})$. Therefore, we still have

$$\hat{\mu}_1^{(2)} = \left(1 + O(\sqrt{\gamma} \log^6 n)\right) \mu_1 + O\left(\log d \sqrt{\frac{\ell}{n}}\right).$$

This proves the second part of the theorem.

Remark. We use only 1 observation for each day because our analysis relies on different $\Pi_1^{(i)}(t)$ and $\Pi_2^{(i)}(t)$ are being independent. The FlipSign algorithm does not need more samples because K is near low-rank (Theorem 3.8.5).

3.10 Estimating $g(\cdot)$ with boosting

As shown in Alg. 3 and Fig. 3.6, LIN-PVEL's weak learner first performs a variable selection (i.e., selects the 3 features that correlate the most with the residual returns), and then fits a linear model with both linear and quadratic interaction terms over the selected variables. The final model is a **linear** one with features and their interactions terms.

Algorithm 3 LIN-PVEL

Input $\mathbf{X}$, $\mathbf{Y}$, $\hat{K}$, η, b

Output $\{g_m(\cdot)\}_{m \le b}$

procedure BOOSTING-ALGORITHM($\mathbf{Y}$, $\mathbf{X}$, $\hat{K}$, η, b)
  $\mathbf{Y}_{\text{Res}} \leftarrow \mathbf{Y}$ ▷ η is the learning rate
  for all $m \leftarrow 1$ **to** b **do**
    $g_m \leftarrow$ LINEAR-FIT($\mathbf{Y}_{\text{Res}}, \mathbf{X}, \hat{K}$)
    $(\hat{\mathbf{Y}})_m \leftarrow \hat{K} g_m(\mathbf{X})$.
    $\mathbf{Y}_{\text{Res}} \leftarrow \mathbf{Y}_{\text{Res}} - \eta(\hat{\mathbf{Y}})_m$
  end for
  return $\{g_m(\cdot)\}_{m \le b}$
end procedure
procedure LINEAR-FIT($\mathbf{Y}$, $\mathbf{X}$, $\hat{K}$) ▷ $\mathbf{x}_{t,i} \in \mathbf{R}^k$ and $\mathbf{F}^{(t)} \in \mathbf{R}^{k \times d}$
  for all $i \leftarrow 1$ **to** d **do**
    $\mathbf{F}^{(t)}_{:,i} = \sum_{j \in [d]} \hat{K}_{i,j} \mathbf{x}_{t,j}$
  end for
  for all $j \leftarrow 1$ **to** k **do**
    $r_j = \sum_{t \le n} \text{corr}(\mathbf{F}^{(t)}_{j,:}, \mathbf{y}_t)$.
  end for
  Let j_1, j_2, j_3 be the indices with the largest r_j.
  $g(\cdot) = \arg\min_{\beta_1, \ldots, \beta_6} \sum_{\substack{t \le n \\ i \in [d]}} (\mathbf{y}_{t,i} - \sum_{j \in [d]}$
    $\hat{K}_{i,j} (\beta_1 (\mathbf{x}_{t,j})_{j_1} + \cdots + \beta_6 (\mathbf{x}_{t,j})_{j_2} \cdot (\mathbf{x}_{t,j})_{j_3}))^2$. ▷ Fits a linear model with linear and quadratic interaction terms.
  return $g(\cdot)$
end procedure

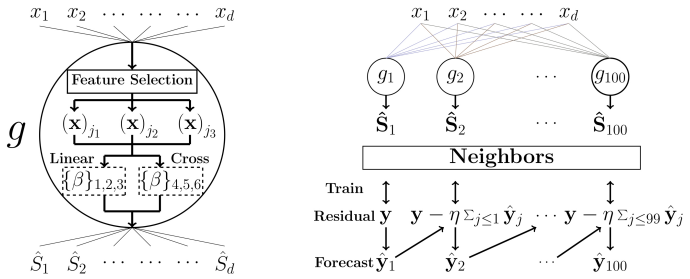


Figure 3.6: Left: training one weak learner. We first choose three features that are the most correlated with the residualized response. Then we run a linear regression using linear and interaction terms. Right: boosting. When we train learners sequentially, we can control $g_m(\cdot)$ but not the neighbors' structure.

3.11 Consolidation/Ensemble model

We next describe how we consolidate forecasts generated by multiple models. We do not intend to design a new consolidation algorithm. Instead, we use a "folklore" algorithm that weighs each model forecast by its recent historical performance. Specifically, we let $C = \{\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_m\}$ be the set of models to be consolidated. Our consolidated forecast is a linear combination of all forecasts $\hat{\mathbf{y}} = \sum_{j=1}^{m} w_j \hat{\mathbf{y}}_j$, where w_j is simply the t-statistics of the j-th model computed through the Newey-West estimation algorithm from the in-sample data. For example, when $m = 2$ and the t-statistics for $\hat{\mathbf{y}}_1$ and $\hat{\mathbf{y}}_2$ are 3 and 5 respectively, we set the consolidated forecast be proportional to $3 \times \hat{\mathbf{y}}_1 + 5 \times \hat{\mathbf{y}}_2$. The consolidated forecast needs to be properly re-scaled (e.g., set the daily standard deviation to be constant). In the forecasting models, correlation with the ground-truth is more important than MSE. Therefore, the scale of a forecast is less important than its direction. This t-statistics based consolidation algorithm is used in the following two situations.

Allowing nparam-gEST to use all factors. Our theoretically sound algorithm in Sec. 3.3 allows us to use only a small number of features. Now we may use a two-step procedure to let this algorithm simultaneously use hundreds of technical factors constructed in-house. *Step 1.* For each improvable factor, we build a model that uses only this factor as the feature. *Step 2.* After we obtain multiple models (the number of models is the same as the number of improvable factors), we use the above consolidation algorithm to produce the final forecast.

Consolidating multiple models. We also use the consolidation trick to aggregate the forecasts of all our models (Lin-PVEL, nparam-gEST, MLP). The result in Table 3.2 (last line) shows that the consolidated signal is stronger than any individual signal. Even if a model may not have the best out-of-sample performance, it may still be useful for constructing consolidated signals.

3.12 Additional proofs and calculations

In this section we give some additional proofs and calculations for App. 3.8 and App. 3.9.

3.12.1 Proof of Proposition 3.8.7

We can see that

$$\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y} = \frac{1}{n}\left(K^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}\mathbf{S}K + K^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}E + E^{\mathrm{T}}\mathbf{S}K + E^{\mathrm{T}}E\right) \tag{3.27}$$

$$= K^{\mathrm{T}}K + \mathcal{E}_1 + \mathcal{E}_2 + \mathcal{E}_3 + \mathcal{E}_4, \tag{3.28}$$

where $\mathcal{E}_1 = K^{\mathrm{T}}\left(\frac{\mathbf{S}^{\mathrm{T}}\mathbf{S}}{n} - I\right)K$, $\mathcal{E}_2 = \frac{K^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}E}{n}$, $\mathcal{E}_3 = \frac{E^{\mathrm{T}}\mathbf{S}K}{n}$, and $\mathcal{E}_4 = \frac{E^{\mathrm{T}}E}{n}$

We next show that each $\mathcal{E}_i$ $(i \leq 4)$ is small.

Bounding $\mathcal{E}_1$. We need the following lemma.

Lemma 3.12.1. *Let $\mathbf{S} \in \mathbf{R}^{n\times d}$ be such that each row $\mathbf{S}_{i,:}$ is an i.i.d. random vector $\|S_{i,:}\|_\infty \leq 1$ and $\mathbb{E}[\mathbf{S}_{i,:}^{\mathrm{T}}\mathbf{S}_{i,:}] = I$. We have*

$$\Pr\left[\left\|\frac{\mathbf{S}^{\mathrm{T}}\mathbf{S}}{n} - I_{d\times d}\right\|_2 \geq \epsilon\right] \leq 2n^2 \exp\left(-\frac{n\epsilon^2}{\log^4 n}\right), \tag{3.29}$$

where ϵ is a tunable parameter.

Here, we shall set $\epsilon = \frac{\log^3 n}{\sqrt{n}}$. This implies that with high probability $\left\|\frac{\mathbf{S}^{\mathrm{T}}\mathbf{S}}{n} - I_{d\times d}\right\|_2 = O\left(\frac{\log^3 n}{\sqrt{n}}\right)$. On the other hand, we can see that $\|K\|_F^2 = \Theta(d^2)$ and $\|K\|_F = \Theta(d)$. This implies

$$\|\mathcal{E}_1\|_F = \left\|K^{\mathrm{T}}\left(\frac{\mathbf{S}^{\mathrm{T}}\mathbf{S}}{n} - I_{d\times d}\right)K\right\|_F \leq \left\|\frac{\mathbf{S}^{\mathrm{T}}\mathbf{S}}{n} - I_{d\times d}\right\|_2 \|K\|_F^2 = \Theta\left(\frac{d^2\log n}{\sqrt{n}}\right) \tag{3.30}$$

Bounding $\mathcal{E}_2$ and $\mathcal{E}_3$. Recall that $\mathcal{E}_2 = \frac{K^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}E}{n}$ and $\mathcal{E}_3 = \frac{E^{\mathrm{T}}\mathbf{S}K}{n}$. We have the following lemma.

Lemma 3.12.2. *Let $\mathbf{S} \in \mathbf{R}^{n\times d}$ be such that each row $\mathbf{S}_{i,:}$ is an i.i.d. random vector with $\|\mathbf{S}_{i,:}\|_\infty \leq 1$ and $\mathbb{E}[\mathbf{S}_{i,:}^{\mathrm{T}}\mathbf{S}_{i,:}] = I$. Let $E \in \mathbf{R}^{n\times d}$ be such that $E_{i,j}$ are i.i.d. Gaussian with standard deviation σ_ξ. We have with overwhelming probability*

$$\|\mathbf{S}^{\mathrm{T}}E\|_F^2 \leq c_0\sigma_\xi d^2 n \tag{3.31}$$

for some constant c_0.

Proof of Lemma 3.12.2. First, note that

$$\mathbb{E}[\|\mathbf{S}^{\mathrm{T}}E_{:,i}\|_F^2 \mid \mathbf{S}] = \sigma_\xi^2\|S\|_F^2.$$

Therefore, we have $\mathbb{E}[\|\mathbf{S}^{\mathrm{T}}E_{:,i}\|_F^2] = \sigma_\xi^2 dn$. This also implies that

$$\mathbb{E}[\|\mathbf{S}^{\mathrm{T}}E\|_F^2] = \sum_{i\le d}\mathbb{E}[\|\mathbf{S}^{\mathrm{T}}E_{:,i}\|^2] = \sigma_\xi^2 d^2 n.$$

By a standard Chernoff bound, we have whp

$$\|\mathbf{S}^{\mathrm{T}}E\|_F^2 \le c_0\sigma_\xi^2 d^2 n \tag{3.32}$$

for some constant c_0, i.e., whp $\|\mathbf{S}^{\mathrm{T}}E\|_F = O(\sigma_\xi d\sqrt{n})$. □

We next use Lemma 3.12.2 to bound $\mathcal{E}_2$ and $\mathcal{E}_3$:

$$\|\mathcal{E}_2\|_F = \|\mathcal{E}_3\|_F = \frac{1}{n}\|K^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}E\|_F = \frac{1}{n}\|K\|_2\|\mathbf{S}E\|_F. \tag{3.33}$$

Now we have $\|K\|_2 = O(d)$ and $\|\mathbf{S}E\|_F = O(d\sqrt{n})$ whp. Therefore, with high probability

$$\|\mathcal{E}_2\|_F = \|\mathcal{E}_3\|_F = O\left(\frac{d^2}{\sqrt{n}}\right).$$

Bounding $\mathcal{E}_4$. With the assumption that $d = O(n)$, we have

$$\left\|\frac{E^{\mathrm{T}}E}{n}\right\|_F^2 \le \frac{\mathrm{Rank}(E^{\mathrm{T}}E)\|E\|_2^2}{n} = O\left(\frac{\sigma_\xi^2 dn}{n}\right) = O(\sigma_\xi^2 d). \tag{3.34}$$

Above, we used a finite sample version of semi-circle law (i.e., $\|E\|_2^2 = O(n)$ whp [134]).

Summing up above and using that $n < d^2$ and $\sigma_\xi = O(\sqrt{d})$, we have $\left\|\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y} - K^{\mathrm{T}}K\right\|_F = O\left(\frac{d^2\log^3 n}{\sqrt{n}}\right)$.

3.12.2 Anti-concentrations

Theorem 3.12.3. *(Littlewood-Offord-Erdos; e.g., [87]) Let $L_1, \ldots, L_d \ge 1$. Let $\xi_1, \ldots \xi_n$ be independent Bernoulli ± 1 unbiased random variables such that $\Pr[\xi_i = 1] = \frac{1}{2}$. Let $S = \sum_{i\le n}\xi_i L_i$. For any open interval I of length 2, we have*

$$\Pr[S \in I] = O(n^{-\frac{1}{2}}). \tag{3.35}$$

Lemma 3.12.4. *Let $\ell \leq d/\log^2 d$. Let $L_1, L_2, \ldots, L_d$ be positive numbers such that $L_i = \Omega(1)$. Define a random variable*

$$Z_i = \begin{cases} L_i & \text{with probability } \frac{1}{\ell} \\ -\frac{L_i}{\ell-1} & \text{with probability } 1 - \frac{1}{\ell}. \end{cases}$$

There exist constants c_0 and c_1 such that

$$\Pr\left[\sum_{i \leq d} Z_i \geq \frac{c_0}{\log d}\sqrt{\frac{d}{\ell}}\right] \geq c_1$$

Proof. We shall use Theorem 3.12.3 to prove Lemma 3.12.4. Theorem 3.12.3 requires that random variables ξ_i (or Z_i in our setting) to be symmetric, which is violated in our setting. Our goal is to reduce our problem to the original setting.

We now show that this can be done through "debiasing" the walk. We first define $\{B_i\}_{i\in[d]}$ such that B_i is a random binary indicator variable with $\Pr[B_i = 1] = \frac{\ell-2}{\ell}$ and $\Pr[B_i = 0] = \frac{2}{\ell}$.

We may generate Z_i by using B_i, i.e., when $B_i = 1$, we set $Z_i = -\frac{L_i}{\ell-1}$, and when $B_i = 0$, we set $Z_i = L_i$ with half of the probability and $Z_i = -\frac{L_i}{\ell-1}$ with the other half of the probability. Note that when $B_i = 0$, the probability that Z_i takes one of the possible values in $\frac{1}{2}$ (thus is uniform).

Next, let $\mathcal{B} = \{B_i : B_i = 1\}$ and $\bar{\mathcal{B}} = \{B_i : B_i = 0\}$. Let also that $T = |\bar{\mathcal{B}}|$. One can see that $\mathbb{E}[T] = \frac{2d}{\ell}$. In addition, because $d = \omega(\ell \log \ell)$, with overwhelming probability that $T \geq \frac{d}{\ell}$.

We now can see that

$$\mathbb{E}\left[\sum_{i\in\mathcal{B}} Z_i\right] = -\left(\sum_{i\leq d} L_i\right)\frac{\ell-2}{\ell(\ell-1)}$$

In addition, $\mathbb{E}[Z_i \mid i \in \bar{\mathcal{B}}] = L_i\left(1 - \frac{1}{\ell-1}\right)\frac{1}{2}$ for any $i \in \bar{\mathcal{B}}$. Next, we define a random variable to "debias" Z_i, conditioned on $i \notin \mathcal{B}$, i.e., for any $i \in \bar{\mathcal{B}}$

$$\tilde{Z}_i = \begin{cases} L_i - L_i\left(1 - \frac{1}{\ell-1}\right)\frac{1}{2} & \text{with probability } \frac{1}{2} \\ -\frac{L_i}{\ell-1} - L_i\left(1 - \frac{1}{\ell-1}\right)\frac{1}{2} & \text{with probability } \frac{1}{2}. \end{cases} \tag{3.36}$$

Note that $\mathbb{E}[\tilde{Z}_i] = 0$ and $L_i - L_i\left(1 - \frac{1}{\ell-1}\right)\frac{1}{2} = \frac{L_i}{\ell-1} + L_i\left(1 - \frac{1}{\ell-1}\right)\frac{1}{2}$. Next, we have

$$\begin{aligned}\sum_{i\le d} Z_i &= \sum_{i\in\mathcal{B}}\left[-\frac{L_i}{\ell-1}\right] + \sum_{i\in\bar{\mathcal{B}}}\left(\tilde{Z}_i + L_i\left(1-\frac{1}{\ell-1}\right)\frac{1}{2}\right)\\ &= \underbrace{\left(\sum_{i\in\mathcal{B}}\left(-\frac{L_i}{(\ell-1)}\right) + \sum_{i\notin\mathcal{B}} L_i\left(1-\frac{1}{\ell-1}\right)\frac{1}{2}\right)}_{\Psi_1} + \underbrace{\left(\sum_{i\notin\mathcal{B}}\tilde{Z}_i\right)}_{\Psi_2}.\end{aligned}$$

One can see that *(i)* the sign of Ψ_2 is independent of the sign of Ψ_1, and *(ii)* one of $\Pr[\Psi_1 \ge 0] \ge \frac{1}{2}$ and $\Pr[\Psi_1 \le 0] \ge \frac{1}{2}$ must hold. Wlog, assume that $\Pr[\Psi_1 \ge 0] \ge \frac{1}{2}$. By Theorem 3.12.3, we have $\Pr\left[\Psi_2 \ge \frac{c_1}{\log d}\sqrt{T}\right] = \Omega(1)$.

Finally, we have

$$\begin{aligned}\Pr\left[\Psi_1 + \Psi_2 \ge \frac{c_1}{\log d}\sqrt{\frac{d}{\ell}}\right] &\ge \Pr[\Psi_1 \ge 0]\Pr\left[\Psi_2 \ge \frac{c_2}{\log d}\sqrt{T} \mid \Psi_1 \ge 0\right]\\ &\ge \Pr[\Psi_{\ge} 0]\Pr\left[\Psi_2 \ge \frac{c_1}{\log d}\sqrt{\frac{d}{\ell}} \mid \Psi_1 \ge 0\right]\\ &= \Omega(1).\end{aligned}$$

The second inequality uses $T \ge \frac{d}{\ell}$ whp. □

3.13 Experiments

We evaluate our algorithms on the Chinese market, the second largest market in the world (by value). We describe the dataset collection and setup of experiments in Sec. 4.5.2, and the evaluation metrics and additional explanation of baselines in Sec. 3.13.2. This is followed by the improvement over individual factors, analysis for our performance and trading simulation results, and visualization for our learned latent space and the importance of factors in Sec. 4.6.

3.13.1 Datasets collection and experimental setup

Datasets collection. The specific description of the used dataset is as follows:

(1) Chinese stock data: Our data set consists of daily prices and trading volumes of approximately 3,600 stocks between 2009 and 2018. We use open prices to compute the

returns and we aim to predict the next 5-day returns, in which the last three years are out-of-sample. We examine two universes. (i) *Universe 800* is equivalent to the S&P 500 and consists of 800 stocks, and (ii) *Full universe* consists of all stocks except for illiquid ones. The average "size" (in either capital or trading volume) in *Universe 800* is larger than the average "size" of the *Full universe*.

(2) Technical factors: We manually build 337 technical factors based on previous studies [63, 34, 78, 6, 125]. All these factors are derived from price and dollar volume. See also App. 3.15.

(3) Barra factor dataset: We use a third-party risk model known as the Barra factor model [122]. The model uses 10 real-valued factors and 1 categorical variable to characterize a stock. The real-valued factors known as "style factors" include beta, momentum, size, earnings yield, residual volatility, growth, book-to-price, leverage, liquidity, and non-linear size. The categorical variable represents the industrial sector the stock is in. We do not use the categorical variable in our experiments. Table 3.3 defines the style factors.

Barra factors name	**Beta**	**Momentum**	**Size**	**Earnings Yield**	**Residual Volatility**
Description	Measure of volatility.	Rate of acceleration of a security's price or volume.	Total equity value in market.	The percentage of how much a company earned per share.	The volatility of daily excess returns.
Barra factors name	**Growth**	**Book-to-Price**	**Leverage**	**Liquidity**	**Non-linear Size**
Description	Measure of the growth rate.	firm's book value to its market capitalization.	Measure of a firm's leverage rate.	Measure of a firm's liquidity.	Non-linear transformation of size factor.

Table 3.3: Barra style factors from [122].

(4) News dataset: We crawled financial news between 2012 and 2018 from a major Chinese news website Sina. We collected a total number of 2.6 million news articles. Each article can refer to one or multiple stocks. On average, a piece of news refers to 2.94 stocks. We remark that our way to use news data sets deviates from standard news-based models for predicting equity returns [39, 72]. Most news-based models aim to extract sentiments and events that could directly impact one or more related stocks' prices. Rather than building links between events and the stock fluctuation, we use news dataset to identify similarities between stocks. i.e., when two stocks are mentioned often, they are more likely to be similar. This is orthogonal to how the news itself impacts the movement of stock

prices.

Model and training. We use three years of data for training, 10 months of data for validation and one year of data for testing. We re-train the model every testing year. For example, the training set starts from Jan. 1, 2012, to Dec. 31, 2014. The corresponding validation period is from Jan. 15, 2015, to Dec. 16, 2015. We use the validation set to select the hyperparameters and build the model. Then we use the trained model to forecast returns of equity in the same universe from Jan. 1, 2016, to Dec. 31, 2016, where we set 10 trading days as the "gap". Then we re-train the model by using data in the second training period (Jan. 1, 2013, to Dec. 17, 2015). We set a "gap" between the training and validation periods, and the validation periods testing dataset to avoid looking-ahead issues.

3.13.2 Additional explanation about evaluation matrices and baselines

Computing t-statistics. Recall that $\mathbf{y}_t \in \mathbf{R}^d$ is a vector of responses and $\hat{\mathbf{y}}_t \in \mathbf{R}^d$ is the forecast of a model to be evaluated. We examine whether the signals are correlated with the responses, i.e., for each t we run the regression model $\mathbf{y}_t = \beta_t \hat{\mathbf{y}}_t + \epsilon$ and test whether we can reject the null hypothesis that the series $\beta_t = 0$ for all t. Note that the noises in the regression model are serially correlated so we use Newey-West [119] estimator to adjust serial correlation issues. Consider, for example, a coin-tossing game, in which we make one dollar if our prediction of a coin toss is correct or lose one dollar otherwise. When our forecast has 51% accuracy, we are guaranteed to generate positive returns in the long run by standard concentration results. Testing whether our forecast has better than 51% accuracy needs many trials because, e.g., when there are only 100 tosses, there is a $\approx 40\%$ probability that a random forecast has a $\geq 51\%$ accuracy rate.

An example of compare correlation vs MSE. Consider a case where the true returns of Google and Facebook are +2% and +4%, respectively. Let forecast A be -1% (Google) and -1% (Facebook), and let forecast B be +20% (Google) and +40% (Facebook). While forecast A has a smaller MSE, forecast B is more accurate and more profitable (e.g., the directions of the returns are predicted correctly).

Sharpe Ratio. The popular Sharpe Ratio measures the performance of an investment by adjusting for its risk.

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p}, \tag{3.37}$$

where R_p is the return of the portfolio, R_f is the risk-free rate, and σ_p is the standard deviation of the portfolio's excess return.

PnL. Profit & Loss (PnL) is a standard performance measure used in trading and captures the total profit or loss of a portfolio over a specified period. The PnL of all forecasts made on day t is given by

$$\text{PnL} = \frac{1}{d}\sum_{i}^{d} sign(\hat{\mathbf{y}}_{t,i}) * \mathbf{y}_{t,i}, \quad t = 1, \ldots, n, \tag{3.38}$$

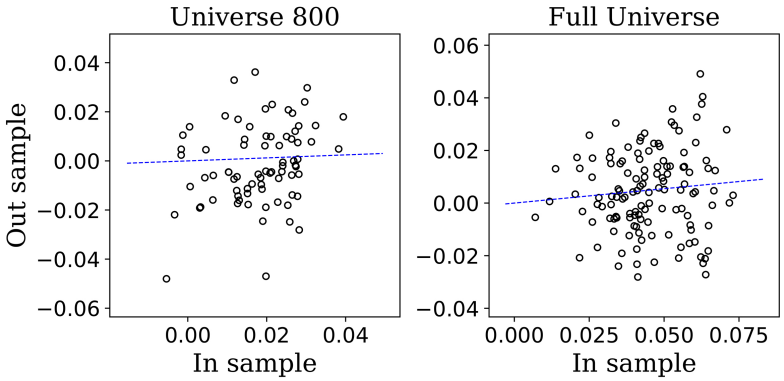
Additional explanation for recent CAMs

• **SFM [167].** SFM decomposes the hidden states of an LSTM [132] network into multiple frequencies by using Discrete Fourier Transform (DFT) so the model can capture signals at different horizons.

• **HAN [72].** This work introduces a so-called hybrid attention technique that translates news into signals.

• **AlphaStock [155].** This work is proposed by [155]. AlphaStock integrates deep attention networks reinforcement learning with the optimization of the Sharpe Ratio. For each stock, AlphaStock uses LSTM [135] with attention on hidden states to extract the stock representation. Then AlphaStock uses CAAN, which is a self-attention layer, to capture the interrelations among stocks. Specifically, CAAN takes the stock representations as inputs to generate the stock's winning score. We implement LSTM with basic CAAN and change the forecast into return instead of winning scores.

• **ARRR** ARRR [161] is a new regularization technique designed to address the overfitting issue in vector regression under the high-dimensional setting. Specifically, ARRR involves two SVD, the first SVD is for estimating the precision matrix of the features, and the second SVD is for solving the matrix denoising problem.



Correlation	*Universe 800*	*Full universe*
In-sample	0.0178	0.0206
In-sample with NN (our method)	0.0368	0.0662
Out-of-sample	0.0110	0.0141
Out-of-sample with NN	0.0129	0.0191
Average improvement (out,%)	17.27	35.46
% of improved factors	7.91	12.86

Figure 3.7: Improving factor level forecasting power by the nparam-gEST algorithm.

3.13.3 Experiment evaluation

Improvement over individual factors We fit the nparam-gEST model, where $\mathbf{x}_{t,i}$ is a stand-alone technical factor to understand whether a technical factor's forecasting power can be improved by using neighborhood information (see App. 3.15). We say a factor is "improvable" when its forecasting power in the in-sample data is better by a certain margin (in correlation). We examine the behaviors of these improvable factors. Fig. 3.7 shows that the average (out-of-sample) improvement in correlation is 17.27% for *Universe 800*, and 35.46% for *Full universe*.

Detailed results for each testing year Tables 3.4 and 3.5 list the results for each testing year in *Universe 800* and *Full universe*. The bold fonts denote the best performance in each group. The results are consistent with the Table 3.2. Note that we also report weighted correlation and weighted t-statistic. The weights are determined by the historical dollar volume of the asset. These statistics are useful because the positions taken by the optimizer are sensitive to historical dollar volumes.

		2016				2017				2018			
	Our CAMs	core	w_corr	t-stat	w_t-stat	corr	w_corr	t-stat	w_t-stat	corr	w_corr	t-stat	w_t-stat
Lin-PVEL	Opt.	0.1084	**0.1149**	9.9081	**7.9814**	**0.0388**	**0.0624**	**3.0441**	**3.8233**	**0.0820**	**0.1037**	**7.4293**	**7.2038**
	DD	0.1064	0.1109	9.7619	7.4212	0.0284	0.0541	2.1949	3.1475	0.0729	0.0972	6.9855	6.8733
nparam-gEST	Opt.	0.0800	**0.0595**	**6.1201**	**2.9453**	-0.0067	**-0.0095**	-0.4554	**-0.4800**	0.0604	0.0461	**4.2237**	2.2608
	DD	**0.0805**	0.0577	6.0357	2.7922	**-0.0051**	-0.0102	**-0.3472**	-0.5496	0.0582	**0.0446**	4.1457	**2.3772**
MLP	Opt.	**0.0958**	0.0917	**7.4846**	5.0039	**0.0050**	0.0182	0.3610	**0.9769**	**0.0641**	0.0602	5.7370	3.4793
	DD	0.0940	**0.0919**	7.3239	**5.0924**	0.0047	0.0165	0.3308	0.8749	0.0634	**0.0604**	6.0943	3.5154
LSTM	Opt.	**0.0662**	**0.0762**	5.7290	**4.5106**	**-0.0216**	-0.0144	-1.5466	-0.7624	**0.0413**	**0.0423**	**3.7099**	**2.4316**
	DD	0.0606	0.0682	4.8167	4.0641	-0.0025	**0.0017**	**-0.1520**	**0.0803**	0.0110	0.0356	0.9228	1.8596
Linear	Opt.	**0.0726**	**0.0708**	**5.1011**	**3.5324**	**0.0054**	0.0166	0.3429	0.8720	**0.0567**	**0.0679**	**4.1086**	**4.1605**
UM: poor man Lin-PVEL		**0.1093**	0.1128	**10.1352**	7.5786	0.0242	0.0499	1.7650	2.9580	0.0688	0.0970	6.3840	6.6570
UM: poor man nparam-gEST		0.0788	0.0579	6.0820	2.8561	-0.0061	-0.0096	-0.4083	-0.4862	0.0569	0.0442	3.7779	2.1036
UM: MLP		0.0861	0.0812	5.8771	4.2409	0.0052	0.0132	**0.3800**	0.6764	0.0609	0.0571	**6.1635**	**3.7053**
UM: LSTM		0.0619	0.0632	**6.5873**	4.1299	-0.0253	-0.0215	-1.7873	-1.1504	0.0169	0.0183	1.4487	1.0374
UM: Lasso		-0.0046	0.0088	-0.3889	0.5531	0.0282	**0.0333**	**2.1633**	**2.0936**	0.0083	0.0153	1.2997	1.6726
UM: Ridge		0.0290	0.0406	3.4617	2.8301	-0.0064	-0.0161	-1.3527	-1.3455	0.0091	0.0066	1.5421	0.5618
UM: GBRT		0.0655	0.0601	9.9051	5.4083	0.0419	0.0565	5.6987	5.0517	0.0476	0.0606	7.1179	6.4332
UM: SFM		0.0114	0.0102	0.9237	0.6828	0.0097	0.0081	0.6644	0.4479	0.0078	-0.0133	0.6194	-0.8263
Existing CAM: Alpha		0.0132	0.0165	2.3632	1.8841	0.0135	0.0133	2.5594	1.6109	-0.0062	-0.0110	-1.3995	-1.3236
Existing CAM: HAN		0.0096	0.0056	1.0205	0.4777	0.0060	0.0088	0.4980	0.5455	0.0160	0.0101	1.9352	0.7273
Existing CAM: VR		0.0207	0.0038	1.8590	0.2582	0.0087	0.0192	0.9239	1.6069	0.0174	0.0248	1.6513	1.3219
Existing CAM: ARRR		0.0593	0.0657	3.8366	3.2866	-0.0083	-0.0043	0.3975	0.578	0.0432	0.0533	3.3669	3.9343

Table 3.4: The by year results for *Universe 800*

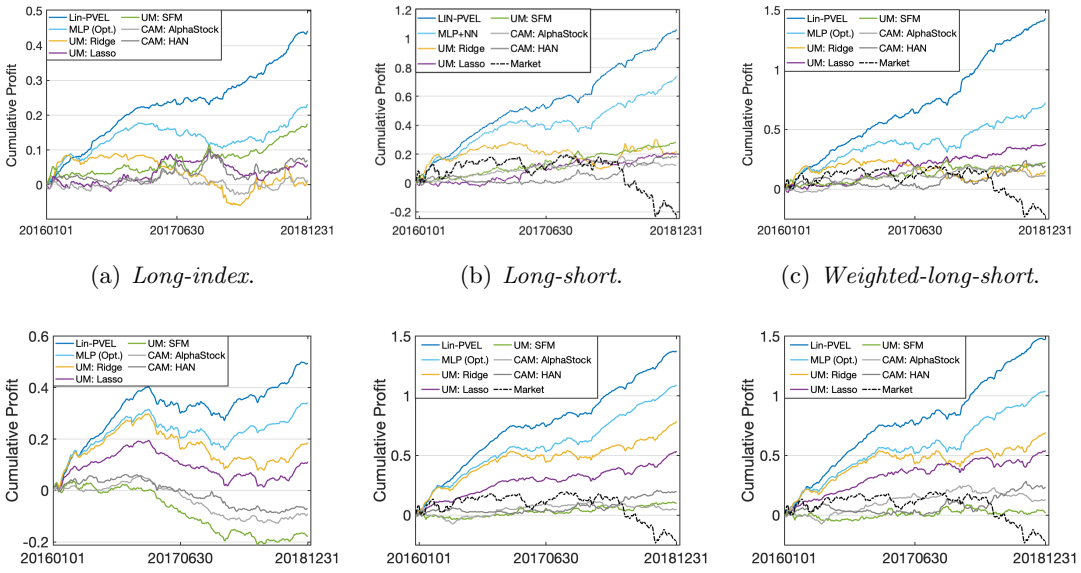
		2016				2017				2018			
Method	Our CAMs	corr	w_corr	t-stat	w_t-stat	corr	w_corr	t-stat	w_t-stat	corr	w_corr	t-stat	w_t-stat
Lin-PVEL	Opt.	0.1328	**0.1316**	12.1131	8.9092	0.0564	0.0590	4.4505	3.3487	**0.0939**	**0.1122**	8.2186	7.0727
	DD	**0.1358**	0.1308	**12.7510**	**9.0186**	**0.0584**	**0.0632**	**4.8204**	**3.5678**	0.0859	0.1062	**9.5940**	**8.1365**
nparam-gEST	Opt.	**0.1045**	**0.0969**	**10.3212**	**6.6829**	0.0159	**0.0129**	1.2465	**0.7205**	**0.0650**	**0.0559**	**5.6303**	**3.1603**
	DD	0.1039	0.0941	8.9599	6.1395	**0.0174**	0.0118	**1.3356**	0.6298	0.0596	0.0463	4.8991	2.3960
MLP	Opt.	**0.1072**	**0.0983**	8.3802	6.1198	0.0290	**0.0219**	1.9765	**1.0954**	**0.0851**	**0.0876**	**11.0013**	**5.9272**
	DD	0.0935	0.0921	**8.4685**	**6.6603**	**0.0303**	0.0212	2.1287	1.0544	0.0776	0.0788	8.4386	5.0757
LSTM	Opt.	**0.0744**	**0.0750**	**7.0918**	**4.5892**	0.0210	0.0200	1.3738	0.8513	**0.0465**	**0.0523**	**5.6732**	**3.3210**
	DD	0.0476	0.0532	4.1179	3.9801	**0.0327**	**0.0292**	**2.7048**	**1.4664**	0.0441	0.0462	4.5315	2.5036
Linear	Opt.	**0.0995**	**0.0956**	7.6410	5.7275	0.0123	0.0041	0.7983	0.1940	**0.0527**	**0.0684**	**5.1804**	**4.4595**
UM: poor man Lin-PVEL		0.1279	0.1214	11.4010	8.2082	0.0488	0.0517	3.6993	2.8182	0.0713	0.0920	7.1887	5.9714
UM: poor man nparam-gEST		0.1002	0.0957	8.8982	6.2661	0.0169	0.0112	1.2822	0.5872	0.0580	0.0457	4.8490	2.4000
UM: MLP		0.0837	0.0830	6.3470	5.4216	0.0286	0.0123	**2.3251**	0.6869	0.0697	0.0449	8.1207	2.5859
UM: LSTM		0.0684	0.0577	5.7502	3.6079	-0.0007	-0.0057	-0.0401	-0.2410	0.0379	0.0370	3.4094	1.8486
UM: Lasso		0.0589	0.0612	**9.4412**	**8.2446**	-0.0032	-0.0028	-0.3080	-0.1819	0.0313	0.0169	2.6735	0.8430
UM: Ridge		0.0631	0.0636	5.9308	4.4291	**0.0152**	**0.0168**	**0.9798**	**0.8296**	0.0290	0.0413	2.7536	2.2439
UM: GBRT		0.0898	0.0842	13.6203	9.5775	0.0531	0.0687	5.8328	7.0454	0.0588	0.0711	8.5605	7.0563
UM: SFM		-0.0055	-0.0058	-0.4868	-0.4281	0.003	0.0096	0.3578	0.708	0.0107	0.0057	1.2447	0.4851
Existing CAM: Alpha		0.0076	0.0109	1.2236	1.3496	0.0093	0.0123	2.3817	1.9694	0.003	0.008	0.778	1.4379
Existing CAM: HAN		0.0135	0.0081	1.7515	0.7924	-0.0008	-0.0020	-0.0791	-0.1253	0.0114	0.0098	1.5316	0.7547
Existing CAM: VR		0.0031	0.0033	0.3887	0.2949	-0.0056	-0.0245	-0.7108	-1.868	0.0148	0.0138	2.0156	0.8331
Existing CAM: ARRR		0.0527	0.0714	2.9072	3.2284	-0.0067	-0.0169	0.6266	-0.2307	0.0205	0.0275	2.0334	1.2328

Table 3.5: Yearly results for *Full universe*.

Simulation and PnL. Fig. 3.8 shows three ways to simulate investments on our signals for testing years from 2016 to 2018 for *Universe 800* and *Full universe*. (i) Long-index portfolio: Long-only minus the market index. (ii) Long-short portfolio[1]: By allowing short-selling, we can execute on negative forecasts to understand the overall forecasting quality. (iii) Weighted-Long-short portfolio: We weight an investment by the historical turnover of the asset. We conduct the trading in the daily granularity and select the stocks from the top 20% strongest forecast signals. We can see that our signals are consistently better than other baselines in both long/long-short. The results confirm that our method

[1]Short is implementable in the Chinese market only under special circumstances, e.g., through brokers in Hong Kong under special arrangements.

generates stronger and more robust signals for trading.



(a) *Long-index.* (b) *Long-short.* (c) *Weighted-long-short.*

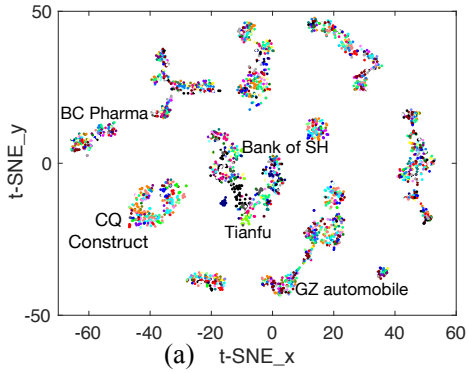
(d) *Long-index.* (e) *Long-short.* (f) *Weighted-long-short.*

Figure 3.8: Cumulative PnL (Profit & Loss) curves of the top quintile portfolio (i.e., on any given day, we consider a portfolios with only the top 20% strongest in magnitude predictions, against future market excess returns). (a)-(c) are for the *Universe 800* and (d)-(f) are for the *Full universe*.

Visualization/Qualitative examination

(1) Visualization for learned stock latent space. We examine the latent positions we learned, and draw two observations. *(i) Latent positions are not driven by sectors.* One possible explanation of our models' forecasting power is that they capture sector-related signals, e.g., growth of one airline implies the growth of others. Our visualization in Fig. 3.9 shows this is not the case. *(ii). Interactions are fine-grained.* We also present the neighbors uncovered by our pipeline, and also those found by AlphaStock for five stocks (all well known to the public). Our algorithm picks up different embeddings for these five stocks compared to AlphaStock, which indicates we discover an orthogonal signal.

(2) Factor importance. The linear boosting algorithm is based on a large collection of "linear models" $\{g_m(\cdot)\}_{m\leq b}$. Each base learner is a linear model with 3 linear terms (3 features) and 3 quadratic interaction terms. Here one feature can be selected by different



(a)

Center	Algorithm	Neighbor 1	Neighbor 2	Neighbor 3
CQ Construct 重庆建工 (construction)	Ours	Hunan Friendship	North China Pharma	Citic Helicopter
	AlphaStock	Yuntianhua	Ningbo Marine	Tianjin Teda
Tianfu Energy 天富能源 (utility)	Ours	Top Energy	Fuxing Tech	HuaYuan Property
	AlphaStock	SH Industrial	GZ Develop	Jilin Electric
GZ Automobile 广汽集团 (car)	Ours	SAIC Motor	SH Port	Huayu Automotive
	AlphaStock	Huayu Automobile	Hengyi Petrochemical	Kangde Material
BC Pharma 步长制药 (medicine)	Ours	Mantis Construct	China Meheco	ZJ Juhua
	AlphaStock	CR Pharma	National Medicines	Lingrui Pharma
Bank of SH 上海银行(bank)	Ours	Bank of Jiangsu	Bank of Beijing	Sinopec
	AlphaStock	Commercial Bank	Tianjin Realty	China Coal

(b)

Figure 3.9: (a): t-SNE for our latent embedding (colors are coded by sectors); (b): Examples of stocks and their neighbors.

base learners (See Sec. 3.3 for details). We examine the "feature importance" by counting the number of times a feature is selected. Table 3.6 and Table 3.7 show the 10 most important features in our model for different universes and different test years. We can observe some factors, such as Amihuds_3 (an indicator to measure "illiquidity"), are consistently important for different test years and different universes.

Year	Features 1	Features 2	Features 3	Features 4	Features 5	Features 6	Features 7	Features 8	Features 9	Features 10
2016	**Amihuds_3**	**Amihuds_6**	**Vol_regress_1m**	**alpha028**	CO	CCI_1	CR_5	PSY_12	PSY_26	**VROC_15**
2017	**Amihuds_3**	**Amihuds_6**	**Vol_regress_1m**	**alpha028**	alpha039	**alpha074**	**VROC_15**	vmacd_5_slow	MI	alpha072
2018	**Amihuds_3**	**Vol_regress_1m**	Vol_regress_4m	mf_020	**alpha028**	**alpha074**	**VROC_15**	mf_038	VCR_med	mf_005

Table 3.6: Feature Importance of Linear Boosting (*Universe 800*).

Year	Features 1	Features 2	Features 3	Features 4	Features 5	Features 6	Features 7	Features 8	Features 9	Features 10
2016	**Amihuds_3**	**Amihuds_6**	**Vol_regress_1m**	**alpha039**	CO	RSV_3	VCR_quick	vrsi_26	DMI_10	PSY_12
2017	**Amihuds_3**	**Amihuds_6**	**Vol_regress_1m**	**alpha039**	**alpha044**	D_10	K_5	**MI**	**alpha078**	mf_038
2018	**Amihuds_3**	**Vol_regress_1m**	**Vol_regress_4m**	alpha041	**alpha044**	alpha050	**MI**	VROC_15	**alpha078**	KDJ

Table 3.7: Feature Importance of Linear Boosting (*Full universe*).

3.14 FAQ

Why focus on technical features but fundamental features are not included?

Prediction models can be categorized into the following four groups based on the technical and non-technical features used by the model. *Group 1. Technical models.* Technical models rely on technical factors to predict stock returns. A technical feature is constructed from information related to trading activities, such as a stocks historical

price/trading volume, the evolution of the bid-ask spreads, or the number of transactions worth 1 million dollars or more on the last trading day. A non-technical feature could be the revenues of the underlying company (see models in group 2 discussed below), news about the company, or postings on the social networks of its major shareholders. *Group 2. Fundamental models.* Fundamental models rely on statistics related to a company's financial status, such as revenues, research expenses, and price-earning ratios. These technical factors, also called fundamental factors, can characterize a company's long-term behavior, such as whether its products will stay competitive in the next few years. Fundamental factors are usually extremely sparse. For example, quarterly revenue information provides four datapoints points for one stock each year [36, 127, 81]. *Group 3. Macro models.* Macro models rely on macroeconomic conditions' of the nation which a company calls home, or is registered to conduct business. For example, if a nation subsidizes renewable energy, an electric vehicle company may benefit from a rise in its stock price. *Group 4. Event models.* Event models capture price fluctuation due to the release/announcement of new information to the market. For example, replacing the CEO can cause a positive price fluctuation, whereas breaking news about merger and acquisition can drive down the acquiring companys stock price.

Not using fundamental and macro factors. These two types of factors are usually not used for building machine learning models forecasting 5-day returns. *(i)* fundamental and macro factors have long-term (multiple months) impact to stock prices so they are not suitable for forecasting 5-day returns. *(ii)* Simple linear regressions are widely used for fundamental and macro models partly because of data sparsity.

Why do you perform experiments only on the China market? *(i)* The US and Chinese markets are the two largest equity markets by trading volume [102], and the Chinese market has attracted increasing attentions. *(ii)* Among important recent results, many were evaluated soly in the Chinese market (e.g., [155, 49, 26, 161, 64, 39, 167, 72]). *(iii)* The license cost for China market data is significantly lower than that of the US

market.

Why not making money, instead of publishing? Empirical asset pricing is a long and established area [63, 77, 43, 45, 34], in which the predictability of equity returns is extensively studied. The reasons we share our research results are similar to those of researchers in the same field. Some of our considerations include:

- *Signal is only a part of a large system.* Building a profitable system requires substantially more optimization and engineering efforts than merely discovering the signals [50]. In other words, while identifying "alpha" signals is an important part of a trading system, other parts (such as data acquisition and execution) also play an important role in building a reliable and profitable trading system.
- *Execution quality.* Hundreds of stocks need to be traded every day. While building this trading system (using standard brokers such as Interactive Broker or its equivalence in China) is feasible, it involves a highly non-trivial amount of engineering work. An effective and robust execution system without incurring large slippage cost is itself a highly nontrivial component and is beyond the scope of this paper.
- *Fees and robustness* Although our strategy is stable and has a high Sharpe, we note that we did not account for the the transaction cost and slippage, which are nontrivial since the holding period is only 5-days. We estimate that after accounting for the transaction cost and slippage, with approximately 5% to 10% probability that an individual may still experience a loss in any specific year. This may not be acceptable for extremely risk averse investors. Hence, to obtain a robust industry level strategy, many components of our strategy need to be optimized.

3.15 Factor list

In this section, we provide a table to explain the technical factors we build. The last column shows the number of features for one factor category since we can build several factors for different time scales.

Index	Factor	Description	Number
1	AD [34]	Accumulation/Distribution is a cumulative indicator that uses price and volume to assess whether an equity is cumulative or distributed.	3
2	MACD [34]	Moving Average Convergence Divergence (MACD) is a trend-following momentum indicator that shows the relationship between two moving averages of an equity's price.	9
3	RSI [34]	Relative strength index is a momentum indicator to measure the magnitude of one equity's recent price changes.	3
4	SRMI [34]	Shifted relative momentum index.	4
5	VRSI [34]	VRSI is an indicator similar to RSI, but it measures volume rather than price.	5
6	Amihud [6]	Amihud illiquidity measure is mainly used to calculate the level of illiquidity of a stock.	6
7	Volume Std	Standard deviation of volume.	2
8	CR	Cumulative return	3

Continued on next page

Index	Factor	Description	Number
9	ER [34]	Elder-Ray indicator defines bull and bear power by 13 days exponential moving average of highest and lowest price.	1
10	DDI [34]	The DDI indicator measures the relative relationship of one stock's highest and lowest prices.	3
11	BBI [34]	BBI measures the difference between the moving averages of stock indices closing prices in different periods.	3
12	EoM [34]	Exponential moving average of BBI indicator.	2
13	KDJ [34]	KDJ indicator is a technical indicator used to analyze and predict changes in stock trends and price patterns in a traded asset.	3
14	Hurst [74]	The Hurst indicator is used to do chaotic fractal analysis for the capital market. It reflects the result of a long series of interconnected events.	3

Continued on next page

Index	Factor	Description	Number
15	Klinger [34]	Klingers volume swing indicator determines the long-term capital flow trend while maintaining sensitivity to short-term capital flows, so it can be used to predict short-term price inflection points.	2
16	LQ [45]	An indicator to measure one equity's liquidity for a time period.	8
17	MF [34]	Money flow.	2
18	Close regression [34]	Linear regression on one equitys close prices.	2
19	High regression [34]	Linear regression on one stocks highest prices.	2
20	Low regression [34]	Linear regression on one stocks lowest prices.	2
21	Volume regression [34]	Linear regression on one stocks trading volume.	2

Continued on next page

Index	Factor	Description	Number
22	MFI [34]	The MFI indicator combines price and quantity and includes it in the scope of comprehensive consideration.	6
23	ROC [34]	A momentum oscillator that measures the percent change in price from one period to the next.	5
24	SRMI [34]	SRMI indicator revised momentum indicator based on the relation of closing prices of the day and the previous trading day.	3
25	TEMA [34]	Triple exponential moving average on closing prices.	3
26	TRIX [34]	Smoothed TEMA indicator.	3
27	ULCER [34]	ULCER indicator measures the relationship between the highest price and closing price of a stock over a period of time.	3
28	Volume Std [45]	Standard deviation of trading volume over a period of time.	3

Continued on next page

Index	Factor	Description	Number
29	Price Std [45]	Standard deviation of closing prices over a period of time.	3
30	Volume Mean [45]	Average of trading volume over a period of time.	3
31	Price Mean [45]	Average of closing prices over a period of time.	3
32	ASI [34]	The cumulative volatility index uses the stock's opening price, closing price, highest price, and lowest price to measure long-term trends.	3
33	EMV [34]	Ease of Movement is an oscillator to measure the ease of stock moving.	3
34	ATR [34]	The Average True Range is the average of the price range over the past n periods.	3
35	CO [34]	The CO indicator measures the difference between the index moving averages of stock AD indicators in different periods.	1

Continued on next page

Index	Factor	Description	Number
36	CV [34]	CV measures the difference between the exponential moving averages of stock price ranges in different periods.	1
37	CMO [34]	The Chande momentum oscillator is a price volume oscillator which is closely related to RSI indicator.	2
38	Ccurve [34]	Exponential moving average of past 7 and 14days return.	1
39	DMI [34]	The DMI indicator provides a basis for judging trends by analyzing changes in the balance of power between buyers and sellers during the rise and fall of stock prices.	13
40	VROC [34]	Volume change rate indicator measures the range of changes in trading volume at a certain moment.	3
41	M2C [34]	M2C indicator measures the relationship between stocks closing price and its exponential moving average.	4

Continued on next page

Index	Factor	Description	Number
42	OBV [34]	On-Balance Volume speculates on stock price trends by counting the 'number of trends' in volume changes.	2
43	PSY [34]	PSY is an indicator to count the number of rising days over a period of time.	4
44	PVT [34]	Price Volume Trend not only measured the price-volume trend relative to the previous day but also measures the relative price-volume trend N days ago.	2
45	RVI [34]	RVI is used to predict the direction of volatility, which predicts the strength of prices by calculating volatility rather than changes in prices.	6
46	StochRSI [34]	StochRSI is an oscillator indicator to measure the RSI indicator.	3
47	VOSC [34]	The volume moving average indicator measures the running trend of the volume and determines the direction of the trend change.	2

Continued on next page

Index	Factor	Description	Number
48	VEMA [34]	Exponential moving average of trading volume over a period of time.	2
49	RSV [34]	Stochastic value for each period.	5
50	WR [34]	Williams overbought/oversold index.	5
51	VR [34]	A measure that helps investors follow the volatility of a stock's price.	3
52	VCR [34]	Volume cumulative return.	3
53	EMA [34]	Exponential moving average.	1
54	Momentum Std [45]	Standard deviation of trading momentum over a period of time.	3
55	Momentum Mean [45]	Average of trading momentum over a period of time.	3
56	101 ALPHA [78]	101 ALPHA is published recently which includes a collection of 101 technical indicators. We choose 77 factors from them.	77

Continued on next page

Index	Factor		Description	Number
57	MF 1 [89]	Series	Money flow indicators series one. This group compares the volume-weighted average prices of trading orders in different groups (extra large, large, medium, and small).	12
58	MF 2 [89]	Series	Money flow indicators series two. This group compares the trading volume among different group of buy and57 sell orders (extra large, large, middle,small).	36
59	MF 3 [89]	Series	Money flow indicators series three. This group compares the volume of buy and sell orders over different periods (extra large, large, middle,small).	30
	Summary			337

Chapter 4

Equity2Vec: End-to-end Deep Learning Framework for Cross-sectional Asset Pricing

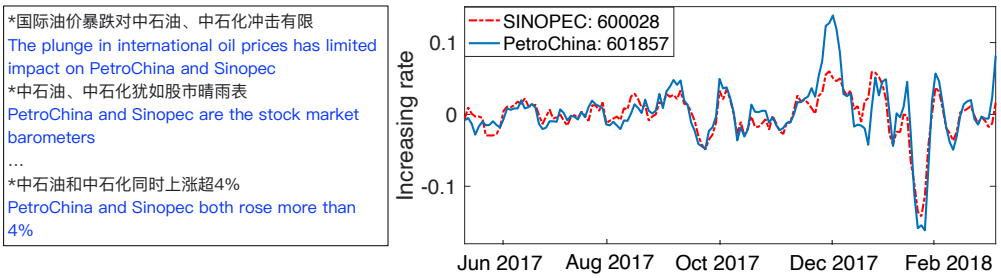
4.1 Introduction

It is widely acknowledged that forecasting stock prices is a difficult task. Most traditional efforts rely on time series analysis models, such as Autoregressive models [91], Kalman Filters [115], and technical analysis [83, 116, 117]. Deep neural networks, especially recurrent neural networks (RNN) [72, 149, 39, 59, 167, 132, 38] recently emerged as an effective solution for stock prediction tasks. Such lines of work have significantly increased in popularity in recent years, mostly fueled by the fact that a large collection of high-quality financial data sets have become available.

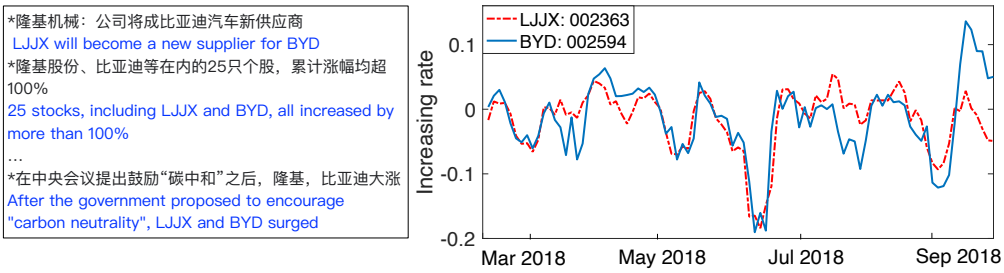
We observe two fundamental limitations in the prior works: *1. Cross-sectional effects are not properly leveraged.* Most existing approaches treat each stock independently and overlook the cross-sectional effect. The cross-sectional effect posits the fact that the information from one stock may influence/impact another stock's price change in both static and dynamic aspects. Statically, the stocks that share the same intrinsic properties

may move synchronously. For instance, when Twitter goes up, Facebook is more likely to go up because they are in the same sector (social media advertising). Dynamically, stock relation is also temporally evolving. For example, in early 2021, AMC theatres, GameStop, and BlackBerry suddenly exhibit co-movement driven by investors on social media who are buying up these stocks.

2. Heterogeneous data sets are not leveraged to their fullest extent. Most models use only one type of data (i.e., either textual information or "technical factors" [111, 72, 112, 161, 123] in numeric form). The latter is derived from prices and traded volumes. It remains open to building a model that leverages heterogeneous data sources. This model needs to reconcile and aggregate information from different data sources.



(a) The co-mentions capture sector relation between stocks



(b) The co-mentions capture supply-chain relation between stocks

Figure 4.1: Examples showing our key observations: When the news mention stocks frequently, the stocks are 1) likely to reflect relations, such as sector and supply-chain, 2) likely to have similar movement on prices.

An efficient stock embedding scheme that addresses the first limitation must determine: 1) what data source includes the co-movement information, 2) how to extract the cross-sectional signals from the data source, and 3) how to incorporate both the static and dynamic stock relations into the stock embedding. We propose EQUITY2VEC that answers

the three questions.

Ideally, the stock representation should reflect comprehensive relations such as sector, supply chain, value, growth, business cycles, volatility, and analysts' confidence towards the stock. One possible way is to collect such information manually from experts and analysts, but it is inefficient and costly. Further, there are no widely agreed upon standard approaches to converts people's opinions into stock representations. *Since millions of investors, analysts, and financial experts share opinions, events, comments, and transactions about stocks in the news, we consider using news as a data source to learn stock representations.*

Next, we make two key observations by analyzing news on stocks. When two stocks are frequently co-mentioned, 1) they are likely to share common characteristics such as sector and supply-chain relation, 2) their prices tend to have a similar trend. For example, the co-mentioned stocks in Figure 4.1 (a) are in the same sector (energy), while Figure 4.1 (b) shows they have a supply-chain relationship (i.e., LJJX is a supplier of BYD). In both cases, the prices of these co-mentioned stocks often move synchronously and most often in the same direction. *Based on our observations, we use news co-mention*[1] *to learn the stock representation.*

Moreover, Equity2Vec extracts the long-term (static) and evolving (dynamic) stock relations by the following approach. To capture the long-term relation, we build a global stock co-occurrence matrix with a long observation window. We extract the stocks' long-term representations via matrix factorization of the co-occurrence matrix. To learn the evolving relation, we build a stock graph that reflects the dynamic neighboring relations, where Equity2Vec propagates the embedding of a stock to its neighbors to capture the cross-sectional information of the stocks effectively.

To leverage the heterogeneous data sources (i.e., second limitation of prior works), we propose a framework that integrates the learned stock embeddings, news signals, and technical factors into a neural network model to make the final prediction. We perform extensive experiments on real-world data that contains more than 3,600 stocks in the

[1]Through the statistic analysis of real-world online media, a piece of news refers to 3.0 stocks on average.

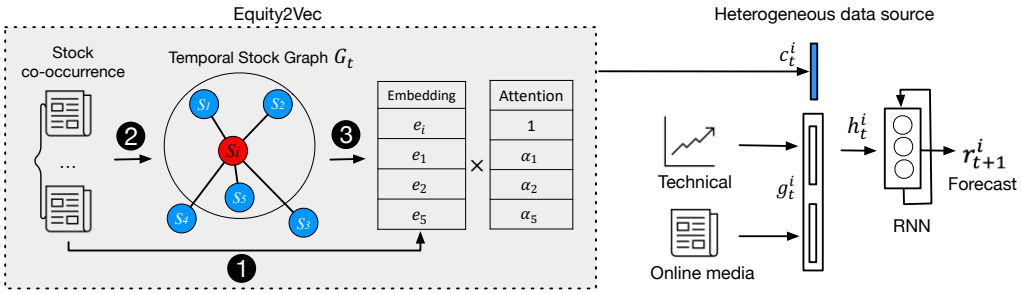


Figure 4.2: The illustration of our end-to-end framework. It contains the EQUITY2VEC component (Section 4.3) and heterogeneous data source component (Section 4.4).

Chinese stock market from 2009 to 2018. The experimental results demonstrate the effectiveness of stock representations extracted by EQUITY2VEC outperforms existing state-of-the-art works. The market trading simulation illustrates that EQUITY2VEC along with the proposed framework increases profit significantly.

In summary, this paper makes the following contributions:

- We propose EQUITY2VEC that incorporates news into stock embeddings. To the best of our knowledge, EQUITY2VEC is the first work that mines the stock representation from the news co-mention. Moreover, EQUITY2VEC captures both long-term (static) and evolving (dynamic) relations between stocks.
- We forecast stock prices using multiple categories of signals (i.e., heterogeneous data sets), including cross-sectional embeddings, technical signals, and financial news signals.
- Extensive experiments on real-world data sets confirm the efficacy of our approach, comparing favorably to state-of-the-art methods.

4.2 Preliminaries and Framework overview

Problem setting. Given a universe of n stocks $s_1, s_2, ..., s_n$, the stock price trend is log return for a given stock i on day t

$$r_t^i = \log\left(\frac{p_t^i}{p_{t-1}^i}\right) \tag{4.1}$$

where p_t^i denotes the open price of stock i on day t. We formulate the task of predicting the future price trend as a regression problem. The response is the future return r_{t+1}^i, and

$\mathbf{x}_t^i$ denotes the vector of features associated with stock i on day t. The historical features up to time t are defined as $\mathbf{x}_{\leq t}^i$. We aim to learn the function

$$r_{t+1}^i = f(\mathbf{x}_{\leq t}^i). \tag{4.2}$$

Framework Overview. Figure 4.2 shows our overall framework, which includes the EQUITY2VEC component and the heterogeneous data source component. The EQUITY2VEC component first learns the stocks' long-term relations from the global stock co-occurrence matrix, and then extracts the static stock embedding as e_i (❶). Then, at time t, we build the temporal graph G_t dynamically based on the local stock co-occurrence matrix to capture the evolving relations (❷). Within graph G_t, each solid circle represents a stock. Stocks close to one another are likely to be associated with a similar moving trend. Finally, our approach obtains the final stock representation c_t^i by propagating its neighbors' basic embedding via an attention mechanism (❸).

In the heterogeneous data source component, we integrate the stock embedding (c_t^i) with dynamic input (g_t^i) (generated from technical factors and online textual data), and denote the heterogeneous output as h_t^i. Finally, the RNN model forecasts future return r_{t+1}^i based on the input h_t^i.

4.3 Equity2Vec from news

We propose EQUITY2VEC, which mines the stock embeddings from the news, since such data comprises a valuable knowledge repository with rich relation information between stocks from the crowd of financial experts/journalists. Our approach is inspired by the observation (as depicted in Figure 4.1) that stocks frequently co-mentioned by the same news are likely to share similar properties and exhibit co-movements in their price trends. We formulate the stock co-occurrence matrix, and use matrix factorization to extract the *static* stock representation in Section 4.3.1. To explicitly leverage the cross-sectional signals and circumvent the challenge that the relations between stocks are evolving, we build a temporal stock graph and fine-tune the stock representations by *dynamically* infusing neighbors latent representations (Section 4.3.2).

4.3.1 Capturing long-term stock relations

We now focus on learning the long-term relation, and discuss how to address the dynamic relations in the next subsection.

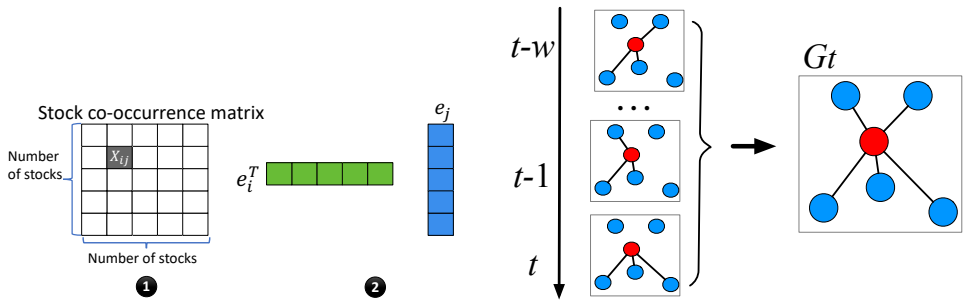


Figure 4.3: (a) Our approach to build the stock co-occurrence matrix and calculate the static embedding for stocks. (b) The construction of temporal graph.

Co-occurrence Matrix. We build a stock co-occurrence matrix by counting the stock co-occurrence within each news. We prefer the stock co-occurrence matrix instead of the entire news and stock matrix for the following two reasons: (1) The size of the stock co-occurrence matrix only depends on the number of stocks, and is much smaller than the news and stock matrix. (2) We only care about the stock representation, which thus renders news representations as not necessary. We use the global co-occurrence matrix to obtain static representations that reflect the essential relations between stocks in the long term.

Formally, suppose we have n stocks and $\mathbf{X} \in \mathbf{R}^{n \times n}$ denotes the stock co-occurrence matrix. Figure 4.3(a) (❶) shows $\mathbf{X}_{i,j}$, counting the number of articles that mention both s_i and s_j before testing phase. We associate a vector $e_i \in \mathbf{R}^d$ to represent the stock i's static representation, where d is the dimension of the stock representation. In this way, the similarity between stock i and stock j can be formulated as the inner product of e_i and e_j, given by $e_i^{\mathrm{T}} e_j$.

Matrix Factorization. Figure 4.3(a) (❷) shows that we adopt matrix factorization [85] to learn the embedding of the stocks. Here, matrix factorization works as a collaborative filtering method. The idea behind matrix factorization is to learn the latent representation

where stocks near each other will likely obtain similar embeddings. Given the co-occurrence matrix reflects the similarity between stocks, we obtain the latent representation of stocks through fitting the training data by optimizing the objective function

$$J_s = \sum_{i,j=1}^{n} (e_i^{\mathrm{T}} e_j - \mathbf{X}_{ij})^2. \tag{4.3}$$

In reality, there could exist prior bias of stocks as the prior preference of financial journalists/experts. Hence, we use b_i and b_j as the bias of stock i and stock j and introduce them to the objective function,

$$J_s = \sum_{i,j=1}^{n} (e_i^{\mathrm{T}} e_j + b_i + b_j - \mathbf{X}_{ij})^2 + \beta(||\theta||^2) \tag{4.4}$$

where $||\theta|| = (||e_i||^2 + ||e_j||^2) + b_i^2 + b_j^2$ and $\beta(||\theta||^2)$ is the regularization term that prevents overfitting.

4.3.2 Capturing evolving stock relations

The latent representation learned from the above global occurrence matrix reflects the static stock relations. Motivated by the fact that the stock relations are changing over time and cross-asset signals are beneficial towards stock price prediction, we further fine-tune the stock representation by building a temporal stock graph and infusing the neighbors' embedding dynamically.

Temporal stock graph. As shown in Figure 4.3(b), the construction of G_t consists of two steps. (1) Construct the stock graph using the co-occurrence matrix. Assume $\tilde{G}_t = \{\mathcal{V}, \mathcal{E}_t\}$ is the stock graph at time t, where $\mathcal{V} = \{s_1, \ldots, s_n\}$ is the set of stocks. $(s_i, s_j) \in \mathcal{E}_t$ if and only if s_i and s_j are co-mentioned by the news collected at time t. The edge eight over s_i and s_j is the number of co-occurrences across all news on date t. (2) Due to insufficient number of news about specific stocks in time t, we maintain a sliding window w (w is a hyper-parameter) to collect a sequence of stock graphs and then construct G_t by taking an exponential moving average of $\tilde{G}_{t-w}, ..., \tilde{G}_t, \tilde{G}_t$, where we assign a nearby graph a larger weight.

Propagation of neighbors' embedding via a stock attention mechanism. Given

the temporal graph (G_t) identifies the dynamic stock structure, it is crucial to appropriately update the stock representation by infusing the current neighbors' embedding. Motivated by the fact that not all the neighbors contribute to the current stock trend, we filter out the stocks that are too far away from the current stock (See Section 4.6 for more details). Specifically, for stock s_i, we focus on the k nearest neighbors when sorting by the edge weight (which reflects the magnitude of the co-occurrence), where k is a hyper-parameter. Formally, let $S_t(i)$ denote the set of k nearest neighbors of s_i in G_t at time t.

We introduce an attention mechanism [152] to infuse the neighbors embedding weighted by an assigned attention value. In this way, we reward the stocks offering more forecasting power by assigning them larger attention values.

$$c_t^i = \sum_{j \in S_t(i)} \alpha_{ij} e_j, \tag{4.5}$$

$$\sum_{j \in S_t(i)} \alpha_{ij} = 1 \text{ for } j \in S_t(i), \tag{4.6}$$

where c_t^i denotes the fine-tuned stock representation for stock i at time t, and $\alpha_{i,j} \in \mathbf{R}$ is the attention weight on the embedding e_j, which is given by

$$\alpha_{ij} = \frac{exp(f(e_i, e_j))}{\sum_{l \in S(i)} exp(f(e_i, e_l))}. \tag{4.7}$$

The weights define which neighboring stocks are more significant. $f(e_i, e_j)$ measures the compatibility between embeddings e_i and e_j, and is parameterized by a feed-forward network with a single hidden layer, which is jointly trained with other parts of the model. We let $f(\cdot, \cdot)$ have the following functional form

$$f(e_i, e_j) = \mathbf{v}_a^{\mathrm{T}} \tanh(\mathbf{W_a}[e_i; e_j] + b_a), \tag{4.8}$$

where $\mathbf{v}_a$ and $\mathbf{W}_a$ are weight matrices, and b_a is the bias vector [152], obtained during model training via backpropagation.

4.4 Leverage heterogeneous data sources

In this section, we show how to integrate heterogeneous data sources for making forecasts, and how we gather different sources of signals.

4.4.1 Sequential modeling

Finally, we integrate the stock embedding from EQUITY2VEC with stock dynamic features into the neural net model to predict the future return. The stock dynamic input is given by g_t^i, which stems from technical factors and news data. We overlay the stock vector c_t^i with g_t^i as an input. Specifically, let h_t^i be the hidden state at time t for stock i

$$h_t^i = [c_t^i, g_t^i], \tag{4.9}$$

where $[\cdot, \cdot]$ denotes a direct concatenation.

Keeping in mind that stock trends are highly influenced by a variety of time-series market signals, it is intuitive to take the historical features of a stock as the most influential input to predict its future trend. Therefore, we use Recurrent Neural Networks [110, 137] as the neural net model. LSTM is a variant of the recurrent net, which is capable of learning long-term dependencies. The final output is given by

$$v_{t-T}^i, ..., v_{t-1}^i, v_t^i = \text{LSTM}(h_1^i, h_2^i ... h_T^i; \theta_l), \tag{4.10}$$

where θ_l denotes the parameters from LSTM.

Temporal attention layer. Since a stock's historical data contributes to its price trend unequally, we adopt the attention mechanism at the temporal level. We consider

$$\begin{aligned} r_{t+1}^i &= \textstyle\sum_p \beta_p v_p^i, \\ \beta_p &= \frac{exp(f(v_p^i, v_q^i))}{\sum_q exp(f(v_p^i, v_q^i))}, \end{aligned} \tag{4.11}$$

where β_p is the attention weight for prior date p indicating the importance of the date. We then compute the weighted sum to incorporate the sequential data and temporal attention.

Assume we have m trading days and n stocks. We use the mean squared error as the loss function for gradient descent, given by

$$J = \frac{1}{mn} \sum_{i=1}^{n} \sum_{t=1}^{m} (r_{t+1}^{i} - \hat{r}_{t+1}^{i})^2. \tag{4.12}$$

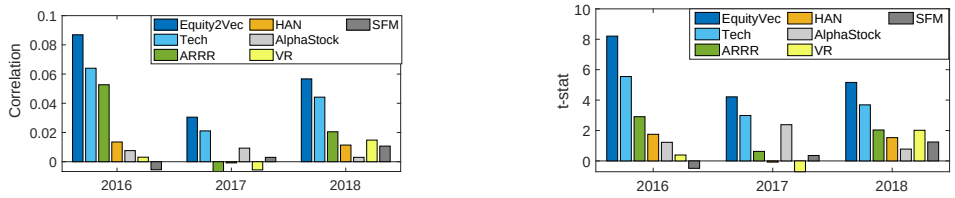
4.4.2 Gathering difference sources of alphas

Financial studies have attributed stock movements to three types of market information, i.e., cross-sectional signals, numerical technical indicators, and news features. To the best of our knowledge, the proposed framework is the first one that fuses technical factors, financial news and stock embedding together for stock predictions.

Stock graph: leveraging cross-sectional signals. Trading on cross-sectional signals (i.e., when Google goes up, Facebook is more likely to go up) is remarkably difficult because we need to examine all possible relations. We leverage news articles that mention multiple stocks to detect correlations between stock prices. Detecting co-movements by news appears to be much more effective than existing methods. Our EQUITY2VEC learns both long-term and evolving relations.

Technical factors: hand-built features are more effective. We note that features extracted by deep learning [40, 95] are often less effective than features (technical factors) crafted by financial professionals [34]. Thus, we overlay an LSTM over technical factors, so that we can simultaneously leverage expertise from financial professionals, and also extract serial correlations from deep learning models.

Financail news Advancing development of Natural Language Processing techniques has inspired increasing efforts on stock trend prediction by automatically analyzing stock-related articles [72, 31, 5]. We pre-trained WORD2VEC [108, 109] on the news corpus from the training data set to produce word embeddings. We average all the word vectors in a piece of news to represent the news vector. For a given date t and stock i, we compute the daily news vector by extracting the news related to stock i and averaging the news vectors within date t.



(a) (b)

Figure 4.4: Performance comparison in terms of correlation (a) and t-statistic (b) among our EQUITY2VEC, ARRR, HAN, AlphaStock, VR, and SFM. For both correlation and t-statistic, higher scores are better.

4.5 Evaluation Methodology

We now evaluate the methodology proposed. We focus on the Chinese market, whose value is the second largest in the world.

4.5.1 Data Collection

Chinese Equity Market. Our data set consists of daily prices and trading volumes of approximately 3,600 stocks between 2009 and 2018. We consider the universe with all the stocks except for the very illiquid ones. We use open prices (at 9:30 am) to compute the daily returns, and we focus on predicting the next 5-day returns. The last three years of the period are out-of-sample.

Technical Factors. We manually build 337 technical factors based on the previous studies [63, 34, 78, 6, 125]. "Technical factor" is a broad term encompassing indicators constructed directly from data related to trading activities. Specifically, all these factors are derived from price and dollar volume by mathematical calculation. Table 4.1 shows a set of popular technical factors.

News Dataset. We crawled all the financial news between 2009/01/01 and 2018/08/30 from a major Chinese news website Sina[2]. It has a total number of 2.6 million articles. Each article can refer to one or multiple stocks. On average, a piece of news refers to 2.94 stocks. We link each of the collected news articles to a specific stock if the news mentions the stock in the title or content. The timestamps of news published online are usually unreliable (the dates are reliable, but the hour or minute information is usually inaccurate).

[2]https://finance.sina.com.cn

Table 4.1: A set of popular technical indicators and the corresponding description.

Factors	Description
EMA	Exponential moving average over price or dollar volume. [34]
RSI	The magnitude of one equity's recent price changes. [34]
ROC	Price variation from one period to the next. [57]
Volume Std	Standard deviation of volume. [54]
VCR	Volume cumulative return [34]

We use news signals on the next trading day or later to avoid look-ahead issues.

4.5.2 Experimental settings

Training and Testing Data. We use three years of data for training, one year of data for validation, and one year for testing. The model is re-trained every testing year. For example, the training set starts from Jan 1, 2012 to Dec 31, 2014. The corresponding validation period is from Jan 15, 2015, to Dec 16, 2015. We use the validation set to tune the hyperparameters and build the model. Then we use the trained model to forecast returns of equity in the same universe from Jan 1, 2016 to Dec 31, 2016, where we set 10 trading days as the "gap". We set a "gap" between training and validation periods, and validation periods and testing periods to avoid look-ahead issues. The model is then re-trained by using data in the second training period (2013 to 2015) to make forecast on the second testing year 2017.

Parameter Setting. We use the standard grid search to select the hyper-parameters in our experiments. We build the global co-occurrence matrix in Section 4.3.1 by using all the news before the first day of the testing year. To learn the stocks' embedding, we tune the dimension of stocks' representation within {32, 64, 128, 256}. We explore the number of LSTM cell within {2, 5, 10, 20}. We greedily search the number of neighbors for the stock graph from no neighbors to all the neighbors. The sliding window for temporal graph w is tuned within {2, 5, 10, 20, 60}. In addition, we tune the learning rate within {0.001, 0.01} with the Adam optimizer [82], and set the batch size within {128, 256}.

Evaluation Metrics. We evaluate our performance in terms of correlation, t-statistic, and PnL.

Correlation. Unlike the other regression tasks, correlation [17] is a preferable metric in stock price prediction compared to MSE since the direction instead of magnitude is more crucial for forecasting return.

Significance test (t-statistics). The use of t-statistics estimators [119] can account for the serial and cross-sectional correlations. Recall that $\mathbf{r}_t \in \mathbf{R}^n$ is a vector of responses and $\hat{\mathbf{r}}_t \in \mathbf{R}^n$ is the forecast of a model to be evaluated. We examine whether the signals are correlated with the responses, i.e., for each t we run the regression model $\mathbf{r}_t = \beta_t \hat{\mathbf{r}}_t + \epsilon$, and test whether we can reject the null hypothesis that the series $\beta_t = 0$ for all t. Note that the noises in the regression model are serially correlated, and thus we use the Newey-West [119] estimator to adjust for serial correlation issues.

PnL. Profit & Loss (PnL) is a standard performance measure used in trading. PnL captures the total profit or loss of a portfolio over a specified period. The PnL of all forecasts made on day t is given by

$$\text{PnL} = \frac{1}{n} \sum_{i}^{n} sign(\hat{r}_t^i) * r_t^i, \quad t = 1, \ldots, m, \tag{4.13}$$

4.5.3 Baselines for comparison

To test our proposed deep learning framework, we compare our model against state-of-art baselines, as described below. For all the baselines, we use the validation data set to configure the hyper-parameters.

SFM [167]. SFM is designed for stock prediction. It decomposes the hidden states of an LSTM [132] network into multiple frequencies by using Discrete Fourier Transform (DFT) in order for the model to capture signals at different horizons.

HAN [72]. This work introduces a so-called hybrid attention technique that translates news into signals. HAN uses Word2Vec to transfer news into vectors, and uses RNN as

the sequential modeling method.

AlphaStock [155]. AlphaStock integrates deep attention networks reinforcement learning with the optimization of the Sharpe Ratio. For each stock, AlphaStock uses LSTM [135] with attention on hidden states to extract the stock representation. Next, it relies on CAAN, a self-attention layer, to capture the inter-relations among stocks. We implement LSTM with basic CAAN, and change the forecast into returns, instead of winning scores.

Vector Autoregression. We include a standard linear vector autoregression (VAR) [118]. VAR is a typical stock forecast baseline. Formally, it assumes $\mathbf{r}_{t+1} = f(\mathbf{x}_t)+\xi_t$, where $\mathbf{x}_t$ denotes the features of all stocks, and $\mathbf{r}_{t+1} \triangleq (r_{t+1,1}, \ldots, r_{t+1,n})$ denotes the future return of all stocks in the universe.

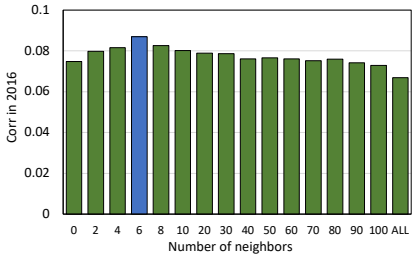
ARRR [161]. ARRR is a new regularization technique designed to address the overfitting issue in vector autoregression under the high-dimensional setting. Stock prediction is one of its applications. Specifically, ARRR involves two SVD steps; the first SVD is for estimating the precision matrix of the features, and the second SVD is for solving the matrix denoising problem.

4.6 Performance and Discussion

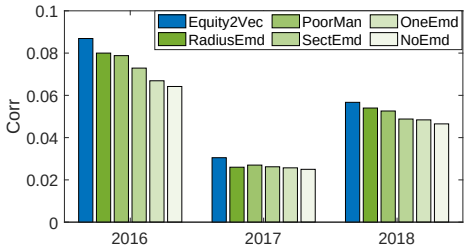
In this section, we discuss our overall performance, analyze the effectiveness of k-nearest neighbors, the learned stock embedding, and market trading simulation results.

Overall Performance on Correlation and t-statistic. Figure 4.4(a) and Figure 4.4(b) report the comprehensive analysis on all compared methods, for each testing year in terms of both correlation and t-statistic. The results confirm that our EQUITY2VEC method consistently outperforms all other baselines, for each testing year and across all metrics.

Impact of Different Number of Neighbors. We investigate the number of neighbors k against the correlation metric. Figure 4.5(a) shows that with the increase of k, the out-of-sample correlation first increases and then decreases. This indicates the performance



(a) *Varying k in testing year 2016.*



(b) *Different embedding methods.*

Figure 4.5: The effects of using different number of neighbors and effects of learned stock representation.

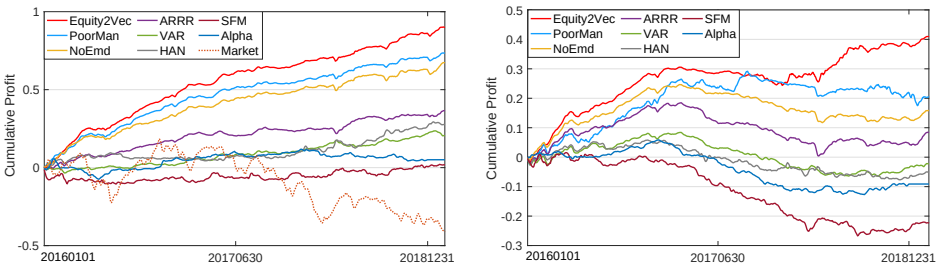
gains from our choices on k-nearest neighbors graph in Section 4.3.2.

Effect of Learned Stock Representation. To demonstrate the effects of learned stock representations for stock prediction, we investigate the performance of Equity2Vec by replacing the learned stock representations with the following representations:

- NoEmd: Remove the Equity2Vec module.
- PoorMan: Remove the influence from neighbors.
- SectEmd: Replace the embedding with the aggregated embedding from the same sector.
- OneEmd: Replace the embedding with the embedding from all the neighbors.
- RadiusEmd: Instead of using k-nearest neighbors, we also used the radius to select neighbors and the radius is a hyper-parameter.

As shown in Figure. 4.5(b), Equity2Vec achieves better performance than the above five baselines. We have the following observations: (1) Comparing with NoEmd proves the effectiveness of the information aggregated through stock embedding. (2) Comparing with PoorMan and OneEmd enables us to confirm the efficacy of learning evolving relations from k-nearest neighbors. (3) Comparing with SectEmd shows that Equity2Vec not just capture merely the sector information. (4) The RadiusEmbd is our variation and also achieved competitive results.

Market Trading Simulation. To further evaluate our method's effectiveness, we conduct a back-testing by simulating the stock trading for three out-of-sample years.



(a) *Long-short PnL.* (b) *Long-only PnL.*

Figure 4.6: The cumulative PnL (Profit and Loss) curves of the top quintile portfolio. For example, on any given day, we consider a portfolio with only the top 20% strongest predictions in magnitude, against future *market excess returns*. We simulate the investment on both (a) *Long-short portfolio* and (b) *Long-index* portfolio.

We simulate investments on our signals in two ways:(i) *Long-short* portfolio. (ii) *Long-index* portfolio: Long-only minus the market index. We conduct the trading in the daily granularity and select the stocks from the top 20% strongest forecast signals. The position of each stock is proportional to the signal (i.e., the dollar position of i-th stock is proportional to our forecast $\hat{r}_i^t$). The holding period is 5 days. [3]By allowing short-selling, we can execute on negative forecasts to understand the overall forecasting quality. Figure. 4.6 shows the cumulative PnL for our approach and baselines. We can see that our signals are consistently better than other baselines in both *Long-short* and *Long-index* portfolios, suggesting that our method generates stronger and more robust signals for trading.

4.7 Interpretation Analysis

In this section, we assess the interpretability for stock relationships, temporal weights in LSTM, and news as follows.

Visualizing Learned Stock Embedding. As depicted in Figure 4.7, we use t-SNE [105] on our final stock representation to assess the interpretation of the universe of stocks. Each dot represents a stock, and the color denotes the largest sector from Barra [42] associated with the given stock, while the text annotations represent the detailed stock categories. It shows that our EQUITY2VEC learns the interpretable stock representations that are well aligned with the Barra sectors.

[3]Short is implementable in the Chinese market only under particular circumstances, e.g., through brokers in Hong Kong under special arrangements.

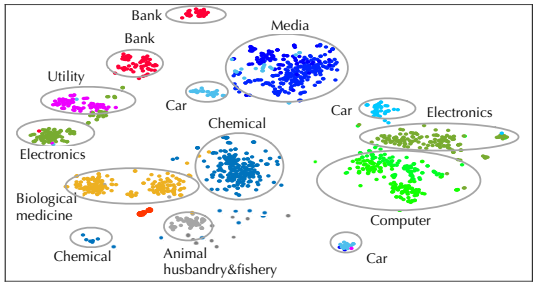


Figure 4.7: t-SNE of final stock representations (colors code industry sectors).

News Interpretation. To understand the news predictive ability, we track back the news from two groups in the test dataset. We focus the samples within the smallest 5% and the samples within the largest 5% errors based on Equation 4.12. We extract the corresponding news and show the detailed results in Figure 4.8. We focus on the demonstrative news from these two groups of samples. One can see that the news in the high accuracy group contains significant events with predictive ability, while the news in the low accuracy group mainly has no apparent influence on the stock forecast.

粤传媒: 2017年度资产盈利预测 (Annual Earning projection/ forecast) 合力泰: 增持公司股份公告 (Increase company holding) 雅致股份: 净收入与净利润增长分析 (Profit express and analysis) 天马股份:重大事项停盘 (Important event and trade suspension) 东方明珠:董事即将回购部分股权 (Buyback equity /share repurchase) [High accurate: important events]
经济日报: 18年我国经济结构持续优化 (Good domestic economic structure) 中投研究: 2018全球宏观前景怎么看 (Global Economic Analysis) 航发科技: 公司举行迎新年升旗仪式 (New Year flag-raising ceremony) [Low accurate: not predictive news]

Figure 4.8: The demonstrative news from high accuracy and low accuracy performance.

Temporal Attention Explanation. Next, we show the overall attention weights from the testing data set in Table 4.2 when we use the past 5 days' historical data. The hyperparameter 5 is set by the performance in the validation dataset. The recent days have larger weights indicating the recent days play more significant roles in the prediction.

	-5day	-4day	-3day	-2day	-1day
Weights	0.0055	0.0265	0.1662	0.3064	0.4954

Table 4.2: The temporal overall attention weights.

4.8 Related work

Predicting equity returns (i.e., empirical asset pricing) is an extensively studied academic disciplinary that can date back to the beginning of the 20th century [22, 60], so it is impossible to provide a comprehensive review here. Instead, we focus on recent work on using machine learning to forecast asset prices.

Linear Model. Empirical asset pricing (e.g., estimating the "true price" or forecasting the future price) is an important area in Finance [12]. Linear regression has been a dominating methodology to forecast equity returns, especially for intraday tradings [69, 21, 86, 46]. Because these linear models usually use a large number of features, regularizations are usually needed [73, 81, 118, 161]. For example, recently researchers examined regularization for "ultra-high dimensional" setting, in which the number of features could be significantly larger than the number of observations [161].

Deep Learning. There are two major approaches to forecast equity returns. *Approach 1. ANN as a blackbox for standard "factors."* First, "factors" that are known to be correlated with returns are constructed. These factors can be viewed as features constructed by financial experts. Second, the factors are fed into standard ANN black boxes so that non-linear models are learned (see e.g., [70, 62] and references therein). Little effort is made to optimize ANN's architecture or algorithm. *Approach 2. Forecasting the price time-series.* This approach views the price, trading volume, and other statistics representing trading activities as time series and designs specialized deep learning models to extract signals from the time series. Little feature engineering is done for these models. See e.g., [51, 132, 40, 95, 138, 93]. Approach 1 represents the line of thought that feature engineering is critical in building machine learning models, whereas Approach 2 represents the mindset that deep learning can automatically extract features so effort on feature engineering should be avoided. The co-movement effect is often ignored by the previous studies. Only a few works on stock predictions have explored this effect [26, 155, 49]. However, [26] relies on non-public dataset and learns the relations in a static way. [49]

consider only consider the relation to a particular type (sector and supply chain) and ignore the other relations, such as stocks affected by the same event. [155] has unsatisfying performance (even in their own reports on experiments) and only consider the co-movements of historical prices.

News. NLP-based techniques are developed to correlate news with the movement of stock prices. Earlier works use matrix factorization approaches (see e.g., [111, 145]) whereas more recent approaches use deep learning methods [39, 72, 31, 5]. These methods exclusively use the news to predict equity returns, and they do not consider any other "factors" that can impact the stock prices.

Factor Model (Cross-sectional Returns). The movement of two or more stocks usually can be explained by a small subset of factors. For example, Facebook and Google often co-move because their return can be explained by the technical factors. The so-called "factor model" (e.g., [44, 142, 167, 161]) can effectively capture the co-movement of prices but these methods usually rely on PCA/SVD techniques and are not computationally scalable.

Comparison. *1. Comparing to existing linear models.* Our method is more effective at extract non-linear signals. *2. Comparing to existing DL models.* We find that we need both careful feature engineering and optimizing DL techniques to use technical factors in the most effective manner, moreover, we do not restrict the stock relation into a particular type and learn the evolving relations. *3. Comparing with News/NLP-based techniques.* We do not exclusively rely on the news. Instead, we explicitly model the interaction between news and other factors so that our model avoids low quality signals (e.g., news-based signals could be essentially trading momentum), and *4. Comparing with factor models.* A key innovation of our model is the introduction of Equity2Vec component. This component models the interaction between stocks and circumvents SVD computation on large matrices.

4.9 Conclusion and future work

This paper presents a novel approach to answer two research questions. *(i)* How can we interpret the relationship between stocks? *(ii)* How can we leverage heterogeneous data sources to extract high-quality forecasting power? Through extensive evaluation against the state-of-art baselines, we confirm that our method achieves superior performance. Meanwhile, the results from different trading simulators demonstrate that we can effectively monetize the signals. In addition, we interpret the stock relationships highlighting they align well with the sectors defined by commercial risk models, extract important technical factors, and explain what kind of news has more predictive power.

We identify several potential future directions. First, it is worth exploring more effective features from social media such as financial discussion forums. As individual investors often engage in insightful discussions on finance topics and stock movements, the large volume of such discussions could indicate potential upcoming major events. Second, the proposed Equity2Vec can generalized to other problems, such as mining the relations between futures.

Chapter 5

Conclusions and Future Work

5.1 Summary of Contributions

Predicting the assets prices or return is of fundamental importance to the financial technology community as the successful prediction of assets' future price could yield significant profit [71, 157, 103]. This thesis investigates using machine learning techniques to simultaneously forecast the future return for a large number of stocks traded in a region. For example, in the US market, we generally build models to predict the next 5-day returns for the S&P 500 or the Russell 3000.

We aim to tackle three key challenges that are not properly addressed in prior works [155, 72, 167, 92, 20, 25, 84, 118, 73]:

C1. High-dimensional interactions between assets. The current state of one asset could potentially impact the future state of another. For example, Amazon's disclosure of its revenue change in cloud services could indicate that revenues also could change in other cloud providers. The number of possible interactions is excessively large and can be even significantly larger than the number of observations. For example, in a portfolio of 3,000 stocks, the total number of potential links between pairs of stocks is $3,000 \times 3,000 \approx 10^7$, whereas we typically have 10 years of daily data with only 2,500 data points [124, 62, 81, 28]. This setting is also referred to as the high dimensional setting and is prone to have severe overfitting issues. It requires careful analysis of a models theoretical

properties before fitting the data.

C2. Non-linearity of the hypothesis class. Linear models are usually insufficient to characterize the relationship between the response/label and the available information (features), so techniques beyond simple linear regressions are heavily needed.

C3. Data scarcity for training individual asset model. While we usually have a large volume of data, the size of the data associated with an individual asset could be small and is insufficient for properly train the model for the individual asset. For example, a typical daily forecasting model based on technical factors uses 10 years (approx. 2500 trading days) of data. We collect one data point for each day so only 2500 observations are available for each asset. We develop the following works to address these challenges.

1. Adaptive reduced rank regression (addressing C1) Adaptive-RRR studies the high-dimensional regression problem $\mathbf{y} = M\mathbf{x} + \epsilon$ with a low signal-to-noise ratio which is known to suffer from severe overfitting. First, we analyze reduced rank regression and its overfitting problem. Second, we propose adaptive reduced rank regression with better generalization guarantees. Our method leverages the spectral properties of $\mathbf{x}$ and can adapt the model to signal quality. Third, we prove the optimality of our algorithm. Additionally, our approach either outperforms or is competitive with existing baselines in the synthetic experiments and achieves the best performance in real datasets (forecasting equity returns and predicting users' popularity).

2. On embedding stocks (addressing C1 & C2) To address the overffiting issues and design algorithms effective in extracting non-linear signals, we proposed on embedding stocks, and we highlight the three main contributions of our work. First, we propose the additive influence model for equity returns that enables us to orchestrate mathematically rigorous high-dim techniques with practically effective machine learning algorithms. Our model assumes that each stock i is associated with a vector representation $\mathbf{z}_i$ in a (latent) Euclidean space, and characterizes the interactions between stocks in the form of $\mathbf{y}_{t,i} = \sum_{j\in[d]} \kappa(\mathbf{z}_i, \mathbf{z}_j) g(\mathbf{x}_{j,t}) + \xi_{t,i}$. Our proposed model allows for feature interactions through

$g(\cdot)$, and addresses the overfitting problem arising from stock interactions. Second, to learn the $\mathbf{z}_i$'s, we design a simple algorithm that uses low-rank approximation of $\mathbf{y}_t$'s covariance matrix to find the closeness of the stocks, and develop a novel theoretical analysis based on recent techniques from high-dim and kernel learning [15, 147, 161].

Third, to learn $g(\cdot)$, we generalize major machine learning techniques, including neural nets, non-parametric, and boosting methods, to the additive influence model when estimates of $\mathbf{z}_i$'s are known. We specifically develop a moment-based algorithm for non-parametric learning of $g(\cdot)$, and a computationally efficient boosting algorithm based on linear learners by using the domain knowledge of equity data sets.

Equity2Vec (addressing C2 & C3). We develop a specialized neural net model for each asset (e.g., train $g_i(\cdot)$ for asset i) but there is insufficient data to properly train g_i with data only from i (because of C3). Our idea is to shrink $g_i(\cdot)$'s toward one or more centroids to reduce model (sample) complexities. Specifically, we train a neural net model $g(\mathbf{x}_i, W, W_i)$, where W is shared across all entities, W_i is entity-specific and is learned through embedding, and $g_i(\mathbf{x}_i) = g(\mathbf{x}_i, W, W_i)$. When entities i and j are close, then W_i and W_j are close. Consequently, g_i and g_j will be similar when entity i and entity j are similar.

Based on our observation, we propose an end-to-end deep learning framework to price the assets. Our framework possesses two main properties: 1) We propose Equity2Vec, a graph-based component that effectively captures both long-term and evolving cross-sectional interactions. 2) The framework simultaneously leverages all the available heterogeneous alpha sources including technical indicators, financial news signals, and cross-sectional signals.

5.2 Future Research Direction

Our future work targets to relax the theoretical guarantee requirement. We aim to find a set of principles under which the deep architectures do not suffer from overfitting problems. Our research plan is motivated by the following investigations:

Principles of using deep learning for high-dim problems. Although previous efforts [167, 72, 92, 155] to build deep learning-based cross-asset models were unsuccessful, deep learning techniques were effective in solving high-dim problems in both recommender system [154, 30, 168] and graph-learning [61, 156, 164]. In our future work, we aim to answer the research question: how can we use and generalize deep learning techniques for recommender systems and graphs to produce stock embedding?

Bibliography

[1] Hasan Abasi, Nader H Bshouty, and Hanna Mazzawi. On exact learning monotone dnf from membership queries. In *International Conference on Algorithmic Learning Theory*, 2014.

[2] Ittai Abraham, Shiri Chechik, David Kempe, and Aleksandrs Slivkins. Low-distortion inference of latent similarities from a multiplex social network. *SIAM Journal on Computing*, 44(3):617–668, 2015.

[3] Anish Agarwal, Devavrat Shah, Dennis Shen, and Dogyoon Song. On robustness of principal component regression. In *NeurIPS*, 2019.

[4] Gernot Akemann, Jonit Fischmann, and Pierpaolo Vivo. Universal correlations and power-law tails in financial covariance matrices. *Physica A: Statistical Mechanics and its Applications*, 2010.

[5] Ryo Akita, Akira Yoshihara, Takashi Matsubara, and Kuniaki Uehara. Deep learning for stock prediction using numerical and textual information. In *ICIS*, 2016.

[6] Yakov Amihud. Illiquidity and stock returns: cross-section and time-series effects. *Journal of financial markets*, 2002.

[7] Theodore Wilbur Anderson et al. Estimating linear restrictions on regression coefficients for multivariate normal distributions. *The Annals of Mathematical Statistics*, 22(3):327–351, 1951.

[8] Andrew Ang. *Asset management: A systematic approach to factor investing*. 2014.

[9] Sanjeev Arora, Aditya Bhaskara, et al. More algorithms for provable dictionary learning. *arXiv preprint*, 2014.

[10] Douglas Azevedo and Valdir A Menegatto. Eigenvalues of dot-product kernels on the sphere. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, 2015.

[11] Zhidong Bai and Jack W Silverstein. *Spectral analysis of large dimensional random matrices*, volume 20. 2010.

[12] Turan G Bali, Robert F Engle, and Scott Murray. *Empirical asset pricing: The cross section of stock returns*. John Wiley & Sons, 2016.

[13] David Bamman, Jacob Eisenstein, and Tyler Schnoebelen. Gender identity and lexical variation in social media. *Journal of Sociolinguistics*, 2014.

[14] Carolina Batis, Michelle A Mendez, Penny Gordon-Larsen, Daniela Sotres-Alvarez, Linda Adair, and Barry Popkin. Using both principal component analysis and reduced rank regression to study dietary patterns and diabetes in chinese adults. *Public health nutrition*, 2016.

[15] Mikhail Belkin. Approximation beats concentration? An approximation view on inference with smooth radial kernels. In *COLT*, 2018.

[16] Jennifer Bender, Remy Briand, Dimitris Melas, and Raman Aylur Subramanian. Foundations of factor investing. *Available at SSRN 2543990*, 2013.

[17] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[18] Srinadh Bhojanapalli, Anastasios Kyrillidis, and Sujay Sanghavi. Dropping convexity for faster semi-definite optimization. In *COLT*, 2016.

[19] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986.

[20] Florentina Bunea, Yiyuan She, and Marten H Wegkamp. Optimal selection of reduced rank estimators of high-dimensional matrices. *ANN STAT*, 2011.

[21] Nusret Cakici, Kalok Chan, and Kudret Topyan. Cross-sectional stock return predictability in china. *The European Journal of Finance*, 2017.

[22] John Y Campbell, John J Champbell, John W Campbell, Andrew W Lo, Andrew W Lo, and A Craig MacKinlay. *The econometrics of financial markets*. princeton University press, 1997.

[23] Emmanuel J Candes et al. The restricted isometry property and its implications for compressed sensing. *Comptes rendus mathematique*, 2008.

[24] LJ Cao, Kok Seng Chua, WK Chong, HP Lee, and QM Gu. A comparison of pca, kpca and ica for dimensionality reduction in support vector machine. *Neurocomputing*, 2003.

[25] Gavin C Cawley and Nicola LC Talbot. Reduced rank kernel ridge regression. *Neural Processing Letters*, 2002.

[26] Chi Chen, Li Zhao, Jiang Bian, Chunxiao Xing, and Tie-Yan Liu. Investment behaviors can tell what inside: Exploring stock intrinsic properties for stock trend prediction. In *KDD*, 2019.

[27] Kun Chen, Hongbo Dong, and Kung-Sik Chan. Reduced rank regression via adaptive nuclear norm penalization. *Biometrika*, 2013.

[28] Luyang Chen, Markus Pelger, and Jason Zhu. Deep learning in asset pricing. *Available at SSRN 3350138*, 2019.

[29] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, 2016.

[30] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, and Mustafa Ispir. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[31] Raymond Chiong, Zongwen Fan, Zhongyi Hu, Marc TP Adam, Bernhard Lutz, and Dirk Neumann. A sentiment analysis-based machine learning approach for financial market prediction via news disclosures. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 278–279, 2018.

[32] Fan Chung and Linyuan Lu. Concentration inequalities and martingale inequalities: a survey. *Internet Mathematics*, 3(1):79–127, 2006.

[33] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 2009.

[34] Robert W Colby and Thomas A Meyers. *The encyclopedia of technical market indicators*. Dow Jones-Irwin Homewood, IL, 1988.

[35] Chen Dan, Kristoffer Arnsfelt Hansen, He Jiang, Liwei Wang, and Yuchen Zhou. Low rank approximation of binary matrices: Column subset selection and generalizations. *arXiv preprint arXiv:1511.01699*, 2015.

[36] Somnath Das, Carolyn B Levine, and Konduru Sivaramakrishnan. Earnings predictability and bias in analysts' earnings forecasts. *Accounting Review*, 1998.

[37] Chandler Davis and William Morton Kahan. The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 1970.

[38] Marcos Lopez De Prado. *Advances in financial machine learning*. John Wiley & Sons, 2018.

[39] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *IJCAI*, 2015.

[40] Jonathan Doering, Michael Fairbank, and Sheri Markose. Convolutional neural networks applied to high-frequency market microstructure forecasting. In *CEEC*.

[41] David Donoho, Matan Gavish, et al. Minimax risk of matrix denoising by singular value thresholding. *ANN STAT*, 2014.

[42] Frank J Fabozzi and Harry M Markowitz. *The theory and practice of investment management: Asset Allocation, Valuation, Portfolio Construction, and Strategies*, volume 198. John Wiley & Sons, 2011.

[43] Eugene F Fama et al. Multifactor explanations of asset pricing anomalies. *J.Finance*, 1996.

[44] Eugene F Fama and Kenneth R French. The Cross-Section of Expected Stock Returns. *Journal of Finance*, 1992.

[45] Eugene F Fama and Kenneth R French. Common risk factors in the returns on stocks and bonds. *JFE*, 1993.

[46] Eugene F Fama and Kenneth R French. Dissecting anomalies with a five-factor model. *The Review of Financial Studies*, 29(1):69–103, 2016.

[47] Fan Fan, Yong Ma, Chang Li, Xiaoguang Mei, Jun Huang, and Jiayi Ma. Hyperspectral image denoising with superpixel segmentation and low-rank representation. *Information Sciences*, 2017.

[48] Jianqing Fan, Yuan Liao, and Han Liu. An overview of the estimation of large covariance and precision matrices. *The Econometrics Journal*, 2016.

[49] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. Temporal relational ranking for stock prediction. *TOIS*, 2019.

[50] Guanhao Feng, Stefano Giglio, and Dacheng Xiu. Taming the factor zoo: A test of new factors. *The Journal of Finance*, 2020.

[51] Guanhao Feng, Nicholas G Polson, and Jianeng Xu. Deep learning in asset pricing. *arXiv preprint*, 2018.

[52] Laura Frank, Franziska Jannasch, Janine Kröger, George Bedu-Addo, Frank Mockenhaupt, Matthias Schulze, and Ina Danquah. A dietary pattern derived by reduced rank regression is associated with type 2 diabetes in an urban ghanaian population. *Nutrients*, 2015.

[53] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. 2001.

[54] Tak-Chung Fu, Chi-Pang Chung, and Fu-Lai Chung. Adopting genetic algorithms for technical analysis and portfolio management. *Computers & Mathematics with Applications*, 66(10):1743–1757, 2013.

[55] Matan Gavish and David L Donoho. The optimal hard threshold for singular values is 4/sqrt(3). *IEEE Transactions on Information Theory*, 2014.

[56] Rong Ge, Chi Jin, et al. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. In *ICML*, 2017.

[57] Ramazan Gencay and Thanasis Stengos. Moving average rules, volume and the predictability of security returns with feedforward networks. *Journal of Forecasting*, 1998.

[58] Ali Ghodsi. Dimensionality reduction a short tutorial. *Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada*, 2006.

[59] Mustafa Göçken, Mehmet Özçalici, Asli Boru, and Ayşe Tuğba Dosdoğru. Integrating metaheuristics and artificial neural networks for improved stock price prediction. *Expert Systems with Applications*, 44:320–331, 2016.

[60] Richard C Grinold and Ronald N Kahn. Active portfolio management. 2000.

[61] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[62] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. Technical report, 2018.

[63] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 2020.

[64] Shihao Gu, Bryan T Kelly, and Dacheng Xiu. Autoencoder asset pricing models. *Available at SSRN*, 2019.

[65] Chung-Wei Ha. Eigenvalues of differentiable positive definite kernels. *SIAM Journal on Mathematical Analysis*, 1986.

[66] AJS Hamilton and Max Tegmark. Decorrelating the power spectrum of galaxies. *Monthly Notices of the Royal Astronomical Society*, 2000.

[67] Fang Han, Huanran Lu, and Han Liu. A direct estimation of high dimensional stationary vector autoregressions. *The Journal of Machine Learning Research*, 16(1):3115–3150, 2015.

[68] Yufeng Han, Ai He, David Rapach, and Guofu Zhou. What firm characteristics drive us stock returns? *Available at SSRN 3185335*, 2018.

[69] Campbell R Harvey, Yan Liu, and Heqing Zhu. and the cross-section of expected returns. *The Review of Financial Studies*, 29(1):5–68, 2016.

[70] JB Heaton, NG Polson, and Jan Hendrik Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 2017.

[71] Michael G Hertzel, Zhi Li, Micah S Officer, and Kimberly J Rodgers. Inter-firm linkages and the wealth effects of financial distress along the supply chain. *Journal of Financial Economics*, 87(2):374–387, 2008.

[72] Ziniu Hu, Weiqing Liu, et al. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *WSDM*, 2018.

[73] Dashan Huang, Jiaen Li, and Guofu Zhou. Shrinking factor dimension: A reduced-rank approach. *Available at SSRN 3205697*, 2019.

[74] HE Hurst, RP Black, and YM Simaika. Long-term storage: an experimental study constable. *London UK*, 1965.

[75] Mohamed Assem Ibrahim, Hongyuan Liu, Onur Kayiran, and Adwait Jog. Analyzing and leveraging remote-core bandwidth for enhanced performance in gpus. In *2019 28th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 258–271. IEEE, 2019.

[76] Alan Julian Izenman. Reduced-rank regression for the multivariate linear model. *Journal of multivariate analysis*, 5(2):248–264, 1975.

[77] Narasimhan Jegadeesh and Sheridan Titman. Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of finance*, 1993.

[78] Zura Kakushadze. 101 formulaic alphas. *Wilmott*, 2016.

[79] Tosio Kato. Variation of discrete spectra. *Communications in Mathematical Physics*, 1987.

[80] Guolin Ke, Zhenhui Xu, Jia Zhang, Jiang Bian, and Tie-Yan Liu. Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks. In *KDD*, 2019.

[81] Bryan T Kelly, Seth Pruitt, et al. Characteristics are covariances: A unified model of risk and return. *JFE*, 2019.

[82] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[83] Charles D Kirkpatrick II and Julie A Dahlquist. *Technical analysis: the complete resource for financial market technicians*. FT press, 2010.

[84] Vladimir Koltchinskii, Karim Lounici, et al. Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion. *ANN STAT*, 2011.

[85] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[86] Serhiy Kozak, Stefan Nagel, and Shrihari Santosh. Shrinking the cross-section. *Journal of Financial Economics*, 135(2):271–292, 2020.

[87] MANJUNATH Krishnapur. Anti-concentration inequalities. *Lecture notes*, 2016.

[88] Olivier Ledoit and Michael Wolf. Honey, i shrunk the sample covariance matrix. *The Journal of Portfolio Management*, 2004.

[89] Wassily Leontief and Andras Brody. Money-flow computations. *Economic Systems Research*, 1993.

[90] Cheng Li, Felix Wong, Zhenming Liu, and Varun Kanade. From which world is your graph. In *NeurIPS*, 2017.

[91] Lili Li, Shan Leng, Jun Yang, and Mei Yu. Stock market autoregressive dynamics: A multinational comparative study with quantile regression. *Mathematical Problems in Engineering*, 2016, 2016.

[92] Zhige Li, Derek Yang, Li Zhao, Jiang Bian, Tao Qin, and Tie-Yan Liu. Individualized indicator for all: Stock-wise technical indicator optimization with stock embedding. In *KDD*, 2019.

[93] Bryan Lim, Stefan Zohren, and Stephen Roberts. Enhancing time-series momentum strategies using deep neural networks. *The Journal of Financial Data Science*, 1(4):19–38, 2019.

[94] Huanxin Lin, Cho-Li Wang, and Hongyuan Liu. On-gpu thread-data remapping for branch divergence reduction. *ACM Transactions on Architecture and Code Optimization (TACO)*, 15(3):1–24, 2018.

[95] Tao Lin, Tian Guo, and Karl Aberer. Hybrid neural networks for learning the trend in time series. In *IJCAI*, 2017.

[96] Ao Liu, Qiong Wu, Zhenming Liu, and Lirong Xia. Near-neighbor methods in random preference completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[97] Han Liu, Lie Wang, and Tuo Zhao. Multivariate regression with calibration. In *NeurIPS*, 2014.

[98] Hongyuan Liu, Mohamed Ibrahim, Onur Kayiran, Sreepathi Pai, and Adwait Jog. Architectural support for efficient large-scale automata processing.

In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 908–920. IEEE, 2018.

[99] Hongyuan Liu, King Tin Lam, Huanxin Lin, Cho-Li Wang, and Junchao Ma. Lightweight dependency checking for parallelizing loops with non-deterministic dependency on gpu. In *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pages 884–893. IEEE, 2016.

[100] Hongyuan Liu, Bogdan Nicolae, Sheng Di, Franck Cappello, and Adwait Jog. Accelerating DNN Architecture Search at Scale Using Selective Weight Transfer. In *Proceedings of the IEEE International Conference on Cluster Computing*, 2021.

[101] Hongyuan Liu, Sreepathi Pai, and Adwait Jog. Why gpus are slow at executing nfas and how to make them faster. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 251–265, 2020.

[102] Jianan Liu, Robert F Stambaugh, and Yu Yuan. Size and value in china. *JFE*, 2019.

[103] Guanyi Lu and Guangzhi Shang. Impact of supply base structural complexity on financial performance: Roles of visible and not-so-visible characteristics. *Journal of Operations Management*, 53:23–44, 2017.

[104] Zongming Ma and Tingni Sun. Adaptive sparse reduced-rank regression. *arXiv*, 2014.

[105] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[106] David JC MacKay. *Information theory, inference and learning algorithms*. 2003.

[107] Marcelo C Medeiros and Eduardo F Mendes. Estimating high-dimensional time series models. Technical report, Texto para discussão, 2012.

[108] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint*, 2013.

[109] Tomas Mikolov, Kai Chen, Gregory S Corrado, and Jeffrey A Dean. Computing numeric representations of words in a high-dimensional space, May 19 2015. US Patent.

[110] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[111] Felix Ming, Fai Wong, Zhenming Liu, and Mung Chiang. Stock market prediction from wsj: text mining via sparse matrix factorization. In *ICDM*, 2014.

[112] Marc-Andre Mittermayer and Gerhard F Knolmayer. Newscats: A news categorization and trading system. In *ICDM'06*, pages 1002–1007. Ieee, 2006.

[113] Tobias J Moskowitz, Yao Hua Ooi, and Lasse Heje Pedersen. Time series momentum. *Journal of financial economics*, 104(2):228–250, 2012.

[114] Ashin Mukherjee and Ji Zhu. Reduced rank ridge regression and its kernel extensions. *Statistical analysis and data mining: the ASA data science journal*, 2011.

[115] Howard Musoff and Paul Zarchan. *Fundamentals of Kalman filtering: a practical approach*. American Institute of Aeronautics and Astronautics, 2009.

[116] Christopher J Neely. Technical analysis in the foreign exchange market: a layman's guide. *Review-Federal Reserve Bank of St. Louis*, 79(5):23, 1997.

[117] Christopher J Neely and Paul A Weller. Technical analysis in the foreign exchange market. *Federal Reserve Bank of St. Louis Working Paper No*, 2011.

[118] Sahand Negahban and Martin J Wainwright. Estimation of (near) low-rank matrices with noise and high-dimensional scaling. *ANN STAT*, 2011.

[119] Whitney K Newey and Kenneth D West. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. Technical report, 1986.

[120] Ashadun Nobi, Seong Eun Maeng, Gyeong Gyun Ha, and Jae Woo Lee. Random matrix theory and cross-correlations in global financial indices and local stock market indices. *Journal of the Korean Physical Society*, 2013.

[121] Roberto Oliveira et al. Sums of random hermitian matrices and an inequality by rudelson. *Electronic Communications in Probability*, 2010.

[122] DJ Orr and Igor Mashtaler. Supplementary notes on the barra china equity model (cne5). 2012.

[123] Jigar Patel, Sahil Shah, Priyank Thakkar, and Ketan Kotecha. Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4):2162–2172, 2015.

[124] Christopher Polk, Samuel Thompson, and Tuomo Vuolteenaho. Cross-sectional forecasts of the equity premium. *JFE*, 2006.

[125] Richard A Posner. *Economic analysis of law*. 2014.

[126] Jean-Luc Prigent. *Portfolio optimization and performance analysis*. 2007.

[127] Mahesh Pritamani and Vijay Singal. Return predictability following large price changes and information releases. *Journal of Banking & Finance*, 2001.

[128] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1986.

[129] Mehdi Rahim, Bertrand Thirion, and Gaël Varoquaux. Multi-output predictions from neuroimaging: assessing reduced-rank linear models. In *2017 International Workshop on Pattern Recognition in Neuroimaging (PRNI)*, 2017.

[130] Anca Ralescu, Mojtaba Kohram, et al. Spectral regression with low-rank approximation for dynamic graph link prediction. *IEEE intelligent systems*, 2011.

[131] Riccardo Rastelli, Nial Friel, and Adrian E Raftery. Properties of latent variable network models. *Network Science*, 2016.

[132] Akhter Mohiuddin Rather, Arun Agarwal, and VN Sastry. Recurrent neural network and a hybrid model for prediction of stock returns. *EXPERT SYST APPL*, 2015.

[133] Pradeep Ravikumar, Martin J Wainwright, John D Lafferty, et al. High-dimensional ising model selection using l1-regularized logistic regression. *ANN STAT*, 2010.

[134] Mark Rudelson and Roman Vershynin. Non-asymptotic theory of random matrices: extreme singular values. In *Proceedings of the International Congress of Mathematicians (ICM)*, 2010.

[135] Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014.

[136] Robert E Schapire and Yoav Freund. Boosting: Foundations and algorithms. *Kybernetes*, 2013.

[137] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[138] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020.

[139] Jianfeng Si, Arjun Mukherjee, Bing Liu, Qing Li, Huayi Li, and Xiaotie Deng. Exploiting topic based twitter sentiment for stock prediction. In *ACL*, 2013.

[140] Shiladitya Sinha, Chris Dyer, Kevin Gimpel, and Noah A Smith. Predicting the nfl using twitter. *arXiv preprint arXiv:1310.6998*, 2013.

[141] Gilbert W Stewart. Matrix perturbation theory. 1990.

[142] James H Stock and Mark Watson. Dynamic factor models. *Oxford Handbooks Online*, 2011.

[143] James H Stock and Mark W Watson. Forecasting using principal components from a large number of predictors. *Journal of the American statistical association*, 2002.

[144] James H Stock and Mark W Watson. Implications of dynamic factor models for var analysis. Technical report, National Bureau of Economic Research, 2005.

[145] Andrew Sun, Michael Lachanski, and Frank J Fabozzi. Trade the tweet: Social media text mining and sparse matrix factorization for stock market prediction. *International Review of Financial Analysis*, 48:272–281, 2016.

[146] Daniel L Sussman, Minh Tang, and Carey E Priebe. Consistent latent position estimation and vertex classification for random dot product graphs. *TPAMI*, 2013.

[147] Minh Tang, Daniel L Sussman, Carey E Priebe, et al. Universally consistent vertex classification for latent positions graphs. *ANN STAT*, 2013.

[148] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1996.

[149] Jonathan L Ticknor. A bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications*, 2013.

[150] Alexandre B Tsybakov. *Introduction to nonparametric estimation*. 2008.

[151] Thomas Chinwe Urama, Patrick Oseloka Ezepue, and Chimezie Peters Nnanwa. Analysis of cross-correlations in emerging markets using random matrix theory. In *CMCGS*, 2017.

[152] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[153] Raja Velu and Gregory C Reinsel. *Multivariate reduced-rank regression: theory and applications*. 2013.

[154] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1235–1244, 2015.

[155] Jingyuan Wang, Yang Zhang, Ke Tang, Junjie Wu, and Zhang Xiong. Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks. In *KDD*, 2019.

[156] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, and Chao Ma. Deep graph library: Towards efficient and scalable deep learning on graphs. *arXiv preprint arXiv:1909.01315*, 2019.

[157] Jing Wu and John R Birge. Supply chain network structure and firm returns. *Available at SSRN 2385217*, 2014.

[158] Qiong Wu, Wen-Ling Hsu, Tan Xu, Zhenming Liu, George Ma, Guy Jacobson, and Shuai Zhao. Speaking with actions - learning customer journey behavior. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 279–286, 2019.

[159] Qiong Wu, Lucas CK Hui, Cheuk Yu Yeung, and Tat Wing Chim. Early car collision prediction in vanet. In *2015 International Conference on Connected Vehicles and Expo (ICCVE)*, pages 94–99. IEEE, 2015.

[160] Qiong Wu and Zhenming Liu. Rosella: A self-driving distributed scheduler for heterogeneous clusters. *International Conference on Mobility, Sensing and Networking (MSN)*, 2021.

[161] Qiong Wu, Felix M.F. Wong, Yanhua Li, Zhenming Liu, and Varun Kanade. Adaptive Reduced Rank Regression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[162] Qiong Wu, Zhang Zheng, et al. A deep learning framework for pricing financial instruments. *arXiv preprint*, 2019.

[163] Yun-Jhong Wu, Elizaveta Levina, and Ji Zhu. Generalized linear models with low rank effects for network data. *arXiv preprint arXiv:1705.06772*, 2017.

[164] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[165] Beat Wüthrich, D Permunetilleke, Steven Leung, W Lam, Vincent Cho, and J Zhang. Daily prediction of major stock indices from textual www data. *Hkie transactions*, 1998.

[166] Yumo Xu and Shay B Cohen. Stock movement prediction from tweets and historical prices. In *ACL*, 2018.

[167] Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. Stock price prediction via discovering multi-frequency trading patterns. In *KDD*, 2017.

[168] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[169] Xinfeng Zhou and Sameer Jain. *Active Equity Management*. 2014.

[170] Xiaofeng Zhu, Heung-Il Suk, Heng Huang, and Dinggang Shen. Low-rank graph-regularized structured sparse regression for identifying genetic biomarkers. *IEEE transactions on big data*, 2017.

VITA

Qiong Wu

Qiong Wu is a Ph.D. candidate in the Department of Computer Science at the William & Mary advised by Prof. Zhenming Liu. Her research on developing robust machine learning algorithms under harsh conditions, such as excessively large number of features and notoriously low signal-to-noise ratio. Her Ph.D. research has been published in TIST 2021, NeurIPS 2020, ICAIF 2020, AAAI 2019, and ICSC 2019. Her current and past efforts include collaborations with AT&T on customer care analysis, Instacart on conceptual graph construction, and The Alan Turing Institute on high-dimensional regularization methods and developing forecasting models for financial instruments. Before joining William & Mary, she received her B.Eng degree from the Dalian University of Technology in 2014 and an M.Sc degree from the University of Hong Kong in 2016.