Dimensionality Reduction for Supervised and Unsupervised Learning: New Algorithms, Analysis and Applications

Hexuan Liu

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2022

Reading Committee:
Aleksandr Aravkin, Chair
Anne Greenbaum, Chair
J. Nathan Kutz

Program Authorized to Offer Degree:
Applied Mathematics

University of Washington

Abstract

Dimensionality Reduction for Supervised and Unsupervised Learning: New Algorithms, Analysis and Applications

Hexuan Liu

Co-Chairs of the Supervisory Committee:
Professor Aleksandr Aravkin
Department of Applied Mathematics

Professor Anne Greenbaum
Department of Applied Mathematics

Dimensionality reduction is an essential topic in data science, particularly when data are high-dimensional or have more features than samples. The process of reducing the data dimension usually involves solving an eigenvalue problem. For example, the ubiquitously used principal component analysis obtains the principal subspace by solving a standard eigenvalue problem, and linear discriminant analysis obtains a discriminative subspace by solving a generalized eigenvalue problem. A vast array of real-world data problems can be framed mathematically as variants of eigenvalue problems, including eigenvalue problems with sparsity constraints and penalties, and nonlinear eigenvalue problems. In this thesis, I present new formulations of penalized and constrained eigenvalue problems for dimensionality reduction, propose new provably convergent algorithms to solve these formulations, and present real-world applications spanning many fields, including examples from both supervised and unsupervised learning.

This thesis is divided into three parts. In the first part, I reformulate the Wasserstein Discriminant Analysis (WDA) as a ratio trace problem and present an eigensolver-based algorithm to compute the discriminative subspace of WDA. The new formulation, along with the proposed algorithm, can serve as an efficient and more stable alternative to the original

trace ratio formulation and its gradient-based algorithm. A rigorous convergence analysis is provided for the proposed algorithm under the self-consistent field framework, which is crucial but missing in the literature. In the second part, I develop an efficient Truncated Orthogonal Iteration (TOrth) algorithm to simultaneously compute multiple leading eigenvectors with sparsity constraints. Convergence results are then established using a matrix perturbation framework. Numerical simulations show that TOrth and its variant require minimal tuning and are competitive with state-of-the-art algorithms for sparse principal components analysis. The proposed algorithms can be extended to perform unsupervised learning tasks such as K-subspace clustering. In the third part, I give examples of the methods applied to the neuronal network of C. elegans to obtain a graphical model, and to the Covid-19 death rate time series for clustering and forecasting.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Number Page

ACKNOWLEDGMENTS

I would like to first express my gratitude to my advisors, Anne Greenbaum and Aleksandr (Sasha) Aravkin, who have advised me and inspired me greatly during my graduate school research. Anne has guided me into the world of numerical analysis and always reminds me the standards for good research. Sasha has led me into the world of optimization and always encourages me with his optimistic and cheerful spirit. With the guidance and constant encouragement from my advisors, my journey in graduate school has been richly rewarding.

I would like to thank my other committee members, Nathan Kutz, Sam Burden and Zelda Zabinsky for taking time to attend my general and final exam, and for providing insights and feedback to my work. I am also indebted to many of my mentors, co-authors and colleagues, both inside and outside the department. I am thankful for Eli Shlizerman and Jimin Kim for getting me started in graduate school research and introducing me to the interesting worm C.elegans. I would like to thank Ping Li, Yunfeng Cai and Youlin Chen for a wonderful experience in industrial research, with whom I have collaborated during my internship at Baidu Research. I also want to thank Tyler Chen for collaboration and research chats. I am also grateful for Tom Trogdon, Laura Balzano and John Lipor for helpful discussions. Along with many others, they have given me broader perspectives and have shown me how mathematics can be applied to the real-world.

Last but not least, I am deeply grateful for my parents, Ming He and Huizhi Liu, for giving me unconditional support and love. They have provided me the opportunity to study abroad for ten years and never questioned my decision to pursue any career direction. I am also thankful for my boyfriend, Zhengqi Yang, who has always been there for me through the ups and downs. Without their support I would never be able to start or finish this degree.

DEDICATION

To my family

Chapter 1
INTRODUCTION

1.1 Overview

Many linear feature projection techniques involve solving an eigenvalue problem. It could be a standard eigenvalue problem, such as the principal component analysis, or a generalized eigenvalue problem, such as the linear discriminant analysis. The problem could also be framed as some variations of eigenvalue problems by adding sparsity constraints or structural penalties, or an eigenvector-dependent eigenvalue problem. The focus of our research is to present new formulations of penalized and constrained eigenvalue problems to achieve dimensionality reduction, and to propose new algorithms and improve existing algorithms that solve such eigenvalue problems with numerical convergence guarantees. We also provide real-world applications of the proposed methods that span many fields, from clustering text datasets to extracting coherent signals from brain imaging datasets. In this first chapter we give some definitions and motivations of dimensionality reduction. We then classify the broad topic into several categories and present some common techniques in each category.

1.2 Dimensionality Reduction: Feature Selection vs. Feature Projection

Given a dataset $X \in \mathbb{R}^{n \times p}$, where n is the number of samples and p is the number of features, dimensionality reduction aims to reduce the number of features p. When p is large, dimensionality reduction can help to avoid the curse of dimensionality [11] and to increase the generalization ability of models by reducing overfitting [47]. Dimensionality reduction can be achieved via feature selection or feature projection. Feature selection returns a subset of the original features and discards the features that are either irrelevant or redundant [53] and can be further categorized into three main categories: filters, wrappers and embedded

methods [25]. Filter methods use statistical measures such as the Pearson correlation and mutual information [53] to select important and independent features. Wrapper methods choose a predictor in advance and use the predictor performance as the objective function to evaluate a subset of features. Embedded methods incorporate feature selection in the training process, and examples of this approach are the LASSO [130] and the elastic-net regularization [154]. Feature projection transforms the original features using a function $f : \mathbb{R}^{n\times p} \to \mathbb{R}^{n\times d}$, where f can be either linear or nonlinear.

We mainly concern ourselves with linear feature projections, i.e. $f(X) = XU$ for some matrix $U \in \mathbb{R}^{p\times d}$. Adding sparsity penalty or constraints into feature projection incorporates feature selection simultaneously as done in the embedded methods. Linear feature projection techniques can be further categorized as supervised and unsupervised, and we explain their difference below.

1.3 Supervised Dimensionality Reduction

Supervised dimensionality reduction techniques require class label information y in addition to the data matrix X or the covariance matrix Σ. The linear discriminant analysis (LDA) is one of the most commonly-used supervised dimensionality reduction techniques. LDA looks for low-dimensional and discriminative subspace represented by an orthonormal matrix V by maximizing the between-class separation of data while minimizing the within-class scatter. More specifically, consider n data points $(x_1, \ldots, x_n)$ and data labels $(y_1, \ldots, y_n)$ such that each label y_i belongs to one of the classes. In LDA, the within-class and between-class matrices are

$$S_w = \frac{\sum_k \sum_{i:y_i=k}(x_i - \mu_k)(x_i - \mu_k)^\top}{n}, S_b = \frac{\sum_k n_k(\mu_k - m)(\mu_k - m)^\top}{n} \tag{1.1}$$

where n_k is the number of data in class k, $\mu_k = \frac{1}{n_k}\sum_{i\in C_k} x_i$ is the mean of k-th class and $m = \frac{1}{n}\sum_i x_i$. The criterion of LDA is to maximize the following objective function:

$$J(V) = \text{Trace}(V^\top S_b V(V^\top S_w V)^{-1}) \tag{1.2}$$

The solution V can be obtained by the generalized eigenvalue decomposition with respect to S_b, S_w:

$$S_b V = S_w V \Lambda. \tag{1.3}$$

1.4 Unsupervised Dimensionality Reduction

In practice, true label information is often lacking, and thus we turn to unsupervised techniques. By "unsupervised" we mean that we do not use the label information and only require the data matrix X or its covariance matrix Σ to perform dimensionality reduction. Common techniques include the principal component analysis (PCA), the nonnegative matrix factorization (NMF) [82,83], and the independent component analysis (ICA) [62]. PCA performs the Singular Value Decomposition (SVD) on the de-meaned data matrix $\hat{X}$, i.e. $\hat{X} = USV^*$. The columns of US are the principal components, and the columns of V are the loading vectors. V is also the matrix of eigenvectors of the covariance matrix $\hat{X}^T\hat{X}/n$. NMF decomposes a non-negative data matrix to the product of two non-negative matrices, i.e. $X \approx WH$ where all entries in $W \in \mathbb{R}^{n\times d}$ and $H \in \mathbb{R}^{d\times p}$ are non-negative. ICA assumes that the data are non-Gaussian and aims to find components that are statistically independent.

We illustrate the difference between PCA and LDA with a simple 2D example, as shown in figure 1.1. The two classes of data are generated using the multivariate Gaussian distribution, with mean $(-3, 0)$ and $(3, 0)$ respectively, and covariance matrix $\begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix}$ for both. We generate 500 samples in each class. We then apply PCA and LDA to obtain a 1D subspace. If we ignore the class label information and apply PCA to the dataset, it will find the direction that best explains the observed data variance, which is approximately the vector $(0, 1)$, i.e. the vertical line (the y-axis). If we project the two classes of data onto the vertical line, they will overlap and we will not able to distinguish one class from the other. On the other hand, if we apply LDA and utilize class label information, we will find the horizontal line (the x-axis) and projection onto this line maximally separates the two classes.

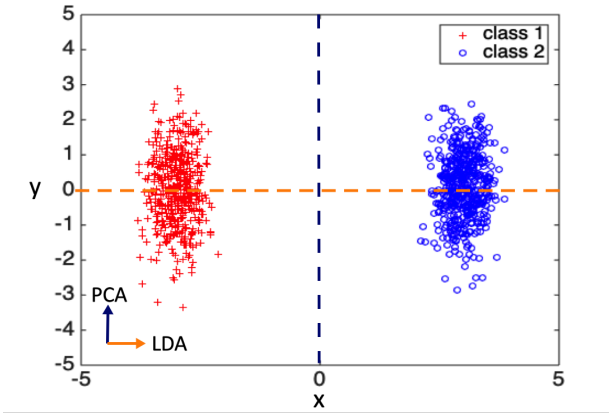


Figure 1.1: **Lines obtained by PCA and LDA on a 2D dataset.** We generate a simple 2D dataset consisting of two clusters using multivariate Gaussian distribution, and apply PCA and LDA to obtain a 1D subspace, which are the vertical and horizontal dotted lines. The vertical line is the first principal component obtained by PCA, which gives the direction of the maximum spread (variation) of the data. Taking the class labels into account, LDA finds a feature subspace (the horizontal line), and projecting data onto it maximally separates the data.

1.5 Organization of Thesis

In chapter 2, we study a recently proposed supervised dimensionality reduction algorithm, the Wasserstein Discriminant Analysis (WDA) [44]. WDA is a supervised linear dimensionality reduction technique that generalizes the classical Linear Discriminant Analysis (LDA). The original WDA adopts a trace ratio formulation and is solved by a gradient-based algorithm. We reformulate WDA as a ratio trace problem, which involves solving an eigenvector-dependent eigenvalue problem. We solve the problem using the Self-consistent field iteration [23], and show that it achieves better accuracy and efficiency compared to its original formulation. We also extend the supervised technique to unsupervised learning by subspace clustering. Numerical experiments on real datasets show promising results of the ratio trace formulation of WDA in both classification and clustering tasks.

In chapter 3 we turn to the most-widely used dimensionality reduction technique, the

principal components analysis (PCA) [104], and review some sparse variants of PCA. Sparse PCA (SPCA) was first proposed by [70, 155] to find sparse loading vectors. We first review some popular SPCA methods in the literature, including but not limited to the methods presented in [34,71,90,155]. We then present our proposed algorithm, the Truncated Orthogonal Iteration, and provide a convergence analysis with numerical studies. We also apply the algorithms to real-world applications such as the MNIST dataset, the sea surface temperature dataset, and the 20 newsgroup dataset.

In chapter 4 and 5 we focus on applications of the proposed algorithms. In chapter 4, we use sparse principal components of neuronal network time series to represent the network dynamics as a graphical model. Specifically, we use a model of Caenorhabditis elegans (C. elegans) somatic nervous system as an example of our approach. The resulting graphical model enables us to obtain and verify underlying neuronal pathways for known behavioral scenarios and detect possible pathways for novel scenarios. In chapter 5, we study a data-driven approach to combine existing forecasts for COVID-19 death rates. We apply dimensionality reduction techniques to the metafeatures extracted from the time series. By reducing the dimension of metafeatures and clustering similar time series in the lower-dimensional subspace, we obtain a better ensemble model that is more robust across time, location and forecast horizon compared to the individual models and the top ensemble models.

Chapter 2

WASSERSTEIN DISCRIMINANT ANALYSIS

2.1 Introduction

Wasserstein Discriminant Analysis (WDA) [44] is a supervised linear dimensionality reduction technique that generalizes the classical Fisher Discriminant Analysis (FDA) [47] using the optimal transport distances [136]. Many existing works [19, 42, 102, 142] have addressed the issue that FDA only considers global information. In particular, [153] proposed a new formula relaying on worst-case distance; [126] developed a localized version of FDA; [85] provided an adaptive method for learning local structure from data. The recently proposed WDA [44] has the advantage of adaptively capturing both local and global information, and shows competitive performance in classification tasks compared to other supervised dimensionality reduction techniques.

WDA as developed in [44] used the trace ratio formulation to maximize the ratio of the inter-class's regularized Wasserstein distances to the intra-class's regularized Wasserstein distances. Formally, they aimed to solve $\max_{\mathbf{P}} \text{Trace}(\mathbf{P}^T\mathbf{C}_b(\mathbf{T})\mathbf{P})/\text{Trace}(\mathbf{P}^T\mathbf{C}_w(\mathbf{T})\mathbf{P})$ where $\mathbf{C}_b$ and $\mathbf{C}_w$ are the inter-class and intra-class covariance matrices, respectively, and are functions of the optimal transport matrix $\mathbf{T}$. The optimal transport matrix $\mathbf{T}$ quantifies how important the distance between two samples should be in order to obtain a good projection matrix $\mathbf{P}$. The authors derived the gradient of the objective function with respect to $\mathbf{P}$ and also utilized automatic differentiation to compute the gradients. The difficulties of their approach are 1) the optimization objective is non-convex and non-smooth; and 2) $\mathbf{C}_b$ and $\mathbf{C}_w$ are functions of $\mathbf{T}$ and $\mathbf{T}$ is an implicit function on $\mathbf{P}$. Thus WDA is a bi-level optimization problem [30] and requires solving an optimal transport problem in every step of gradient descent. Due to these complications, theoretical guarantees on the convergence

are lacking. Vanilla gradient descent gets stuck easily in the non-smooth region, especially for real datasets, due to the natural structure of the data such as low rank or sparsity. In practice, the approach introduced in [44] can be sensitive to initialization and may take many iterations or even fail to reach convergence. All these issues raise concerns when WDA is applied to real data.

In this chapter, we circumvent the aforementioned challenges by reformulating WDA as a ratio trace problem, which has a closed-form solution and can be solved by the generalized eigenvalue decomposition if $\mathbf{T}$ is given. For algorithms of dimensionality reduction, it is common to use ratio trace formulation to approximate trace ratio problems [145]. For example, in Fisher Discriminant Analysis (FDA), these two formulations are both defined and are both served as criterion to maximize inter-class distance while minimizing intra-class distance [50]. Although there are many comparisons between these two formulations when the inter-class and intra-class covariance matrices are fixed [65, 99, 101, 141], they do not concern with the case of the covariance matrices being functions of the discriminative subspace as in WDA. We give numerical comparisons between these two formulations in terms of classification accuracy on simulated as well as real data, on which the proposed formulation either is comparable or outperforms the original formulation.

Specifically, we solve the ratio trace problem: $\arg\max_{\mathbf{P}} \text{Trace}(\mathbf{P}^T\mathbf{C}_b(\mathbf{T})\mathbf{P}(\mathbf{P}^T\mathbf{C}_w(\mathbf{T})\mathbf{P})^{-1})$ instead of the original WDA formulation: $\arg\max_{\mathbf{P}} \text{Trace}(\mathbf{P}^T\mathbf{C}_b(\mathbf{T})\mathbf{P})/\text{Trace}(\mathbf{P}^T\mathbf{C}_w(\mathbf{T})\mathbf{P})$. We propose an algorithm: **WDA-eig**, to solve the ratio trace problem using the self-consistent field iteration (SCF), and establish a convergence analysis for the general SCF framework with specific application to the WDA context. The SCF iteration was originally used for solving Kohn-Sham equation arising in electronic structure calculations [23]. Most works on SCF concern with the standard eigenvalue problem [22, 89, 146], while convergence analysis for the generalized eigenvalue problem has not appeared in current literature. Our numerical examples demonstrate that the algorithm based on SCF iteration usually converges within a few iterations in practice and is less sensitive to initialization compared to the original approach. We also give a convergence analysis under the SCF framework,

which not only provides convergence guarantee to the ratio trace WDA problem but is also applicable to other eigenvector-dependent generalized eigenvalue problems.

As an application, we extend **WDA-eig** to unsupervised clustering. Since WDA requires class labels to calculate the inter- and intra-class Wasserstein distances, a natural solution is to combine WDA with low-dimensional clustering techniques, which requires iteratively applying WDA given updated label information. The new algorithm has a fast convergence compared to the original approach and aid in iteratively applying WDA to find the most discriminative subspace. Several methods [37, 139, 147] that are closely related to our work leverage label information by combining FDA with Kmeans. Our numerical experiments show that the WDA-Kmeans has promising performance compared to existing subspace clustering techniques on real-world datasets.

Our contribution in this project is three-fold. First, we present a ratio trace formulation of the WDA problem. Second, we propose to solve the problem using the SCF iteration, and provide a convergence analysis for the SCF framework as well as specific application to the WDA context. Last but not least, we iteratively apply WDA and low-dimensional clustering technique to perform clustering. We emphasize that we do not attempt to solve the original trace ratio formulation of WDA with the proposed algorithm. A better solution to the original formulation is not the focus of this project.

Notations. We use $\|\cdot\|$ to denote the 2-norm of a matrix or vector. $\mathbf{I}_n$ is used to denote the identity matrix of order n. For any matrix $\mathbf{X}$, let x_i denote its ith column vector and $x_{i,j}$ denote the (i,j)th entry. For any $\mathbf{A}$, $\mathbf{B} \in \mathbb{R}^{m\times n}$, $\langle \mathbf{A}, \mathbf{B}\rangle$ is the inner product of $\mathbf{A}$ and $\mathbf{B}$, i.e., $\langle \mathbf{A}, \mathbf{B}\rangle = \text{trace}(\mathbf{A}^T\mathbf{B})$. Let $\mathbb{S}^n = \{A \in \mathbb{R}^{n\times n} | A = A^T\}$ be the set of symmetric matrices. For a symmetric matrix pair $(\mathbf{A}, \mathbf{B})$, $\mathbf{A}, \mathbf{B} \in \mathbb{S}^n$ with $\mathbf{B}$ being positive definite, we denote the generalized eigenvalues of $(\mathbf{A}, \mathbf{B})$ by $\lambda_{\min}(\mathbf{A}, \mathbf{B}) = \lambda_n(\mathbf{A}, \mathbf{B}) \leq \cdots \leq \lambda_1(\mathbf{A}, \mathbf{B}) = \lambda_{\max}(\mathbf{A}, \mathbf{B})$. Let $\mathbb{O}^{d\times p}$ represent the set of orthogonal $d \times p$ matrices, i.e., $\mathbb{O}^{d\times p} = \{\mathbf{A} \in \mathbb{R}^{d\times p} \mid \mathbf{A}^T\mathbf{A} = \mathbf{I}_p\}$.

2.2 Methodology

In this section, we first review the existing supervised WDA problem and its gradient-based solver, and reformulate the problem as a nonlinear generalized eigenvalue problem. We then present an algorithm that solves the problem using the self-consistent field iteration.

2.2.1 Background

Wasserstein distance (also known as the optimal transport distance, earth mover distance) is a distance between probability measures that preserves the underlying geometry of the space based on principles from the optimal transport theory [136]. The regularized Wasserstein distance is the solution of the following entropy-smoothed optimal transport problem:

$$\mathbf{T}_\lambda \triangleq \underset{\mathbf{T}\in\mathbb{U}_{nm}}{\arg\min}\, \lambda\langle \mathbf{T}, \mathbf{M}_{\mathbf{X},\mathbf{Z}}\rangle - \Omega(\mathbf{T}), \tag{2.1}$$

where $\lambda \geq 0$ is the Wasserstein regularization parameter, $\mathbf{M}_{\mathbf{X},\mathbf{Z}}$ denotes the pairwise squared Euclidean distance matrix between samples in $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $\mathbf{Z} \in \mathbb{R}^{n\times d}$: $\mathbf{M}_{\mathbf{X},\mathbf{Z}} \triangleq [\|x_i - z_j\|_2^2]$, and $\Omega(\mathbf{T})$ is the entropy of $\mathbf{T}$: $\Omega(\mathbf{T}) \triangleq -\sum_{ij} t_{ij}\log(t_{ij})$. $\mathbb{U}_{mn}$ is the polytope of $m \times n$ nonnegative matrices with row and column sums being equal to $\mathbf{1}_m/m$ and $\mathbf{1}_n/n$ respectively: $\mathbb{U}_{mn} \triangleq \{\mathbf{T} \in \mathbb{R}_+^{m\times n} \mid \mathbf{T}\mathbf{1}_n = \mathbf{1}_m/m, \mathbf{T}^T\mathbf{1}_m = \mathbf{1}_n/n\}$.

As the entropy-smoothed optimal transport problem is strictly convex, the solution to (2.1) exists and is unique. Numerically, $\mathbf{T}_\lambda$ can be obtained very efficiently using algorithms such as the Sinkhorn's fixed-point iterations [32, 74], the Greenkhorn algorithm [3, 4], or APDAMD [86]. The regularization parameter λ can be used to balance the global (at the distribution scale) and the local (at the samples' scale) interactions between different classes.

The original Wasserstein Discriminant Analysis solves the following bi-level optimization

problem:

$$\begin{aligned}
&\max_{\mathbf{P}\in\mathbb{O}^{d\times p}} J(\mathbf{P},\ \mathbf{T}(\mathbf{P})) = \frac{\sum_{c,c'>c}\langle \mathbf{P}\mathbf{P}^T, \mathbf{C}^{c,c'}\rangle}{\sum_c \langle \mathbf{P}\mathbf{P}^T, \mathbf{C}^{c,c}\rangle} = \frac{\langle \mathbf{P}\mathbf{P}^T, \mathbf{C}_b\rangle}{\langle \mathbf{P}\mathbf{P}^T, \mathbf{C}_w\rangle}, \\
&\text{where } \mathbf{C}^{c,c'} = \sum_{i,j} t_{i,j}^{c,c'}(x_i^c - x_j^{c'})(x_i^c - x_j^{c'})^T, \quad \forall c, c', \\
&\text{s.t. } \mathbf{T}^{c,c'} = \arg\min_{\mathbf{T}\in U_{n_c n_{c'}}} \lambda\langle \mathbf{T}, \mathbf{M}_{\mathbf{X}^c\mathbf{P},\mathbf{X}^{c'}\mathbf{P}}\rangle - \Omega(\mathbf{T}),
\end{aligned} \tag{2.2}$$

where $\mathbf{X}^c \in \mathbb{R}^{n_c\times d}$ is the data matrix of the samples from class c, and $\mathbf{X}^c\mathbf{P}$ is the matrix of projected samples from class c. $\mathbf{C}_b = \sum_{c,c'>c}\mathbf{C}^{c,c'}$ and $\mathbf{C}_w = \sum_c \mathbf{C}^{c,c}$ are the between and within cross-covariance matrices, and they both depend on $\mathbf{T}(\mathbf{P})$.

In [44], the gradient $G^k = \nabla_{\mathbf{P}} J(\mathbf{P},\ \mathbf{T}(\mathbf{P}))$ at iteration k was computed using automatic differentiation [92], and the optimization problem is solved using pymanopt solvers such as the projected gradient descent and trust region methods on the Stiefel manifold [12, 132]. In practice, due to the complication of the problem formulation and the structures of data, the gradient-based approach often has a slow convergence and is sensitive to parameters and initialization. We will illustrate these difficulties in Section 2.4 with numerical experiments.

2.2.2 The Nonlinear Eigensolver-based Approach

For (2.2), once $\mathbf{T}^{c,c'}$ is computed, the problem becomes a trace ratio problem:

$$\max_{\mathbf{P}\in\mathbb{O}^{d\times p}} J(\mathbf{P}) = \frac{\text{Trace}(\mathbf{P}^T\mathbf{C}_b\mathbf{P})}{\text{Trace}(\mathbf{P}^T\mathbf{C}_w\mathbf{P})}, \tag{2.3}$$

where $\mathbf{C}_b$ and $\mathbf{C}_w$ depend on $\mathbf{P}$. We approximate the problem by solving a ratio trace problem:

$$\max_{\mathbf{P}\in\mathbb{R}^{d\times p}} J_{rt}(\mathbf{P}) = \text{Trace}((\mathbf{P}^T\mathbf{C}_b\mathbf{P})(\mathbf{P}^T\mathbf{C}_w\mathbf{P})^{-1}), \tag{2.4}$$

Problem (2.4) can be efficiently solved by the generalized eigenvalue decomposition:

$$\mathbf{C}_b(\mathbf{P})\mathbf{P} = \mathbf{C}_w(\mathbf{P})\mathbf{P}\Lambda, \tag{2.5}$$

where the optimal $\mathbf{P}$ is the matrix of eigenvectors corresponding to the p largest generalized eigenvalues. The generalized eigenvector-dependent nonlinear eigenvalue problem (which we refer to as NLEP from now on) can be solved via the self-consistent field (SCF) iteration [95, 114]: given $\mathbf{P}_{t-1}$, we first construct $\mathbf{C}_b(\mathbf{P}_{t-1})$ and $\mathbf{C}_w(\mathbf{P}_{t-1})$, then solve the generalized eigenvalue problem $\mathbf{C}_b(\mathbf{P}_{t-1})v = \mu\mathbf{C}_w(\mathbf{P}_{t-1})v$. Let v_j be the eigenvector corresponding to the jth largest generalized eigenvalue, then $\mathbf{P}_t$ is updated as an orthonormal basis for $[v_1, \ldots, v_p]$. Compared to the gradient-based approach, the new formulation with SCF iteration could drastically reduce the number of iterations. We therefore propose Algorithm 1 for solving supervised WDA.

Algorithm 1 WDA-eig algorithm

Input: De-meaned data X, class labels $\hat{y}$, initial subspace $\mathbf{P}_0 \in \mathbb{O}^{d\times p}$, tolerance ϵ, maximum number of iterations N

for $k = 1$ **to** N **do**

  for each pair of classes c, c' **do**

    Compute $\mathbf{T}^{c,c'}(\mathbf{P}_{k-1})$ using the Sinkhorn iteration

  end for

  Construct $\mathbf{C}_b(\mathbf{P}_{k-1})$ and $\mathbf{C}_w(\mathbf{P}_{k-1})$

  Compute the generalized eigenvalue problem: $\mathbf{C}_b(\mathbf{P}_{k-1})\mathbf{P} = \mathbf{C}_w(\mathbf{P}_{k-1})\mathbf{P}\Lambda$, and obtain $\mathbf{P}_k \in \mathbb{O}^{d\times p}$ as an orthonormal basis for the eigenvector matrix corresponding to the p largest generalized eigenvalues

  if the change in $\mathbf{P}_k$ is sufficiently small **then**

    Break

  end if

end for

From a computational complexity point of view, suppose that for each class there are n samples and d features. For each SCF iteration, complexity is dominated by constructing $\mathbf{C}_b$ and $\mathbf{C}_w$, which are $\mathcal{O}(n^2d^2)$. Solving the generalized eigenvalue problem has complexity

$\mathcal{O}(d^3)$, but it is possible to run only a few iterations to reach a certain tolerance. Each Sinkhorn iteration is of $\mathcal{O}(n^2)$ and we run a fixed number of iterations. The memory complexity is $\mathcal{O}(d^2)$ by storing the matrices $\mathbf{C}_b$ and $\mathbf{C}_w$.

2.3 Analysis

In this section we first give a convergence analysis for the SCF framework for solving generalized NLEP, followed by an analysis for the proposed **WDA-eig** in Algorithm 1.

2.3.1 Convergence of SCF

Consider the generalized NLEP $A(\mathbf{P})\mathbf{V} = B(\mathbf{P})\mathbf{V}\Lambda$, where $\mathbf{V} = [v_1, \ldots, v_p]$ and $\mathbf{P}$ is an orthonormal basis of $\mathbf{V}$ that spans the same subspace as $\mathbf{V}$. $A(\mathbf{P})$, $B(\mathbf{P})$ are symmetric matrix-valued functions and $B(\mathbf{P})$ is positive definite. $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_p)$, where $\lambda_1 \geq \cdots \geq \lambda_p$ are the p largest eigenvalues of $(A(\mathbf{P}), B(\mathbf{P}))$ corresponding to eigenvectors $v_1, \ldots, v_p$. We emphasize that $A(\mathbf{P})$, $B(\mathbf{P})$ are invariant to orthogonal transformation of $\mathbf{P}$, i.e., $A(\mathbf{P}) \equiv A(\mathbf{P}Q)$, $B(\mathbf{P}) \equiv B(\mathbf{P}Q)$ for any orthogonal matrix $Q \in \mathbb{R}^{p\times p}$.

Definitions. Let $\mathcal{X}$ and $\mathcal{Y}$ be two p-dimensional subspaces of $\mathbb{R}^n$. Let the columns of X form an orthonormal basis for $\mathcal{X}$ and the columns of Y form an orthonormal basis for $\mathcal{Y}$. We use $\|\sin\Theta(\mathcal{X}, \mathcal{Y})\|$ as in [124] to measure the distance between $\mathcal{X}$ and $\mathcal{Y}$, where

$$\Theta(\mathcal{X}, \mathcal{Y}) = \operatorname{diag}(\theta_1(\mathcal{X}, \mathcal{Y}), \ldots, \theta_p(\mathcal{X}, \mathcal{Y})). \tag{2.6}$$

Here, $\theta_j(\mathcal{X}, \mathcal{Y})$'s denote the *canonical angles* between $\mathcal{X}$ and $\mathcal{Y}$ [p. 43] [124], which is defined as

$$0 \leq \theta_j(\mathcal{X}, \mathcal{Y}) \triangleq \arccos\sigma_j \leq \frac{\pi}{2} \quad \text{for } 1 \leq j \leq k, \tag{2.7}$$

where σ_j's are the singular values of X^TY. Similar to the Crawford number for symmetric definite matrix pair (A, B) [Chapter 8.7] [133], we define the Crawford number for the generalized NLEP as

$$c \triangleq \min_{\mathbf{P}\in\mathbb{O}^{d\times p}} \min_{x\in\mathbb{C}^d, \|x\|=1} (x^T(A(\mathbf{P}) + iB(\mathbf{P}))x),$$

where i is the imaginary unit. Define $C \triangleq \max_{\mathbf{P}\in\mathbb{O}^{d\times p}} \sqrt{\|A(\mathbf{P})^2 + B(\mathbf{P})^2\|}$. At the kth SCF iteration, one computes an approximation to the eigenvector matrix $\mathbf{V}_k$ associated with the p largest eigenvalues of $(A(\mathbf{P}_{k-1}), B(\mathbf{P}_{k-1}))$, where $\mathbf{P}_{k-1}$ is an orthonormal basis for $\mathbf{V}_{k-1}$, and then $\mathbf{V}_k$ is used as the next approximation to the solution. Let $A_k = A(\mathbf{P}_k)$, $B_k = B(\mathbf{P}_k)$, $\mu_{i,k} = \arctan\lambda_i(A(\mathbf{P}_k), B(\mathbf{P}_k))$. We also define $s_k \triangleq \|\sin\Theta(\mathbf{P}_k, \mathbf{P}_{k-1})\|$ as the distance between subspaces $\mathbf{P}_k$ and $\mathbf{P}_{k-1}$.

We study the convergence of SCF iteration under the following assumptions:

A1: For any $\mathbf{P}_1, \mathbf{P}_2 \in \mathbb{O}^{d\times p}$, assume that there exist positive constants ξ_a, ξ_b such that

$$\|A(\mathbf{P}_1) - A(\mathbf{P}_2)\| \le \xi_a\|\sin\Theta(\mathbf{P}_1, \mathbf{P}_2)\|, \quad \|B(\mathbf{P}_1) - B(\mathbf{P}_2)\| \le \xi_b\|\sin\Theta(\mathbf{P}_1, \mathbf{P}_2)\|;$$

A2: For $k = 1, 2, \cdots$, there exists an $\eta > 0$ such that

$$\mu_{p,k} - \mu_{p+1,k} \ge \eta.$$

We state the convergence theorems below and give proofs in the supplementary material. By global convergence we mean that the algorithm converges to some stationary points [120] and does not guarantee convergence to a global optimum for all initial points. The algorithm converges when the change in subspace is sufficiently small, i.e., s_k is within some user-specified tolerance.

Theorem 1. ***(Global Convergence)*** *Let* $s_1 = \|\sin\Theta(\boldsymbol{P}_0, \boldsymbol{P}_1)\|$. *Assume* ***A1*** *and* ***A2****, and* $s_1\sqrt{\xi_a^2 + \xi_b^2} < c$. *If*

$$\eta > \arcsin(\rho C\sqrt{\xi_a^2 + \xi_b^2}/c^2) + \arctan(s_1\sqrt{\xi_a^2 + \xi_b^2}/c)$$

for some constant $\rho > 1$*, then SCF converges linearly at the rate of* $\frac{1}{\rho}$.

With relaxed assumption on the arctangent gap, we can show local convergence if the initial subspace is close enough to the true subspace $\mathbf{P}^*$:

A3: Let μ_i^* denote $\arctan\lambda_i(A(\mathbf{P}^*), B(\mathbf{P}^*))$. There exists an $\eta > 0$ such that

$$\mu_p^* - \mu_{p+1}^* \ge \eta.$$

Theorem 2. ***(Local Convergence)*** *Let* $\hat{s}_0 = \|\sin\Theta(\boldsymbol{P}_0, \boldsymbol{P}^*)\|$. *Assume* ***A1*** *and* ***A3****, and* $\hat{s}_0\sqrt{\xi_a^2+\xi_b^2} < c$. *If*

$$\eta > \arcsin(\rho C\sqrt{\xi_a^2+\xi_b^2}/c^2) + \arctan(\hat{s}_0\sqrt{\xi_a^2+\xi_b^2}/c)$$

for some constant $\rho > 1$*, then SCF is locally convergent at* $\boldsymbol{P}^*$ *at the rate of* $\frac{1}{\rho}$.

Theorems 1 and 2 characterize how the eigenspace varies when the matrix pair undergoes a small perturbation. The sensitivity of the matrix pair as functions of $\mathbf{P}$ is quantified by the Lipschitz constants in A1. A2 and A3 are assumptions to guarantee that a discriminative subspace exists. In the following section we give more concrete examples in the WDA context for these assumptions.

2.3.2 Analysis for Supervised WDA

In the context of WDA, $A(\mathbf{P})$ is the inter-class covariance matrix $\mathbf{C}_b(\mathbf{P})$ and $B(\mathbf{P})$ is the intra-class covariance matrix $\mathbf{C}_w(\mathbf{P})$. For each iteration in **WDA-eig**, a fixed number of Sinkhorn iterations is computed to obtain an approximation to the optimal transport distance $\mathbf{T}$. $\mathbf{T}(\mathbf{P})$ can be expressed as an implicit function using the optimality conditions of the equation defining the optimal $\mathbf{T}$, and $\frac{\partial \mathbf{T}}{\partial \mathbf{P}}$ exists and is bounded. Therefore it is safe to assume that $\mathbf{T}$ is Lipschitz continuous in $\mathbf{P}$.

Corollary 1. *Suppose that the optimal transport matrix* $\boldsymbol{T}^{c,c'}$ *satisfies a Lipschitz-like condition:*

$$\|\boldsymbol{T}^{c,c'}(\boldsymbol{P}_1) - \boldsymbol{T}^{c,c'}(\boldsymbol{P}_2)\| \leq \xi^{c,c'}\|\sin\Theta(\boldsymbol{P}_1, \boldsymbol{P}_2)\|,$$

For a given p, let

$$\eta = \min_k\{\eta_k | \eta_k = \mu_{p,k} - \mu_{p+1,k}\}.$$

Denote $\xi_a = \sum_{c,c'>c}\xi^{c,c'}\|\sum_{i,j}(x_i^c - x_j^{c'})(x_i^c - x_j^{c'})^T\|$, $\xi_b = \sum_c \xi^{c,c}\|\sum_{i,j}(x_i^c - x_j^c)(x_i^c - x_j^c)^T\|$. *If*

$$\eta > \arcsin(\rho C\sqrt{\xi_a^2+\xi_b^2}/c^2) + \arctan(s_1\sqrt{\xi_a^2+\xi_b^2}/c)$$

for some constant $\rho > 1$*, then* ***wda-eig*** *converges linearly at the rate* $\frac{1}{\rho}$.

Corollary 1 implies that given a data matrix, the convergence rate of **WDA-eig** depends on the initialization, the subspace dimension p and ξ_a, ξ_b. ξ_a and ξ_b are functions of $\xi^{c,c'}$ and depends on the Wasserstein regularization parameter λ. When $\lambda = 0$, $t^{c,c'}$ is a constant matrix and $\xi^{c,c'} = 0$. For a fixed λ, the arctangent gap η depends on the inherent structure of the data matrix and whether a discriminative subspace exists. For example, given two clusters of data generated from 2D normal distributions as shown in Figure 2.1, η depends on the separation of these two clusters. We can calculate $\eta^* \triangleq \mu_p^* - \mu_{p+1}^*$ since we know the true subspace $\mathbf{P}^*$, and we also run **WDA-eig** on a random initialization to get η. We observe that η is close to 0 when the clusters overlap and is a monotonically increasing function of the Euclidean distance between the mean of the two clusters.

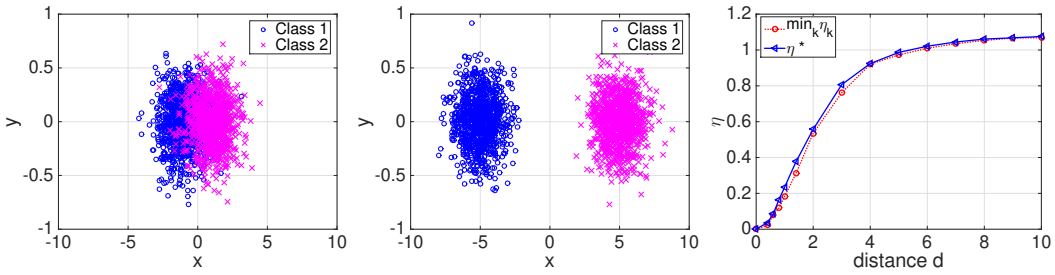


Figure 2.1: **Arctangent gap as a function of the distance between clusters.** Left and middle: two classes of data generated from two random normal distributions: $X_i \mathcal{N} \in (\mu_i, \Sigma)$, $\mu = (\pm x, 0)$ where $x \in [0, 5]$. $x = 1$ in the Left and $x = 5$ in the Middle. Right: Arctangent gap η as a function of the distance between the means $d \triangleq \|\mu_1 - \mu_2\|$.

By applying the algorithm to a simulated dataset with 3 classes and 2 discriminative dimensions, we draw log plots of the distances between the subspaces subject to these components in Figure 2.2 to illustrate linear convergence rates. On the left we show s_k with different values of the subspace dimension p and with $\lambda = 0.1$ fixed. With $p = 2$ the algorithm achieves the fastest rate because the dimension of the true discriminative subspace is 2. In FDA, since $\mathbf{C}_b$ has rank $Nc - 1$ (where Nc is the number of classes), p has to be $\leq Nc - 1$. In **WDA-eig**, p is less restrictive, but choosing $p \geq Nc$ may still slow down

or prevent convergence if λ is small. In the middle we show s_k with different values of the Wasserstein regularizer λ and with $p = 2$ fixed. When λ is small, the matrices $\mathbf{C}_w$ and $\mathbf{C}_b$ in WDA can be viewed as the matrices in FDA with a small perturbation, and in such cases the Lipschitz constants ξ_a and ξ_b are close to zero so the algorithm is guaranteed to converge. We also observe that a larger λ corresponds to a slower convergence rate. On the right we illustrate the effect of initialization for local convergence. We use the converged solution as an approximation to the true discriminative subspace $\mathbf{P}^*$ and plot the distance $\|\sin\Theta(\mathbf{P}^*, \mathbf{P}_{k-1})\|$ for each iteration k, with varying $\hat{s}_0 = \|\sin\Theta(\mathbf{P}^*, \mathbf{P}_0)\|$. We observe that initialization has little effect on the convergence rate and that the algorithm converges in most cases except for the case where $\hat{s}_0 \approx 1$.

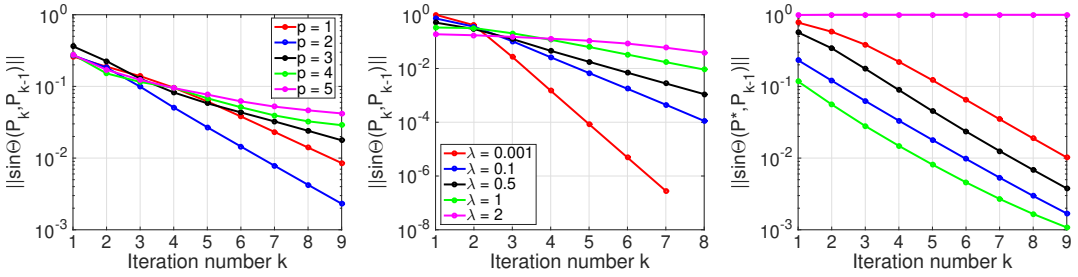


Figure 2.2: **Distances between subspaces with varying parameters.** Left and Middle: Distances between subspaces $\|\sin\Theta(\mathbf{P}_k, \mathbf{P}_{k-1})\|$ as a function of iteration number k, with varying λ and p, respectively. Right: $\|\sin\Theta(\mathbf{P}^*, \mathbf{P}_{k-1})\|$ as a function of iteration number k, with varying initialization $\mathbf{P}_0$.

2.4 Numerical Experiments

In this section we evaluate the performance of the proposed Algorithm 1 on classification tasks by applying it to a simulated dataset and the MNIST dataset. We refer to our proposed algorithm as **WDA-eig** and refer to the original implementation in [43] with projected gradient descent as **WDA**. **WDA** converges when the norm of the gradient is below 10^{-6}, and **WDA-eig** converges when the distance between two consecutive subspaces is less than 10^{-6}.

2.4.1 Simulated dataset

We first compare **WDA-eig** with **WDA** on a simulated dataset. We use the same setup as given in [44], where the data belongs to 3 non-linearly separable classes and is generated using 2 discriminant features and 8 dimensions of Gaussian noise. We apply these two algorithms with varying regularization parameter λ, and compare their computational efficiency and classification accuracy with a K-Nearest-Neighbors classifier (KNN) on the projected data ($k = 10$). For each λ, we run each algorithms for 100 randomly-initialized trials, and the results are shown in Table 2.1. The third column of the table shows the probability of convergence over 100 trials, and the fourth column shows the accuracy averaged over trials. For $\lambda = 0.1,\ 1,\ 5$, **WDA-eig** converges in all the trials with zero standard deviations and achieves higher accuracy scores on average, while **WDA** has high standard deviation due to the low probability of convergence. The fifth column shows the accuracy averaged only for the converged trials, and **WDA-eig** and **WDA** have comparable performances in accuracy, which indicates that the ratio trace formulation can serve as a good approximation to the trace ratio formulation. The last column shows the efficiency measured by averaged CPU time in seconds over 100 trials (all these codes are run on a local machine with 2.3 GHz Dual-Core Intel Core i5 CPU and 8 GB RAM). **WDA-eig** takes shorter running time than **WDA** since the former only requires a few iterations to converge and the running time per SCF iteration is comparable to the running time per gradient descent iteration. Even in cases where most trials converge for both solvers (e.g., when $\lambda = 1$), **WDA** takes more iterations to converge on average.

2.4.2 MNIST dataset

Next, we test the classification performance on a real dataset and also evaluate the generalization ability of the proposed approach. To measure the robustness and generalizability of our approach on small training sample size despite high-dimensionality of the problem, we extract 1000 samples in the MNIST dataset as the training set and test on another 10,000

Table 2.1: Comparison between **WDA** and **WDA-eig**

Param λ	Algo	Prob. of Convergence	Avg. Acc.(std.)	Converged Acc.(std.)	CPU time
$\lambda = 0.1$	**wda**	25%	0.712(0.152)	0.977(0)	72
	wda-eig	100%	0.968(0)	0.968(0)	0.677
$\lambda = 1.0$	**wda**	78%	0.908(0.149)	0.987(0)	6.37
	wda-eig	100%	0.986(0)	0.986(0)	1.17
$\lambda = 5.0$	**wda**	73%	0.885(0.164)	0.985(0)	7.04
	wda-eig	100%	0.985(0)	0.985(0)	1.61

samples. We measure the KNN prediction error on the projected data as a function of the subspace dimension p, the number of nearest neighbors K, and the Wasserstein regularization parameter λ respectively in Figure 2.3. On the left we show the prediction error of full data/PCA/FDA/**WDA**/**WDA-eig**+KNN applied to the original data as a function of p, with $\lambda = 0.01$ and $K = 10$ fixed. In implementation of FDA/**WDA-eig** we add a small perturbation term ϵI_p on $\mathbf{C}_w$ to make the denominator positive definite, and we choose $\epsilon = 2$ in this setting, which removes the restriction of $p \leq 9$ for FDA. In the middle we show the performance of these methods as a function of K. Another approach to avoid $\mathbf{C}_w$ being semidefinite is to project away the null space of the data matrix before applying discriminant analysis. To achieve this end, we first apply PCA to the original data matrix and retain only the first 20 principal components. We then apply PCA/FDA/LFDA [126]/**WDA**/**WDA-eig** on the dimension-reduced data to obtain a subspace of dimension $p = 9$ without any regularization on $\mathbf{C}_w$, and the results are shown on the right.

2.5 Unsupervised WDA

Since WDA is a dimensionality reduction technique, it could also be integrated with a low-dimensional clustering technique to do high-dimensional clustering. Here we propose Algorithm 2 to extend WDA to the unsupervised setting.

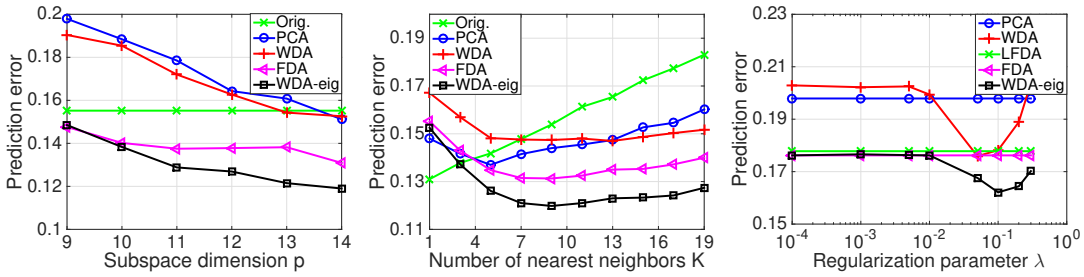


Figure 2.3: **Comparisons on the MNIST dataset.** We apply dimensionality reduction techniques and KNN to the MNIST dataset, and plot the prediction error as a function of the subspace dimension p, the number of nearest neighbors K, and the Wasserstein regularization parameter λ.

2.5.1 Clustering Algorithm

Algorithm 2 Iterative WDA clustering

Input: De-meaned dataset X, $\mathbf{P}_0 \in \mathbb{O}^{d\times p}$, tolerance ϵ, max number of iterations N

for $k = 0, 1, \cdots, N$ **do**

Compute $y = X\mathbf{P}_k \in \mathbb{R}^{n\times p}$

Cluster y into K classes and obtain the class labels $\hat{y}$

Call Algorithm 1 to update $\mathbf{P}$: $\mathbf{P}_{k+1} = \mathbf{WDA\text{-}eig}(X, \hat{y}, \mathbf{P}_k)$

if the change in $\mathbf{P}_k$ is sufficiently small **then**

Break

end if

end for

We start with an initial guess and adaptively improve its labeling by performing clustering in the projected space. The goal is to converge to a discriminative subspace that will render the most accurate labels. Algorithm 2 solves the following optimization problem:

$$\max_{\mathbf{P}} \hat{J}(\mathbf{P}, \mathbf{T}, \hat{y}) \quad \text{s.t. } \mathbf{T}^{c,c'} = \arg\min_{\mathbf{T}\in U_{n_c n_{c'}}} E_1(\mathbf{T}, \mathbf{P}, \hat{y}),\ \hat{y} = \arg\min_{\hat{y}} E_2(\mathbf{P}, \hat{y}), \tag{2.8}$$

and E_2 is the objective of any specific low-dimensional clustering technique. The algorithm uses an alternating optimization scheme: for each iteration, given the class labels $\hat{y}$, $\mathbf{P}$ is chosen to maximize the ratio-trace problem $\hat{J}$, and then given the subspace, it finds the optimal labeling according to the clustering objective E_2. The objectives E_2 and $\hat{J}$ do not always align. A special case is FDA-Kmeans (or LDA-Kmeans) [37], where minimizing E_2 is equivalent to maximizing $\hat{J}$. It is derived that iteratively applying FDA and K-means is the same as alternating optimization in a unified objective [139], and that combining FDA and K-means is equivalent to kernel K-means in the original space with a specific kernel Gram matrix [147].

However, there is no theoretical guarantee that a larger objective value corresponds to a better clustering result in terms of external evaluation criteria. We observe that for FDA-Kmeans, the adjusted random index (ARI) [108] does not increase monotonically with the iteration number and could even converge to a worse result compared to the initial guess. For WDA, K-Means in the projected space does not maximize $\hat{J}$, but empirically we observe that several iterations with K-Means does improve clustering result in terms of external evaluation criteria such as ARI. Since the performance of FDA degrades when class distributions are multimodal, FDA could perform poorly given the wrong labels even if the true underlying distribution is Gaussian. On the other hand, we numerically observe that WDA is more robust to noisy labels due to a balance of local and global information.

2.5.2 Sensitivity to Noisy Labels

For iterative subspace clustering, performing K-Means on the projected data may not render accurate labels in the first few iterations, especially if we initialize with random subspace. We therefore investigate how the subspace changes when we perturb the labels. The results in Table 2.2 illustrate the sensitivity to noisy labels of FDA (same as **WDA-eig** with $\lambda = 0$), **WDA-eig** ($\lambda = 1.0$) and local FDA (LFDA) [126] with the number of neighbors= 1. We use the simulated dataset introduced in the Main Paper, Section 4.1 and add noisy labels to the data. The first column of Table 2.2 shows the percentage of wrong labels added. The rest

of the columns show the distance of the subspace $\mathbf{P}$ obtained by FDA/**WDA-eig**/LFDA under the noisy labels to the original subspaces $\mathbf{P}^*$ measured by $\|\sin\Theta(\mathbf{P},\mathbf{P}^*)\|$, where the original subspaces are approximated by the converged solution of FDA/**WDA-eig**/LFDA under true labels. The results are averaged over 20 trials. We observe that WDA is more robust to noisy labels than both FDA and local FDA.

Table 2.2: Sensitivity to Noisy Labels.

% wrong labels	FDA dist. to $\mathbf{P}^*$	WDA dist. to $\mathbf{P}^*$	LFDA dist. to $\mathbf{P}^*$
1%	0.21	**0.01**	0.04
5%	0.32	**0.02**	0.08
10%	0.59	**0.05**	0.11
20%	0.84	**0.07**	0.15

2.5.3 Experiments on WDA Clustering

In this section we evaluate the proposed Algorithm 2 and compare with other subspace clustering techniques. In what follows, let Nc denote the number of classes, n be the number of observations and d be the number of features.

We use four real world datasets to evaluate the proposed method: the MNIST dataset for digits recognition, the 15-scene dataset [81] for multi-class image recognition, the KTH action recognition database [115] for multi-class video recognition, and the 20 newsgroup dataset for text classification. To avoid the singularity of the $\mathbf{C}_w$ matrix in FDA and WDA, we first do a dimension reduction on the original dataset using PCA and retain the first $2 \times Nc$ principal components. We refer to this data as the dimension-reduced data. We apply four different clustering methods to the four dataset: (1) K-means on the original data (Baseline); (2) K-means on dimension-reduced data (PCAKm); (3) FDA-Kmeans (FDAKm) on the dimension-reduced data; (4) WDA-kmeans (Algorithm 2 combined with K-means)

Table 2.3: Clustering results.

Method	Dataset	ARI	NMI	Homogeneity	Completeness	FMI
Baseline	MNIST	0.334 ± 0.007	0.475 ± 0.006	0.473 ± 0.006	0.477 ± 0.007	0.403 ± 0.007
PCAKm	MNIST	0.334 ± 0.005	0.471 ± 0.005	0.469 ± 0.004	0.473 ± 0.007	0.402 ± 0.005
FDAKm	MNIST	0.360 ± 0.006	0.500 ± 0.007	0.497 ± 0.006	0.503 ± 0.008	0.426 ± 0.005
WDAKm	MNIST	$\mathbf{0.398 \pm 0.008}$	$\mathbf{0.526 \pm 0.006}$	$\mathbf{0.524 \pm 0.006}$	$\mathbf{0.528 \pm 0.006}$	$\mathbf{0.459 \pm 0.008}$
Baseline	KTH	0.424 ± 0.035	0.576 ± 0.035	0.556 ± 0.035	0.598 ± 0.033	0.535 ± 0.028
PCAKm	KTH	0.470 ± 0.017	0.616 ± 0.009	0.596 ± 0.011	0.637 ± 0.007	0.571 ± 0.011
FDAKm	KTH	0.481 ± 0.022	0.635 ± 0.018	0.614 ± 0.019	0.657 ± 0.017	0.580 ± 0.016
WDAKm	KTH	$\mathbf{0.488 \pm 0.020}$	$\mathbf{0.643 \pm 0.015}$	$\mathbf{0.623 \pm 0.016}$	$\mathbf{0.663 \pm 0.014}$	$\mathbf{0.584 \pm 0.014}$
Baseline	15scene	0.161 ± 0.009	0.352 ± 0.009	0.336 ± 0.009	0.369 ± 0.009	0.233 ± 0.008
PCAKm	15scene	0.160 ± 0.007	0.351 ± 0.006	0.334 ± 0.007	0.368 ± 0.006	0.232 ± 0.005
FDAKm	15scene	0.150 ± 0.009	0.350 ± 0.011	0.324 ± 0.010	0.376 ± 0.014	0.234 ± 0.010
WDAKm	15scene	$\mathbf{0.170 \pm 0.010}$	$\mathbf{0.366 \pm 0.012}$	$\mathbf{0.350 \pm 0.011}$	$\mathbf{0.384 \pm 0.012}$	$\mathbf{0.243 \pm 0.009}$
Baseline	20ng	0.081 ± 0.011	0.235 ± 0.021	0.217 ± 0.020	0.255 ± 0.023	0.151 ± 0.009
PCAKm	20ng	0.097 ± 0.003	0.247 ± 0.005	0.238 ± 0.005	0.255 ± 0.005	0.152 ± 0.003
FDAKm	20ng	0.113 ± 0.013	0.298 ± 0.019	0.275 ± 0.019	0.322 ± 0.020	0.185 ± 0.010
WDAKm	20ng	$\mathbf{0.128 \pm 0.011}$	$\mathbf{0.302 \pm 0.014}$	$\mathbf{0.283 \pm 0.013}$	$\mathbf{0.322 \pm 0.015}$	$\mathbf{0.194 \pm 0.010}$

(WDAKm) on the dimension-reduced data. For (3) and (4) we use the subspace obtained by PCA as initialization and $p = Nc - 1$ as the subspace dimensions. No regularization term is added to $\mathbf{C}_w$. The Wasserstein regularizer λ is coarsely tuned, where we choose $\lambda = 0.01$ for MNIST and 15-scene, $\lambda = 10$ for KTH, and $\lambda = 5$ for 20ng. The results are averaged over 20 trials. We use five external evaluation criteria to evaluate the quality of the clustering solutions:

- Adjusted random index (ARI; [59]): the Rand Index computes a similarity measure between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. The raw RI score is then "adjusted for chance" into the ARI score.

- Normalized mutual information (NMI; [125]): the Mutual Information is a function that measures the agreement of the two assignments, ignoring permutations.

- Homogeneity [113]: the measurement of whether each cluster contains only members of a single class.

- Completeness [113]: the measurement of whether all members of a given class are assigned to the same cluster.

- Fowlkes-Mallows scores (FMI; [45]): FMI is defined as the geometric mean of the pairwise precision and recall:

$$FMI = \frac{TP}{\sqrt{(TP + FP)(TP + FN)}}$$

For all these measurement, a random labeling would give a value close to 0 and perfect labeling corresponds to a value of 1.

The results in Table 2.3 show that WDAKm achieves the best performance on all four datasets, in terms of the 5 external metrics we use. We also notice that in 15-scene dataset the performance of FDAKm is worse than the baseline method, which means with some wrong

tags, FDA tends to overly separate data and decrease the clustering quality. In contrast, WDA always improve the clustering even in the difficult case such as the 15-scene dataset.

2.6 Conclusion

In this paper, we present a ratio trace formulation of the Wasserstein Discriminant Analysis and an eigensolver-based algorithm: **WDA-eig** to solve the problem. Unlike the original trace ratio formulation, the ratio trace formulation has a closed-form solution that is readily obtainable by the generalized eigenvalue decomposition once the regularized optimal transport problem is solved. We give a convergent analysis for **WDA-eig** under the SCF framework and numerically test the efficiency and convergence properties of the proposed algorithm. Although **WDA-eig** solves a slightly different problem, the ratio trace formulation can serve as an efficient alternative for the trace ratio formulation of WDA. **WDA-eig** also takes less time to converge on average and is less sensitive to initialization and parameters compared to **WDA**. As a supervised dimensionality reduction technique, WDA can also be combined with clustering techniques and applied iteratively to perform unsupervised learning. Numerical experiments show that the WDA clustering algorithm performs well on a set of real-world problems.

2.7 Appendix

Proof of Theorem 1. (Global Convergence of SCF) Consider the generalized NLEP $A(\mathbf{P})\mathbf{V} = B(\mathbf{P})\mathbf{V}\Lambda$, where $\mathbf{V} = [v_1, \ldots, v_p]$ and $\mathbf{P}$ is an orthonormal basis of $\mathbf{V}$. $A(\mathbf{P})$, $B(\mathbf{P})$ are symmetric matrix-valued function and $B(\mathbf{P})$ is positive definite. $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$, where $\lambda_1 \geq \cdots \geq \lambda_p$ are the p largest eigenvalues of $(A(\mathbf{P}), B(\mathbf{P}))$ corresponding to eigenvectors $v_1, \ldots, v_p$. We emphasize that $A(\mathbf{P})$, $B(\mathbf{P})$ are invariant to orthogonal transformation of $\mathbf{P}$, i.e., $A(\mathbf{P}) \equiv A(\mathbf{P}Q)$, $B(\mathbf{P}) \equiv B(\mathbf{P}Q)$ for any orthogonal matrix $Q \in \mathbb{R}^{p\times p}$.

Definitions. Let $\mathcal{X}$ and $\mathcal{Y}$ be two p-dimensional subspaces of $\mathbb{R}^n$. Let the columns of X form an orthonormal basis for $\mathcal{X}$ and the columns of Y form an orthonormal basis for $\mathcal{Y}$.

We use $\|\sin\Theta(\mathcal{X},\mathcal{Y})\|$ as in [124] to measure the distance between $\mathcal{X}$ and $\mathcal{Y}$, where

$$\Theta(\mathcal{X},\mathcal{Y}) = \mathrm{diag}(\theta_1(\mathcal{X},\mathcal{Y}),\ldots,\theta_p(\mathcal{X},\mathcal{Y})).$$

Here, $\theta_j(\mathcal{X},\mathcal{Y})$'s denote the *canonical angles* between $\mathcal{X}$ and $\mathcal{Y}$ [p. 43] [124], which is defined as

$$0 \le \theta_j(\mathcal{X},\mathcal{Y}) \triangleq \arccos\sigma_j \le \frac{\pi}{2} \quad \text{for } 1 \le j \le k,$$

where σ_j's are the singular values of X^TY. Similar to the Crawford number for symmetric definite matrix pair (A,B) [Chapter 8.7] [133], we define the Crawford number for the generalized NLEP as

$$c \triangleq \min_{\mathbf{P}\in\mathbb{O}^{d\times p}} \min_{x\in\mathbb{C}^d,\|x\|=1} (x^T(A(\mathbf{P}) + iB(\mathbf{P}))x).$$

Define $C \triangleq \max_{\mathbf{P}\in\mathbb{O}^{d\times p}} \sqrt{\|A(\mathbf{P})^2 + B(\mathbf{P})^2\|}$. At the kth SCF iteration, one computes an approximation to the eigenvector matrix $\mathbf{V}_k$ associated with the p largest eigenvalues of $(A(\mathbf{P}_{k-1}), B(\mathbf{P}_{k-1}))$, where $\mathbf{P}_{k-1}$ is an orthonormal basis for $\mathbf{V}_{k-1}$, and then $\mathbf{V}_k$ is used as the next approximation to the solution. Let $A_k = A(\mathbf{P}_k)$, $B_k = B(\mathbf{P}_k)$, $\mu_{i,k} = \arctan\lambda_i(A(\mathbf{P}_k), B(\mathbf{P}_k))$.

We study the convergence of SCF iteration under the following assumptions:

A1: For any $\mathbf{P}_1, \mathbf{P}_2 \in \mathbb{O}^{d\times p}$, assume that there exist positive constants ξ_a, ξ_b such that

$$\|A(\mathbf{P}_1) - A(\mathbf{P}_2)\| \le \xi_a\|\sin\Theta(\mathbf{P}_1,\mathbf{P}_2)\|, \quad \|B(\mathbf{P}_1) - B(\mathbf{P}_2)\| \le \xi_b\|\sin\Theta(\mathbf{P}_1,\mathbf{P}_2)\|;$$

A2: For $k = 1, 2, \cdots$, there exists an $\eta > 0$ such that

$$\mu_{p,k} - \mu_{p+1,k} \ge \eta.$$

Theorem 1. ***(Global Convergence)*** *Let $s_1 = \|\sin\Theta(\boldsymbol{P}_0, \boldsymbol{P}_1)\|$. Assume **A1** and **A2**, and $s_1\sqrt{\xi_a^2 + \xi_b^2} < c$. If*

$$\eta > \arcsin(\rho C\sqrt{\xi_a^2 + \xi_b^2}/c^2) + \arctan(s_1\sqrt{\xi_a^2 + \xi_b^2}/c)$$

for some constant $\rho > 1$, then SCF converges linearly at the rate of $\frac{1}{\rho}$.

In order to show Theorem 1, we need the following three lemmas. The first lemma gives some fundamental results for $\|\sin\Theta(X,Y)\|$, which can be verified via definition.

Lemma 1. *Let $[X,\, X_{\rm c}]$ and $[Y,\, Y_{\rm c}]$ be two orthogonal matrices with $X, Y \in \mathbb{R}^{n\times k}$. Then*

$$\|\sin\Theta(X,Y)\| = \|X_{\rm c}^T Y\| = \|X^T Y_{\rm c}\| = \|XX^T - YY^T\|.$$

The next lemma gives perturbation bound for the eigenvalues of definite matrix pair.

Lemma 2. *[Theorem 8.7.3] [133] Let A, B be symmetric, B be positive definite. Let the eigenvalues of (A,B) be $\lambda_1 \geq \cdots \geq \lambda_n$. Let $c(A,B)$ be the Crawford number of $\{A,B\}$:*

$$c(A,B) \equiv \min_{x\in\mathbb{C}^n, \|x\|=1} |x^T(A+iB)x|.$$

Suppose E and F are symmetric matrices that satisfy

$$\epsilon^2 = \|E\|^2 + \|F\|^2 < c^2(A,B).$$

Then $B+F$ is positive definite, and the eigenvalues $\tilde{\lambda}_1 \geq \ldots \tilde{\lambda}_n$ of $(A+E, B+F)$ satisfy

$$|\arctan\tilde{\lambda}_i - \arctan\lambda_i| \leq \arctan\frac{\epsilon}{c(A,B)},\ \forall 1 \leq i \leq n.$$

The following lemma gives perturbation bound for the eigenspace of definite matrix pair, which is rewritten from [Theorem 2.1] [127].

Lemma 3. *Let A, B, $\widetilde{A}$, $\widetilde{B}$ be symmetric, B and $\widetilde{B}$ be positive definite. Let the eigenvalues of (A,B) and $(\widetilde{A},\widetilde{B})$ be $\tan\mu_1 \geq \cdots \geq \tan\mu_n$, $\tan\tilde{\mu}_1 \geq \cdots \geq \tan\tilde{\mu}_n$, respectively, the*

corresponding eigenvectors be $v_1, \ldots, v_n$, *and* $\tilde{v}_1, \ldots, \tilde{v}_n$, *respectively. Assume that there are* $\alpha \geq 0$ *and* $\delta > 0$ *satisfying* $\alpha + \delta \leq 1$, *and a real number* γ *such that*

$$|\sin(\gamma - \mu_i)| \leq \alpha, \text{ for } i = 1, \ldots, p,$$
$$|\sin(\gamma - \tilde{\mu}_j)| \geq \alpha + \delta, \text{ for } j = p+1, \ldots, n$$

(or vice-versa). Let $V_1 = [v_1, \ldots, v_p]$, $\widetilde{V}_1 = [\tilde{v}_1, \ldots, \tilde{v}_p]$. *Then*

$$\|\sin\Theta(V_1, \widetilde{V}_1)\| \leq \frac{p(\alpha, \delta; \gamma)\sqrt{\|A^2 + B^2\|}}{c(A, B)c(\widetilde{A}, \widetilde{B})} \times \frac{\sqrt{\|(\widetilde{A} - A)^2 + (\widetilde{B} - B)^2\|}}{\delta},$$

where

$$p(\alpha, \delta; \gamma) \triangleq \frac{q(\gamma)(\alpha + \delta)\sqrt{1 - \alpha^2} + \alpha\sqrt{1 - (\alpha + \delta)^2}}{2\alpha + \delta},$$

with $q(\gamma) = \sqrt{2}$ *for* $\gamma \neq 0$ *and* $q(0) = 1$.

Now we are ready to prove **Theorem 1**.

Proof of Theorem 1. Denote $A_k = A(\mathbf{P}_k)$, $B_k = B(\mathbf{P}_k)$, $E_k = A_k - A_{k-1}$, $F_k = B_k - B_{k-1}$. Without loss of generality, we assume that A_k is also positive definite. Otherwise, we let $A_k = A_k + tB_k$ for sufficiently large t, then A_k is positive definite and the sequence $\{\mathbf{V}_k\}$ produced by SCF iteration remains unchanged. Let $\Lambda_k = \text{diag}(\lambda_{1,k}, \ldots, \lambda_{p,k})$, $\mathbf{V}_k = [v_{1,k}, \ldots, v_{p,k}]$, where $\lambda_{i,k}$ is the i^{th} largest eigenvalue of (A_k, B_k), $v_{i,k}$ is the corresponding eigenvector. Also denote $s_k = \|\sin\Theta(\mathbf{P}_k, \mathbf{P}_{k-1})\| = \|\mathbf{P}_k\mathbf{P}_k^T - \mathbf{P}_{k-1}\mathbf{P}_{k-1}^T\|$ as the distance between subspaces.

By assumption **A1**, we have

$$\begin{aligned}&\sqrt{\|A_k - A_{k-1}\|^2 + \|B_k - B_{k-1}\|^2}\\ \leq &\sqrt{\xi_a^2 + \xi_b^2}\|\sin\Theta(\mathbf{P}_k, \mathbf{P}_{k-1})\| = s_k\sqrt{\xi_a^2 + \xi_b^2}.\end{aligned}$$

Now consider $k = 1$. By assumption, $s_1\sqrt{\xi_a^2 + \xi_b^2} < c$, then we may apply Lemma 2,

which gives

$$\begin{aligned}|\mu_{i,1}-\mu_{i,0}| \leq & \arctan \frac{\sqrt{\|A_1-A_0\|^2+\|B_1-B_0\|^2}}{c} \\ \leq & \arctan(s_1\sqrt{\xi_a^2+\xi_b^2}/c),\ \forall 1 \leq i \leq n.\end{aligned}$$

It follows that

$$\begin{aligned}\mu_{p,1}-\mu_{p+1,0} =& \mu_{p,1}-\mu_{p+1,1}+\mu_{p+1,1}-\mu_{p+1,0} \\ \geq & \eta - \arctan(s_1\sqrt{\xi_a^2+\xi_b^2}/c) \\ > & \arcsin(\rho C\sqrt{\xi_a^2+\xi_b^2}/c^2) \geq 0, \end{aligned} \tag{2.9}$$

where the last inequality uses the assumption

$$\eta > \arcsin(\rho C\sqrt{\xi_a^2+\xi_b^2}/c^2) + \arctan(s_1\sqrt{\xi_a^2+\xi_b^2}/c).$$

Now let

$$\gamma = \frac{\mu_{1,1}+\mu_{p,1}}{2},\ \alpha = \sin\frac{\mu_{1,1}-\mu_{p,1}}{2},\ \alpha+\delta = \sin(\frac{\mu_{1,1}+\mu_{p,1}}{2}-\mu_{p+1,0}),$$

then for all $1 \leq i \leq p$ and $p+1 \leq j \leq n$, we have

$$|\sin(\gamma-\mu_{i,1})| \leq |\sin\frac{\mu_{1,1}-\mu_{p,1}}{2}| = \alpha, \tag{2.10a}$$

$$|\sin(\gamma-\mu_{j,0})| \geq |\sin(\frac{\mu_{1,1}+\mu_{p,1}}{2}-\mu_{p+1,0})| = \alpha+\delta, \tag{2.10b}$$

$$\alpha+\delta \leq 1, \qquad \gamma > 0, \tag{2.10c}$$

$$\begin{aligned}\delta =& \sin(\frac{\mu_{1,1}+\mu_{p,1}}{2}-\mu_{p+1,0}) - \sin\frac{\mu_{1,1}-\mu_{p,1}}{2} \\ =& 2\cos\frac{\mu_{1,1}-\mu_{p+1,0}}{2}\sin\frac{\mu_{p,1}-\mu_{p+1,0}}{2}. \end{aligned} \tag{2.10d}$$

By calculations, we obtain

$$\begin{aligned} p(\alpha,\delta;\gamma) &= \frac{(\alpha+\delta)\sqrt{1-\alpha^2}+\alpha\sqrt{1-(\alpha+\delta)^2}}{2\alpha+\delta} \\ &= \frac{\sin\frac{\mu_{1,1}-\mu_{p,1}}{2}\cos(\frac{\mu_{1,1}+\mu_{p,1}}{2}-\mu_{p+1,0})+\cos\frac{\mu_{1,1}-\mu_{p,1}}{2}\sin(\frac{\mu_{1,1}+\mu_{p,1}}{2}-\mu_{p+1,0})}{\sin\frac{\mu_{1,1}-\mu_{p,1}}{2}+\sin(\frac{\mu_{1,1}+\mu_{p,1}}{2}-\mu_{p+1,0})} \\ &= \frac{\sin(\mu_{1,1}-\mu_{p+1,0})}{\sin\frac{\mu_{1,1}-\mu_{p,1}}{2}+\sin(\frac{\mu_{1,1}+\mu_{p,1}}{2}-\mu_{p+1,0})} \\ &= \frac{2\sin\frac{\mu_{1,1}-\mu_{p+1,0}}{2}\cos\frac{\mu_{1,1}-\mu_{p+1,0}}{2}}{2\sin\frac{\mu_{1,1}-\mu_{p+1,0}}{2}\cos\frac{\mu_{p,1}-\mu_{p+1,0}}{2}} \\ &= \frac{\cos\frac{\mu_{1,1}-\mu_{p+1,0}}{2}}{\cos\frac{\mu_{p,1}-\mu_{p+1,0}}{2}}. \end{aligned} \tag{2.11}$$

Using Lemma 3, we have

$$s_2 \le \frac{p(\alpha,\delta;\gamma)C}{c^2}\cdot\frac{\sqrt{\|(A_1-A_0)^2+(B_1-B_0)^2\|}}{\delta} \le \frac{p(\alpha,\delta;\gamma)C}{c^2}\cdot\frac{\sqrt{\xi_a^2+\xi_b^2}}{\delta}s_1. \tag{2.12}$$

Substituting (2.10d), (2.11) into (2.12), and using (2.9), we have

$$s_2 \le \frac{1}{\rho}s_1. \tag{2.13}$$

where

$$\rho = \frac{c^2\sin(\mu_{p,1}-\mu_{p+1,0})}{C\sqrt{\xi_a^2+\xi_b^2}} > 1. \tag{2.14}$$

For general $k=2$, noticing the following holds

$$\begin{aligned} \eta &> \arcsin(\rho C\sqrt{\xi_a^2+\xi_b^2}/c^2)+\arctan(s_1\sqrt{\xi_a^2+\xi_b^2}/c) \\ &> \arcsin(\rho C\sqrt{\xi_a^2+\xi_b^2}/c^2)+\arctan(s_2\sqrt{\xi_a^2+\xi_b^2}/c). \end{aligned}$$

Similar to the proof for $k=1$, we can conclude $s_3 \le \frac{1}{\rho}s_2$. By induction, $s_{k+1} \le \frac{1}{\rho}s_k$, thus completing the proof. □

Proof of Theorem 2. (Local Convergence of SCF) With relaxed assumptions, we can show local convergence if the initial subspace is close enough to the true subspace $\mathbf{P}*$:

A3: Let μ_i^* denote $\arctan \lambda_i(A(\mathbf{P}^*), B(\mathbf{P}^*))$. There exists an $\eta > 0$ such that

$$\mu_p^* - \mu_{p+1}^* \geq \eta.$$

Theorem 2. ***(Local Convergence)*** *Let* $\hat{s}_0 = \|\sin\Theta(\boldsymbol{P}_0, \boldsymbol{P}^*)\|$. *Assume* ***A1*** *and* ***A3****, and* $\hat{s}_0\sqrt{\xi_a^2 + \xi_b^2} < c$. *If*

$$\eta > \arcsin(\rho C\sqrt{\xi_a^2 + \xi_b^2}/c^2) + \arctan(\hat{s}_0\sqrt{\xi_a^2 + \xi_b^2}/c)$$

for some constant $\rho > 1$*, then SCF is locally convergent at* $\boldsymbol{P}^*$ *at the rate of* $\frac{1}{\rho}$.

Proof. By assumption **A1**, we have

$$\sqrt{\|A_0 - A^*\|^2 + \|B_0 - B^*\|^2} \leq \sqrt{\xi_a^2 + \xi_b^2}\|\sin\Theta(\mathbf{P}_0, \mathbf{P}^*)\| = \hat{s}_0\sqrt{\xi_a^2 + \xi_b^2}.$$

Applying Lemma 2, we have

$$\begin{aligned} |\mu_{p,0} - \mu_p^*| &\leq \arctan\frac{\sqrt{\|A_0 - A^*\|^2 + \|B_0 - B^*\|^2}}{c} \\ &\leq \arctan(\hat{s}_0\sqrt{\xi_a^2 + \xi_b^2}/c),\ \forall 1 \leq p \leq n. \end{aligned}$$

By assumption **A3** it follows that

$$\begin{aligned} \mu_{p,0} - \mu_{p+1}^* &= \mu_{p,0} - \mu_p^* + \mu_p^* - \mu_{p+1}^* \geq \eta - \arctan(\hat{s}_0\sqrt{\xi_a^2 + \xi_b^2}/c) \\ &> \arcsin(\rho C\sqrt{\xi_a^2 + \xi_b^2}/c^2) \geq 0. \end{aligned} \tag{2.15}$$

Following the same procedures as in the proof for **Theorem 1**, we arrive that

$$\|\sin\Theta(\mathbf{P}_{k-1}, \mathbf{P}^*)\| \leq \frac{1}{\rho}\|\sin\Theta(\mathbf{P}_k, \mathbf{P}^*)\|,$$

where $\rho = \frac{c^2\sin(\mu_{p,0} - \mu_{p+1}^*)}{C\sqrt{\xi_a^2 + \xi_b^2}}$. □

Proof of Corollary 1. (Convergence of WDA-eig)

Corollary 1. *Suppose that the optimal transport matrix* $\boldsymbol{T}^{c,c'}$ *satisfies a Lipschitz-like condition:*

$$\|\boldsymbol{T}^{c,c'}(\boldsymbol{P}_1) - \boldsymbol{T}^{c,c'}(\boldsymbol{P}_2)\| \leq \xi^{c,c'}\|\sin\Theta(\boldsymbol{P}_1, \boldsymbol{P}_2)\|,$$

For a given p*, let*

$$\eta = \min_k\{\mu_{p,k} - \mu_{p+1,k}\}.$$

Denote $\xi_a = \sum_{c,c'>c}\xi^{c,c'}\|\sum_{i,j}(x_i^c - x_j^{c'})(x_i^c - x_j^{c'})^T\|$, $\xi_b = \sum_c \xi^{c,c}\|\sum_{i,j}(x_i^c - x_j^c)(x_i^c - x_j^c)^T\|$. *If*

$$\eta > \arcsin(\rho C\sqrt{\xi_a^2 + \xi_b^2}/c^2) + \arctan(s_1\sqrt{\xi_a^2 + \xi_b^2}/c)$$

for some constant $\rho > 1$*, then* ***wda-eig*** *converges linearly at the rate* $\frac{1}{\rho}$*.*

Proof. We first note that in **WDA-eig**, $\mathbf{C}_b$ and $\mathbf{C}_w$ are invariant to orthogonal transformation of $\mathbf{P}$, i.e., $\mathbf{C}_b(\mathbf{P}) \equiv \mathbf{C}_b(\mathbf{P}Q)$, $\mathbf{C}_w(\mathbf{P}) \equiv \mathbf{C}_w(\mathbf{P}Q)$ for any orthogonal matrix $Q \in \mathbb{R}^{p\times p}$, since

$$\mathbf{C}_b(\mathbf{P}) = \sum_{c,c'>c}\sum_{i,j} t_{ij}^{c,c'}(\mathbf{P})(x_i^c - x_j^{c'})(x_i^c - x_j^{c'})^T,$$
$$\mathbf{C}_w(\mathbf{P}) = \sum_c \sum_{i,j} t_{ij}^{c,c}(\mathbf{P})(x_i^c - x_j^c)(x_i^c - x_j^c)^T,$$

and $\mathbf{T}^{c,c'}(\mathbf{P})$ and $\mathbf{T}^c(\mathbf{P})$ are invariant to orthogonal transformation of $\mathbf{P}$:

$$\begin{aligned}
\mathbf{T}^{c,c'}(\mathbf{P}Q) &\triangleq \underset{\mathbf{T}\in U_{n_c n_{c'}}}{\arg\min}\, \lambda\langle \mathbf{T}, \mathbf{M}_{X^c\mathbf{P}Q, X^{c'}\mathbf{P}Q}\rangle + \sum_{i,j} t_{ij}\log(t_{ij}) \\
&= \underset{\mathbf{T}\in U_{n_c n_{c'}}}{\arg\min}\, \lambda\sum_{i,j} t_{ij}\|(x_i^c - x_j^{c'})^T\mathbf{P}Q\| + \sum_{i,j} t_{ij}\log(t_{ij}) \\
&= \underset{\mathbf{T}\in U_{n_c n_{c'}}}{\arg\min}\, \lambda\sum_{i,j} t_{ij}\|(x_i^c - x_j^{c'})^T\mathbf{P}\| + \sum_{i,j} t_{ij}\log(t_{ij}) = \mathbf{T}^{c,c'}(\mathbf{P}).
\end{aligned}$$

By the assumption on $\mathbf{T}^{c,c'}$, $\mathbf{C}_b$ satisfies

$$\begin{aligned}\|\mathbf{C}_b(\mathbf{P}_1) - \mathbf{C}_b(\mathbf{P}_2)\| &= \| \sum_{c,c'>c} \sum_{i,j} (t_{ij}^{c,c'}(\mathbf{P}_1) - t_{ij}^{c,c'}(\mathbf{P}_2))(x_i^c - x_j^{c'})(x_i^c - x_j^{c'})^T)\| \\ &\leq \sum_{c,c'>c} \max_{i,j} |t_{ij}^{c,c'}(\mathbf{P}_1) - t_{ij}^{c,c'}(\mathbf{P}_2)| \| \sum_{i,j} (x_i^c - x_j^{c'})(x_i^c - x_j^{c'})^T\| \\ &\leq \sum_{c,c'>c} \xi^{c,c'} \| \sin\Theta(\mathbf{P}_1, \mathbf{P}_2)\| \| \sum_{i,j} (x_i^c - x_j^{c'})(x_i^c - x_j^{c'})^T\| \\ &\triangleq \xi_a \| \sin\Theta(\mathbf{P}_1, \mathbf{P}_2)\|.\end{aligned}$$

The last inequality holds since

$$\max_{i,j} |t_{ij}^{c,c'}(\mathbf{P}_1) - t_{ij}^{c,c'}(\mathbf{P}_2)| \leq \|\mathbf{T}^{c,c'}(\mathbf{P}_1) - \mathbf{T}^{c,c'}(\mathbf{P}_2)\| \leq \xi^{c,c'} \| \sin\Theta(\mathbf{P}_1, \mathbf{P}_2)\|.$$

Similarly,

$$\|\mathbf{C}_w(\mathbf{P}_1) - \mathbf{C}_w(\mathbf{P}_2)\| \leq \sum_c \xi^{c,c} \| \sum_{i,j} (x_i^c - x_j^c)(x_i^c - x_j^c)^T\| \| \sin\Theta(\mathbf{P}_1, \mathbf{P}_2)\| \triangleq \xi_b \| \sin\Theta(\mathbf{P}_1, \mathbf{P}_2)\|.$$

For all iteration number k, $\mu_{p,k} - \mu_{p+1,k} \geq \eta$. Since **A1** and **A2** are satisfied, the result follows directly from **Theorem 1**. □

Chapter 3

TRUNCATED ORTHOGONAL ITERATION

3.1 Introduction

Sparse eigenvector problems arise in many applications where localized and structured eigenvectors are desired, such as sparse principal component analysis (PCA), sparse dictionary learning and densest $k-$subgraphs recovery. The goal of sparse coding/dictionary learning is to represent the input signal as a sparse linear combination of the dictionary elements. The densest $k-$subgraph can also be formulated as a sparse eigenvector problem [149] to find a set of k vertices with maximum average degree in the subgraph induced by the set.

Among these applications, one of the most widely used and well-studied methods is sparse PCA. Classical PCA [58, 68, 104] is a fundamental dimensionality reduction technique that finds principal directions of data variance, and can be computed via the eigendecomposition on the sample covariance matrix. Unfortunately, in high-dimensional settings, classical PCA suffers from inconsistency [67] and poor interpretability [35]. Inconsistency means that the estimated principal components of the sample covariance matrix deviate from the true principal component of the population covariance matrix, and this occurs when the number of features grows faster than the number of samples with a constant signal-to-noise ratio. When the number of features is too large, it is also hard to interpret each principal component obtained by classical PCA because it is a linear combination of all the original features. Sparse PCA aims to solve these two issues by selecting a relatively small subset of the original features. Over the last two decades many algorithms and theory for sparse PCA have been proposed to mitigate these issues, see e.g. the survey [156]. In this paper we analyze the sparse eigenvector problem in general, and then focus on the sparse PCA application.

To formalize the sparse eigenvector problem, we assume that the leading eigenvectors

corresponding to the largest m eigenvalues of a positive semidefinite matrix $\bar{A}$ are sparse, i.e.

$$\bar{P} = \underset{P^T P = I_m}{\arg\max} \operatorname{Tr}(P^T \bar{A} P), \quad \bar{P} \text{ is sparse.} \tag{3.1}$$

Here, *sparse* means that each column of $\bar{P}$ has a lot of entries that are either exactly zero or sufficiently close to zero. In practice, $\bar{A}$ is often unknown and we are given a perturbed positive semidefinite matrix A: $A = \bar{A} + E$. Our goal is to recover the true eigenvectors $\bar{P}$ from A.

A straightforward formulation for the sparse eigenvector problem given A is as follows:

$$\bar{Q} = \underset{Q^T Q = I_m}{\arg\max} \operatorname{Tr}(Q^T A Q), \quad \text{subject to } \|q_i\|_0 \leq k_i,\ i = 1, \ldots, m, \tag{3.2}$$

where q_i is the i^{th} column vector of Q and $\|\cdot\|_0$ denotes the ℓ_0 "norm" which is the number of nonzeros in a vector. To find a solution to eq. (3.2) is an NP-hard problem. Moreover, the solution for eq. (3.2) is not a good approximation for eq. (3.1) unless some additional assumptions are imposed on E, e.g. when $E = \sigma^2 I$. In this work we do not impose additional assumptions on A and do not aim to solve eq. (3.2) directly. We focus instead on the orthogonal iteration for solving the standard eigenvalue problems, and present two algorithms. In the first algorithm, we relax the sparsity constraint, while in the second, we relax the orthogonality constraint. We analyze whether truncation at each iteration yields a better approximation to the true eigenvectors than those of the standard orthogonal iteration.

Many existing algorithms for sparse eigenvector recovery focus on recovering a single eigenvector and then using a deflation scheme to generalize to multiple components [34, 69, 91, 116, 149, 155]. The downside of this approach is that deflation adds extra perturbation error to the original problem, with estimates of latter components accumulating errors from each deflation, as observed and analyzed in the first numerical example in section 3.5. When the leading eigenvalues have multiplicity larger than 1, it is also difficult to identify the corresponding eigenvectors, and in such case a unique subspace spanned by these eigenvectors is preferred over individual eigenvectors [133, 137]. To avoid these issues, we use the orthogonal iteration, which is a block generalization of the power method and outputs an orthonormal basis of the subspace spanned by the leading eigenvectors.

Another line of relevant work focuses on finding a row-sparse principal subspace, assuming multiple eigenvectors share the same sparsity pattern (or "support set") [129,138,140]. This is equivalent to first selecting a sparse subset of features (an NP-hard problem [129]) and then applying PCA. A practical limitation of the row-sparse formulation is that we cannot always assume the same support set across all leading eigenvectors of interest. For example, in face recognition [64], brain image diagnosing [36], and natural language processing [152], the goal is often to find different localized and interpretable patterns in different eigenvectors. Here, we consider the general problem of column-sparse subspace estimation, without assuming a common support set.

Contributions. We propose a general framework for estimating sparse eigenvectors and differentiate between the deflation scheme and block scheme. We develop two new algorithms: TOrth and TOrthT, based on the orthogonal iteration to obtain several leading eigenvectors with sparsity constraints, and also propose a new sparsity parameter tuning scheme. We provide a deterministic convergence analysis for methods within the block framework, without additional assumptions on the matrix A, and extend the analysis to other sparse eigenvector algorithms. For application, we demonstrate the accuracy and efficiency of the proposed algorithms by applying them to simulated and real-world datasets, including the PitProps, Sea Surface Temperature, and 20 Newsgroup datasets. Furthermore, we extend TOrth to improve K-subspace clustering and show that it is more robust to noise and achieves better clustering accuracy.

Notation. Let $\mathbb{S}^p = \{A \in \mathbb{R}^{p\times p} | A = A^T\}$ denote the set of symmetric matrices. For any $A \in \mathbb{S}^p$, we denote its eigenvalues by $\lambda_{\min}(A) = \lambda_p(A) \leq \cdots \leq \lambda_1(A) = \lambda_{\max}(A)$. We use $\rho(A)$ and $\|A\|_2$ to denote the spectral norm of A, which is $\max\{|\lambda_{\max}(A)|, |\lambda_{\min}(A)|\}$. For vectors, $\|\cdot\|_2$ or $\|\cdot\|$ denote the 2-norm, and $\|\cdot\|_0$ denotes the ℓ_0 "norm" which is the number of nonzeros in a vector. Let $\lambda_{\max}(A, k) = \max_{x\in\mathbb{R}^p} x^T Ax$, such that $\|x\| = 1$, $\|x\|_0 \leq k$. Define $\rho(E, k) := \sqrt{\lambda_{\max}(E^T E, k)}$. For a squared matrix A, we let $\mathrm{Tr}(A) = \sum_i a_{ii}$ be its trace. We use $\|\cdot\|_F$ to denote the Frobenius norm, and $\|A\|_F = \sqrt{\sum_i \sum_j a_{ij}^2} = \sqrt{\mathrm{Tr}(A^T A)}$. For two matrices A, B of the same size, we use $\langle A, B\rangle = \mathrm{Tr}(A^T B) = \sum_{i,j} a_{ij} b_{ij}$ to denote their

matrix inner product.

3.2 Methodology

Many methods for finding an approximate solution to eq. (3.2) use a power iteration-like scheme, either using deflation to obtain the leading vectors sequentially [34, 149], or using block approaches to compute several vectors at once [40, 71, 90]. The deflation approach works best if the eigenvectors of interest correspond to eigenvalues that are all separated by large gaps compared to the remaining eigenvalues. If several leading eigenvectors are desired and their corresponding eigenvalues are clustered, the block approach is preferable since it recovers a principal subspace instead of identifying individual eigenvectors separately.

For the single vector case, power iteration-based methods iterate the following steps:

1. Update the current vector: $\tilde{v}_{t+1} = Av_t$.

2. Truncate or threshold the vector based on some penalty, usually ℓ_1 or ℓ_0.

3. Re-normalize the vector: $v_{t+1} = \frac{\hat{v}_{t+1}}{\|\hat{v}_{t+1}\|}$.

To recover several eigenvectors at once, one natural extension of the truncated power method is the truncated orthogonal iteration, see e.g. ITSPCA [90]. However, it cannot enforce orthogonality and sparsity at the same time. Performing an orthogonalization step (by computing a QR factorization or a singular value decomposition) is crucial to the algorithm stability, but destroys the sparsity pattern. On the other hand, performing truncation afterwards gives a sparse solution, but loses orthogonality. We propose the following framework based on the orthogonal iteration, where a post-processing step may be used to enforce sparsity.

1. Update the current vectors: $Q'_{t+1} = AQ_t$.

2. Truncate or threshold (usually process each column of Q'_{t+1} separately).

3. Re-orthogonalize: QR: $Q_{t+1} = \mathbf{qr}(Q'_{t+1})$. SVD: $U, S, V = \mathbf{svd}(Q'_{t+1})$, $Q_{t+1} = UV^T$.

4. (Optional) Post-processing: in each iteration, truncate or threshold Q_{t+1}.

We propose two variations of the Truncated Orthogonal Iteration in this framework. The first approach, formalized in algorithm 3, is similar to the ITSPCA algorithm [90], but with key differences in implementation and analysis. First, we use a direct truncation scheme induced by ℓ_0 constraint, where we keep the largest k entries and truncate the rest of the entries to zero for each column of the obtained matrix. More specifically, for a column vector x, let $F = \text{supp}(x, k)$ be the indices of x with the largest k absolute values. The procedure *Truncate* is defined as:

$$[\text{Truncate}(x, F)]_i = \begin{cases} [x]_i, & \text{if } i \in F \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

For a matrix Q, we define Truncate$(Q, \mathcal{F})$ as the matrix obtained by truncating each column of Q, where $\mathcal{F} = \{F_i\}$ and F_i is the indices of q_i with the largest k absolute values. Whereas thresholding techniques can be more general and require estimating parameters associated with sparsity, for example, the commonly-used soft thresholding [155] derived from the proximal operator of ℓ_1 norm:

$$[\text{Threshold}(x, F)]_i = \begin{cases} [x]_i - t \text{ sign}([x]_i), & \text{if } |[x]_i| > t \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

Second, we give a deterministic numerical analysis in theorem 3, while for ITSPCA, [20, 90] established statistical convergence analyses under the spiked covariance model [66] and did not analyze the numerical convergence of the algorithm. Third, we use a different initialization scheme, i.e. warm initialization as discussed in Section 3.5, as opposed to the "diagonal thresholding" initialization [67], which also relies on the spiked covariance model and requires extra parameters. Our second approach, formalized in algorithm 4, uses a greedy approach to get sparse vectors after each iteration of algorithm 3.

We construct a test problem that is used throughout the paper to illustrate the efficiency and accuracy of the proposed algorithms:

Test Problem Setup: $\bar{A} = \bar{V}\Lambda\bar{V}^T \in \mathbb{R}^{100\times 100}$ with the first three columns of $\bar{V}$, denoted as V, being sparse orthonormal vectors, and the remaining columns of $\bar{V}$ are randomly generated orthonormal vectors such that $\bar{V}^T\bar{V} = I$. The eigenvalues are set to be $\lambda_1 = 1$, $\lambda_2 = 0.9$, $\lambda_3 = 0.8$, $\lambda_j = 0.1$ for $j = 4, \cdots, 100$. E is a positive semidefinite matrix, $A = \bar{A} + E$ and $\rho(E) \approx 0.21$. We aim to recover the eigenvectors corresponding to the largest 3 eigenvalues. We construct three simple scenarios: a) the leading eigenvectors that we are trying to recover have completely overlapping nonzero indices; b) the leading eigenvectors have partially overlapping nonzero indices; and c) the indices are non-overlapping. In all three cases, the true sparsity level $[\bar{k}_1, \bar{k}_2, \bar{k}_3] = [10, 10, 10]$. We keep $K = [p, p, p] = [100, 100, 100]$ in the first 20 iterations, $K = [50, 50, 50]$ in iteration $21 - 40$, $K = [25, 25, 25]$ in iteration $41 - 60$, and $K = [10, 10, 10]$ in iteration $61 - 80$.

Initialization and Tuning the Cardinality Parameter K. We use a refined warm initialization strategy based on the one specified in [149]. Unless the cardinality parameter K is pre-specified, we start with no sparsity constraint and gradually tighten the cardinality parameter K. Once the subspace converges for K, we proceed to the next truncation level with cardinality parameter $K/2$ and keep track of the distance $\|\sin\Theta(Q_t, Q_{t-1})\|_F^2$. Specifically, we check the distance between Q_t, the matrix obtained at the beginning of a new truncation level, and Q_{t-1}, the matrix obtained at the end of the old truncation level, and compare it to a user-defined threshold ϵ_2. We want to restrict the truncation error such that it does not exceed the perturbation error when no truncation is occurred. A heuristic is to choose ϵ_2 on the order of $\rho(E)$, based on [Theorem 8.1.10] [133] and theorem 3. If K is too small, $\|\sin\Theta(Q_t, Q_{t-1})\|_F^2$ is likely to have a big jump, indicating that an over-truncation has occurred, as illustrated in fig. 3.1. In such case, we return to the previous level K by setting $Q_t = Q_{t-1}$ and terminate the algorithm.

Convergence Criterion. Since we are interested in not only recovering the eigenspace but also recovering the exact eigenvectors, the convergence criterion for the proposed algo-

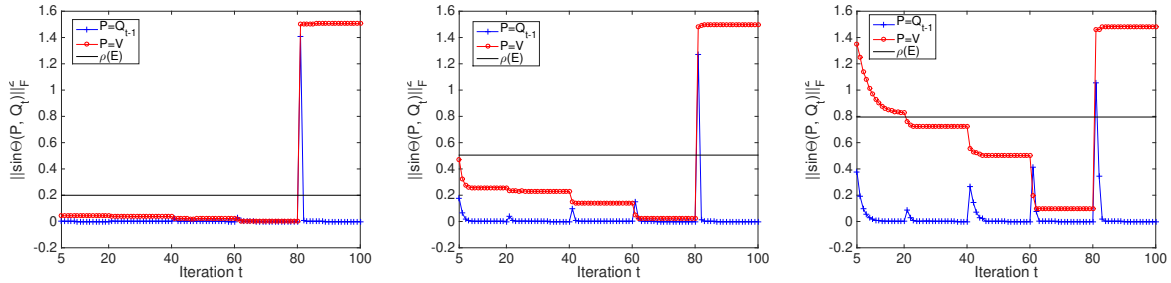


Figure 3.1: **Effect of gradually decreasing the truncation level.** To measure the truncation effect on accuracy, we keep track of the distance between the output matrix and the true eigenvectors (Q_t, V), shown as the red line. To estimate when to truncate to the next level when the true eigenvectors are unknown, we also keep track of the distance between consecutive updates (Q_t, Q_{t-1}), shown as the blue line. We use the test problem (a) with varying noise magnitude, i.e. $\rho(E) = 0.2,\ 0.5,\ 0.8$, from left to right. The truncation level is decreased approximately by half every 20 iterations until iteration 80. At iteration 81, the truncation level is further decreased to a half of the true sparsity level, and we observe a huge spike in both lines, indicating that an over-truncation has occurred.

rithm is based on the size of $\|Q_t - Q_{t-1}\|_2$. Unless specified, the default threshold ϵ is set to 10^{-4}, and the algorithm is forced to terminate after a maximum of 200 iterations.

Complexity Analysis. For sparse eigenvector problems, we are given a positive semidefinite matrix $A \in \mathbb{S}^p$. For sparse PCA problems, suppose that we are given a data matrix $X \in \mathbb{R}^{n\times p}$, and its covariance matrix is calculated by $\Sigma = X^T X \in \mathbb{S}^p$. In high-dimensional settings, $p \gg n \geq m$. In each iteration, matrix-matrix multiplication is $\mathcal{O}(npm)$. Sorting m vectors of length p in order to identify the largest entries is $\mathcal{O}(mp \log p)$. Since the dimension of matrix $\hat{P}_t$ is $p \times m$, QR factorization is $\mathcal{O}(pm^2)$. Using the Householder algorithm, the MATLAB **qr** function takes $2m^2p - (2/3)m^3$ flops. Compared to the single vector case, QR factorization is more expensive than normalizing m single vectors (requiring $\mathcal{O}(mp)$ operations), but the deflation steps can be avoided, thus saving $\mathcal{O}(npm)$ operations.

Algorithm 3 Truncated Orthogonal Iteration (TOrth)

Input: Symmetric positive semidefinite matrix $A \in \mathbb{S}^p$, initial vectors $Q_0 \in \mathbb{R}^{p \times m}$

Output: An orthogonal matrix Q_t

Parameters: Cardinalities for each column vector $K = [k_1, \cdots, k_m]$

$t = 0$

while $Distance < \epsilon_1$ **do**

  $k \leftarrow k/2$ unless otherwise specified

  $counter = 0$

  while $Distance < \epsilon_2$ or $counter < MaxIter$ **do**

    Compute $P_t = AQ_{t-1}$. Denote the i^{th} column of P_t as p_i.

    Construct $\hat{P}_t$ with column $\hat{p}_i = \text{Truncate}(p_i, F_i)$.

    Reorthogonalize $Q_t = \mathbf{qr}(\hat{P}_t)$.

    if $counter = 0$ and $t \neq 0$ and $Distance > \epsilon_1$ **then**

      $Q_t = Q_{t-1}$

      Break

    end if

    $counter \leftarrow counter + 1$

    $t \leftarrow t + 1$

  end while

end while

Algorithm 4 Truncated Orthogonal Iteration with Post Truncation (TOrthT)

Input: Symmetric positive semidefinite matrix $A \in \mathbb{S}^p$, initial vector $Q_0 \in \mathbb{R}^{p\times m}$, cardinality vector $K \in \mathbb{R}^m$

Output: A sparse and near-orthogonal matrix Q_t

while $Distance < \epsilon_1$ **do**

$k \leftarrow k/2$ unless otherwise specified

$counter = 0$

while $Distance < \epsilon_2$ or $counter < MaxIter$ **do**

Compute $P_t = AQ_{t-1}$. Denote the i^{th} column of P_t as p_i.

for $i = 1, \cdots, m$ **do**

Let $F_i = \text{supp}(p_i, k_i)$ be the indices of p_i with the largest k_i absolute values.

Compute $\hat{p}_i = \text{Truncate}(p_i, F_i)$.

$\hat{P}_t[:, i] = \hat{p}_i$

end for

Reorthogonalize $\tilde{Q}_t = \mathbf{qr}(P_t)$. Denote the i^{th} column of $\tilde{Q}_t$ as $\tilde{q}_i$.

for $i = 1, \cdots, m$ **do**

Let $F_i = \text{supp}(\tilde{q}_i, k_i)$ be the indices of $\tilde{q}_i$ with the largest k_i absolute values.

Compute $\hat{q}_i = \text{Truncate}(\tilde{q}_i, F_i)$.

Compute $q_i = \frac{\hat{q}_i}{\|\hat{q}_i\|}$.

$Q_t[:, i] = q_i$

end for

if $counter = 0$ and $t \neq 0$ and $Distance > \epsilon_1$ **then**

$Q_t = Q_{t-1}$

Break

end if

$counter \leftarrow counter + 1$

$t \leftarrow t + 1$

end while

end while

3.2.1 Truncated Orthogonal Iteration with Strict Sparsity Constraint

In real scenarios, final output vectors are often required to be sparse for better interpretation. Sparsity constraints are not enforced in TOrth or in ITSPCA [90], since by doing QR decomposition, the m^{th} vector needs to be orthogonalized against all previously obtained vectors $q_1, \cdots, q_{m-1}$. If the previous vectors are partially overlapping, the m^{th} vector is likely to be dense. Therefore we also present algorithm 4 that performs a truncation step on the output matrix obtained by QR, which we denote as Q below. Specifically, we solve the following problem:

$$Q_{\text{truncate}} = \arg\min_{\hat{Q}} \|Q - \hat{Q}\|_F^2, \quad \text{subject to: } \|\hat{q}_i\|_0 \leq k_i, \ \|\hat{q}_i\|_2 = 1. \tag{3.5}$$

The optimal solution to eq. (3.5) can be obtained by truncating and normalizing each column of Q, since

$$\|Q - \hat{Q}\|_F^2 = \|Q\|_F^2 + \|\hat{Q}\|_F^2 - 2\langle Q, \hat{Q}\rangle = 2m - 2\langle Q, \hat{Q}\rangle = 2m - \sum_i q_i^T \hat{q}_i. \tag{3.6}$$

To test the impact of this post-truncation step, we apply algorithm 4 to the three scenarios in the test problem. $\|Q_{\text{truncate}} - Q\|_F^2$ is kept relatively low ($\approx 1e-4$) when the algorithm converges in all scenarios, and we include the result figures in Appendix A4. For the non-overlapping case, the algorithm is able to recover the true support set before truncation. The orthogonality loss of the final output matrix Q_t is also calculated, and $\|I - Q_t^T Q_t\|_F^2 = 1.57e{-}4$ in case a), 1.17e-4 in case b) and 0 in case c). The post-truncation step does not have a significant impact on the quality or orthogonality of the obtained vectors, especially when the true eigenvectors have non-overlapping support sets. We also observe this minimal impact in section 3.5.

3.3 Analysis

In this section, we analyze what happens in each iteration of algorithm 3 when the matrix is perturbed and a truncation step is performed.

3.3.1 Preliminaries.

We use the standard $\sin\Theta$ definition [124, 133] to measure the distance between subspaces:

Definition. Let $\mathcal{X}$ and $\mathcal{Y}$ be two m-dimensional subspaces of $\mathbb{R}^p$. Let the columns of X form an orthonormal basis for $\mathcal{X}$ and the columns of Y form an orthonormal basis for $\mathcal{Y}$. We use $\|\sin\Theta(\mathcal{X},\mathcal{Y})\|_F$ to measure the distance between $\mathcal{X}$ and $\mathcal{Y}$, where

$$\Theta(\mathcal{X},\mathcal{Y}) = \mathrm{diag}(\theta_1(\mathcal{X},\mathcal{Y}),\ldots,\theta_m(\mathcal{X},\mathcal{Y})). \tag{3.7}$$

Here, $\theta_j(\mathcal{X},\mathcal{Y})$'s denote the *canonical angles* between $\mathcal{X}$ and $\mathcal{Y}$ [p. 43] [124], which is defined as

$$0 \le \theta_j(\mathcal{X},\mathcal{Y}) = \arccos\sigma_j \le \frac{\pi}{2} \quad \text{for } 1 \le j \le m, \tag{3.8}$$

where σ_j's are the singular values of X^TY. Note that this definition is independent of which orthonormal bases X and Y are chosen for the spaces $\mathcal{X}$ and $\mathcal{Y}$.

For the ease of notation, we use $\Theta(X,Y) = \Theta(\mathcal{X},\mathcal{Y})$, where X, Y are the orthonormal bases for the subspaces $\mathcal{X}$, $\mathcal{Y}$, respectively. It has been shown [124, 133] that the following relations hold:

$$\|\cos\Theta(X,Y)\|_{ui} = \|X^TY\|_{ui}, \tag{3.9}$$

$$\|\sin\Theta(X,Y)\|_{ui} = \|X^{\perp T}Y\|_{ui} = \|X^TY^\perp\|_{ui}, \tag{3.10}$$

$$\|\sin\Theta(X,Y)\|_2 = \|XX^T - YY^T\|_2, \tag{3.11}$$

$$\|\sin\Theta(X,Y)\|_F = \frac{1}{\sqrt{2}}\|XX^T - YY^T\|_F = \sqrt{p - \|X^TY\|_F^2}. \tag{3.12}$$

where $\|\cdot\|_{ui}$ denotes any unitary invariant norm such as the 2-norm and the Frobenius norm. $X^\perp$ denotes the orthogonal complement of X.

Throughout the paper, we use the following well-known properties of matrix norms:

$$\|A\|_2 \le \|A\|_F \le \mathrm{rank}(A)\|A\|_2, \tag{3.13}$$

$$\|AB\|_2 \le \|A\|_2\|B\|_2, \tag{3.14}$$

$$\|AB\|_F \le \|A\|_F\|B\|_2. \tag{3.15}$$

3.3.2 Convergence Analysis

We now establish our main result and key consequences.

Theorem 3. *Let $\bar{P}$ be the matrix of eigenvectors corresponding to the m largest eigenvalues of $\bar{A}$, with $\lambda_1(\bar{A}) \geq \lambda_2(\bar{A}) \geq \cdots \geq \lambda_m(\bar{A}) > \lambda_{m+1}(\bar{A}) > 0$. Let $A = \bar{A} + E$. Assume $\lambda_1(\bar{A}) = 1$. Define $\gamma := \frac{\lambda_{m+1}}{\lambda_m} < 1$. Let Q_t be the matrix obtained at iteration t by algorithm 3. Then*

$$\|\bar{P}^T Q_t\|_2^2 \geq \frac{\|\bar{P}^T Q_{t-1}\|_F^2}{(1-\gamma^2)\|\bar{P}^T Q_{t-1}\|_F^2 + m\gamma^2} - \delta_E - \delta_{Truncate}, \tag{3.16}$$

where

$$\delta_E = \frac{4\rho(E,K)}{\lambda_m^2(1 - \|\sin\Theta(\bar{P}, Q_{t-1})\|_2^2)}, \quad \delta_{Truncate} = 2m\sqrt{\frac{\min\{\bar{k}_{\max}, p - k_{\min}\}}{p}}.$$

$$\rho(E,K) = \max_{Q^TQ=I_m} \|EQ\|_2 \text{ subject to } \|q_i\|_0 \leq k_i,$$

$$\bar{k}_{\max} = \max_i\{\bar{k}_i\} = \max_i\{\|p_i\|_0\},\ k_{\min} = \min_i\{k_i\}.$$

Assume that $\|P^T Q_t\|_F^2 = c\|P^T Q_t\|_2^2$, where $c \in [1, m]$. Then we have:

$$\|\sin\Theta(\bar{P}, Q_t)\|_F^2 \leq \frac{\gamma^2\|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2 + \frac{m-c}{m}\|\bar{P}^T Q_{t-1}\|_F^2}{1 - (1-\gamma^2)\|\sin\Theta(\bar{P}, Q_{t-1})\|_2^2} + c\delta_E + c\delta_{Truncate}. \tag{3.17}$$

When $c = m$, we have:

$$\|\sin\Theta(\bar{P}, Q_t)\|_F^2 \leq \frac{\gamma^2\|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2}{1 - (1-\gamma^2)\|\sin\Theta(\bar{P}, Q_{t-1})\|_2^2} + m\delta_E + m\delta_{\text{Truncate}}. \tag{3.18}$$

For $m = 1$, the algorithm reduces to the Truncated Power method [149] and the inequality eq. (3.18) holds exactly. We derive a uniform convergence bound for $\|\sin\Theta(\bar{P}, Q_t)\|_F^2$, as stated in the following corollary:

Corollary 2. *Let $\bar{p}$ be the eigenvector corresponding to the largest eigenvalue of $\bar{A}$. $A = \bar{A} + E$. Define $\gamma := \frac{\lambda_2}{\lambda_1} < 1$. Let q_t be the matrix obtained at iteration t by the Truncated Power method. Then*

$$|\sin\angle(\bar{p}, q_t)|^2 \leq \frac{\gamma^2|\sin\angle(\bar{p}, q_{t-1})|^2}{1 - (1-\gamma^2)|\sin\angle(\bar{p}, q_{t-1})|^2} + \delta_E + \delta_{Truncate}.$$

And a uniform bound is given by

$$|\sin\angle(\bar{p}, q_t)| \leq \mu^t |\sin\angle(\bar{p}, q_0)| + \sqrt{\delta_E + \delta_{Truncate}} \frac{1-\mu^t}{1-\mu},$$

where

$$\mu = \frac{\gamma}{\sqrt{1-(1-\gamma^2)|\sin\angle(\bar{p}, q_0)|^2}} \leq 1,\ \delta_E = \frac{4\rho(E,k)}{\lambda_2^2(1-|\sin\angle(\bar{p}, q_0)|^2)},\ \delta_{Truncate} = 2\sqrt{\frac{\min\{\bar{k}, p-k\}}{p}}.$$

For $m > 1$, as $Q_t \to \bar{P}$, $c \to m$ and $\|\sin\Theta(\bar{P}, Q_t)\|_F^2$ converges at the asymptotic rate $\gamma = \frac{\lambda_{m+1}}{\lambda_m}$.

Remark. A natural question that arises is whether truncating the vector only at the last step would give a better result. Besides computational concerns, truncating at each step helps to reduce the perturbation error, which is proportional to $\rho(E,k)$. If we keep $k = p$ at each iteration, there is no truncation error but $\rho(E,k) = \rho(E)$ can be large. On the simulated problem in section 3.5, we observe that the proposed algorithms outperforms the simple truncation strategy (i.e. truncating only at the last step). On the other hand, if we truncate to k nonzeros with $k \approx \bar{k} \ll p$, the truncation error could potentially be large although $\rho(E,k) \approx \rho(E,\bar{k}) \ll \rho(E)$. We recommend keeping k close to p in the first few iterations to avoid truncating the true nonzeros. At later steps, when the nonzero indices of x_t include the nonzero indices of $\bar{x}$, it is safe to truncate to a smaller k without much truncation error and the perturbation error is kept low at the same time. This truncation strategy is illustrated in fig. 3.1.

To prove theorem 3, we need the following lemmas: lemma 4 measures the progress made by each standard orthogonal iteration, lemma 5 accounts for the perturbation error, and lemma 6 analyzes the truncation step.

We first measure the progress made by the standard orthogonal iteration without any truncation or perturbation. In [133] it has been shown that the distance between the t^{th} updated matrix $Q_t \in \mathbb{R}^{p\times m}$ and the matrix of first m eigenvectors $\bar{P}$ converges at a rate $\gamma = |\lambda_{m+1}/\lambda_m|$:

$$\|\sin\Theta(Q_t, \bar{P})\|_2 \leq \gamma^t \frac{\|\sin\Theta(Q_0, \bar{P})\|_2}{\sqrt{1-\|\sin\Theta(Q_0, \bar{P})\|_2^2}}. \tag{3.19}$$

When $m = 1$, define $\theta_t \in [0, \pi/2]$ by $\cos(\theta_t) = |p^T q_t|$ and this reduces to

$$\sin\theta_t \leq \gamma^t \tan\theta_0. \tag{3.20}$$

An equivalent bound measured in Frobenius norm can be derived from [133] (the proof can be found in the Appendix):

$$\|\sin\Theta(Q_t, \bar{P})\|_F \leq \gamma^t \frac{\|\sin\Theta(Q_0, \bar{P})\|_F}{\sqrt{1 - \|\sin\Theta(Q_0, \bar{P})\|_2^2}}. \tag{3.21}$$

A one-step bound can also be derived from [133]:

$$\|\sin\Theta(Q_t, \bar{P})\|_F \leq \gamma \frac{\|\sin\Theta(Q_{t-1}, \bar{P})\|_F}{\sqrt{1 - \|\sin\Theta(Q_{t-1}, \bar{P})\|_2^2}}. \tag{3.22}$$

We provide the following lemma for a similar approximation of the distance update in each iteration:

Lemma 4. *Let $\bar{P}$ be the matrix of eigenvectors corresponding to the largest m (in absolute value) eigenvalues of a symmetric matrix $\bar{A}$, and $\Lambda_m = \mathrm{diag}(\lambda_1, \cdots, \lambda_m)$, and let $\gamma = |\lambda_{m+1}/\lambda_m|$. Given any $Q_{t-1} \in \mathbb{R}^{p\times m}$ such that $Q_{t-1}^T Q_{t-1} = I$, let Q_t be the orthogonal matrix obtained by QR factorization of $\bar{A}Q_{t-1}$, i.e. $Q_t R_t = \bar{A}Q_{t-1}$, then*

$$\|\bar{P}^T Q_t\|_2^2 \geq \frac{\|\bar{P}^T Q_{t-1}\|_F^2}{(1-\gamma^2)\|\bar{P}^T Q_{t-1}\|_F^2 + m\gamma^2}. \tag{3.23}$$

Assume that $\|\bar{P}^T Q_t\|_F^2 = c\|\bar{P}^T Q_t\|_2^2$, where $c \in [1, m]$. Then we have:

$$\|\sin\Theta(\bar{P}, Q_t)\|_F^2 \leq \frac{\gamma^2\|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2 + \frac{m-c}{m}\|\bar{P}^T Q_{t-1}\|_F^2}{1 - (1-\gamma^2)\|\sin\Theta(\bar{P}, Q_{t-1})\|_2^2}. \tag{3.24}$$

Proof. We can decompose Q_{t-1} as $Q_{t-1} = \bar{P}X + \bar{P}^\perp Y$, where $\bar{P}^\perp$ is the orthogonal complement of $\bar{P}$ and its columns are the eigenvectors of $\bar{A}$ corresponding to the $m+1, \cdots p$ eigenvalues, i.e. $\bar{A}\bar{P} = \bar{P}\Lambda_m$, $\bar{A}\bar{P}^\perp = \bar{P}^\perp\Lambda'$, where $\Lambda' = \mathrm{diag}(\lambda_{m+1}, \cdots, \lambda_p)$. We have the following equations:

$$Q_{t-1}^T Q_{t-1} = X^T X + Y^T Y = I \Rightarrow \|X\|_F^2 + \|Y\|_F^2 = m, \tag{3.25}$$

$$\|\bar{P}^T Q_t R_t\|_F^2 = \|\bar{P}^T \bar{A} Q_{t-1}\|_F^2 = \|\bar{P}^T \bar{A}(\bar{P}X + \bar{P}^\perp Y)\|_F^2 = \|\Lambda_m X\|_F^2 \geq \lambda_m^2 \|X\|_F^2. \tag{3.26}$$

Since $\|Q_t R_t\| = \|R_t\| = \|\bar{A} Q_{t-1}\|$, and

$$\|\bar{A} Q_{t-1}\|_F^2 = \|\bar{P} \Lambda_m X + \bar{P}^{\perp} \Lambda' Y\|_F^2 = \|\Lambda_m X\|_F^2 + \|\Lambda' Y\|_F^2, \tag{3.27}$$

We have

$$\begin{aligned}
\|\bar{P}^T Q_t\|_2^2 &\geq \frac{\|\bar{P}^T Q_t R_t\|_F^2}{\|R_t\|_F^2} = \frac{\|\bar{P}^T Q_t R_t\|_F^2}{\|\bar{A} Q_{t-1}\|_F^2} \text{ by eq. (3.15)} && (3.28)\\
&\geq \frac{\|\Lambda_m X\|_F^2}{\|\Lambda_m X\|_F^2 + \|\Lambda' Y\|_F^2} \text{ by eq. (3.26) and eq. (3.27)} && (3.29)\\
&\geq \frac{\lambda_m^2 \|X\|_F^2}{\lambda_m^2 \|X\|_F^2 + \lambda_{m+1}^2 \|Y\|_F^2} \text{ by eq. (3.26)} && (3.30)\\
&= \frac{\lambda_m^2 \|X\|_F^2}{\lambda_m^2 \|X\|_F^2 + \lambda_{m+1}^2 (m - \|X\|_F^2)} \text{ by eq. (3.25)} && (3.31)\\
&= \frac{\|\bar{P}^T Q_{t-1}\|_F^2}{(1-\gamma^2)\|\bar{P}^T Q_{t-1}\|_F^2 + m\gamma^2}. && (3.32)
\end{aligned}$$

Assume that $\|\bar{P}^T Q_t\|_F^2 = c\|\bar{P}^T Q_t\|_2^2$, where $c \in [1, m]$, then we have

$$\begin{aligned}
\|\sin\Theta(\bar{P}, Q_t)\|_F^2 &= m - \|\bar{P}^T Q_t\|_F^2 = m - c\|\bar{P}^T Q_t\|_2^2 && (3.33)\\
&\leq \frac{m\gamma^2 \|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2 + (m-c)\|\bar{P}^T Q_{t-1}\|_F^2}{m - (1-\gamma^2)\|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2} \text{ by eq. (3.32)} && (3.34)\\
&\leq \frac{\gamma^2 \|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2 + \frac{m-c}{m}\|\bar{P}^T Q_{t-1}\|_F^2}{1 - (1-\gamma^2)\|\sin\Theta(\bar{P}, Q_{t-1})\|_2^2} \text{ by eq. (3.13).} && (3.35)
\end{aligned}$$

When $c = m$,

$$\|\sin\Theta(\bar{P}, Q_t)\|_F^2 \leq \frac{\gamma^2 \|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2}{1 - (1-\gamma^2)\|\sin\Theta(\bar{P}, Q_{t-1})\|_2^2}. \tag{3.36}$$

□

lemma 4 measures the progress made by the orthogonalization step, and the QR factorization can be replaced by other factorization methods, as long as the updated matrix Q is orthogonal.

Lemma 5. *Suppose that $A = \bar{A} + E$, where A and $\bar{A}$ are symmetric positive semidefinite matrices. Suppose that $\bar{P}$ is the matrix of eigenvectors of $\bar{A}$ corresponding to the largest m*

eigenvalues. Assume $\lambda_1(\bar{A}) = 1$. *Let* $Q \in \mathbb{R}^{p\times m}$ *be any sparse matrix such that* $Q^TQ = I$, $\|q_i\|_0 \le k_i$. *Then*

$$\frac{\|\bar{P}^T AQ\|_F^2}{\|AQ\|_F^2} \ge \frac{\|\bar{P}^T \bar{A}Q\|_F^2}{\|\bar{A}Q\|_F^2} - \frac{4m\rho(E,K)}{\|\bar{A}Q\|_F^2}, \tag{3.37}$$

where $\rho(E,K) = \max_{Q^TQ=I_m} \|EQ\|_2$ *subject to* $\|q_i\|_0 \le k_i,\ K = [k_1, \cdots, k_m]$.

Proof. Since $A = \bar{A} + E$, using triangle inequality of norms, we have

$$\frac{\|\bar{P}^T AQ\|_F}{\|AQ\|_F} = \frac{\|\bar{P}^T(\bar{A}+E)Q\|_F}{\|(\bar{A}+E)Q\|_F} \ge \frac{\|\bar{P}^T\bar{A}Q\|_F - \|\bar{P}^TEQ\|_F}{\|\bar{A}Q\|_F + \|EQ\|_F} \tag{3.38}$$

$$\ge \frac{\|\bar{P}^T\bar{A}Q\|_F - \|\bar{P}^TEQ\|_F - \|EQ\|_F}{\|\bar{A}Q\|_F} \tag{3.39}$$

$$\ge \frac{\|\bar{P}^T\bar{A}Q\|_F - 2\|EQ\|_F}{\|\bar{A}Q\|_F} \tag{3.40}$$

$$\Rightarrow \frac{\|\bar{P}^T AQ\|_F^2}{\|AQ\|_F^2} \ge \frac{\|\bar{P}^T\bar{A}Q\|_F^2 - 4\|\bar{P}^T\bar{A}Q\|_F\|EQ\|_F + 4\|EQ\|_F^2}{\|\bar{A}Q\|_F^2} \tag{3.41}$$

$$\ge \frac{\|\bar{P}^T\bar{A}Q\|_F^2 - 4\sqrt{m}\|EQ\|_F}{\|\bar{A}Q\|_F^2} \tag{3.42}$$

$$\ge \frac{\|\bar{P}^T\bar{A}Q\|_F^2}{\|\bar{A}Q\|_F^2} - \frac{4m\rho(E,K)}{\|\bar{A}Q\|_F^2}. \tag{3.43}$$

□

Remark. In algorithm 3, $Q_tR_t = \text{Truncate}(AQ_{t-1}, \mathcal{F})$, and in theory Q_t can be dense. If the columns of Truncate(AQ_{t-1}) mostly have nonzeros in non-overlapping sets of rows, the columns of this matrix will be almost orthogonal, and Q_t will be approximately sparse. The first column of Q_t has the same sparsity as the first column of Truncate$(AQ_{t-1}, \mathcal{F})$, but later columns are orthogonalized against more vectors and may become denser. In the spiked covariance matrix model where the matrix A is the empirical covariance matrix of n samples drawn from normal distribution, we have $\rho(E,k_i)^2 = O(k_i \log p/n)$ (shown in the Appendix A2) and

$$\|EQ\|_F^2 = \sum_i \rho(E,k_i)^2 = O(\frac{(\sum_i k_i)\log p}{n}). \tag{3.44}$$

To measure the loss incurred during truncation, we establish the lemma below.

Lemma 6. *Consider a unit vector $\bar{x} \in \mathbb{R}^p$ with support set $supp(\bar{x}) = \bar{F}$, and $\bar{k} = |\bar{F}|$. Consider a vector y with the $k-$largest absolute values with indices in set F. Then*

$$\frac{|\,Truncate(y,F)^T\bar{x}|}{\|\,Truncate(y,F)\|} \geq \frac{|y^T\bar{x}|}{\|y\|} - \sqrt{\frac{\min\{\bar{k}, p-k\}}{p}}. \tag{3.45}$$

Proof. Let $F_1 = \bar{F}\backslash F$, $F_2 = \bar{F} \cap F$, $F_3 = F\backslash\bar{F}$, then

$$\bar{x} = \bar{x}_{F_1} + \bar{x}_{F_2},\ \text{Truncate}(y,F) = y_{F_2} + y_{F_3}.$$

Let $\bar{\alpha} = \|\bar{x}_{F_1}\| \leq 1$, $\alpha = \|y_{F_1}\|$, $k_1 = |F_1|$. Note that if $\bar{F} \subseteq F$, then $F_1 = \emptyset$ and $k_1 = 0$, $\bar{\alpha} = 0$. If $k_1 \neq 0$ we have:

$$\frac{\alpha^2}{k_1} = \frac{\sum_{i\in F_1} y_i^2}{|F_1|} \leq \frac{\|y\|^2}{p}, \tag{3.46}$$

since y_{F_1} contains the k_1-smallest entries in y. We know $k_1 = |\bar{F}\backslash F| \leq \min\{\bar{k}, p-k\}$, therefore

$$\alpha \leq \sqrt{\frac{k_1}{p}}\|y\| \leq \min\{\sqrt{\frac{\bar{k}}{p}}, \sqrt{\frac{p-k}{p}}\}\|y\|, \tag{3.47}$$

$$|\text{Truncate}(y,F)^T\bar{x}| \geq |y^T\bar{x}| - \bar{\alpha}\alpha \geq |y^T\bar{x}| - \alpha \tag{3.48}$$

$$\Rightarrow |\text{Truncate}(y,F)^T\bar{x}| \geq |y^T\bar{x}| - \min\{\sqrt{\frac{\bar{k}}{p}}, \sqrt{\frac{p-k}{p}}\}\|y\|. \tag{3.49}$$

□

Remark. In TPower [149], the authors established a truncation error bound that depends on the inner product $y^T\bar{x}$, assuming $y^Tx > \sqrt{\frac{\bar{k}}{\bar{k}+k}}$:

$$|\text{Truncate}(y,F)^T\bar{x}| \geq |y^T\bar{x}| - \sqrt{\frac{\bar{k}}{k}}\min\{\sqrt{1-(y^T\bar{x})^2}, (1+\sqrt{\frac{\bar{k}}{k}})(1-(y^T\bar{x})^2)\} \tag{3.50}$$

$$\geq |y^T\bar{x}| - \sqrt{\frac{\bar{k}}{k}}\min\{1-\frac{(y^T\bar{x})^2}{2}, (1+\sqrt{\frac{\bar{k}}{k}})(1-(y^T\bar{x})^2)\} \tag{3.51}$$

The limitation of eq. (3.50) is that it requires either $y^T\bar{x}$ to be large enough when $k \approx \bar{k}$, or $k \gg \bar{k}$ for the bound to be non-trivial.

Truncation error of the matrix product $Q^T\bar{P}$. We denote each column of Q by q_i and each column of $\bar{P}$ by $\bar{p}_j$. Let $\|\bar{p}_j\|_0 = \bar{k}_j$, $|F_i| = k_i$, and $\mathcal{F} = \{F_i\}$. The truncation error of the matrix product $\bar{P}^T Q$ is given by:

$$(\text{Truncate}(q_i, F_i)^T \bar{p}_j)^2 \geq (q_i^T \bar{p}_j)^2 - 2\sqrt{\frac{\min\{\bar{k}_j, p - k_i\}}{p}}\|q_i\|^2, \tag{3.52}$$

$$\sum_{i,j}(\text{Truncate}(q_i, F_i)^T \bar{p}_j)^2 \geq \sum_{i,j}(q_i^T \bar{p}_j)^2 - 2\sum_{i,j}\sqrt{\frac{\min\{\bar{k}_j, p - k_i\}}{p}}\|q_i\|^2 \tag{3.53}$$

$$\Rightarrow \frac{\|\text{Truncate}(Q, \mathcal{F})^T \bar{P}\|_F^2}{\|\text{Truncate}(Q, \mathcal{F})\|_F^2} \geq \frac{\|Q^T \bar{P}\|_F^2}{\|Q\|_F^2} - 2m\sqrt{\frac{\min\{\bar{k}_{\max}, p - k_{\min}\}}{p}}, \tag{3.54}$$

where $\bar{k}_{\max} = \max_j\{\bar{k}_j\}$ and $k_{\min} = \min_i\{k_i\}$.

Remark. This bound is a worst-case bound, and it is only achieved when the truncated support set does not contain any of the true indices. If we assume that $\bar{F}_i \subseteq F_i^{(t)}$ $\forall i$, which is often the case when $k > \bar{k}$, there is no truncation error and

$$\frac{\|\text{Truncate}(Q, \mathcal{F})^T \bar{P}\|_F^2}{\|\text{Truncate}(Q, \mathcal{F})\|_F^2} \geq \frac{\|Q^T \bar{P}\|_F^2}{\|Q\|_F^2}. \tag{3.55}$$

Furthermore, when $p \gg k$, $\|\text{Truncate}(Q, \mathcal{F})\|_F^2$ is likely to be much less than $\|Q\|_F^2$. Due to these two factors, the inequality eq. (3.55) usually holds in practice when we use a warm initialization and gradually decrease k. We consider the test example in section 3.2 with varying $\rho(E)$ and plot an empirical estimate for the truncation error: $\hat{\delta}_{\text{Truncate}} = \frac{\|Q^T \bar{P}\|_F^2}{\|Q\|_F^2} - \frac{\|\text{Truncate}(Q,\mathcal{F})^T \bar{P}\|_F^2}{\|\text{Truncate}(Q,\mathcal{F})\|_F^2}$, and the result is included in the Appendix. In all cases, $\hat{\delta}_{\text{Truncate}}$ remains negative regardless of $\rho(E)$, indicating that eq. (3.55) holds.

Putting everything together, we can now prove theorem 3:

Proof. Based on algorithm 3, Q_t is obtained by doing **qr** on $\text{Truncate}(AQ_{t-1}, \mathcal{F})$, i.e.

$$Q_t R_t = \text{Truncate}(AQ_{t-1}, \mathcal{F}). \tag{3.56}$$

$$\|\bar{P}^T Q_t\|_2^2 \geq \frac{\|\bar{P}^T \text{Truncate}(AQ_{t-1}, \mathcal{F})\|_F^2}{\|R_t\|_F^2} = \frac{\|\bar{P}^T \text{Truncate}(AQ_{t-1}, \mathcal{F})\|_F^2}{\|\text{Truncate}(AQ_{t-1}, \mathcal{F})\|_F^2} \text{ by eq. (3.15)} \quad (3.57)$$

$$\geq \frac{\|\bar{P}^T AQ_{t-1}\|_F^2}{\|AQ_{t-1}\|_F^2} - \delta_{\text{Truncate}} \text{ by eq. (3.54)} \quad (3.58)$$

$$\geq \frac{\|\bar{P}^T \bar{A}Q_{t-1}\|_F^2}{\|\bar{A}Q_{t-1}\|_F^2} - \frac{4m\rho(E,K)}{\|\bar{A}Q_{t-1}\|_F^2} - \delta_{\text{Truncate}} \text{ by lemma 5} \quad (3.59)$$

$$\geq \frac{\|\bar{P}^T \bar{A}Q_{t-1}\|_F^2}{\|\bar{A}Q_{t-1}\|_F^2} - \frac{4\rho(E,K)}{\lambda_m^2(1 - \|\sin\Theta(\bar{P}, Q_{t-1})\|_2^2)} - \delta_{\text{Truncate}} \quad (3.60)$$

$$\geq \frac{\|\bar{P}^T Q_{t-1}\|_F^2}{(1-\gamma^2)\|\bar{P}^T Q_{t-1}\|_F^2 + m\gamma^2} - \delta_E - \delta_{\text{Truncate}} \text{ by lemma 4} \quad (3.61)$$

The inequality eq. (3.60) holds due to the fact that:

$$\frac{m}{\|\bar{A}Q_{t-1}\|_F^2} \leq \|(\bar{A}Q_{t-1})^{-1}\|_2^2 \leq \frac{1}{\lambda_m^2(1 - \|\sin\Theta(\bar{P}, Q_{t-1})\|_2^2)} \text{ by eq. (3.78).}$$

Assume that $\|\bar{P}^T Q_t\|_F^2 = c\|\bar{P}^T Q_t\|_2^2$, where $c \in [1, m]$. Then we have:

$$\|\sin\Theta(\bar{P}, Q_t)\|_F^2 = m - \|\bar{P}^T Q_t\|_F^2 = m - c\|\bar{P}^T Q_t\|_2^2 \quad (3.62)$$

$$\leq m - c\frac{\|\bar{P}^T Q_{t-1}\|_F^2}{(1-\gamma^2)\|\bar{P}^T Q_{t-1}\|_F^2 + m\gamma^2} + c\delta_E + c\delta_{\text{Truncate}} \quad (3.63)$$

$$= \frac{m\gamma^2\|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2 + (m-c)\|\bar{P}^T Q_{t-1}\|_F^2}{m - (1-\gamma^2)\|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2} + c\delta_E + c\delta_{\text{Truncate}} \quad (3.64)$$

$$\leq \frac{\gamma^2\|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2 + \frac{m-c}{m}\|\bar{P}^T Q_{t-1}\|_F^2}{1 - (1-\gamma^2)\|\sin\Theta(\bar{P}, Q_{t-1})\|_2^2} + c\delta_E + c\delta_{\text{Truncate}} \quad (3.65)$$

When $\|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2 \geq \frac{m-c}{1-\gamma^2}$,

$$\frac{m\gamma^2\|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2 + (m-c)\|\bar{P}^T Q_{t-1}\|_F^2}{m - (1-\gamma^2)\|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2} \leq \|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2. \quad (3.66)$$

When $c = m$, the right-hand side of eq. (3.65) simplifies to

$$\frac{\gamma^2\|\sin\Theta(\bar{P}, Q_{t-1})\|_F^2}{1 - (1-\gamma^2)\|\sin\Theta(\bar{P}, Q_{t-1})\|_2^2} + m\delta_E + m\delta_{\text{Truncate}}. \quad (3.67)$$

□

We have proved that the orthogonal projector $\mathcal{Q}^{(t)} = Q_t Q_t^T$ onto $\text{span}\{Q_t\}$ converges. If the sequence of projectors converges, each column vector of Q_t also converges.

3.4 Relations to Existing Algorithms

When estimating a single sparse eigenvector, theorem 3 can be applied directly to the Truncated Power Method (TPower) [149] using $m = 1$. The Iterative Thresholding Sparse PCA (ITSPCA) proposed by Ma [90] also fits into the framework introduced in Section 3.2 and uses a thresholding step. The thresholding step is performed through a user-specified function η which satisfies:

$$|\eta(y_i, t) - y_i| \leq t,\ \eta(y_i, t)\mathbf{1}_{|y_i|<t} = 0\ \forall y_i \in y,\ t > 0. \tag{3.68}$$

Common thresholding techniques include soft and hard thresholding, as well as a range of operators in between, see e.g. SCAD [41]. The analysis in theorem 3 can be applied to ITSPCA if we substitute lemma 6 by the following lemma:

Lemma 7. *Consider a unit vector $\bar{x} \in \mathbb{R}^p$ with support set $supp(\bar{x}) = \bar{F}$, and $\bar{k} = |\bar{F}|$. Consider a vector y that is thresholded by the user-specified thresholding function η, where η satisfies eq.* (3.68)*. Define $[Thresh(y,t)]_i = \eta(y_i, t)$. Then*

$$\frac{Thresh(y,t)^T \bar{x}}{\|Thresh(y,t)\|_2} \geq \frac{|y^T \bar{x}|}{\|y\|_2} - \frac{t\sqrt{\bar{k}}}{\|y\|_2}. \tag{3.69}$$

Analysis in [90] under the spiked covariance model is also applicable to our algorithm 3 since we can transform our sparsity constraint, which is an upper bound on ℓ_0 norm, to an upper bound on ℓ_1 norm:

$$\|q_j\|_0 \leq k_j,\ \|q_j\|_2 = 1 \Rightarrow \|q_j\|_1 \leq \sqrt{k_j}.$$

3.5 Experimental results

In this section we show several numerical experiments to demonstrate the accuracy, efficiency and interpretability of the proposed algorithms. To illustrate the accuracy of the algorithms, we first apply them on a simulated dataset with known sparse eigenvectors and added perturbations, in which case eigenvalue decomposition and a simple truncation at the end fail to accurately recover the true eigenvectors. Since the Truncated Orthogonal Iteration is an

extension of TPower to recover multiple units, we also compare to TPower. We then use the PitProps dataset, a standard benchmark of sparse PCA algorithms, to evaluate the performance of the proposed algorithms and compare with some state-of-the-art algorithms. For interpretability and efficiency, we apply our algorithm to some large real-world data problems: a sea surface temperature dataset to recover the El Niño mode, and the 20 newsgroup dataset to interpret the projected data in terms of keywords for different topics. In what follows, we refer to algorithm 3 as TOrth and algorithm 4 as TOrthT.

3.5.1 Comparisons on a Simulated Dataset where $\bar{A}$ is known

We first consider a simulated example where we know the true matrix $\bar{A}$ with sparse eigenvectors and we add a small perturbation matrix E (also positive semidefinite) to it. Our goal is to recover the true sparse eigenvectors of $\bar{A}$ from the perturbed matrix $A = \bar{A} + E$. Specifically, we use a 1000×1000 positive semidefinite matrix $\bar{A} = V\Sigma V^T$ with the first three columns of V being sparse orthonormal vectors, and the remaining columns of V are randomly generated orthonormal vectors such that $V^TV = I$. We consider the following three different cases of the support set:

- Case I: the support sets for the first three eigenvectors are identical. Specifically, for $i = 1, 2, 3$, the first 10 entries of v_i are nonzero and the rest of the entries are zero.

- Case II: the support sets partially overlap.

- Case III: the support sets are completely non-overlapping.

For all three cases, the eigenvalues are set to be $\lambda_1 = 1$, $\lambda_2 = 0.9$, $\lambda_3 = 0.8$, $\lambda_j = 0.1$ for $j = 4, \cdots, 1000$. $\rho(E) \approx 0.89$. For each case, we generate A 100 times by randomly generating V and E, and apply the algorithms to A. We denote the true eigenvectors of $\bar{A}$ as $v_1,\ v_2,\ v_3$ and the recovered eigenvectors as $u_1,\ u_2,\ u_3$. For each algorithm, we want to know *whether* it recovers the true support set, and *how well* it approximates the true

eigenvectors. To measure the recovery, we use the F-score: $\mathrm{tp}/(\mathrm{tp} + 0.5 * (\mathrm{fp} + \mathrm{fn}))$, with F-score$= 1$ indicating perfect recovery. tp, fp, fn are defined as follows:

- A true positive (tp) occurs when the true eigenvector at the index has a nonzero element and we identified it.

- A false positive (fp) occurs when the true eigenvector at the index has a zero element, but we mistakenly identified it as a nonzero element.

- A false negative (fn) occurs when the true eigenvector at the index has a nonzero element, but we missed it (identified as zero).

To measure the quality of the recovery, we record the inner products of the true and the recovered eigenvectors averaged over *all* trials. We also record the success rate, where a trial is counted as success if all three inner products are greater than 0.99. We record the eigenvalue decomposition results, shown as "Eig" in table 3.2. We also apply standard eigenvalue decomposition with a simple truncation step at the end, which is denoted as "Simple Truncation".

In terms of approximation accuracy, TOrth and TOrthT outperform TPower in Case I, and perform on par with TPower in the other two cases. With an extra truncation step at each iteration to ensure the sparsity of the output vectors, TOrthT achieves a better F-score than TPower in Case I, and perfect recovery in Case II and III, while maintaining an accuracy comparable to TOrth. The standard eigenvalue decomposition obtains a much lower accuracy than the sparse methods in all three cases, and adding a truncation step in the end does not improve the accuracy. Simple truncation does not recover as much as TOrthT and TPower in case I and II, but is able to recover all in case III. We also include a comparison with random initialization for all algorithms with $\rho(E) \approx 0.2$, and the result is shown in table 3.1. We observe that TOrth and TOrthT are more robust to random initialization in all cases compared to TPower.

Table 3.1: Results on Simulated Data: Random Initialization

	Algorithms	$\lvert v_1^T u_1 \rvert$	$\lvert v_2^T u_2 \rvert$	$\lvert v_3^T u_3 \rvert$	Success Rate	Recovery Rate
Case I: Completely overlap	Standard	0.9912	0.9888	0.9869	0	0
	TPower	0.9885	0.9501	0.7881	78.7%	39.2%
	TOrth	0.9955	0.9910	0.9872	**98.8%**	50.0%
	TOrthT	0.9935	0.9910	0.9872	**98.8%**	**51.9%**
Case II: Partially overlap	Standard	0.9916	0.9896	0.9863	0	0
	TPower	0.9131	0.8252	0.8141	75.8%	75.8%
	TOrth	0.9161	0.8962	0.9600	**89.2%**	0
	TOrthT	0.8971	0.8712	0.9540	86.8%	**86.8%**
Case III: non-overlap	Standard	0.9915	0.9892	0.9878	0	0
	TPower	0.8690	0.7550	0.7569	66.8%	66.8%
	TOrth	0.8470	0.8020	0.9069	**79.4%**	**79.4%**
	TOrthT	0.8560	0.7960	0.9030	79.0%	79.0%

Table 3.2: Results on Simulated Data: Warm Initialization

	Algorithms	$\lvert v_1^T u_1 \rvert$	$\lvert v_2^T u_2 \rvert$	$\lvert v_3^T u_3 \rvert$	Success Rate	F-score
Case I: Completely overlap	TOrth	0.9955	0.9926	0.9949	74%	0.9267
	TOrthT	0.9955	0.9925	0.9950	74%	0.9243
	TPower	0.9950	0.9913	0.9939	61%	0.9117
	Eig	0.9352	0.9079	0.8724	0%	0.0198
	Simple Truncation	0.9350	0.9074	0.8713	0%	0.8950
Case II: Partially overlap	TOrth	0.9997	0.9996	0.9996	100%	0.7143
	TOrthT	0.9997	0.9996	0.9996	100%	1
	TPower	0.9997	0.9996	0.9996	100%	1
	Eig	0.9366	0.9105	0.8741	0%	0.0198
	Simple Truncation	0.9366	0.9101	0.8738	0%	0.9947
Case III: non-overlap	TOrth	0.9997	0.9996	0.9995	100%	1
	TOrthT	0.9997	0.9996	0.9995	100%	1
	TPower	0.9997	0.9996	0.9995	100%	1
	Eig	0.9368	0.9095	0.8758	0%	0.0198
	Simple Truncation	0.9368	0.9095	0.8758	0%	1

3.5.2 PitProps Dataset

The PitProps dataset [63] is one of the classical examples of PCA interpretability and a benchmark to evaluate the performance of sparse PCA algorithms. The dataset contains 180 observations of props of Corsican pine from East Anglia and 13 variables corresponding to the physical properties of the props. The first six principal components obtained by standard PCA accounts for 87% of total variance. We compute the first six sparse loadings of the data and compare the results with other sparse PCA algorithms. Since the outputs of sparse PCA algorithms are not guaranteed to be uncorrelated, we measure the performances of sparse PCA algorithms by the cumulative percentage of explained variance, which was proposed in [116]. It uses the projection of X onto the $m-$dimensional subspace spanned by the loading vectors V to adjust the non-orthogonality of V, i.e.:

$$X_m = XV(V^TV)^{-1}V^T,$$

the cumulative percentage is then computed as $\text{Trace}(X_m^TX_m)/\text{Trace}(X^TX)$.

We report the results obtained by Algorithm 4 and compare with the results obtained by TPower [149], GPower [71], PathPCA [34], rSVD [116], SPCA [155] and ITSPCA [90] in Table 3.3. TOrthT performs the best among the state-of-the-art algorithms.

3.5.3 Denoising of Synthetic Signals

In this experiment we follow a similar setting as the denoising experiment from [64] and generate signals from the following noisy linear model:

$$x_i = \bar{x}_i + \epsilon_i = u_1V^1 + u_2V^2 + u_3V^3 + \epsilon_i \in \mathbb{R}^{400}$$

where $V = [V^1, V^2, V^3] \in \mathbb{R}^{400\times 3}$ are sparse and structured dictionary elements organized on a 20×20-dimensional grid and are non-overlapping, as shown in the first row of fig. 3.2. Each dictionary element has a structured sparsity consisting of a 10×10 nonzero block. The components of the noise vector ϵ_i are independent and identically distributed according to a centered Gaussian distribution with its variance set to obtain a signal-to-noise ratio (SNR)

Table 3.3: Results on PitProps Dataset

Algorithms	Input Parameters	Output Card.	cumulative % of explained variance
TOrthT	$K = [7, 2, 4, 3, 5, 4]$	25	0.8487
TPower	$K = [7, 2, 4, 3, 5, 4]$	25	0.8377
GPower_{ℓ_1}	$\gamma = 0.22$; see [71]	25	0.8279
PathPCA	$K = [7, 2, 4, 3, 5, 4]$	25	0.8438
rSVD_{ℓ_1}	see Shen and Huang [116]	25	0.8450
SPCA	see Zou et al. [155]	18	0.8022
ITSPCA	$\alpha = 3, \beta = 1.5$	35	0.3601

of 1.0, i.e. $\text{std}(\epsilon_i) = \text{std}(\bar{x}_i)$, $\text{mean}(\epsilon_i) = \text{mean}(\bar{x}_i) = 0$. The linear coefficients $[u_1, u_2, u_3]$ are generated from a normal distribution:

$$[u_1, u_2, u_3] \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{bmatrix}\right).$$

We generate $n = 250$ signals according to the noisy linear model, and decompose the data matrix to obtain the first three dictionary elements using standard PCA, standard PCA+ post truncation (only truncate at the end), Vintage Sparse PCA (vsp) and TOrth. The sparsity level is set to be the true sparsity, i.e. $K = [100, 100, 100]$. The results are shown in fig. 3.2. We observe that the standard PCA and simple truncation are not able to recover the original dictionaries, while TOrth finds the structured sparsity in all three elements.

3.5.4 Sea Surface Temperature Example

The Sea Surface Temperature dataset (SST) is a spatio-temporal dataset that records weekly means of satellite ocean temperature data over $64,800$ spatial grid points from 1990 to

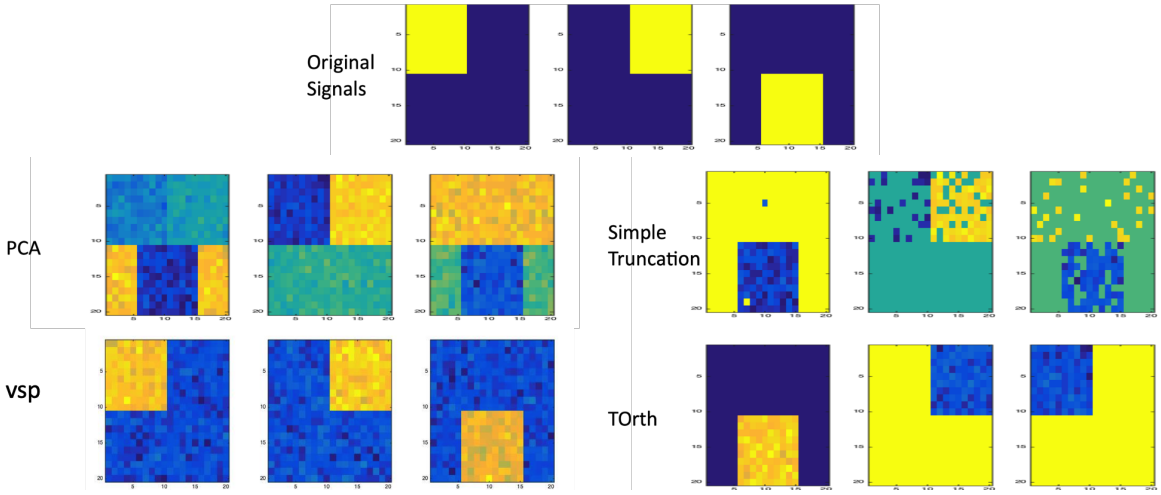


Figure 3.2: **Signal recovery by sparse PCA algorithms.** We generate signals according to the noisy linear model, and decompose the data matrix to obtain the first three dictionary elements using different techniques. The figures show original signals and signals recovered by PCA, PCA with simple truncation, vsp and TOrth.

present [111]. In this experiment, we wish to capture the El Niño events from this time series of sea surface temperature by extracting its spatial modes. The El Niño Southern Oscillation (ENSO) is defined as any sustained temperature anomaly above running mean temperature with a duration of 9 to 24 months. The canonical El Niño is associated with a narrow band of warm water off coastal Peru. In fig. 3.3 left, we see that TOrth successfully recovers the band with only 800 nonzeros in the fourth mode. Standard PCA, as shown in fig. 3.3 right, is unable to separate this band from a global weather pattern across the Pacific and Atlantic.

Moreover, we use this example to demonstrate the computational efficiency of our algorithm. The SST data dimension is 1455×64800, where $n = 1455$ is the number of temporal snapshots and $p = 64800$ is the spatial grid points in each snapshot. We compute the fourth mode that is associated with the canonical El Niño, so we set $m = 4$ and compare with other block (sparse) PCA algorithms. The time complexity and actual running time are recorded in table 3.4. The iteration number needed to reach convergence is denoted as s, which

varies between algorithms. The methods we compared to: GPower [71], TPower [149], and Variable Projection SPCA [40], are already among the top performers in terms of computational speed. We omit comparisons with methods that are much more expensive to compute, for example, the elastic net SPCA algorithm [155] requires $O(np^2 + p^3)$ and takes at least $\times 10$ running time compared to Variable Projection SPCA on the SST dataset [40]. Although GPower [71] is comparable to TOrth in big O notation, GPower requires longer running time because it utilizes the Polar decomposition in each iteration, which takes $3m^2p + m^3 + O(m)$. In comparison, TOrth uses QR, which takes $2m^2p - (2/3)m^3$. The difference between TPower and TOrth is analyzed in section 3.2, and the deflation step can be expensive when the data matrix is large.

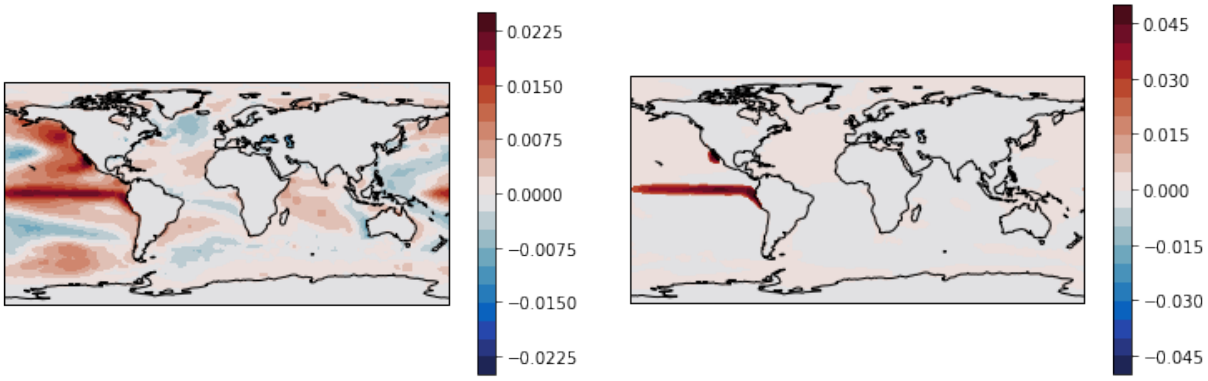


Figure 3.3: **Recovery of the El Niño mode.** Using the sea surface temperature dataset, we are able to extract the dominant spatial modes across time. The first three modes are background and seasonal shift (i.e. summer and winter temperature in the two hemisphere). The figures show the fourth mode recovered by standard PCA (Left), and by TOrth (Right), where $K = [p, p, p, 800]$. The sparse mode obtained by TOrth highlights the band associated with the El Niño event.

3.5.5 Classification Example on the MNIST dataset

We apply algorithm 3 to the MNIST handwritten digit dataset and compare the classification performance with that of standard PCA. The MNIST dataset has 70,000 samples and each

Table 3.4: Comparisons on Complexity and Running Time

	VarProj	GPower$_m$	TPower	TOrth
$n > p$	$O(np^2) + sO(mpn + m^2p)$	$sO(mpn + m^2p)$	$msO(np) + O(mnp)$	$sO(mpn + m^2p)$
$n < p$	$O(pn^2) + sO(mpn + m^2p)$	$sO(mpn + m^2p)$	$msO(np) + O(mnp)$	$sO(mpn + m^2p)$
Run Time	44.79	17.19	6.36	2.89

samples is a 2D image with 28×28 pixels, yielding 748 features. We first use the default train-test split where 60,000 samples are used as the training set and 10,000 samples as the test set. Applying a k-nearest neighbor classifier (KNN) on the original data gives a prediction error of 3%, where the number of nearest neighbors is chosen to be 3 by cross-validation.

We apply standard PCA and TOrth to reduce the dimension from 748 to various subspace dimension p. We use 3 neighbors for both standard and sparse PCA. With PCA, applying KNN on the projected test data achieves the lowest prediction error of 2.47% when $p = 60$, and the performance of KNN starts to decline due to the curse of dimensionality. Fixing $p = 60$, results of TOrth with different k are shown in table 3.5. We observe that by using only $k = 20$ in each loading vectors and 60 loading vectors, TOrth is able to achieve a prediction error comparable to KNN applied to raw data with full dimension. fig. 3.4 displays the top 30 loading vectors obtained by PCA and TOrth, where loading vectors obtained by TOrth captures more local features and includes fractions/strokes of digits.

Table 3.5: Prediction error on MNIST

k	10	20	40	80	160	320	640	748 (p)
Prediction error (%)	4.26	2.95	2.78	2.64	2.63	2.49	2.48	2.47

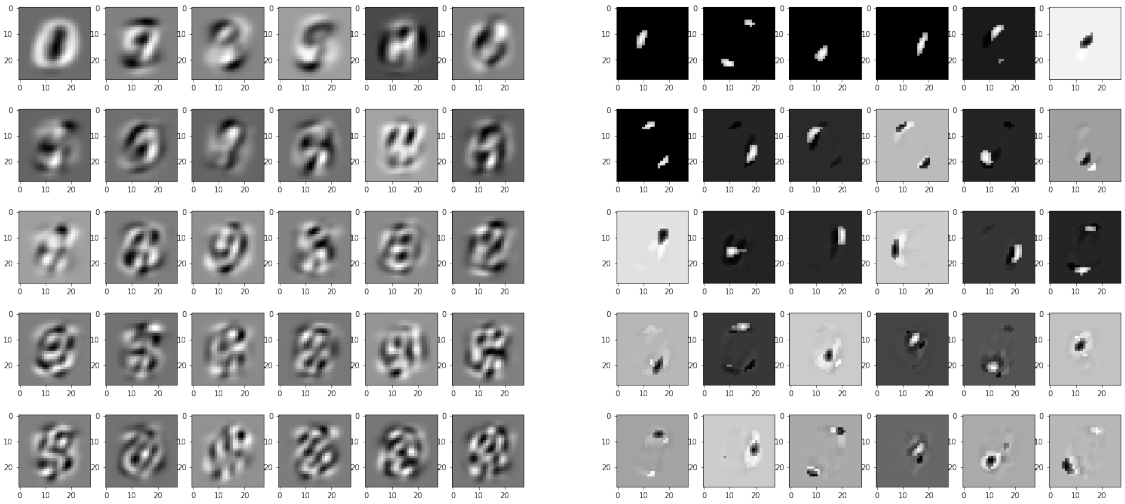


Figure 3.4: **Visualization of the top 30 loading vectors of MNIST training set.** Left: loading vectors obtained by standard PCA. Right: loading vectors obtained by algorithm 3, with $k_i = 20, \forall i$. PCA obtains vectors with overlapping digits, where TOrth captures more local features and includes fractions/strokes of digits.

3.5.6 20 newsgroup dataset

In real data science applications, it is often preferred that we not only project the original data onto a lower-dimensional subspace, but also make meaningful interpretations of the projected data. We take a sample of the 20 newsgroup dataset [80], with binary occurrence data for $p = 100$ keywords across $n = 16242$ postings. The postings come from 4 general topics: computer, recreation, science and talk. We apply sparse PCA using TOrthT with sparsity level $k = 10$ and the standard PCA on the centered data. The 10 nonzero entries in the coefficient vector for the first two PCs are shown in the top of fig. 3.5. The keywords associated with the first PC are more relevant to religion and politics, which are subtopics of "talk" in the dataset, and the keywords associated with the second PC are more relevant to computers. We project the data onto the 2D subspace spanned by standard PCs (fig. 3.5 (a) and (b)), as well as the PCs with sparse loading vectors (fig. 3.5 (c) and (d)). We observe that with the loading vectors obtained from TOrthT, the projections of the "computer"-

themed data are dense in PC2 and sparse in PC1, while the "talk"-themed data projections are dense in PC1 and sparse in PC2. The sparsity in one PC direction indicates that the data lack keywords associated with the PC. In contrast, projections onto the normal PCs are clustered and are dense in both directions, and lacks physical interpretations.

3.6 Application in K-subspace Clustering

Subspace clustering methods are designed to identify the subspaces and group the points according to their nearest subspaces, with the assumptions that data are generated from a union of subspaces and that the clusters have low-rank structure. The well-known K-subspace (KSS) [1, 14] clustering algorithm seeks to minimize the sum of residuals of points to their assigned subspace, i.e.,

$$\min_{\mathcal{C},\,\mathcal{U}} \sum_{k=1}^{K} \sum_{i:x_i \in c_k} \|x_i - U_k U_k^T x_i\|_2^2, \tag{3.70}$$

where $\mathcal{C} = \{c_1, \cdots, c_k\}$ is the set of estimated clusters and $\mathcal{U} = \{U_1, \cdots, U_k\}$ is the the corresponding subspace bases. KSS achieves this objective by alternating between (i) clustering points by the nearest subspace and (ii) obtaining new subspace bases by performing PCA on the points in each cluster. The algorithm decreases the objective value in each iteration and converges to a local minimum, but like K-means, it is highly dependent on initialization. In early stages of the algorithm, the clusters are estimated and can be deterred by outliers and noise, thus making the objective less appropriate and the PCA step faulty. To make the subspace estimation step more robust, especially in the case where the number of points is less than the dimension of the ambient space, the TOrth algorithm can be applied to K-subspace clustering in place of the standard PCA for recovering subspaces. By obtaining sparse basis and setting certain coefficients of the data vector to zeros, TOrth is able to reduce the impact induced by noise. The resulting algorithm, Robust K-subspace clustering (RKSS), is summarized in algorithm 5.

To illustrate the effectiveness of RKSS, we first consider the synthetic example used in [39, 56, 87]. We generate S $d-$dimension subspaces of $D-$dimensional ambient space, with

PC1	Question World	Fact God	Problem Number	Course Human	Case Government
PC2	Help Program	Email University	Problem Computer	System Software	Windows Files

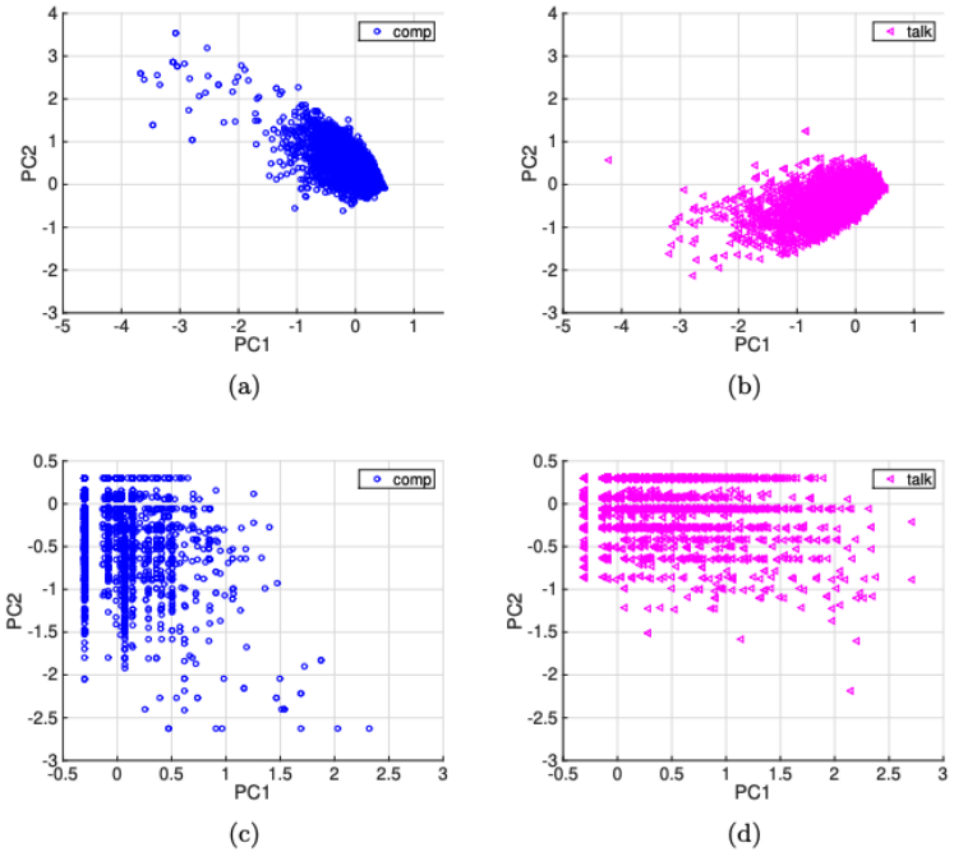


Figure 3.5: **Projection of computer and talk data onto PC1 and PC2.** On the top, we have the nonzero coefficients for the first two principal components obtained by TOrthT. The vector associated with PC1 contains the keywords that are characteristic to religion and politics, such as "God", "World", "Human" and "Government". The vector associated with PC2 contains the keywords that are characteristic to computers, such as "Computer", "Email", "Software" and "Files". The projected data obtained by standard PCA, shown in (a) and (b), are dense along both PC1 and PC2. The projected data obtained by TOrthT, shown in (c) and (d), has a strong sparsity pattern along one axis while dense along the other, thus differentiating between the two topics.

N_k points per subspace corrupted by zero-mean Gaussian noise with variance $\sigma^2 I_d$. fig. 3.6 shows the error distribution obtained by KSS and RKSS over 100 random initialization, and we observe that RKSS outperforms KSS by a large margin. To account for the sensitivity in initialization, we also combine the algorithm with the recently proposed Ensemble K-subspace clustering (EKSS) [87], where KSS is repeated B times to form a co-association matrix. Spectral clustering is then applied to the thresholded co-association matrix. We replace the KSS with RKSS and term the resulting algorithm ERKSS. We test the four algorithms, KSS, KRSS, EKSS and ERKSS on the synthetic data and two real datasets: the USPS [18] dataset and the COIL20 [109] dataset. The clustering error is measured in the same way as in [87], and the results are summarized in table 3.6. The experimental setup and details are included in Appendix A3.

Algorithm 5 Robust K-subspace Clustering

Input: Dataset X, number of subspaces S, subspace dimensions $r_1, \cdots, r_S$, cardinality vector $K_1, \cdots, K_s$.

Output: orthonormal subspace bases $\mathcal{U} = \{U_1, \cdots, U_S\}$, clusters $\mathcal{C} = \{C_1, \cdots, C_k\}$.

randomly initialize orthonormal bases $\mathcal{U}$

while not converged **do**

$c_s \leftarrow \{x \in X : s = \arg\min_l \|x - U_l U_l^T x\|_2\}$ for $s = 1, \cdots, S$

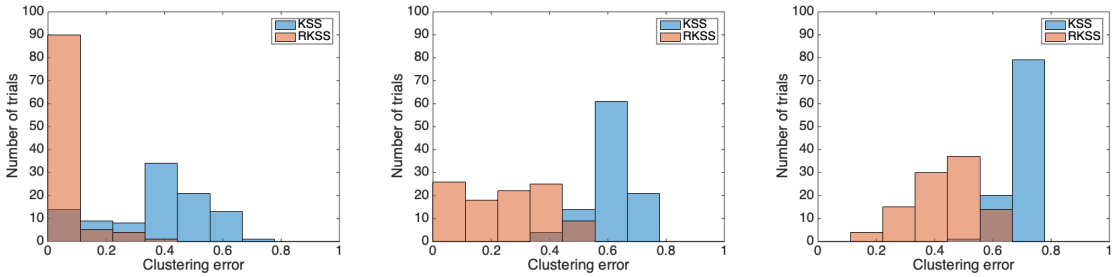
Construct the covariance matrix A_s for data points in c_s for all s

$U_s \leftarrow TOrth(A_s, K_s)$ for $s = 1, \cdots, S$

end while

3.7 Conclusions

In this project, we have proposed two algorithms based on the orthogonal iteration to recover sparse eigenvectors, and established convergence analyses. Our algorithms can be easily implemented and work efficiently and accurately on a wide range of data science applications. Compared to its single-vector counterpart, the block scheme is more robust to



(a) $\sigma^2 = 0.2$. (b) $\sigma^2 = 0.5$. (c) $\sigma^2 = 0.8$.

Figure 3.6: **Clustering error by KSS and RKSS on the synthetic dataset.** We test KSS and RKSS on the synthetic dataset with noise variance $\sigma^2 = 0.2,\ 0.5,\ 0.8$, respectively. We run 100 trials and both algorithms are randomly initiated in each trial. According to the distributions as shown in the histograms, RKSS performs much better than KSS with $mean(error_{RKSS}) < mean(error_{KSS}) - 0.3$.

Table 3.6: Clustering Error

	KSS	RKSS	EKSS	ERKSS
Synthetic data with cov= $0.8I_d$	0.5520	0.1320	0.1760	**0.0760**
Synthetic data with cov= $0.5I_d$	0.3960	**0.0240**	0.0240	0.0240
USPS	0.3033	**0.2195**	0.2766	0.2341
COIL20	0.5674	0.4792	**0.2208**	0.3694

random initialization and achieves better accuracy and sparse recovery rate. The algorithms also extend to other applications, such as the K-subspace clustering, and achieves superior results compared to the standard KSS.

There are still many open problems in this area of research. For example, one challenge for the block approach is how to maximally preserve orthogonality while achieving sparsity. Choosing a proper initialization strategy is also important for better and faster convergence. Lastly, different truncation schemes, including both deterministic and probabilistic approaches, can be explored to preserve the true support set and avoid truncation error.

3.8 Appendix

A1. We give the proof for eq. (3.21) and eq. (3.22), which is based on [Theorem 8.1.10] of [133].

Theorem 4. *Let P be the matrix of eigenvectors corresponding to the $m-$largest eigenvalues of $\bar{A}$. Assume $\lambda_m > \lambda_{m+1}$. Define $\gamma := \frac{\lambda_{m+1}}{\lambda_m}$. Then the matrices Q_t generated by the standard orthogonal iteration satisfy:*

$$\|\sin\Theta(P, Q_t)\|_F \leq \gamma \frac{\|\sin\Theta(P, Q_{t-1})\|_F}{\sqrt{1 - \|\sin\Theta(P, Q_{t-1})\|_2^2}},$$

assuming $\|\sin\Theta(P, Q_{t-1})\|_2 < 1$.

Proof. In the standard orthogonal iteration,

$$\bar{A}Q_{t-1} = Q_t R_t. \tag{3.71}$$

We can decompose Q_t as $Q_t = PX_t + P^{\perp}Y_t$, where $P^{\perp}$ is the orthogonal complement of P and its columns are the eigenvectors of $\bar{A}$ corresponding to the $m+1, \cdots p$ eigenvalues, i.e. $\bar{A}P = P\Lambda_m$, $\bar{A}P^{\perp} = P^{\perp}\Lambda'$, where $\Lambda' = \mathrm{diag}(\lambda_{m+1}, \cdots, \lambda_p)$. We have the following equations:

$$X_t = P^T Q_t, \quad Y_t = {P^{\perp}}^T Q_t \Leftrightarrow \begin{bmatrix} X_t \\ Y_t \end{bmatrix} = \begin{bmatrix} P & P^{\perp} \end{bmatrix}^T Q_t. \tag{3.72}$$

By applying the thin CS decomposition [133] on eq. (3.72) we have

$$1 = \sigma_{\min}(X_t)^2 + \sigma_{\max}(Y_t)^2, \tag{3.73}$$

$$\begin{bmatrix} P & P^{\perp} \end{bmatrix} \begin{bmatrix} \Lambda_m & \\ & \Lambda' \end{bmatrix} \begin{bmatrix} P^T \\ P^{\perp T} \end{bmatrix} Q_{t-1} = Q_t R_t \text{ by eq. (3.71)}, \tag{3.74}$$

$$\begin{bmatrix} \Lambda_m & \\ & \Lambda' \end{bmatrix} \begin{bmatrix} P^T \\ P^{\perp T} \end{bmatrix} Q_{t-1} = \begin{bmatrix} P^T \\ P^{\perp T} \end{bmatrix} Q_t R_t,$$

$$\begin{bmatrix} \Lambda_m & \\ & \Lambda' \end{bmatrix} \begin{bmatrix} X_{t-1} \\ Y_{t-1} \end{bmatrix} = \begin{bmatrix} X_t \\ Y_t \end{bmatrix} R_t$$

$$\Rightarrow \Lambda_m X_{t-1} = X_t R_t, \quad \Lambda' Y_{t-1} = Y_t R_t. \tag{3.75}$$

Assuming R_t and X_{t-1} are nonsingular, we obtain the following equations from eq. (3.75):

$$Y_t = \Lambda' Y_{t-1} R_t^{-1}, \tag{3.76}$$

$$R_t^{-1} = (\Lambda_m X_{t-1})^{-1} X_t = X_{t-1}^{-1} \Lambda_m^{-1} X_t, \tag{3.77}$$

$$\|R_t^{-1}\|_2 \leq \|X_{t-1}^{-1}\|_2 \|\Lambda_m^{-1}\|_2 \|X_t\|_2 \leq \frac{1}{\sqrt{1 - \|Y_{t-1}\|_2^2}} \frac{1}{\lambda_m} \tag{3.78}$$

The last inequality in eq. (3.78) comes from eq. (3.73) and the fact that

$$\|X_{t-1}^{-1}\|_2 \leq \frac{1}{\sigma_{\min}(X_{t-1})} = \frac{1}{\sqrt{1 - \sigma_{\max}(Y_{t-1})^2}}.$$

Based on eq. (3.76) and eq. (3.78), we arrive at the bound given in eq. (3.22):

$$\|\sin\Theta(P, Q_t)\|_F = \|Y_t\|_F \leq \|\Lambda'\|_2 \|Y_{t-1}\|_F \|R_t^{-1}\|_2 \tag{3.79}$$

$$\leq \lambda_{m+1} \cdot \|Y_{t-1}\|_F \cdot \frac{1}{\sqrt{1 - \|Y_{t-1}\|_2^2}} \cdot \frac{1}{\lambda_m} \tag{3.80}$$

$$= \gamma \frac{\|\sin\Theta(P, Q_{t-1})\|_F}{\sqrt{1 - \|\sin\Theta(P, Q_{t-1})\|_2^2}}. \tag{3.81}$$

Similarly, based on

$$\Lambda_m^t X_0 = X_t (R_t \cdots R1), \quad \Lambda'^t Y_0 = Y_t (R_t \cdots R1), \tag{3.82}$$

we can derive the bound in eq. (3.21):

$$\|\sin\Theta(P,Q_t)\|_F = \|Y_t\|_F \le \|\Lambda'\|_2^t \|Y_0\|_F \|(R_t\cdots R1)^{-1}\|_2 \tag{3.83}$$

$$\le \lambda_{m+1}^t \cdot \|Y_0\|_F \cdot \frac{1}{\sqrt{1-\|Y_0\|_2^2}} \cdot \frac{1}{\lambda_m^t} \tag{3.84}$$

$$= \gamma^t \frac{\|\sin\Theta(P,Q_0)\|_F}{\sqrt{1-\|\sin\Theta(P,Q_0)\|_2^2}}. \tag{3.85}$$

□

A2. We give a sketch of proof to show that $\rho(E,s) = O(\sqrt{s\log p/n})$ under the spiked covariance matrix model. In the single spike covariance model, assume that $\|\bar{x}\|_2 = 1$ and the true covariance matrix is given by:

$$\bar{A} = \bar{x}\bar{x}^T + I_p = \Sigma^{1/2}\Sigma^{1/2^*}, \tag{3.86}$$

and A is the empirical covariance matrix,

$$A = \frac{1}{n}(\Sigma^{1/2}X)(\Sigma^{1/2}X)^*, \tag{3.87}$$

where $X \in \mathbb{R}^{p\times n}$ whose entries are i.i.d NORMAL(0,1). Let

$$E = A - \bar{A} = \Sigma^{1/2}(I_p - \frac{XX^*}{n})\Sigma^{1/2}. \tag{3.88}$$

We want to bound the restricted perturbation error, defined by:

$$\rho(E,s) := \max_{\|v\|_2=1, \|v\|_0\le s} v^T E v. \tag{3.89}$$

Define $W = \cup_{v\in\mathbb{R}^p, \|v\|_0\le s}\Sigma^{1/2}v$. $(1 - \min_{y\in W,\|y\|_2=1} y^T\frac{XX^*}{n}y) \ge 0$ with high probability.

$$\rho(E,s) = \max_{\|v\|_2=1, \|v\|_0\le s} v^T\Sigma^{1/2}(I_p - \frac{XX^*}{n})\Sigma^{1/2}v \tag{3.90}$$

$$\le \|\Sigma\|_2 \max_{y\in W,\|y\|_2=1}(1 - y^T\frac{XX^*}{n}y) \tag{3.91}$$

$$= \|\Sigma\|_2(1 - \min_{y\in W,\|y\|_2=1} y^T\frac{XX^*}{n}y) \tag{3.92}$$

Theorem 8.4 in [96] states that

$$\mathbb{P}\{\sigma_{\min}(\frac{X^*}{\sqrt{n}}; W) \leq 1 - \frac{w(W)+1}{\sqrt{n}} - t\} \leq e^{-nt^2/2}, \tag{3.93}$$

where

$$\sigma_{\min}(\frac{X^*}{\sqrt{n}}; W) := \min_{y \in W} \|\frac{X^*}{\sqrt{n}} y\|_2 = \min_{y \in W} \sqrt{y^T \frac{XX^*}{n} y}. \tag{3.94}$$

We first calculate the Gaussian width of W:

$$w(W) := \mathbb{E} \sup_{y \in W} \langle g, y \rangle \text{ where } g \in \mathbb{R}^p \text{ is standard normal} \tag{3.95}$$

$$= \mathbb{E} \sup_{\|v\|_0 \leq s} \|\text{Truncate}(\Sigma^{1/2} g, \text{supp}(v))\|_2 \tag{3.96}$$

$$= \mathbb{E} \sup_{|S|=s} \|\text{Truncate}(\Sigma^{1/2} g, S)\|_2 \tag{3.97}$$

Set $X_s := \|\text{Truncate}(\Sigma^{1/2} g, S)\|_2$. By Gaussian concentration inequality we have

$$\mathbb{P}(|X_s - \mathbb{E}[X_s]| \geq t) \leq 2e^{-t^2/(2\|\Sigma^{1/2}\|_2)}, \tag{3.98}$$

indicating that X_s belongs to the sub-Gaussian family. Then we have

$$\mathbb{E} \max_{|S|=s} (X_s - \mathbb{E}[X_s]) \leq C\sqrt{\log \binom{p}{s}} \tag{3.99}$$

$$\Rightarrow w(W) \leq \max_{|S|=s} \mathbb{E}[X_s] + C\sqrt{s \log \frac{ep}{s}} \tag{3.100}$$

$$= O(\sqrt{s \log p}). \tag{3.101}$$

From eq. (3.93) we can derive that $\rho(E, s) = 1 - \sigma^2_{\min}(\frac{X^*}{\sqrt{n}}; W) = O(\sqrt{s \log p / n})$.

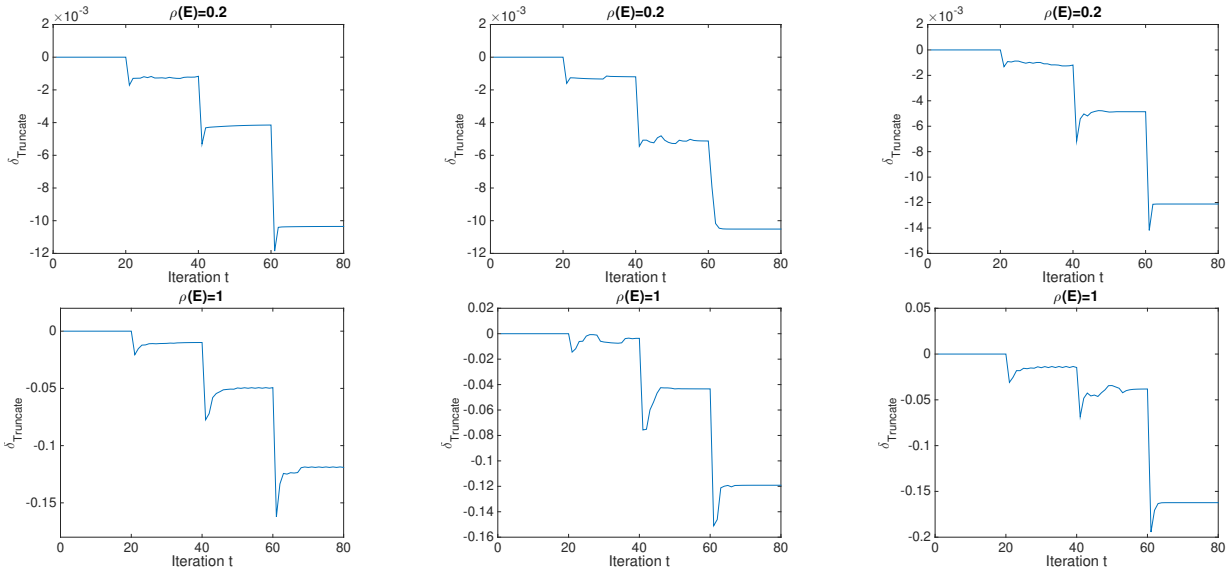
A3. Experiment setup in K-subspace clustering. In all experiments, KSS/RKSS runs 20 iterations in each trial. For EKSS/ERKSS, we repeat KSS/RKSS $B = 100$ times and take the ensemble of the co-association matrix. No thresholding is applied to the co-association matrix. For KSS and RKSS, we run the algorithms 100 times with random initialization, and the clustering errors are shown in the histograms fig. 3.6. We also record the smallest errors among the 100 trials in table 3.6.

For the synthetic dataset, we use the following parameters: the number of clusters and the number of subspaces $S = 5$, the ambient space dimension $D = 100$, the subspace dimension $d = 7$, and the number of true points in each cluster $N_s = 50$.

For the USPS dataset, we use the processed data from [17]. We subsample $n = 9298$ from the USPS handwritten digit database, and each image is 16×16 pixels ($p = D = 16 \times 16 = 256$). We use $S = 10$, $d = 15$. For RKSS and ERKSS, we use $k_i = 200,\ \forall i$, as the sparsity level.

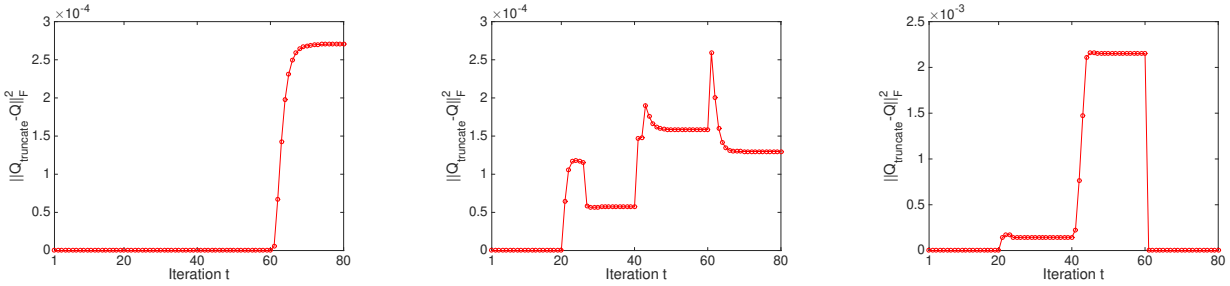
For the COIL dataset, we use the processed data from [18]. The dataset contains 20 objects, and each object has 72 images taken from different rotation angle ($n = 72 \times 20 = 1440$). The size of each image is 32×32 pixels ($p = D = 32 \times 32 = 1024$), with 256 grey levels per pixel. We use $S = 20$, $d = 9$. For RKSS and ERKSS, we use $k_i = 200, \forall i$, as the sparsity level.

A4. Additional numerical experiments.



(a) completely overlapping (b) partially overlapping (c) non-overlapping

Figure 3.7: **Empirical estimate of the truncation error.** (a) (b) (c) correspond to three different cases of support sets of the leading eigenvectors. We plot $\hat{\delta}_{\text{Truncate}}$ vs. iteration number t to give an empirical estimate of the truncation error. We use perturbations with $\rho(E) = 0.2$ and $\rho(E) = 1$ while keeping $\rho(\bar{A}) = 1$. The truncation level starts at p (nothing is truncated) and is decreased at iteration 21, 41 and 61.



(a) completely overlapping (b) partially overlapping (c) non-overlapping

Figure 3.8: **Effect of the extra truncation step on TOrthT.** (a) (b) (c) correspond to three different cases of support sets of the leading eigenvectors. We plot $\|Q_{\text{truncate}} - Q_t\|_F^2$ vs. iteration number t to measure the difference between the output matrix before and after the truncation step. The truncation level starts at p (nothing is truncated) and is decreased at iteration 21, 41 and 61.

Chapter 4

APPLICATION: GRAPHICAL MODEL FOR NEURONAL NETWORK OF C. ELEGANS

We propose an approach to represent neuronal network dynamics as a Probabilistic Graphical Model (PGM). To construct the PGM we collect time series of neuronal responses produced by the neuronal network and use singular value decomposition to obtain a low-dimensional projection of the time series data. We then extract dominant patterns from the projections to get pairwise dependency information and create a graphical model for the full network. The outcome model is a functional connectome which captures how stimuli propagate through the network and thus represents causal dependencies between neurons and stimuli. We apply our methodology to a model of *Caenorhabditis elegans* (*C. elegans*) somatic nervous system to validate and show an example of our approach. The structure and dynamics of *C. elegans* nervous system are well-studied and a model that generates neuronal responses is available. The resulting PGM enables us to obtain and verify underlying neuronal pathways for known behavioral scenarios and detect possible pathways for novel scenarios. In this chapter, we focus primarily on the application of dimensionality reduction techniques, and more details of the background, motivations and discussions can be found in [88].

4.1 Introduction

Probabilistic Graphical Model (PGM) is a statistical model in which a graph maps the conditional dependence structure between multiple random variables [76, 77, 98]. Construction of PGM has been shown as an effective methodology for retrieving dominant trends and analyzing events with very large number of dependent variables. Applications of graphical

models revolutionized a number of fields such as medical diagnosis, natural language processing, and computer vision. The nodes of the PGM correspond to variables in a domain, and edges correspond to probabilistic interactions (conditional dependencies) between the variables. The most commonly used graphical models are Bayesian networks and Markov random fields. Bayesian networks are directed graphs parameterized by conditional probability distributions (CPDs), whereas Markov random fields are non-directed graphs parameterized by factors.

PGMs were successfully applied to problems for which direct inference is intractable [76]. It is thereby appealing to apply them in the context of neuronal networks functionality. However, classical approaches for learning PGM structure are designed for discrete variables and are not compatible with neuronal networks consisting of dynamic neurons interacting through dynamic connections. Both neurons and their connections are typically modeled as nonlinear processes. A possible adaptation of a neuronal network to a statistical model, which captures functionality, is to consider each neuron as a random variable that takes values representing the states of the neuron. For simplicity it is often assumed, and here we assume it as well, that each neuron activity is binary-valued, with 0 being the inactive state and 1 being the active state. The PGM of a neuronal network is thereby a graph with neurons being the nodes and their dependencies being the edges.

There are two main difficulties in learning a graphical model from dynamics. (i) Difficulty in estimating input-output correlations for a network whose responses are time-dependent. (ii) High dimensionality and complexity of neuronal networks typically incorporating recurrent structures. These factors make statistical inference hard to realize. Relevant work has been done in dynamic networks using either experimental data, such as [2, 49, 75], or simulated data, such as [57]. In both cases, "snapshots" that record network dynamics are used to analyze time-series data. For estimating input-output correlations, statistical methods are applied to measure pairwise node correlation. For example, Honey et al. [57], proposed to use transfer entropy (TE) to capture patterns of directed interaction and information flow between pairs of nodes. Butte and Kohane [15] used entropy of gene expression patterns and

the mutual information between RNA expression patterns for each pair of genes to compute pairwise mutual information. To address the problem that the network responses are time-dependent, works assuming that the underlying network is time-invariant, such as [7, 49], or estimate a sequence of graph structures such as [2, 75] have been proposed.

It is also possible to use a machine learning approach, such as to sample the neuronal network multiple times as training data, and to evaluate candidate models according to a scoring function and search for the optimal model [77]. One of the drawbacks of score-based approaches is that in high dimensional and cyclic neuronal networks, the problem of finding an optimal solution becomes NP-hard. Additional approaches have been employed for Bayesian network modeling for human functional network analysis. Zhang L. et. al. and Zhang J. et. al. introduced Dynamic Bayesian Networks, Dynamic Bayesian Variable Partition Model and Dynamic Causal Modeling for fMRI time series snapshots [150, 151]. In these models an underlying assumption for the time series is a Gaussian model, which does not apply for attractor neural dynamics generically appearing in the neuronal networks and require extensive computational cost. While more efficient approaches have been introduced, they still incorporate non-generic assumptions such as low-rank and sparsity of the responses [103].

Here we propose a different approach that circumvents these complexities. Since neurons are random variables, mathematically our approach is based on the concept that if we stimulate them using independently and identically distributed (iid) process, then the construction of the dependencies would be based on measuring how network response deviates from the stimuli distribution. Here we use a simple distribution: the delta distribution, which is single neuron excitation, and record network response to each stimulus. Such an approach is simple to implement with simulated neural dynamics and potentially realizable with the rapid advance of optogenetic technologies to record and stimulate networks at a single neuron resolution [148]. While we employ a single excitation in our proposed algorithm, theoretically the approach is not limited to this stimulation and other distributions of the stimuli can be used. The required assumption is that stimulation of each neuron is

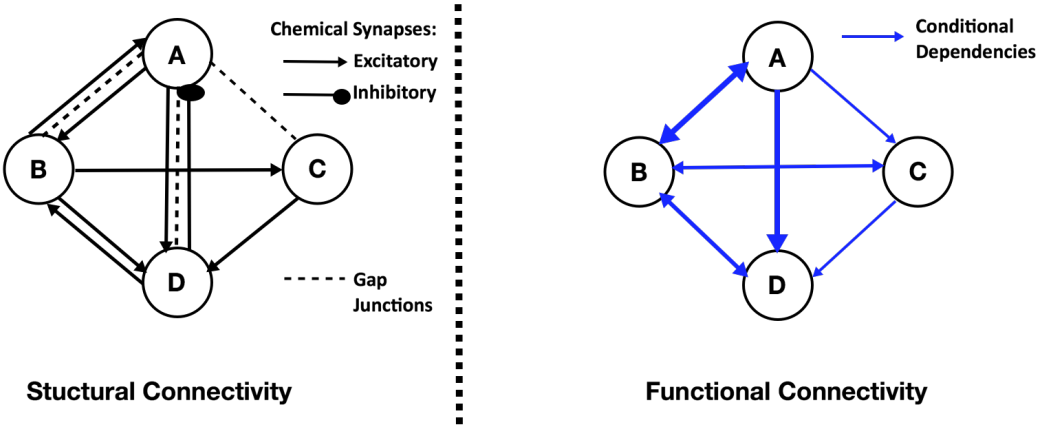


Figure 4.1: **Representation of a neuronal network as a graphical model.** Left: Example of structural/ anatomical connectivity map in which nodes denote neurons and edges map connections, e.g., chemical synapses and electrical gap junctions. Interactions between neurons produce a nonlinear network which dynamically transports stimuli to neuronal behaviors. Right: Example of PGM constructed from the neuronal network governed by the structural connectivity map and nonlinear dynamics. In the PGM nodes are random variables corresponding to neuronal states and edges are conditional probabilities. PGM structure captures functionality of the network and hence typically different from the anatomical connectivity map.

independent and the outcome PGM superimposes response dynamics to independent experiments into a model. To construct the graph, our key idea is to learn the dependencies from low-dimensional projections of time-series instead of learning from sampled data directly. The low dimensional representations capture the dominant dynamics of the network and thus reduce the complexity of the learning algorithm and the computational cost. To be able to capture the dominant patterns we sample the network over a sufficiently long period such that we have captured the typical and dominant dynamics, i.e., the network converged to attractor dynamics, if such dynamics exist. Using this approach we construct a PGM (a Bayesian network) which maps the functional connectivity between the neurons. The model can be used to 'query' the system given evidence regarding activity of some neurons or infer typical relation between activity of neurons. We define these concepts and explain the procedures for these tasks in Section 4.2.

We apply our method to the neuronal network of Caenorhabditis elegans (*C. elegans*) to validate the PGM. *C. elegans* is a well-studied organism; The somatic nervous system of *C. elegans* consists of 279 neurons and the wiring diagram between these neurons was compiled by Varshney et al. consisting of 6393 chemical synapses, 890 gap junctions, and 1410 neuro-muscular junctions [134]. In addition, experimental electrophysiological and optogenetic techniques for stimulating and recording neural activity in-vivo and in-situ have been introduced for *C. elegans* [52, 72, 100]. The connectome representing weights of dynamic interactions between neurons combined with a biophysical model of neural dynamics and their interactions enables us to simulate the full nervous system model [73, 78, 117, 143]. In the following sections, we construct a Bayesian network that represents the functional connectivity of the neuronal network of *C. elegans* and verify the results with experimental data.

4.2 Construction of Graphical Model from Network Dynamics

We first introduce the components of the Bayesian network in the context of neuronal networks. We then illustrate how conditional probabilities are assigned from the simulated neural dynamics. Finally, we show how computed probabilistic dependencies are used for the construction of a Bayesian network.

4.2.1 Representing Neuronal Distributions with Bayesian Networks

A Bayesian network is a representation of a joint probability distribution with graph structure $G = (V, E)$ and conditional probability distributions θ. The graph structure G is a directed acyclic graph (DAG) which vertices, V, correspond to random variables $\{X_1, X_2, \cdots X_n\}$, and edges, E, correspond to connections between random variables, i.e., conditional dependencies. For each variable, θ describes the conditional probability distribution given its parents in G. By applying the probability chain rule and using properties of conditional

independence, joint probability distribution can be decomposed into the product form

$$P(X_1, \cdots, X_n) = \Pi_{i=1}^{n} P(X_i | \mathbf{Pa}^G(X_i))$$

where $\mathbf{Pa}^G(X_i)$ is the set of parents of X_i in G.

PGM is based on this property and graphically represents probability distributions into a graph with parent and children nodes. For neuronal networks, each X_i corresponds to a individual neuron taking discrete values. Thus we can represent $P(X_i|U_1, \cdots, U_k)$ as a table that specifies the probability of values for a neuron X_i for each joint assignment to its parent neurons $U_1, \cdots, U_k$ [49]. In the simplest case we only consider pairwise conditional probability $P(X_i = s_l | X_j = s_k)$, where neuron X_j receives input that drives it to a neural state s_k out of set of states $s = \{s_1, .., s_k, .., s_m\}$, and we would like to estimate the probability of neuron X_i being in state s_l subject to evidence that X_j is in state s_k.

The structure of the PGM is designed to conveniently map the statistical dependencies to allow to 'query' the graph for single neuron excitation in an efficient way. The query procedure, called *posterior inference*, uses the constructed PGM to provide information about probabilities of particular variables and their states, taking into account evidence regarding the states of other variables, i.e., conditional probability. In particular, there are two types of posterior inference tasks in the context of network pathways that can be answered using a graphical model. The first task is to infer conditional probabilities of the type $P(Y|E = e) = \frac{P(Y,e)}{P(e)}$, where the probability distribution over the values $y \in Y$, conditioned on $E = e$ is inferred. For example, in such a case, pathways from upstream neurons belonging to the set E are sought. The second task is to infer the maximum a posteriori (MAP) probability: $\arg\max_y P(y|e)$, where the most likely assignment to the variables in Y is sought given the evidence $E = e$. In this task, dominant pathways, which lead to a set of downstream neurons and reveal upstream neurons that activate them, are sought. PGM is capable of discovering the unstructured information within the distributions since it turns complex distributions into structured information that can be analyzed effectively and efficiently using

statistical tools. The benefit of using a PGM as the functional connectome is that posterior inference can be performed efficiently and circumvent the complexities in direct inference of response pathways in dynamic neuronal networks. In particular, posterior inference reveals the relations and pathways between known stimuli and downstream neurons or allows to discover the stimuli that would trigger downstream neurons.

4.2.2 Collecting Data from Simulated Dynamics

We propose to find conditional probabilities for all pairs of neurons by simulating the network and recording data in snapshot matrices. The snapshot matrix represents finite time-series of whole network dynamics for a particular input to a single neuron. The matrices are computed by injecting a constant input into every neuron in the network independently, thus resulting in total n matrices for the network of n neurons. Network dynamics are modeled by a set of dynamic equations, e.g., conductance-based model, which describe the biophysical processes of neurons and interactions between neurons. A snapshot matrix has the structure $S = [S(t_0)\ S(t_1)\ \cdots S(T)]$ of dimension $n \times T$, where n is the total number of neurons in the network and T is the length of the time span.

4.2.3 Dimension Reduction

Since the the snapshot matrix corresponding to a single neuron excitation has dimension $n \times T$, we wish to first compress it into a single vector in $\mathbb{R}^n$ in order to interpret it as conditional probabilities. We process the time series data by projecting it into a lower dimensional space using singular value decomposition (SVD). The benefits of SVD include reducing dimensionality of the ambient space and thus reduction of computational cost, removing co-linearity and feature redundancy. Furthermore, when SVD-based approach is applied to multi-node time series network dynamics it decomposes the data into spatial modes and their associated time dependent coefficients. The spatial modes are orthogonal vectors representing activity patterns of the nodes [118]. When the recorded data has been conditioned on a particular event, each spatial mode vector is effectively a vector containing

a response score for each node. Since the spatial modes are ranked by singular values to reflect their dominance, the combination of dominant modes will include the most dominant combination of scores as a response to the event. This is the property that we use here to estimate dependence between stimulation and network response. Such an approach is more beneficial than direct estimate of statistical dependence (e.g. correlation) since it separates spatial and time dependent data and ranks the spatial modes. Furthermore, these properties of SVD ensure that the estimation of the dependencies is robust with respect to sample time as long as they have captured the full dynamics to which the network converges.

More precisely, SVD of a $n \times p$ matrix S has the form

$$S = U \cdot \Sigma \cdot V^*,$$

where U and V are $n \times n$ and $p \times p$ unitary matrices, with the columns of U spanning the column space of X (spatial neuronal modes) and the columns of V spanning the row space (time dependent coefficients) [55]. Σ is assumed to have its diagonal entries σ_j which are non-negative and in descending order, i.e., $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$, where $m = \min(n, p)$. Once the original matrix S is decomposed, U_j (column vector of U) and V_j (column vector of V) record the the mode corresponding to singular value and temporal coefficients corresponding to the σ_j, respectively. Each singular value σ_j in Σ corresponds to the significance of the corresponding mode. Using SVD we can find a lower dimensional representation for the matrix S, which captures the dominant features of neural dynamics. Dimension reduction is performed by retaining $k-$dimensional subspace, where $k < n$, spanned by k-modes corresponding to top k singular values.

When using SVD to obtain a low-dimensional representation of the data we need to retain enough modes to approximate the data. In practice, energy criterion on singular values is often used. It specifies the amount of energy included in the chosen modes. For example, typical energy criterion requires 95% of the energy in Σ. That is, the sum of the squares of the retained singular values should be at least 95% of the sum of the squares of all singular values. For snapshot matrix which network responses reach a fixed point, the first dominant mode

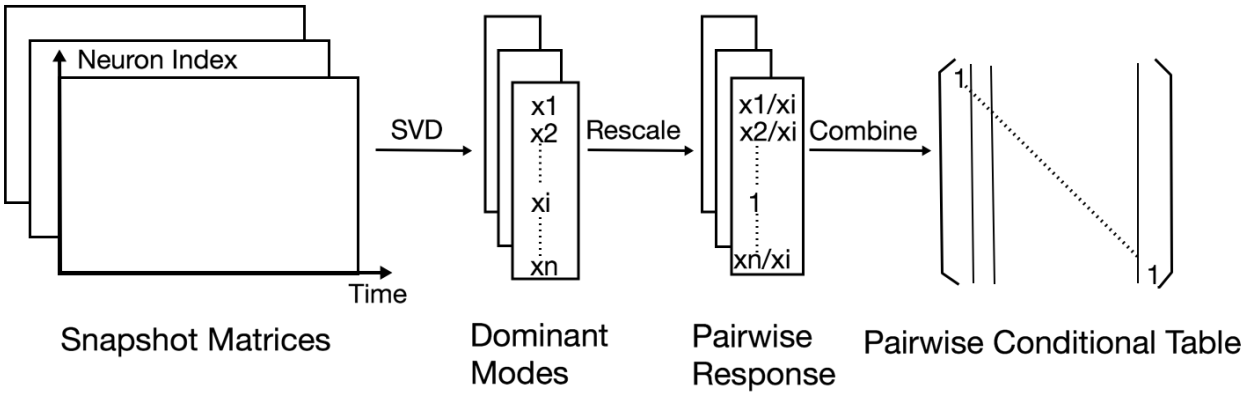


Figure 4.2: **Construction of dependencies between nodes.** By injecting input stimuli into each neuron, we obtain a series of snapshot matrices that record network dynamics. By decomposing the snapshot matrices using SVD we obtain the decomposition modes and compress them into a single vector for each matrix. We then normalize the vector according to its input neuron and get pairwise responses, which constitute the adjacency matrix of pairwise conditional probabilities.

would be sufficient to represent the fixed point. For oscillatory networks, the first two modes together represent oscillatory dynamics. Instead of using the classical k-rank approximation, we propose a new approach that takes the linear combination of modes according to their significance, which results in a one-dimensional column vector. Specifically, we use the sum of all modes weighted by singular values as a one-dimensional representation of the original $\mathbb{R}^{n\times p}$ data:

$$\sum_{i=1}^{n} \frac{\sigma_i^2}{\sum_{j=1}^{n} \sigma_j^2} |U_i| \in \mathbb{R}^n.$$

To improve interpretability and to further reduce the impact induced by noise, we also apply the Truncated Orthogonal Iteration (TOrth) on the covariance matrix to obtain sparse principal components. In such case, U_i is the sparse vector obtained by TOrth with most of its entries being zeros.

Algorithm 6 Constructing Dependencies

$n \leftarrow$ number of *neurons*, $t_0 \leftarrow$ *starting time*, $T \leftarrow$ *final time*, $k \leftarrow$ *time step*

for $i \in 0, 1, \cdots, n$ **do**

Input$[i] = 1$

$X = run\_network(t_0, T, k, \text{Input})$

$[\hat{U}, S, V] = \mathbf{SVD}(X)$

$U = \mathbf{TOrth}(X^T X, \hat{U}, k_i = 10)$

mode $= \sum_j |U_j| \frac{S_j}{\sum_k S_k^2}$

dependency vector = mode/mode$[i]$ and normalize

end for

4.2.4 Visualization: Construction of a Bayesian Network

The vectors extracted from snapshot matrices are used to obtain pairwise dependencies. Each snapshot matrix corresponds to a stimulation of a single neuron. We assume the stimulated neuron is driven into an active state and thus the snapshot matrix reflects the response of other neurons, which we transform to probabilities, conditioned on stimulated neuron being activated. We achieve this by normalizing each mode according to the response of its input neuron, X_i, and to ensure that they are positive and sum up to 1. We interpret the result as the conditional probability $P(X_j = 1|X_i = 1)$ and store it into the i^{th} column of the conditional probability table. The PGM itself consists of probabilities and thus does not indicate whether it is positive causality or negative (anti-) causality. Additional descriptive states of neuron activity could be added in the future by extending neural states, for example, each node state would be active, anti-active and inactive. $P(X_j = 0|X_i = 1)$ can be calculated by $1 - P(X_j = 1|X_i = 1)$, and we assume that $P(X_j = 0|X_i = 0) = 1$ and $P(X_j = 1|X_i = 0) = 0$, i.e. a node cannot be active if there is no input into the network.

The resulting table is a $n \times n$ dependency matrix, which records the complete pairwise dependencies for the entire neuronal network. The matrix itself contains useful informa-

Algorithm 7 Constructing Trees

$maxLayer \leftarrow$ maximum layer of the tree

$threshold \leftarrow$ the threshold for a node being considered as a child

Function $ExtendTree(child, Plist, count)$

if $count > maxLayer$ **then**

$count = 0$

Return

end if

if $parent \in PList$ **then**

Return

end if

if $parent \notin PList$ **then**

$PList.add(parent)$

$count \leftarrow count + 1$

for $i \in (0, n)$ **do**

if $i \notin PList$ **and** $Prob[parent, i] > threshold$ **then**

$childList.add(i)$

end if

end for

for $child \in childList$ **do**

$ExtendTree(child, Plist, count)$

end for

end if

EndFunction

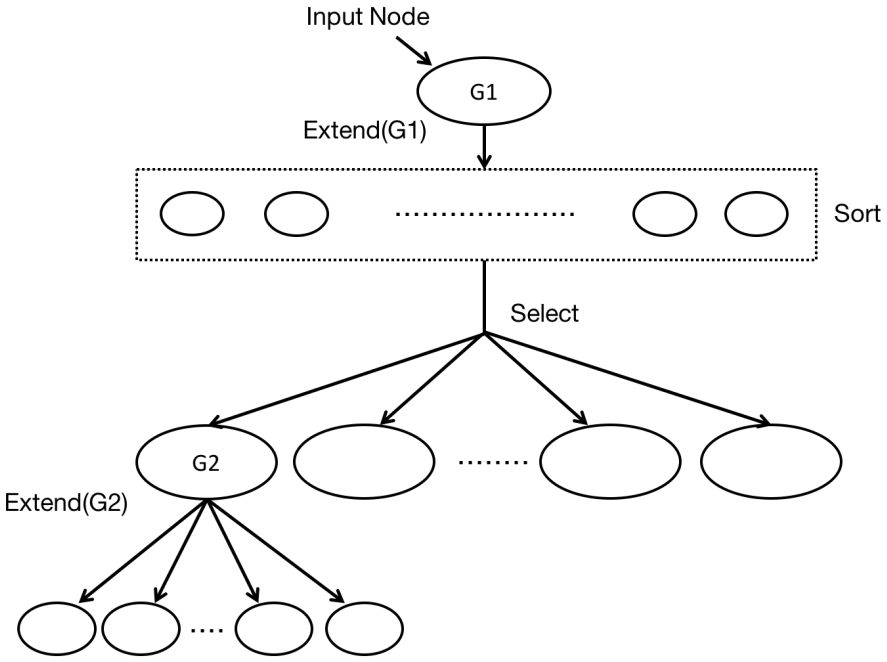


Figure 4.3: **Tree construction from adjacency matrices.** We start with the input node, and sort the other nodes according to their probabilities conditioned on the input node. We ignore nodes with conditional probabilities lower than the threshold, and select the top nodes with the highest conditional probabilities. A constraint can be imposed to limit the number of children that a node can have. From there we extend each node recursively, until it either reaches the maximum tree depth, or reaches neurons in a particular set (e.g. motor neurons). The threshold, maximum tree depth and maximum number of children are pre-set parameters to limit the size of the tree.

tion about the network, and by constructing the PGM we can further extract and visualize this information. Since any joint distribution can be decomposed into a product of conditional probabilities, the dependency matrix when transformed into pairwise conditional distributions records the full joint distributions encoded by the PGM. A natural graphical representation of a neuronal network is a directed cyclic graph since majority of networks incorporate multiple pathways and recurrence. Inference on such graphs is hard to perform as the existence of cycles often leads to non-convergence of the probabilities [76]. We choose to eliminate cycles and propose a Restricted Iterative Deepening algorithm (RID) that builds

a tree for each posterior inference problem on the graph. The tree structure ensures that each neuron has at most one parent and therefore acyclic by construction. A directed spanning tree can be constructed by choosing an arbitrary root and direct edges away from the root [76]. As our goal is to identify functional sub-circuits within the network and their component neurons, tree-structured graph allows us to capture the propagation of neural pathway subject to stimuli.

For constructing the tree, the RID algorithm that we implement starts with designated input neuron and extends the tree according to pairwise conditional dependencies. The process continues until the tree either reaches to the desired set of output neurons, e.g., motor neurons or reaches the maximum depth of the tree, which can be imposed by the user. If using the full PCs, we can restrict the width of the tree (the number of children that a parent can have) by imposing a threshold on the conditional dependencies to avoid the tree growing too wide. If using sparse PCs instead, this thresholding step can be avoided. For each parent neuron X_i, we explore all of its children neurons X_j that have pairwise dependencies $P(X_j|X_i)$ and sort them in descending order.

4.3 Application to Neurobiological Dynamic Connectome

4.3.1 Neuronal Network of Caenorhabditis elegans

The nematode *Caenorhabditis elegans* nervous system is a well-studied system, consisting of 302 neurons identifiable and consistent across individuals. The connections between the neurons are comprised of chemical synapses and gap junctions, whose wiring diagrams, i.e., connectomes, are nearly fully resolved from serial section electron microscopy [134]. In addition to the connectomes, the dynamic model which describes the biophysical processes between neurons has been introduced [78]. Specifically, the dynamical model of the nervous system is governed by a system of nonlinear differential equations:

$$C\dot{V}_i = -G^c(V_i - E_{cell}) - I_i^{Gap}(\overrightarrow{\mathbf{V}}) - I_i^{Syn}(\overrightarrow{\mathbf{V}}) + I_i^{Ext}$$

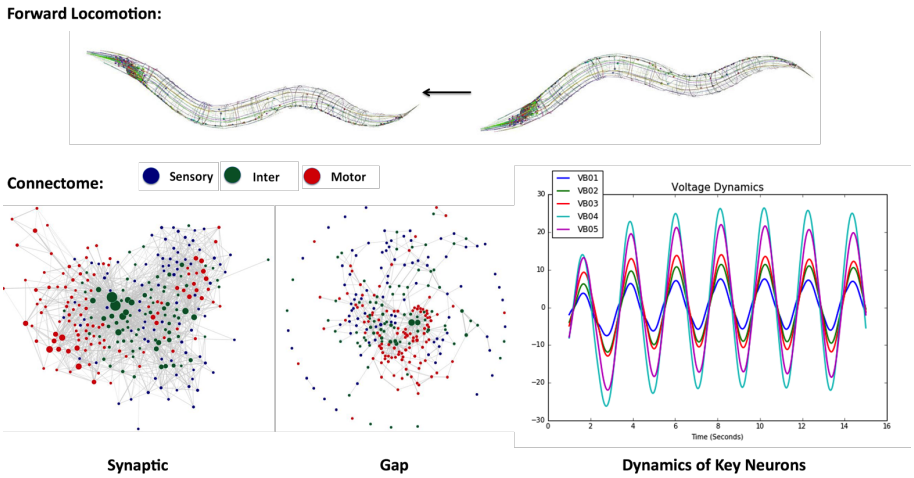


Figure 4.4: **Two layers of the neuronal network of C. elegans.** Top: An example of forward locomotion induced by two layers of the neuronal network of the worm. The first layer (Bottom Left): Connectome of *C. elegans*, consisting of 297 somatic neurons, 6393 chemical synapses and 890 gap junctions. The left part of the connectome shows the chemical synapses between neurons, and the right part shows the gap junctions. The second layer (Bottom Right): Neural dynamics modeled by differential equations. Here we show voltage oscillations of the motor VB group. Combining the two layers we achieve the dynome, a dynamically evolving network, which is the foundation for constructing the PGM.

where C is the whole-cell membrane capacitance, G^c is the membrane leakage conductance and E_{cell} is the leakage potential. I_i^{Ext} is the external input current injected to the i^{th} neuron. I_i^{Gap} and I_i^{Syn} correspond to the input currents modeling gap junctions and synapses, respectively. More details on the biophysical model can be found in [78].

Combining the connectome and the dynamical model constitute the dynome of *C. elegans*. Incorporating both the layers of connectivity and dynamic biophysical processes, *C. elegans* dynome models the nervous system functionality and processing of stimuli that it performs. Indeed, when provided with arbitrary input stimuli *C. elegans* dynome is capable of producing various forms of characteristic dynamics such as static, oscillatory, non-oscillatory and transient voltage patterns consistent with experimentally observed ones [78, 79]. These

simulated dynamics indicate that *C. elegans* dynome is a valuable model for the worm's nervous system and thus a suitable foundation for the construction of its probabilistic graphical model.

4.3.2 Constructing Dependencies

We apply our method to the neuronal network of *C. elegans* nematode by injecting scaled input current into each of the $n = 279$ neurons independently using the Neural Interactome platform (see [73] for further details). We run the simulations for 15 seconds with a time step of $0.01sec$ and record the dynamics of all neurons in snapshot matrices with each snapshot matrix S of dimensions $n \times T = 279 \times 1501$. We then subtract the activation threshold for each neuron from the simulation and exclude the initialization phase of the network (1 *sec*). We then obtain 279 response vector representations of all neurons to the stimulation of the input neurons by performing SVD on each snapshot matrix and taking the weighted sum of all modes as described in Section 4.2. We also apply TOrth on the covariance matrix to obtain sparse principal components. Each of these vectors is normalized according to input neuron response, which yields the conditional probability $P(X_j = 1|X_i = 1)$ as elements of the vector. The vectors are stored as column vectors of the conditional probability table, resulting in a 279×279 dependency matrix, which records the complete pairwise dependencies of the nodes in *C. elegans* neuronal network.

4.4 C. elegans Functional Connectome Represented by PGM

4.4.1 Inference of Functional Sub-circuits

While global features of functional properties could be obtained through visualizing the PGM dependency matrix, additional analysis is needed to retrieve specific features such as pathways of neural information flow from a neuron of interest or a cluster of neurons. Such pathways can be inferred through posterior inference traversing the PGM. We pose two types of inference queries on the pairwise conditional table. (i) Given a set of input

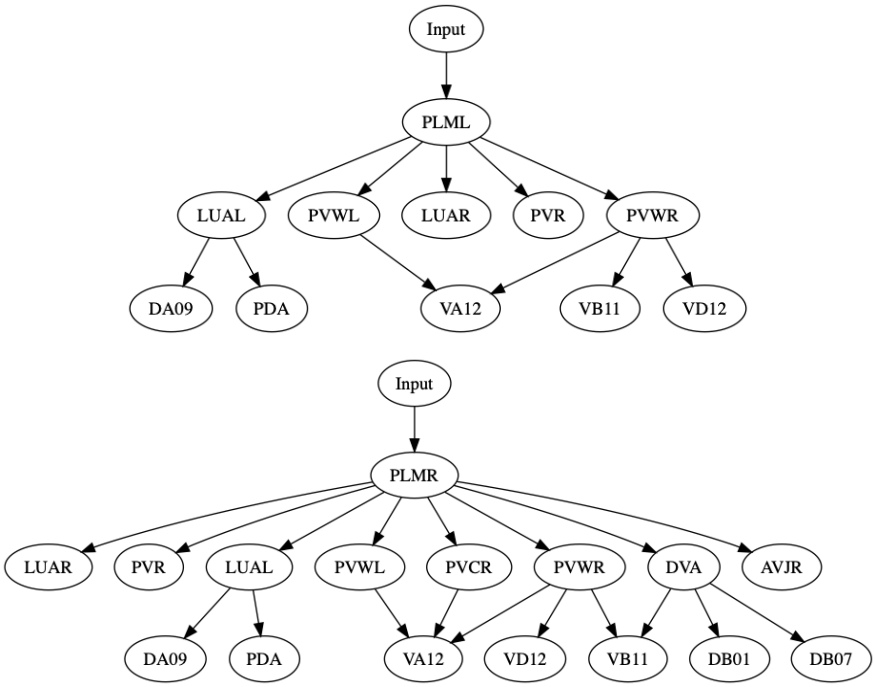


Figure 4.5: **Example of the generated trees.** We construct the trees with parents being the sensory neurons PLML and PLMR, known to trigger forward locomotion upon posterior touch. The resulting trees highlights the key inter and motor neurons reported in experiments and literature.

neurons: What is the set of downstream neurons most likely to be activated? For example, such inference is relevant to identify inter and motor neurons activated by a set of sensory neurons. (ii) Given a set of downstream neurons: What are the subsets of upstream and midstream neurons most likely to activate the downstream neurons? Such inference can reveal, for example, inter neurons and sensory neurons that are most likely to activate motor neurons. Both of these questions can be answered by constructing a graphical model for the network and performing posterior inference. While the first problem follows stimulus propagation forward, the second problem is an inverse problem, often called Maximum A Posteriori (MAP) problem, and requires to examine and optimize over many probable inverse propagation sequences. We summarize our key results below and include individual neuron trees in Figure 4.5.

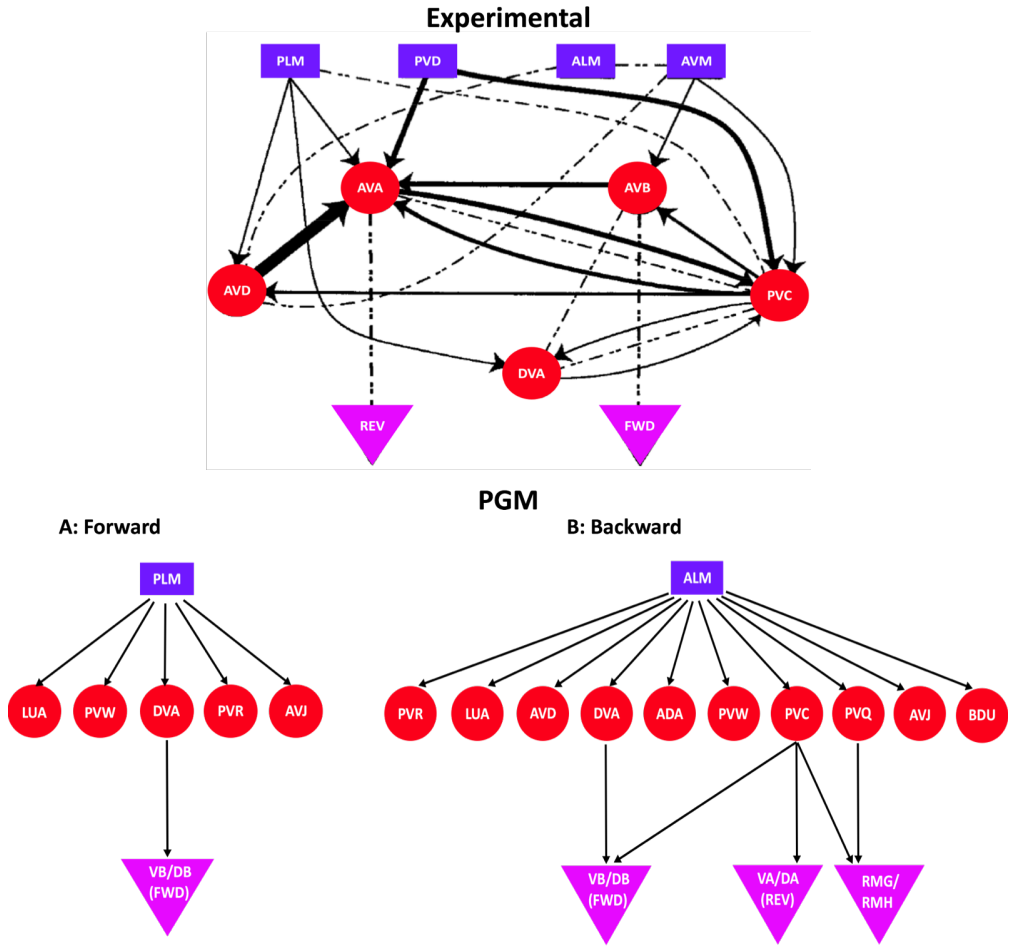


Figure 4.6: **Sub-circuits that lead to forward and backward locomotion in *C. elegans*.** Top: Experimentally proposed connectome sub-circuit of forward and backward locomotion (sensory neurons are blue rectangles, inter neurons are red circles and motor neurons are purple triangles). Lines with arrows refer to chemical synapses and dashed lines refer to gap junctions. Reproduced from [143]. Bottom: sub-circuits inferred from PGM. A: Forward locomotion induced by injecting input into sensory neurons PLML and PLMR, known to trigger forward locomotion upon posterior touch. B: Backward locomotion induced by injecting input into ALML and ALMR, known to trigger backward locomotion upon anterior touch.

To infer downstream neurons, we start with a set of input nodes and use the RID algorithm to construct a response tree for each input node in the set. For each parent node we explore its children in descending order. The number of children is restricted to be less than

10 by the TOrth algorithm. For simplicity, we restrict the tree to have a maximum depth of 3: one layer of sensory, inter and motor neurons each to keep the flow from sensory to motor neurons straightforward. In particular, we focus on well-known experimental scenarios such as forward and backward locomotion, as shown in Figure 4.6. Specifically, we investigate forward locomotion triggered by posterior touch: activation of sensory PLML/R neurons results in the activation of the DB and VB groups, associated with forward locomotion. We investigate this sub-circuit by constructing a tree with input nodes PLML and PLMR (labeled PLM). The set of inter neurons being activated includes LUA, PVR, PVW, AVJ, DVA and PVC. DVA neuron leads to DB01 motor neuron in group B (motor neurons group associated with forward movement [24]), which in turn activates most of the motor neurons in DB and VB groups. These results are consistent with functional stimulus propagation flow reported in the literature.

We also examine anterior touch, triggered by stimulation of ALML and ALMR (labeled ALM) neurons, which leads to backward locomotion. ALM activation leads to a larger set of inter neurons. The set contains all inter neurons associated with forward locomotion and additionally includes AVD, ADA, PVQ and BDU, many of which are indeed associated with backward movement. Indeed, AVD is experimentally known to be associated with backward locomotion, and PVC plays an important role in both forward and backward motion [5,143]. Stimulus flows from PVC neuron to several motor neurons, especially neurons in group A, i.e., DA, VA (experimentally identified associated with backward movement), as well as in DB and VB groups.

Compared with the circuit extracted from the synaptic and gap connectomes in Figure 4.6 top from [143], the PGM successfully recovers neurons participating in this circuit and separates them into two functional sub-circuits. In each of the sub-circuits, motor neurons are reached to induce locomotion. As can be seen from the circuit sketch, separation of it into independent ones is nontrivial. The PGM also identifies inter neurons such as AVD, PVC, and DVA, while the hub neurons AVA and AVB are missing. A possible explanation is that individual activation of ALM or PLM is insufficient for AVA/AVB to reach a strong

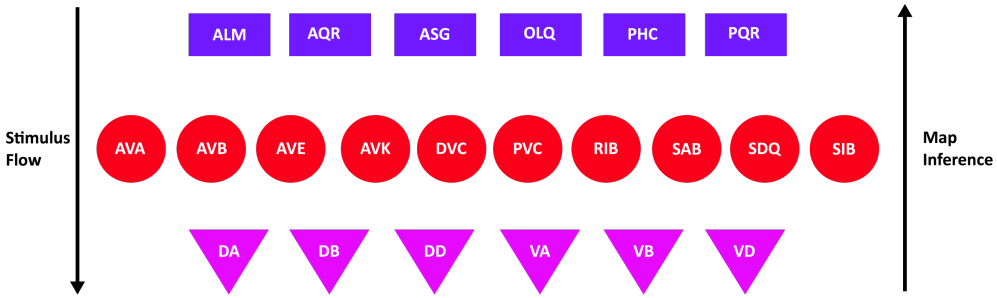


Figure 4.7: **MAP posterior inference: Reverse tracking of A and B motor groups associated with locomotion.** A group of neurons we identify using MAP inference. Given that a specific group of motor neurons are being excited, the set of inter and sensory neurons that leads to such excitation are shown. From top to bottom is the natural stimulus flow, i.e. sensory $\rightarrow$ inter $\rightarrow$ motor. From bottom to top is MAP inference.

state of excitation and they are activated through another stimulation/process.

We use MAP inference to perform reverse tracking of activating neurons of motor neurons associated with locomotion. We, thus, choose motor neurons in the ventral cord members of A and B group shown as purple triangles in Figure 4.7 and labeled as DA, VA, DB, VB, DD, VD. Posterior MAP inference traverses back the PGM to inter and sensory neurons, which are most likely to activate the given motor neurons. Specifically, we apply the RID algorithm and flipped conditional probabilities. Instead of sorting $P(X_i|X_j)$ we sort $P(X_j|X_i)$, with X_j being the parent node. As in forward RID, we limit the depth of the tree to be 3 and keep one layer of the motor, inter and sensory neurons each. We find ten inter neurons and six sensory neurons that most frequently appear in the reverse traversal paths. We list these neurons in Figure 4.7 as red circles (inter) and blue rectangles (sensory). Notably, most of sensory and inter neurons that MAP inference produces were indeed experimentally associated with locomotion, and these are neurons that we identified in PLM and ALM pathways in Figure 4.6. Furthermore, additional neurons known to participate in locomotion are identified as well. Inferred inter neurons now do include AVA and AVB, which were not present in forward inference from ALM and PLM, see Figure 4.6, emphasizing that when additional

paths are considered, these neurons do play a role in sensory-motor neural integration, but not in direct stimulation of ALM and PLM. The power of MAP inference is in its ability to associate additional neurons with the designated motor neurons, without any prior biological knowledge of the network. For example, additional inferred sensory neurons include AQR and PQR, known experimentally to influence locomotion. Their function is currently being studied and conjectured to be associated with oxygen sensation and avoidance [26]. Using MAP inference in PGM we, thereby, able to support this conjecture. The posterior inference from these neurons would provide more details into which pathways the stimulus from AQR and PQR is following to reach locomotion motor neurons.

4.5 Discussion

We presented a new approach to learn a graphical model (PGM) for a neuronal network. Two key components in our approach allow us to construct the PGM that captures network functionality. The first component is that we capture network responses to independent stimuli (single neuron stimulation). The second component is the underlying dimension reduction techniques to obtain a low-dimensional representation of network responses to stimuli. We consider pairwise dependencies instead of full dependencies on the whole network and greatly reduce the number of parameters. This approach is computationally efficient and performs posterior inference when combined with a method for traversing the PGM to produce response trees (RID).

Our framework assumes that the underlying network structure remains isotropic over simulation time, that is, the anatomical connectomes and pairwise conditional probabilities $P(B|A)$ remain similar if the simulation is continued for a longer time, unlike networks which undergo rewiring during simulation time studied in previous works [2, 75]. From dynamical point of view, we expect our approach to perform best when the simulated network dynamics converge to low dimensional attractors. Indeed, for *C. elegans* network, stimuli induced low dimensional attractors were shown to exist in the network and simulation of the network for an efficiently long period (15 *sec*) is expected to capture attractor dynamics [78].

Dimension reduction employed here is based on Singular Value Decomposition (SVD) and Truncated Orthogonal Iteration (TOrth). We collapse all SVD modes into a single vector by computing a weighted sum according to singular values of all modes. In many simulations of *C. elegans*, the first two modes take up more than 90% of the energy and the first four modes take up more than 99%. This is consistent with observation that behavioral maneuvers are spanned by several modes [121]. Indeed, we observe only a minor difference between the weighted sum of all modes compared to taking the weighted sum of the first two modes. As expected, we do see a significant difference between using the weighted sum of first two modes and just the dominant mode, indicating that attractors are spanned by at least a two-dimensional space. By using sparse principal components obtained by TOrth as opposed to the full vectors, we are able to highlight the important connections and automatically limit the width of the resulting trees without imposing extra hyperparameters.

We describe how to apply these techniques to neuronal model of *C. elegans* nervous system whose anatomical connectomes and dynamics have been resolved. The application to *C. elegans* neuronal activity identified key neurons and sub-circuits without any prior knowledge. In particular, our findings suggest additional examinations of functional roles for some of the neurons outlined by PGM. Ultimately, determination and validation of the roles of these neurons should be performed in experimental setting or by coupling neural dynamics with bio-mechanical models of muscles and body. Furthermore, the constructed PGM can be used for extraction of additional pathways associated with stimuli and behaviors that we did not consider here. Notably, the pathways inferred using the PGM indicate that if particular neuron is excited electrophysiologically or optogenetically, the neurons included in the associated pathway tree will be most responsive to this excitation. While we applied the methodology to construct the PGM from simulation-driven data, similar approach could be implemented in experimental setting using single neuron clamping and multi-neuron imaging network dynamics. Likewise, the methodology introduced here can be applied to other network models and organisms.

Chapter 5

APPLICATION: DATA-DRIVEN ENSEMBLE FOR COVID-19 FORECAST

5.1 Introduction

Coronavirus disease 2019 (COVID-19) has developed into a global pandemic since late 2019. From the initial stage of its spread, many organizations and institutions have started developing forecasting methods and made these forecasts available to public. Starting in April 2020, the US COVID-19 Forecast Hub (https://covid19forecasthub.org/) have collected forecasts from more than 90 different academic, industry, and independent research groups. Previous research has suggested that combining forecasts from multiple models into a single ensemble forecast can increase the robustness of forecasts [8, 28, 131]. The US Centers for Disease Control and Prevention (CDC) adopts an equality weighted average of available methods as an ensemble model [110], and it has been shown that this simple ensemble achieves better accuracy consistently compared to individual models [31]. In this project, we utilize a data-driven method to generate the ensemble weights to linearly combine the individual models, and explore various approaches to improve the ensemble results.

In the data-driven process, we use a set of metafeatures calculated from time series of COVID-19 death rates and use them as inputs to a machine learning model. Each forecasting method is treated as a class, and the output is a set of weights to linearly combine the classes. For the meta-learner, we train a gradient tree boosting model of XBGoost [27] based on different model's performance on the training data. The trained meta-learner is then used to output a set of weights for any out-of-sample data based on its metafeatures. The general framework is based on previous work to use time series features combined with meta-learning for forecasting, see for example [84, 97, 107, 112, 128]. However, directly applying these data-

driven approaches to the COVID-19 time series often results in a null model. There are several challenges and issues we need to consider before applying a data-driven ensemble approach to the COVID-19 time series:

- **Forecasting with insufficient training data:** For COVID-19 death rates in the US, we have limited time series available, which spans about 2.5 years. Each forecast date with enough forecast methods is a potential training data point. For each state, the training size is small compared to other common series such as weather forecasting and stock prices. We solve this problem by clustering states with similar time series. When predicting for a testing state, we use the states in the same cluster as training data and train a separate meta-learner for each cluster.

- **Inconsistency of forecasting methods:** The forecasts collected from institutions and organizations rely on different data sources and assumptions, and these assumptions are constantly changing. As a result, two similar time series that are highly correlated can have drastically different forecasts, even from the same method, thus causing confusion to the meta-learner. For this reason, we select a set of methods that consistently perform better compared to a naive baseline model and have shown to have robustness against data anomaly and across different time periods.

- **Preprocessing of meta-features:** The meta-features in [97] capture different characteristics of time series in general, but not all of them are essential to the COVID-19 dataset. We therefore apply the principal component analysis (PCA) and the Truncated Orthogonal Iteration (TOrth) before training to get a more compact set of features. PCA reduces the dimension of the feature space, and TOrth further introduces sparsity to the principal components so that only the important features are selected. When clustering similar time series, a relatively small feature space is strongly preferred since otherwise Euclidean distance will break in high dimensions and the curse of dimensionality will occur.

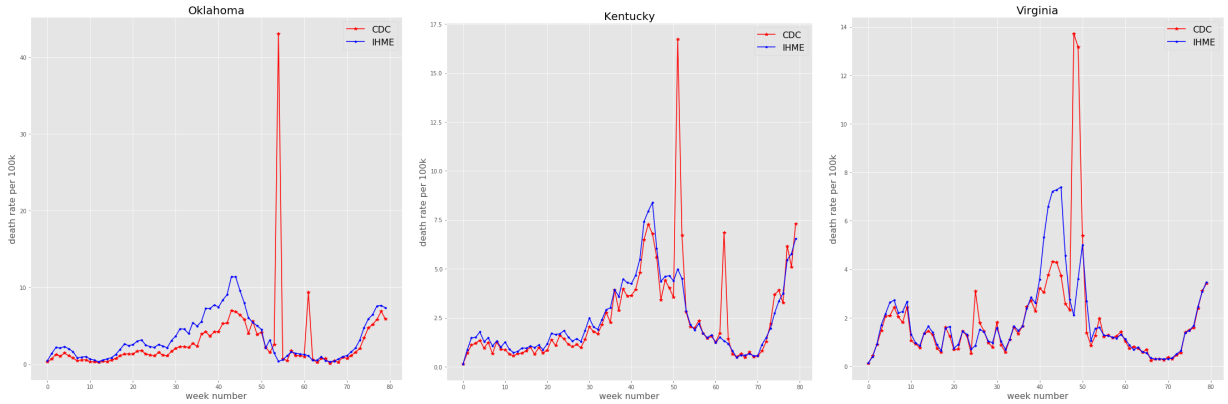


Figure 5.1: **Death rates reported by CDC and IHME** in Oklahoma, Kentucky, and Virginia, from left to right. We observe that the death rates reported by CDC are extremely high in some weeks due to backfill. IHME, on the other hand, smooths the data by distributing the sudden increase of deaths over the past period.

5.2 Data Collection

Truth data. We focus on the COVID-19 death rates per 100k population in the United States, which is calculated by accumulating the daily deaths over a week and dividing by the state's population $\times 10,000$. Although CDC also reports daily death, the data source often has delays and needs to backfill, which results in zero or even negative death rates for some days and a large increase afterwards. To account for such delays, the Institute of Health Metrics and Evaluation (IHME) (https://covid19.healthdata.org/united-states-of-america) provides smoothed data by distributing the sudden increase of deaths over the past period [6]. To illustrate the anomaly in data sources, we include three examples in Figure 5.1, which shows the death rates reported by CDC and IHME in Oklahoma, Kentucky and Virginia. We therefore use the death rates provided by IHME as the ground truth for a more reasonable estimate of truth death rate. We extract the time series between 2020-03-22 to 2021-12-30, and split them for training and testing.

Method Selection for Forecasting Data. We collect forecasting results from the US COVID-19 Forecast Hub (https://covid19forecasthub.org/), which includes submissions

from more than 90 institutions and organizations. The weekly forecasts include 4-week out predictions for all US states and national, including a prediction interval and a point value. [31] provides a thorough evaluation of 27 individual models that submitted complete forecasts of COVID-19 deaths to COVID-19 Forecast Hub, along with the COVIDhub-ensemble. The primary evaluation metric used was the weighted interval score (WIS) [13], which measures how consistent a collection of prediction intervals is with an observed value. Mean absolute error (MAE) is also used to compare the models. The evaluation shows high variability in forecasting models across time, geospatial units and forecast horizon. Among the evaluated methods, the COVIDhub-ensemble is consistently the most accurate model in terms of both WIS and MAE. For our purpose of constructing a data-driven ensemble, we want individual models that performs consistently well across forecast targets, weeks, or locations. We subsample a set of forecasting methods that covers all states and forecasting dates, and also ranks on top in [31]. The selected methods are summarized in Table 5.1. The data sources and website of each model can be found in `https://zoltardata.com/project/44`.

5.3 Feature Preprocessing

To emphasize more on the characteristics of the time series such as the trend, peak and curvature, we use a set of metafeatures as introduced in [97, 128] rather than the actual point values of the time series. Since the features are not linearly independent, we first conduct a principal component analysis (PCA) and decompose the features into linearly independent principal components. We also examine the true dimension of the feature space. Based on Figure 5.2 left, the first three principal components (PCs) already account for 95% of total variance, and thus we only keep these three PCs to represent the original data instead of using the whole set of 34 metafeatures. We also apply the Truncated Orthogonal Iteration (TOrth) on the data matrix to obtain sparse PCs, and we plot the distance between consecutive outputs of TOrth, $distance = \|\sin\Theta(Q_t, Q_{t-1})\|_F^2$, versus the iteration number t in Figure 5.2 right. We decrease the cardinality parameters at iteration 50, 100, and 150, from $K = [34, 34, 34]$ to $K = [4, 8, 6]$. The plot indicates that the algorithm converges and

Table 5.1: Forecasting methods used in the ensemble

Method	Team	Description
Karlen	Karlen Working group	"Discrete-time difference equations with long periods of constant transmission rate."
BPagano	BPagano	"Projects infections and deaths using an SIR model."
CovidComplete	Steve McConnell	"National level and state level, near-term (1-4 week) fatality forecasts."
MOBS	MOBS Lab at Northeastern	"Metapopulation, age structured SLIR model."
CEID-Walk	University of Georgia Center for the Ecology of Infectious Diseases Forecasting Working Group	"A random walk model without drift."

no important features were being truncated.

The features corresponding to nonzero coefficients are highlighted in Table 5.2, where we also include large coefficients in the parentheses. The sum of squares of coefficients for each vector is 1 since each sparse PC vector has unit norm. For sparse PC1 obtained by TOrth, the feature corresponding to the largest coefficient is the "flat_spots", which accounts for the number of flat spots in the time series, calculated by discretizing the series into 10 equal-sized intervals and counting the maximum run length within any single interval. For sparse PC2 obtained by TOrth, it highlights the features "unitroot_pp", "curvature" and "linearity". The "unitroot_pp" [105] feature is the Phillips-Perron test [106] to test the null hypothesis that a time series has a unit root, and the alternative is that the variable was generated by a stationary process. The "curvature" and "linearity" features are calculated

as the coefficients of an orthogonal quadratic regression [61, 128]:

$$T_t = \beta_0 + \beta_1\phi_1(t) + \beta_2\phi_2(t) + \epsilon_t,$$

where T_t is the trend of the time series decomposed using STL [29], and ϕ and ϕ_2 are orthogonal polynomials of orders 1 and 2. "linearity" corresponds to the estimate value of β_1 and "curvature" corresponds to the estimated value of β_2. Sparse PC3 highlights the feature "crossing_points", which corresponds to the number of times the time series crosses the median.

For visualization purpose, we plot the 2D projection of the data obtained by TOrth (Figure 5.3) and by PCA (Figure 5.4). Geographically, the majority of the Southwestern states, such as Arizona, New Mexico, Nevada, Oklahoma, Texas, and Utah, are situated at the lower left corner of the 2D projection, with low values along both PC1 and PC2. On the other hand, the majority of the east cost states, such as Massachusetts, Rhode Island, Connecticut, New York, New Jersey, Delaware, Maryland, are located on the upper half of the 2D projection, corresponding to high values in PC2. In terms of population, we observe that the states with the least populations (e.g. Wyoming, Vermont, North Dakota and South Dakota), as well as the states with the largest population densities (e.g. New Jersey, Rhode Island, Massachusetts, Connecticut, Maryland, Delaware, New York) are situated at the peripheral of the projection and are distant to other states. The states with small population tend to have large PC1, corresponding to a lot of flat spots around zero. The states with large population density tend to have large PC2, corresponding to more variations in the time series. We also plot the projected data along PC1 and PC3. Alaska is the state with the highest value along PC3, since its death rates remain low and oscillate around its median. The discrepancy between the projections obtained by TOrth and PCA mainly lies in the states that are clustered in the lower left corner, which are the states that have relatively low values along PC1, PC2 and PC3. We will illustrate the differences in the following sections when clustering and ensemble evaluations are performed.

Table 5.2: First 3 sparse PCs obtained by TOrth

	Feature set
PC1	**flat_spots (0.99)**, crossing_points, linearity, curvature
PC2	**unitroot_pp (0.70)**, **linearly (0.53)**, **curvature (0.39)**, x_acf10, crossing_points, flat_spots, lumpiness, nonlinearity
PC3	**crossing_points (0.96)**, x_acf10, flat_spots, beta, curvature, unitroot_pp

5.4 Clustering

Since we use a machine learning approach to train the ensemble weights, a large collection of training data is necessary to obtain accurate results. Using the time series for a single state for training is usually insufficient and results in a null tree, whose outputs are equal weights and the resulting ensemble model is an average of all methods. Training with all possible states is also not ideal, since the Covid-19 death rates could be drastically different for two states due to different policies, population density, hospital resources, etc.. We therefore perform a clustering step in advance to group states that have similar time series.

After projecting the metafeatures onto lower-dimensional subspace, we use K-means to obtain clusters. We set the number of clusters $S = 10$. Based on the clusters, we observe that the states that are geographically adjacent usually are clustered together, although there are exceptions such as Virginia and West Virginia, South Carolina and North Carolina. Since we do not have the ground-truth as which states should be clustered together, we adopt some similarity measurements to validate the resulting clusters. For the Covid-19 case, two time series are considered as similar if they have the same peaks and troughs at approximately the same time, up to a few weeks apart. Therefore we use the Pearson correlation to account for the shifts and measure the similarity of two time series. In particular, we focus on the locations that have discrepancies between PCA and TOrth. One such example is Texas,

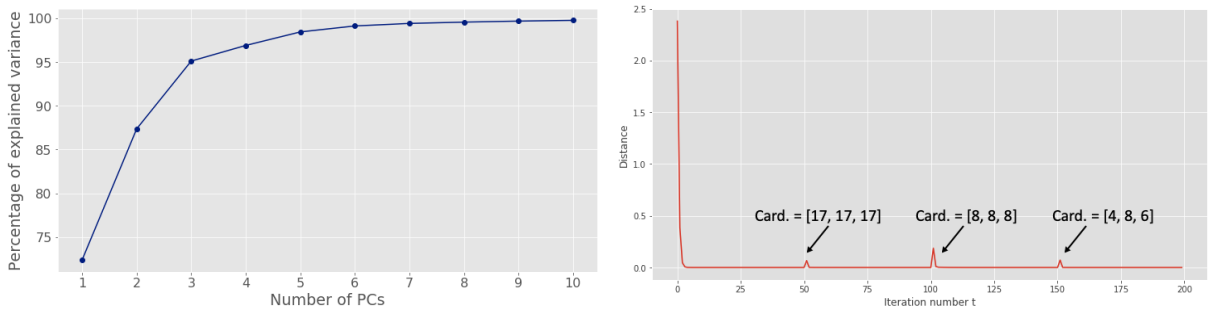


Figure 5.2: **Dimensionality reduction for feature preprocessing.** Left: cumulative explained variance by the first 10 principal components. We see that the first three principal components already take up to 95% of the total variance. Right: the distance between consecutive updates obtained by TOrth as measured by $\|\sin\Theta(Q_t, Q_{t-1})\|_F^2$. The cardinality parameter starts with the full support set and is decreased at iteration 50, 100 and 150. Further decreasing the cardinality parameters would make the algorithm fail to converge.

which is clustered differently by TOrth+K-means and PCA+K-means. We plot the correlation matrix between Texas and other states in the same cluster, and the result is shown in Figure 5.5. All the states clustered with Texas by TOrth+K-means show high (> 0.6) correlation with Texas, whereas some states clustered with Texas obtained by PCA+K-means have low (< 0.5) correlation with Texas. To further understand the discrepancies, we also plot the time series comparisons of the states that are clustered together by TOrth+K-means and PCA+Kmeans. In Figure 5.6, we observe more similarities in the states clustered by TOrth+K-means as they have similar peaks, troughs and flat spots.

5.5 The XGBoost Algorithm for Meta-learner

We use the XGBoost [27] algorithm to train a meta-learner for the ensemble weights, which can be viewed as a nonlinear mapping f that takes input the $x \in \mathbb{R}^p$, which is the vector of metafeatures of each time series, and outputs $y \in \mathbb{R}^M$, which is the vector of weights. Suppose we have N times series and each time series is denoted as t_n. x_n is the vector of metafeatures extracted from series t_n. Suppose we have M classes and each is a forecast method, then the output from the meta-learner $y_n = f(x_n)$ is a vector based on features

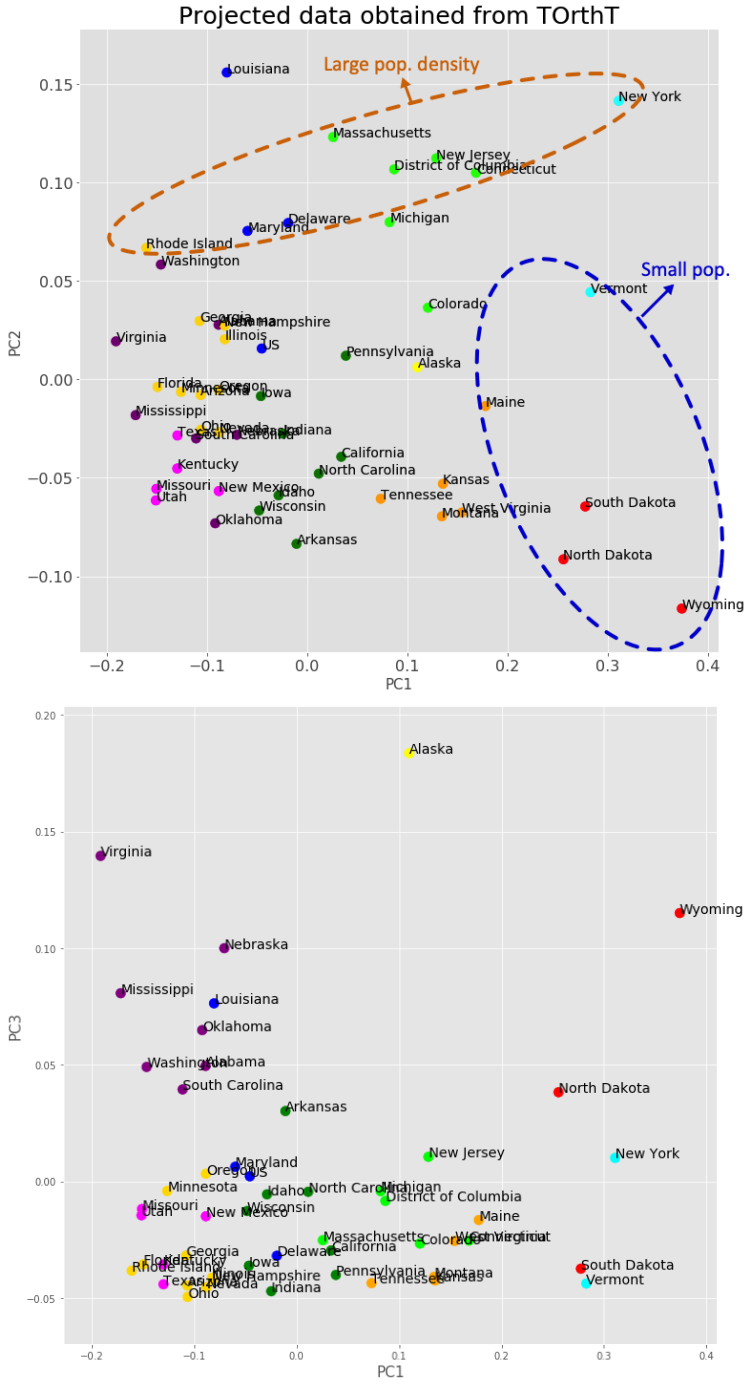


Figure 5.3: **Data projected onto the subspaces obtained by TOrth.** We plot the projected data along two sparse PCs, PC1 vs. PC2 on the top, and PC1 vs. PC3 on the bottom. The colors of the data points correspond to different clusters.

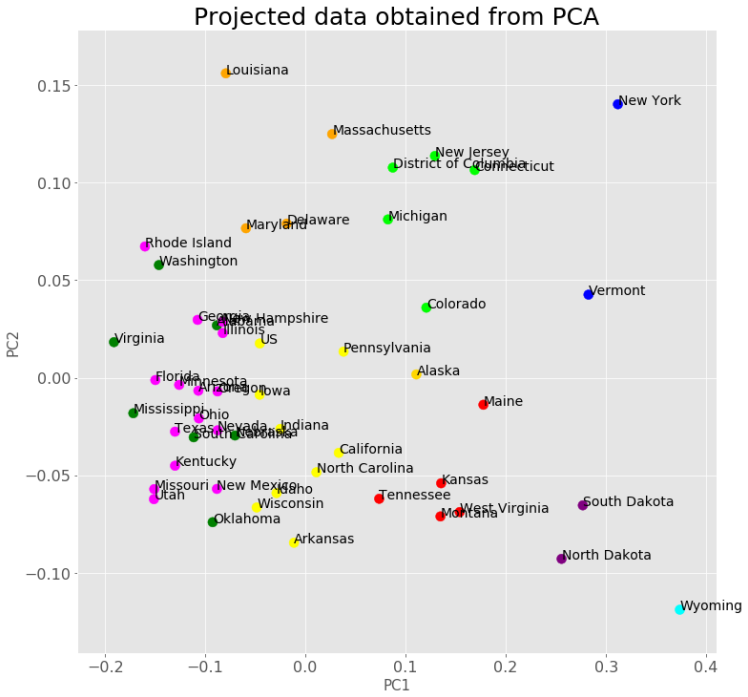


Figure 5.4: **Data projected onto the subspaces obtained by PCA.** We plot the projected data along two PCs, PC1 vs. PC2. The colors of the data points correspond to different clusters.

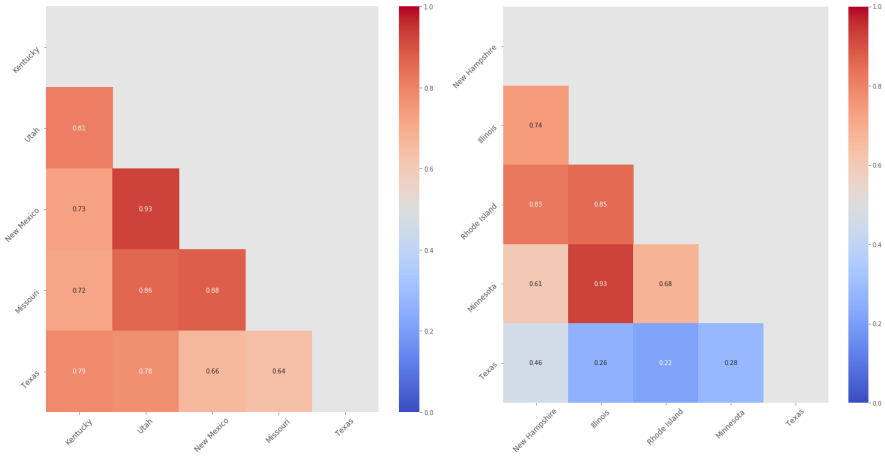


Figure 5.5: **Pearson correlation matrices for the clusters.** Left: correlation matrix of all the states clustered by TOrth+K-means, where all states within the cluster show high (> 0.6) pairwise correlation. Right: correlation matrix of some states clustered with Texas obtained by PCA+K-means but have low (< 0.5) correlation with Texas.

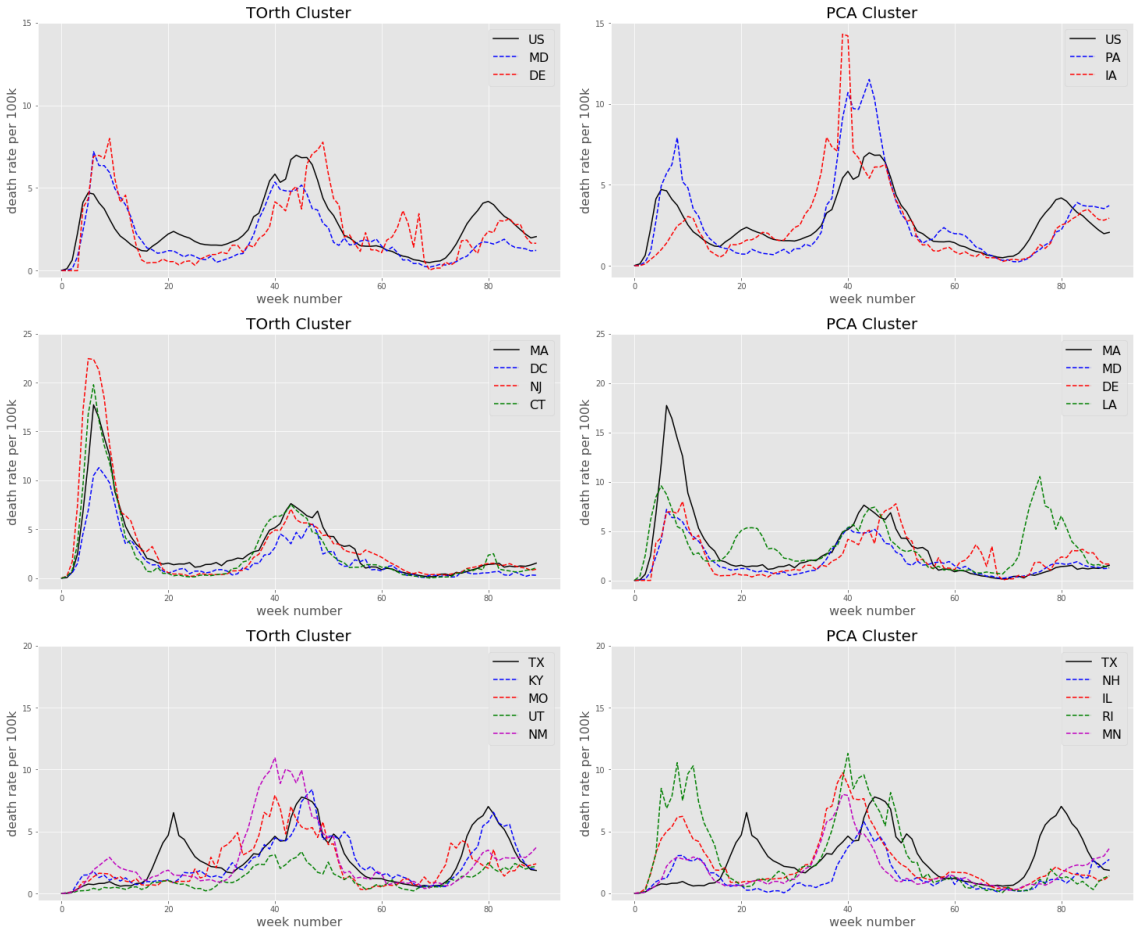


Figure 5.6: **Time series comparisons.** We plot the time series comparisons of the states that are clustered together by TOrth+K-means and PCA+Kmeans, and focus on the states clustered with US (top), Massachusetts (middle), and Texas (bottom). The time series in TOrth clusters show more similarity in time series characteristics such as the strength of the peaks, troughs and flat spots.

x_n with its m^{th} entry corresponding to the m^{th} forecast method. The ensemble weights w_n are then obtained by using a softmax transformation on y_n to ensure that all weights are positive and that they add up to 1:

$$w_{nm} = \frac{\exp(y_{nm})}{\sum_m \exp(y_{nm})}.$$

The objective function for the meta-learner is defined as follows:

$$L = \arg\min_y \sum_{n=1}^{N} \sum_{m=1}^{M} w_{nm} Error_{nm}, \tag{5.1}$$

where the error function is user-supplied and $Error_{nm}$ denotes the error corresponding to the m^{th} forecasting method of the n^{th} time series. We adopt the weighted overall weighted average (OWA; [94]) as the error function, which is a combination of the symmetric mean absolute percentage error (sMAPE; [93]) and the mean absolute scaled error (MASE; [60]). The Gradient and the Hessian of the objective function are then passed to the XGBoost algorithm in order to minimize the objective function. Since the training set is different for each cluster, we train one meta-learner for each cluster, and test on the states within that cluster on out-of-sample forecast dates.

5.6 Summary of the Main Results

For evaluation and comparisons, we use the mean squared error to measure the accuracy of the models, which is given by:

$$MSE = \sqrt{\sum_{t=1}^{n} \frac{1}{n} (A_t - F_t)^2}$$

where A_t refers to the truth data and F_t is the forecast data at time t. We first compare the ensembles generated by different clustering approaches, i.e. Cluster-PCA-ensemble and Cluster-TOrth-ensemble, and we focus on the locations where these two clustering have discrepancies. We observe that Cluster-TOrth-ensemble either outperforms Cluster-PCA-ensemble by a large margin (see Table 5.3) or achieves similar accuracy. We then compare

Table 5.3: MSE comparison between Cluster-PCA-ensemble and Cluster-TOrth-ensemble

Test State	Cluster-PCA-ensemble	Cluster-TOrth-ensemble
National	0.34	0.23
Texas	0.43	0.38
Louisiana	0.81	0.68
Maryland	0.46	0.44
Utah	0.44	0.42

our ensemble approach with an ensemble without clustering, i.e. AllStates-ensemble, as well as with the COVIDhub-ensemble and a simple averaging of the five individual models (Methods Average). We observe that the Cluster-TOrth-ensemble has a relatively consistent performance with increasing forecast horizons compared to the other methods, as measured in MASE for each forecast week, as shown in Figure 5.7. Cluster-TOrth-ensemble also outperforms AllStates-ensemble for most states in terms of MSE. Last but not least, we calculate the relative MSE of the Cluster-TOrth-ensemble with the COVIDhub-ensemble error being 1 across all locations, and the results are shown in Table 5.4. For more than 60% of the states, our ensemble outperforms the COVIDhub-ensemble. Since randomness is incorporated in the XGBoost algorithm, the outputs maybe slightly different when the data is shuffled, and therefore we also count the states that have comparable errors (within 10% error range) to COVIDhub-ensemble. For about 80% of states, Cluster-TOrth-ensemble performs better or comparable to COVIDhub-ensemble. For states where Cluster-TOrth-ensemble performs worse than the COVIDhub-ensemble, we include the percentage of the error differences in the parentheses. Most of these states lie on the peripheral of the 2D projection, which results in clusters with very few members and a small training set for the meta-learner.

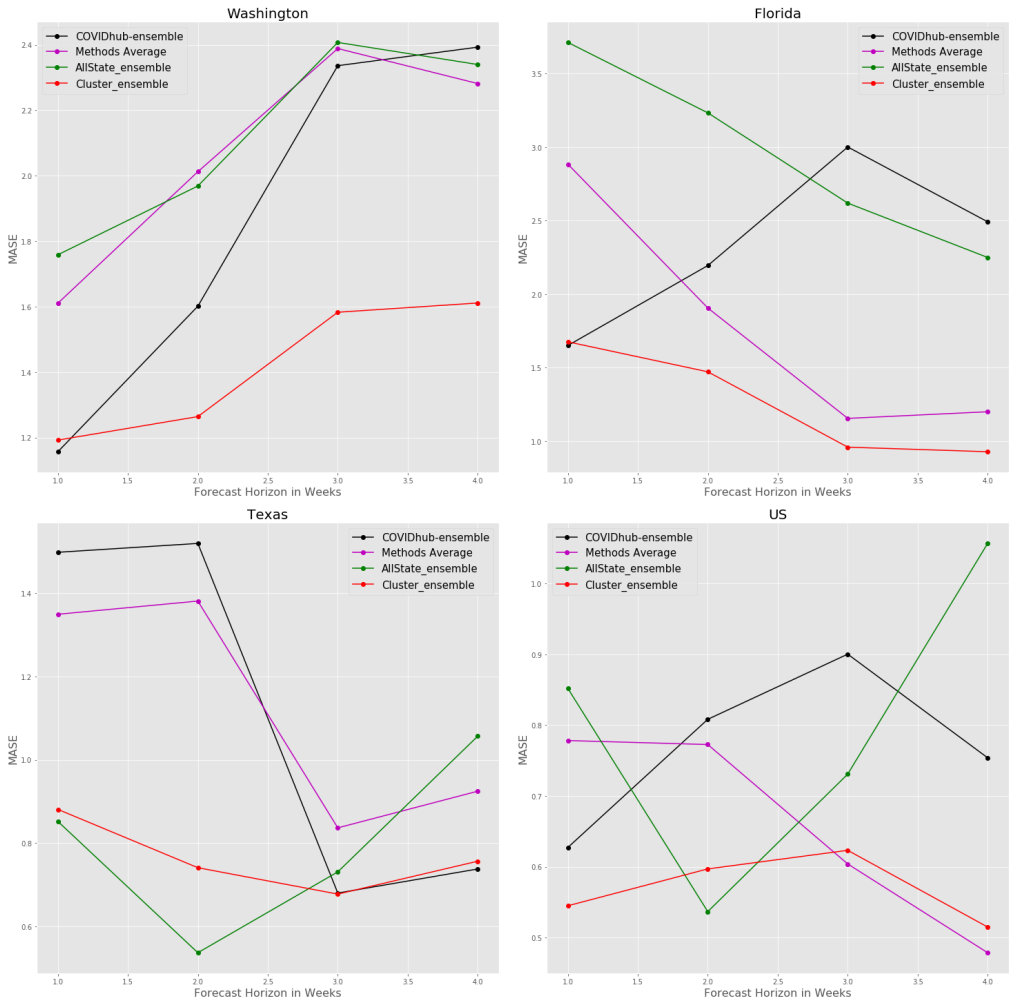


Figure 5.7: **Prediction error comparisons.** MASE comparison between Methods Average, COVIDhub-ensemble, AllStates-ensemble and Cluster-TOrth-ensemble tested on Washington, Florida, Texas and US. The Method Average is an average of the 5 methods used in the ensemble. AllStates-ensemble has the meta-learner trained using all states, and Cluster-ensemble has the meta-learner trained with only the states within a cluster. The forecast error of Cluster-TOrth-ensemble remains relatively flat with increasing horizons compared to the other three methods.

Table 5.4: Relative MSE of Cluster-TOrth-ensemble on each location compared to COVIDhub-ensemble

< 0.9 (Better)	FL, MN, AZ, OH, NV, MA, CT, CO, TX, KY, MO, NM, IA, RI, IN, NC, WI, AR, WA, VA, AL, MS, SC, NE, OK, MT, MD, LA, DE, US.
$\in [0.9, 1.1]$	OR, NJ, PA, ID, KS, TN, WV, SD, VT.
> 1.1 (Worse)	UT (14%), MI (15%), CA (28%), NH (30%), ME (30%), GA (40%), IL (66%), NY (71%), WY (92%), MD (181%).

5.7 Discussion and Conclusions

We have established a data-driven procedure to learn the ensemble weights for the US COVID-19 death rates, which involves four key steps for training: 1) calculate the metafeatures from time series; 2) project the metafeatures onto lower-dimensional space; 3) generate the clusters based on the projected metafeatures; and 4) train a meta-learner using XGBoost for each cluster. We have shown that our ensemble, Cluster-TOrth-ensemble, outperforms the COVIDhub-ensemble in most US states. By projecting the features onto lower-dimensional space and obtaining sparse principal components, we have also improved the interpretability of the time series in each clusters. Our models have shown robustness in the following aspects when compared to other ensemble approaches:

- Robustness in forecast time: since our ensemble method uses a data-driven, machine learning approach, it is trained to learn variations in each time period such as rapidly changing trends, increasing strength in peaks, etc.. The meta-learner outputs a different set of weights for each time series across different time, and these weights takes into accounts how each individual models react to pandemic waves or anomalous data in the past.

- Robustness in forecast locations: we cluster locations by the projections of time series

metafeatures, which enlarges the training set for each location as opposed to using a single location for training and testing. Our ensemble method shows superior accuracy especially in locations where large clusters are formed. It also excludes the states with less similarities and outperforms the ensemble that trains with all states.

- Robustness in forecast horizon: the outlying forecasts that significantly under or overestimate the truth value usually have an increasing impact on the mean with increasing forecast horizon, for example, a random walk model with trend. By assigning weights to models based on their past performances, our ensemble method assigns more weights on models that consistently have good performance in the past and avoids the impact of outlying models.

Our proposed procedure generalizes easily to other types of time series, especially when the training data is insufficient. For COVID-19 forecast, there are still many open areas to help improve the ensemble, especially in the locations where the proposed ensemble performing worse than the averaging method. For example, the set of metafeatures could be expanded to include more COVID-19 exclusive features. The set of forecast methods could also be expanded, and could incorporate some of the auto-generated methods in [97]. Moreover, many clustering techniques are open to explore and could lead to different ensemble results.

BIBLIOGRAPHY

[1] Pankaj K Agarwal and Nabil H Mustafa. K-means projective clustering. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 155–165, 2004.

[2] Amr Ahmed and Eric P Xing. Recovering time-varying networks of dependencies in social and biological studies. *Proceedings of the National Academy of Sciences*, 106(29):11878–11883, 2009.

[3] Mokhtar Z Alaya, Maxime Berar, Gilles Gasso, and Alain Rakotomamonjy. Screening sinkhorn algorithm for regularized optimal transport. In *Advances in Neural Information Processing Systems*, pages 12169–12179, 2019.

[4] Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. In *Advances in Neural Information Processing Systems*, pages 1964–1974, 2017.

[5] ZF Altun and DH Hall. Wormatlas. *URL http://www. wormatlas. org*, 1384, 2002.

[6] Ryan M Barber, Reed JD Sorensen, David M Pigott, Catherine Bisignano, Austin Carter, Joanne O Amlag, James K Collins, Cristiana Abbafati, Christopher Adolph, Adrien Allorant, et al. Estimating global, regional, and national daily and cumulative infections with sars-cov-2 through nov 14, 2021: a statistical analysis. *The Lancet*, 2022.

[7] Katia Basso, Adam A Margolin, Gustavo Stolovitzky, Ulf Klein, Riccardo Dalla-Favera, and Andrea Califano. Reverse engineering of regulatory networks in human b cells. *Nature genetics*, 37(4):382–390, 2005.

[8] John M Bates and Clive WJ Granger. The combination of forecasts. *Journal of the Operational Research Society*, 20(4):451–468, 1969.

[9] Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015.

[10] Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.

[11] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999.

[12] Nicolas Boumal, Bamdev Mishra, P-A Absil, and Rodolphe Sepulchre. Manopt, a matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15(1):1455–1459, 2014.

[13] Johannes Bracher, Evan L Ray, Tilmann Gneiting, and Nicholas G Reich. Evaluating epidemic forecasts in an interval format. *PLoS computational biology*, 17(2):e1008618, 2021.

[14] Paul S Bradley and Olvi L Mangasarian. K-plane clustering. *Journal of Global optimization*, 16(1):23–32, 2000.

[15] Atul J Butte and Isaac S Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Pac Symp Biocomput*, volume 5, page 26, 2000.

[16] Jorge Cadima and Ian T Jolliffe. Loading and correlations in the interpretation of principle components. *Journal of applied Statistics*, 22(2):203–214, 1995.

[17] Deng Cai, Xiaofei He, and Jiawei Han. Speed up kernel discriminant analysis. *The VLDB Journal*, 20(1):21–33, 2011.

[18] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1548–1560, 2010.

[19] Deng Cai, Xiaofei He, Kun Zhou, Jiawei Han, and Hujun Bao. Locality sensitive discriminant analysis. In *IJCAI*, volume 2007, pages 1713–1726, 2007.

[20] Tony Cai, Donggyu Kim, Xinyu Song, and Yazhen Wang. Optimal sparse eigenspace and low-rank density matrix estimation for quantum systems. *Journal of Statistical Planning and Inference*, 213:50–71, 2020.

[21] Tony Cai, Zongming Ma, Yihong Wu, et al. Sparse pca: Optimal rates and adaptive estimation. *The Annals of Statistics*, 41(6):3074–3110, 2013.

[22] Yunfeng Cai, Lei-Hong Zhang, Zhaojun Bai, and Ren-Cang Li. On an eigenvector-dependent nonlinear eigenvalue problem. *SIAM Journal on Matrix Analysis and Applications*, 39(3):1360–1382, 2018.

[23] Eric Cancès. Self-consistent field algorithms for kohn-sham models with fractional occupation numbers. *The Journal of Chemical Physics*, 114(24):10616–10622, 2001.

[24] Martin Chalfie, John E Sulston, JOHN G White, Eileen Southgate, J Nicol Thomson, and Sydney Brenner. The neural circuit for touch sensitivity in caenorhabditis elegans. *Journal of Neuroscience*, 5(4):956–964, 1985.

[25] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[26] Andy J Chang, Nikolas Chronis, David S Karow, Michael A Marletta, and Cornelia I Bargmann. A distributed chemosensory circuit for oxygen preference in c. elegans. *PLoS biology*, 4(9):e274, 2006.

[27] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[28] Robert T Clemen. Combining forecasts: A review and annotated bibliography. *International journal of forecasting*, 5(4):559–583, 1989.

[29] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. Stl: A seasonal-trend decomposition. *J. Off. Stat*, 6(1):3–73, 1990.

[30] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

[31] Estee Y Cramer, Evan L Ray, Velma K Lopez, Johannes Bracher, Andrea Brennen, Alvaro J Castro Rivadeneira, Aaron Gerding, Tilmann Gneiting, Katie H House, Yuxin Huang, et al. Evaluation of individual and ensemble probabilistic forecasts of covid-19 mortality in the united states. *Proceedings of the National Academy of Sciences*, 119(15):e2113561119, 2022.

[32] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300, 2013.

[33] Marco Cuturi and Arnaud Doucet. Fast computation of wasserstein barycenters. In *International Conference on Machine Learning*, pages 685–693, 2014.

[34] Alexandre d'Aspremont, Francis Bach, and Laurent El Ghaoui. Optimal solutions for sparse principal component analysis. *Journal of Machine Learning Research*, 9(Jul):1269–1294, 2008.

[35] Alexandre d'Aspremont, Laurent E Ghaoui, Michael I Jordan, and Gert R Lanckriet. A direct formulation for sparse pca using semidefinite programming. In *Advances in neural information processing systems*, pages 41–48, 2005.

[36] Amicie De Pierrefeu, Tommy Löfstedt, Fouad Hadj-Selem, Mathieu Dubois, Renaud Jardri, Thomas Fovet, Philippe Ciuciu, Vincent Frouin, and Edouard Duchesnay. Structured sparse principal components analysis with the tv-elastic net penalty. *IEEE transactions on medical imaging*, 37(2):396–407, 2017.

[37] Chris Ding and Tao Li. Adaptive dimension reduction using discriminant analysis and k-means clustering. In *Proceedings of the 24th international conference on Machine learning*, pages 521–528. ACM, 2007.

[38] Tianyu Ding, Zhihui Zhu, Manolis Tsakiris, Rene Vidal, and Daniel Robinson. Dual principal component pursuit for learning a union of hyperplanes: Theory and algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 2944–2952. PMLR, 2021.

[39] Ehsan Elhamifar and René Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013.

[40] N Benjamin Erichson, Peng Zheng, Krithika Manohar, Steven L Brunton, J Nathan Kutz, and Aleksandr Y Aravkin. Sparse principal component analysis via variable projection. *SIAM Journal on Applied Mathematics*, 80(2):977–1002, 2020.

[41] Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

[42] Zizhu Fan, Yong Xu, and David Zhang. Local linear discriminant analysis framework using sample neighbors. *IEEE Transactions on Neural Networks*, 22(7):1119–1132, 2011.

[43] Rémi Flamary and Nicolas Courty. Pot python optimal transport library, 2017.

[44] Rémi Flamary, Marco Cuturi, Nicolas Courty, and Alain Rakotomamonjy. Wasserstein discriminant analysis. *Machine Learning*, 107(12):1923–1945, 2018.

[45] Edward B Fowlkes and Colin L Mallows. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 78(383):553–569, 1983.

[46] Jerome Friedman. Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175, 1989.

[47] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[48] Joseph Friedman, Patrick Liu, Christopher E Troeger, Austin Carter, Robert C Reiner, Ryan M Barber, James Collins, Stephen S Lim, David M Pigott, Theo Vos, et al. Predictive performance of international covid-19 mortality forecasting models. *Nature communications*, 12(1):1–13, 2021.

[49] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620, 2000.

[50] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[51] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6):801–806, 1993.

[52] Miriam B Goodman, David H Hall, Leon Avery, and Shawn R Lockery. Active currents regulate sensitivity and dynamic range in c. elegans neurons. *Neuron*, 20(4):763–772, 1998.

[53] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[54] Trevor Hastie, Andreas Buja, and Robert Tibshirani. Penalized discriminant analysis. *The Annals of Statistics*, pages 73–102, 1995.

[55] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[56] Reinhard Heckel and Helmut Bölcskei. Robust subspace clustering via thresholding. *IEEE Transactions on Information Theory*, 61(11):6320–6342, 2015.

[57] Christopher J Honey, Rolf Kötter, Michael Breakspear, and Olaf Sporns. Network structure of cerebral cortex shapes functional connectivity on multiple time scales. *Proceedings of the National Academy of Sciences*, 104(24):10240–10245, 2007.

[58] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[59] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[60] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

[61] Rob J Hyndman, Earo Wang, and Nikolay Laptev. Large-scale unusual time series detection. In *2015 IEEE international conference on data mining workshop (ICDMW)*, pages 1616–1619. IEEE, 2015.

[62] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[63] JNR Jeffers. Two case studies in the application of principal component analysis. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 16(3):225–236, 1967.

[64] Rodolphe Jenatton, Guillaume Obozinski, and Francis Bach. Structured sparse principal component analysis. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 366–373, 2010.

[65] Yangqing Jia, Feiping Nie, and Changshui Zhang. Trace ratio problem revisited. *IEEE Transactions on Neural Networks*, 20(4):729–735, 2009.

[66] Iain M Johnstone. On the distribution of the largest eigenvalue in principal components analysis. *Annals of statistics*, pages 295–327, 2001.

[67] Iain M Johnstone and Arthur Yu Lu. On consistency and sparsity for principal components analysis in high dimensions. *Journal of the American Statistical Association*, 104(486):682–693, 2009.

[68] Ian T Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[69] Ian T Jolliffe. Rotation of iii-defined principal components. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 38(1):139–147, 1989.

[70] Ian T Jolliffe, Nickolay T Trendafilov, and Mudassir Uddin. A modified principal component technique based on the lasso. *Journal of computational and Graphical Statistics*, 12(3):531–547, 2003.

[71] Michel Journée, Yurii Nesterov, Peter Richtárik, and Rodolphe Sepulchre. Generalized power method for sparse principal component analysis. *Journal of Machine Learning Research*, 11(Feb):517–553, 2010.

[72] Saul Kato, Harris S Kaplan, Tina Schrödel, Susanne Skora, Theodore H Lindsay, Eviatar Yemini, Shawn Lockery, and Manuel Zimmer. Global brain dynamics embed the motor command sequence of caenorhabditis elegans. *Cell*, 163(3):656–669, 2015.

[73] Jimin Kim, William Leahy, and Eli Shlizerman. Neural interactome: Interactive simulation of a neuronal system. *Frontiers in computational neuroscience*, 13:8, 2019.

[74] Philip A Knight. The sinkhorn–knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275, 2008.

[75] Mladen Kolar, Le Song, Amr Ahmed, and Eric P Xing. Estimating time-varying networks. *The Annals of Applied Statistics*, pages 94–123, 2010.

[76] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[77] Daphne Koller, Nir Friedman, Lise Getoor, and Ben Taskar. Graphical models in a nutshell. *Introduction to statistical relational learning*, pages 13–55, 2007.

[78] James Kunert, Eli Shlizerman, and J Nathan Kutz. Low-dimensional functionality of complex network dynamics: Neurosensory integration in the caenorhabditis elegans connectome. *Physical Review E*, 89(5):052805, 2014.

[79] James M Kunert-Graf, Eli Shlizerman, Andrew Walker, and J Nathan Kutz. Multistability and long-timescale transients encoded by network structure in a model of C. elegans connectome dynamics. *Frontiers in computational neuroscience*, 11:53, 2017.

[80] Ken Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier, 1995.

[81] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2169–2178. IEEE, 2006.

[82] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[83] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[84] Christiane Lemke and Bogdan Gabrys. Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, 73(10-12):2006–2016, 2010.

[85] Xuelong Li, Mulin Chen, Feiping Nie, and Qi Wang. Locality adaptive discriminant analysis. In *IJCAI*, pages 2201–2207, 2017.

[86] Tianyi Lin, Nhat Ho, and Michael Jordan. On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms. In *International Conference on Machine Learning*, pages 3982–3991, 2019.

[87] John Lipor, David Hong, Yan Shuo Tan, and Laura Balzano. Subspace clustering using ensembles of k-subspaces. *Information and Inference: A Journal of the IMA*, 2020.

[88] Hexuan Liu, Jimin Kim, and Eli Shlizerman. Functional connectomics from neural dynamics: probabilistic graphical models for neuronal network of caenorhabditis elegans. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 373(1758):20170377, 2018.

[89] Xin Liu, Xiao Wang, Zaiwen Wen, and Yaxiang Yuan. On the convergence of the self-consistent field iteration in kohn–sham density functional theory. *SIAM Journal on Matrix Analysis and Applications*, 35(2):546–558, 2014.

[90] Zongming Ma et al. Sparse principal component analysis and iterative thresholding. *The Annals of Statistics*, 41(2):772–801, 2013.

[91] Lester W Mackey. Deflation methods for sparse pca. In *NIPS*, volume 21, pages 1017–1024, 2008.

[92] Dougal Maclaurin, David Duvenaud, and Ryan P Adams. Autograd: Effortless gradients in numpy. In *ICML 2015 AutoML Workshop*, volume 238, 2015.

[93] Spyros Makridakis. Accuracy measures: theoretical and practical concerns. *International journal of forecasting*, 9(4):527–529, 1993.

[94] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020.

[95] Richard M Martin and Richard Milton Martin. *Electronic structure: basic theory and practical methods*. Cambridge university press, 2004.

[96] Per-Gunnar Martinsson and Joel A Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, 2020.

[97] Pablo Montero-Manso, George Athanasopoulos, Rob J Hyndman, and Thiyanga S Talagala. Fforma: Feature-based forecast model averaging. *International Journal of Forecasting*, 36(1):86–92, 2020.

[98] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[99] Thanh T Ngo, Mohammed Bellalij, and Yousef Saad. The trace ratio optimization problem. *SIAM review*, 54(3):545–569, 2012.

[100] Jeffrey P Nguyen, Frederick B Shipley, Ashley N Linder, George S Plummer, Mochi Liu, Sagar U Setru, Joshua W Shaevitz, and Andrew M Leifer. Whole-brain calcium imaging with cellular resolution in freely behaving caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, 113(8):E1074–E1081, 2016.

[101] Feiping Nie, Shiming Xiang, Yangqing Jia, Changshui Zhang, and Shuicheng Yan. Trace ratio criterion for feature selection. In *AAAI*, volume 2, pages 671–676, 2008.

[102] Feiping Nie, Shiming Xiang, and Changshui Zhang. Neighborhood minmax projections. In *IJCAI*, pages 993–998, 2007.

[103] Alp Ozdemir, Edward M Bernat, and Selin Aviyente. Recursive tensor subspace tracking for dynamic brain network analysis. *IEEE Transactions on Signal and Information Processing over Networks*, 3(4):669–682, 2017.

[104] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[105] Bernhard Pfaff, Eric Zivot, Matthieu Stigler, and Maintainer Bernhard Pfaff. Package 'urca'. *Unit root and cointegration tests for time series data. R package version*, pages 1–2, 2016.

[106] Peter CB Phillips and Pierre Perron. Testing for a unit root in time series regression. *Biometrika*, 75(2):335–346, 1988.

[107] Ricardo Prudêncio and Teresa Ludermir. Using machine learning techniques to combine forecasting methods. In *Australasian joint conference on artificial intelligence*, pages 1122–1127. Springer, 2004.

[108] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[109] C Rate and C Retrieval. Columbia object image library (coil-20). *Computer*, 2011.

[110] Evan L Ray, Nutcha Wattanachit, Jarad Niemi, Abdul Hannan Kanji, Katie House, Estee Y Cramer, Johannes Bracher, Andrew Zheng, Teresa K Yamana, Xinyue Xiong, et al. Ensemble forecasts of coronavirus disease 2019 (covid-19) in the us. *MedRXiv*, 2020.

[111] Richard W Reynolds, Nick A Rayner, Thomas M Smith, Diane C Stokes, and Wanqiu Wang. An improved in situ and satellite sst analysis for climate. *Journal of climate*, 15(13):1609–1625, 2002.

[112] John R Rice. The algorithm selection problem. In *Advances in computers*, volume 15, pages 65–118. Elsevier, 1976.

[113] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[114] Yousef Saad, James R Chelikowsky, and Suzanne M Shontz. Numerical methods for electronic structure calculations of materials. *SIAM review*, 52(1):3–54, 2010.

[115] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 32–36. IEEE, 2004.

[116] Haipeng Shen and Jianhua Z Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of multivariate analysis*, 99(6):1015–1034, 2008.

[117] Eli Shlizerman, Konrad Schroder, and J Nathan Kutz. Neural activity measures and their dynamics. *SIAM Journal on Applied Mathematics*, 72(4):1260–1291, 2012.

[118] Eli Shlizerman, Konrad Schroder, and J Nathan Kutz. Neural activity measures and their dynamics. *SIAM Journal on Applied Mathematics*, 72(4):1260–1291, 2012.

[119] Junxiao Song, Prabhu Babu, and Daniel P Palomar. Sparse generalized eigenvalue problem via smooth optimization. *IEEE Transactions on Signal Processing*, 63(7):1627–1642, 2015.

[120] Bharath K Sriperumbudur and Gert RG Lanckriet. On the convergence of the concave-convex procedure. In *Nips*, volume 9, pages 1759–1767. Citeseer, 2009.

[121] Greg J Stephens, Bethany Johnson-Kerner, William Bialek, and William S Ryu. Dimensionality and dynamics in the behavior of c. elegans. *PLoS computational biology*, 4(4):e1000028, 2008.

[122] Gilbert W Stewart. Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM review*, 15(4):727–764, 1973.

[123] Gilbert W Stewart. Perturbation theory for the singular value decomposition. Technical report, 1998.

[124] Gilbert W Stewart and Ji-Guang Sun. *Matrix Perturbation Theory*. Academic Press, Boston, 1990.

[125] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[126] Masashi Sugiyama. Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *Journal of machine learning research*, 8(May):1027–1061, 2007.

[127] Ji-guang Sun. The perturbation bounds for eigenspaces of a definite matrix-pair. *Numerische Mathematik*, 41(3):321–343, 1983.

[128] Thiyanga S Talagala, Rob J Hyndman, George Athanasopoulos, et al. Meta-learning how to forecast time series. *Monash Econometrics and Business Statistics Working Papers*, 6(18):16, 2018.

[129] Lai Tian, Feiping Nie, Rong Wang, and Xuelong Li. Learning feature sparse principal subspace. *Advances in Neural Information Processing Systems*, 33, 2020.

[130] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[131] Allan Timmermann. Forecast combinations. *Handbook of economic forecasting*, 1:135–196, 2006.

[132] James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *The Journal of Machine Learning Research*, 17(1):4755–4759, 2016.

[133] Charles F Van Loan and Gene H Golub. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 4th edition, 2012.

[134] Lav R Varshney, Beth L Chen, Eric Paniagua, David H Hall, and Dmitri B Chklovskii. Structural properties of the caenorhabditis elegans neuronal network. *PLoS computational biology*, 7(2):e1001066, 2011.

[135] A Vespignani, M Chinazzi, JT Davis, K Mu, AP Piontti, N Samay, et al. Modeling of covid-19 epidemic in the united states.

[136] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[137] Vincent Q Vu, Juhee Cho, Jing Lei, and Karl Rohe. Fantope projection and selection: A near-optimal convex relaxation of sparse pca. In *Advances in neural information processing systems*, pages 2670–2678, 2013.

[138] Vincent Q Vu, Jing Lei, et al. Minimax sparse principal subspace estimation in high dimensions. *The Annals of Statistics*, 41(6):2905–2947, 2013.

[139] De Wang, Feiping Nie, and Heng Huang. Unsupervised feature selection via unified trace ratio formulation and k-means clustering (track). In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 306–321. Springer, 2014.

[140] Guanyi Wang and Santanu Dey. Upper bounds for model-free row-sparse principal component analysis. In *International Conference on Machine Learning*, pages 9868–9875. PMLR, 2020.

[141] Huan Wang, Shuicheng Yan, Dong Xu, Xiaoou Tang, and Thomas Huang. Trace ratio vs. ratio trace for dimensionality reduction. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[142] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[143] Stephen R Wicks, Chris J Roehrig, and Catharine H Rankin. A dynamic network simulation of the nematode tap withdrawal circuit: predictions concerning synaptic function using behavioral criteria. *Journal of Neuroscience*, 16(12):4017–4031, 1996.

[144] Daniela M Witten, Robert Tibshirani, and Trevor Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, 2009.

[145] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE transactions on pattern analysis and machine intelligence*, 29(1):40–51, 2006.

[146] Chao Yang, Weiguo Gao, and Juan C Meza. On the convergence of the self-consistent field iteration for a class of nonlinear eigenvalue problems. *SIAM journal on matrix analysis and applications*, 30(4):1773–1788, 2009.

[147] Jieping Ye, Zheng Zhao, and Mingrui Wu. Discriminative k-means for clustering. In *Advances in neural information processing systems*, pages 1649–1656, 2008.

[148] Ofer Yizhar, Lief E Fenno, Thomas J Davidson, Murtaza Mogri, and Karl Deisseroth. Optogenetics in neural systems. *Neuron*, 71(1):9–34, 2011.

[149] Xiao-Tong Yuan and Tong Zhang. Truncated power method for sparse eigenvalue problems. *Journal of Machine Learning Research*, 14(Apr):899–925, 2013.

[150] Jing Zhang, Xiang Li, Cong Li, Zhichao Lian, Xiu Huang, Guocheng Zhong, Dajiang Zhu, Kaiming Li, Changfeng Jin, Xintao Hu, et al. Inferring functional interaction and transition patterns via dynamic bayesian variable partition models. *Human brain mapping*, 35(7):3314–3331, 2014.

[151] Linlin Zhang, Michele Guindani, and Marina Vannucci. Bayesian models for functional magnetic resonance imaging data analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(1):21–41, 2015.

[152] Youwei Zhang, Alexandre d'Aspremont, and Laurent El Ghaoui. Sparse pca: Convex relaxations, algorithms and applications. In *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 915–940. Springer, 2012.

[153] Yu Zhang and Dit-Yan Yeung. Worst-case linear discriminant analysis. In *Advances in Neural Information Processing Systems*, pages 2568–2576, 2010.

[154] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

[155] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.

[156] Hui Zou and Lingzhou Xue. A selective overview of sparse principal component analysis. *Proceedings of the IEEE*, 106(8):1311–1320, 2018.

Appendix A

CODE AVAILABILITY

The code used in this thesis is included in the author's GitHub page. The links can be found below:

- Wasserstein Discriminant Analysis

  `https://github.com/HexuanLiu/WDA_eig`

- Truncated Orthogonal Iteration

  `https://github.com/HexuanLiu/TOrth`

- C. elegans Graphical Model

  `https://github.com/HexuanLiu/PGM`

- Ensemble for COVID-19 Forecast

  `https://github.com/HexuanLiu/COVID-19-Forecast-Ensemble`