Robust Approaches for Unsupervised Learning

Olga Dorabiala

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:

Aleksandr Aravkin, Chair

Nathan J. Kutz

Thomas Trogdon

Program Authorized to Offer Degree:

Applied Mathematics

University of Washington

Abstract

Robust Approaches for Unsupervised Learning

Olga Dorabiala

Chair of the Supervisory Committee:
Aleksandr Aravkin
Department of Applied Mathematics

Today, data science and machine learning are a cornerstone in the engineering, physical, social, and biological sciences. Often, the data available in these fields is large and unlabeled, motivating the development of unsupervised learning methods that can efficiently extract information about object behavior with no human supervision. Unsupervised learning discovers the underlying patterns or structures of unlabeled data through the use of methods such as clustering, dimensionality reduction, and anomaly detection. Although often treated as separate problems, these methods have significant overlap in practice. When dealing with real-world data many traditional techniques are compromised by lack of clear separation between groups, noisy observations, and/or outlying data points. Thus, robust statistical algorithms are required for successful data analytics. In this work, we focus on the robustification of existing cluster analysis and dimensionality reduction techniques. In particular, we propose three new algorithms: Robust Trimmed k-means (RTKM), Spatiotemporal k-means (STKM), and Ensemble Principal Component Analysis (EPCA). RTKM augments the objective function in k-means clustering to create a flexible method that simultaneously identifies outliers and clusters points and can be applied to either single or multi-membership data. Using a similar approach, STKM reframes k-means for the spatiotemporal domain to address the moving cluster problem and proposes a noise robust extension as future work. Finally, EPCA ensembles boostrapped Principal Component Analysis (PCA) with k-means clustering to create a scalable, noise-resistant extension of PCA that lends itself naturally to uncertainty quantification. We introduce each of our methods, demonstrate their effectiveness against current competitors, and discuss potential for future work.

Contents

Chapter 1

Introduction

Data science and machine learning have revolutionized the way that we do science today. Intelligent systems are used in the engineering, physical, social, and biological sciences to take in data and output, among other critical information, actionable decision making capabilities or data analyses that detail correlations between important features [26, 59, 70]. The three major paradigms of machine learning are supervised, unsupervised, and reinforcement learning [65]. These three classes describe the kind of data used to structure learning tasks. In supervised learning, the goal is to generate a learned mapping from inputs to outputs given labeled data. The simplest example is linear regression, while a more complex example with high impact in recent years is deep neural networks [66, 70]. In contrast, unsupervised learning is used to discover the underlying patterns or structures of unlabeled data. This area includes methods for exploratory data analysis, such as dimensionality reduction and clustering. Finally, reinforcement learning learns how to map situations to actions, so as to maximize a numerical (delayed) reward signal [75]. The idea is to capture a learning agent interacting with its environment to achieve a goal through cumulative and delayed rewards.

1.1 Supervised Learning

The most widely used class of machine learning, supervised learning, uses labeled datasets to train algorithms that can create an accurate map from input to outputs [65]. As the model learns from the training data, the accuracy of the map is quantified through some sort of loss function. The weights of the model are iteratively adjusted until the error in

the loss function is minimized. The main goal of supervised learning is to ensure that models create an accurate map from inputs to outputs and also generalize beyond their training set to unseen data. Two sources of error that prevent generalization are bias and variance. Bias is the difference between average model prediction and the true values. Models with high bias oversimplfy the true map from inputs to outputs and suffer from underfitting. On the other hand, variance is the sensitivity of the model to small changes in the training set, and models with high variance overfit the model to the training data [59]. Ideally, practitioners seek a model with both low bias and variance, but the bias-variance tradeoff makes it nearly impossible to achieve both simultaneously. Increasing the complexity of a model reduces bias but increases variance and vice versa.

Cross-validation is a method utilized to address the challenge of model generalization. Instead of simply using train and test data, cross-validation splits the training data to create an additional validation set. The train set is referred to as "seen" data and the validation set as "unseen" data. The model is trained on the seen data and regularly tested on the unseen data. If model accuracy is high on the seen data, but low on the unseen data, this flags the issue of overfitting. Generally, multiple rounds of cross-validation are carried using different partitions of the train set. Examples of cross-validation include Leave-one-out cross validation (LOOCV), Leave-p-out cross validation (LPOCV), and k-fold cross validation [59].

Supervised learning can be broken into two categories: regression and classification. Regression aims to understand the relationship between one or more independent variables and dependent variables and potentially forecast future behavior. Examples of regression include linear and logistic regression. Linear regression is used for continuous data and seeks a line of best fit, typically quantified through the method of least squares, while logistic regression models the probability of a discreet binary outcome [59]. Both these models have high bias, but low variance.

The second category of supervised learning is classification, which has the goal of assigning test data into specific categories. Common examples of classification models include Support Vector Machines (SVM), decision trees, k-nearest neighbors (KNN), and Neural Networks [59]. SVMs construct a hyperplane, known as a decision boundary, that separates classes of data points on either side of the plane. Decision trees create a series of decision rules inferred from independent variables to predict the value of a dependent

variable. KNN is an algorithm that operates on the assumption that similar data points are found near one another and classifies points based on their proximity, and therefore assumed similarity, to one another. Finally, Neural Networks create non-linear maps from inputs to outputs. They consist of layers of nodes, where each node contains inputs, weights, a bias, and an output. If the output exceeds some threshold value, it passes into the next layer [70]. All of these methods are generally known to exhibit low bias, but high variance.

1.2 Reinforcement Learning

The second paradigm of machine learning, reinforcement learning, involves an agent that learns an action using feedback from previous actions and experiences. The "labels" in reinforcement learning are delayed reward signals that define the rewards and punishments for positive and negative behavior. The goal of a given agent is to find an action model, referred to as a policy, that maximizes total cumulative reward, which is measured by a value function [65]. All reinforcement learning agents have explicit goals, can sense all aspects of their environment, and can choose actions that influence their environments [75]. Finding the optimal policy for an agent is complicated by the exploration vs exploitation trade-off. To maximize reward, an agent must prefer actions it has tried in the past and obtained reward for, but to discover such actions, it has to try actions that it has not selected before [75]. The dilemma arises from the fact that the agent cannot do both at the same time.

Almost all reinforcement learning problems can be formulated as Markov Decision Processes (MDPs). The MDP is used to formally define the relationships between an agent's states, actions, and rewards. In a series of discreet time steps $t = 0, 1, 2, 3, \cdots$, at each time t, the agent receives some indication of its state $S_t \in S$ and on that basis selects an action $A_t \in A(s)$. As a consequence, the agent then receives a deylayed numerical reward signal one time step later $R_{t+1} \in R$ and finds a new state S_{t+1}. In a finite, MDP, S, A, and R all have a finite number of elements, and, as a consequence, the random variables R_t and S_t have well defined discrete probability distributions dependent on the preceeding state and action. Mathematically the dynamics of the MDP are defined as

$$p(s', r|s, a) = Pr\Big\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\Big\} \tag{1.1}$$

for all $s', s \in S$ and $a \in A(s)$ [75].

1.3 Unsupervised Learning

The relative successes of supervised and reinforcement learning are directly related to the availability of extensive labeled data. In the absence of labels, these methods are known to perform poorly. Semi-supervised learning attempts to address this issue by augmenting unlabeled data with smaller portions of labeled data [70]. However, it is often infeasible or expensive to manually label even a subset of a high-dimensional dataset. In these cases, unsupervised learning techniques are the only available approach for extracting information.

Unsupervised learning techniques for discovering patterns and trends in data without human intervention include cluster analysis, dimensionality reduction, and anomaly detection. Although often treated as separate problems, these methods have significant overlap in practice. Cluster analysis seeks to divide a set of objects so as to maximize both intra-cluster similarity and inter-cluster differences. There are a wide variety of clustering methods, most of which fall into the categories of hierarchical, density-based, and partitioning [24]. We explore clustering in Chapters 2-4. Next, dimensionality reduction assumes that although a dataset may exist in a high dimensional space, not all dimensions are equally as informative, and finds a low-dimensional representation that contains as much as possible of the variation [59]. We explore dimensionality reduction in Chapters 5 and 6. Both cluster analysis and dimensionality reduction are compromised by lack of clear separation between features, noisy observations, and/or outlying data points and require robustification [26, 59], which is what we aim to address throughout this work. Since the aim of anomaly detection is to identify outliers in the dataset, the intersection within these unsupervised learning techniques is clear.

1.4 Extending Unsupervised Learning to Noisy Data

The poor performance of machine learning algorithms on data with corruption and noise has long been acknowledged. In the 1960s, John Tukey was the first to recognize the need for robust methods, coining the term robust statistics [14, 72]. Tukey was agnostic to any particular procedure, but simply insisted that working with real data required robustification in order to stabilize the performance and predictive power of machine learning and statistical methods. Since that time, scientists have proposed numerous robustification techniques, including for clustering and dimensionality reduction in unsupervised learning.

Our specific algorithmic innovations pertain to developing robust clustering algorithms, as well as noise-agnostic methods for dimensionality reduction. In the second chapter, we introduce the various classes of cluster analysis. We formulate partion-based clustering as an optimization problem based on a least squares objective, and show that the classic k-means algorithm is a particular way to solve it. We show that a broad class of methods can be applied to solve the formulation, but this does not prevent sensitivity to outliers, which is due to the formulation itself.

In Chapter Three, we review state of the art approaches to robust k-means clustering. Current methods that robustify k-means are specialized for either single or multi-membership data, but do not perform competitively in both cases. We propose an extension of the k-means algorithm, which we call Robust Trimmed k-means (RTKM) that simultaneously identifies outliers and clusters points and can be applied to either single- or multi-membership data. We test RTKM on various real-world datasets and show that RTKM performs competitively with other methods on single membership data with outliers and multi-membership data without outliers. We also show that RTKM leverages its relative advantages to outperform other methods on multi-membership data containing outliers.

Chapter Four extends k-means to the spatiotemporal domain. Spatiotemporal data is readily available due to emerging sensor and data acquisition technologies that track the positions of moving objects of interest. Spatiotemporal clustering addresses the need to efficiently discover patterns and trends in moving object behavior without human supervision. One application of interest is the discovery of moving clusters, where clusters

have a static identity, but their location and content can change over time. We propose a two phase spatiotemporal clustering method called Spatiotemporal k-means (STKM) that is able to analyze the multi-scale relationships within spatiotemporal data. Phase 1 of STKM frames the moving cluster problem as the minimization of an objective function unified over space and time. It outputs the short-term associations between objects and is uniquely able to track dynamic cluster centers with minimal parameter tuning and without post-processing. Phase 2 outputs the long-term associations and can be applied to any method that provides a cluster label for each object at every point in time. We evaluate STKM against baseline methods on a recently developed benchmark dataset and show that STKM outperforms existing methods, particularly in the low-data domain, with significant performance improvements demonstrated for common evaluation metrics on the moving cluster problem. We propose a noise robust extension of STKM as future work.

Chapter Five transitions to an introduction on dimensionality reduction. We explain the motivations for low-dimensional representations of high-dimensional data structures and discuss existing dimensionality reduction methods. We focus on linear techniques that are cornerstones for data analysis, particularly the Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). We offer an intuitive geometric interpretation of these methods and showcase their use for both data compression and data separation.

In Chapter Six, we propose an extension of Principal Component Analysis (PCA). Efficient representations of data are essential for processing, exploration, and human understanding, and PCA is arguably the most common dimensionality reduction technique used for the analysis of large, multivariate datasets today. Two well-known limitations of the method include sensitivity to outliers and noise and no clear methodology for the uncertainty quantification of the purely descriptive information it provides. Whereas previous work has focused on each of these problems individually, we propose a scalable method called Ensemble PCA (EPCA) that tackles them simultaneously for data which has an inherently low-rank structure. EPCA combines boostrapped PCA with k-means cluster analysis to handle challenges associated with sign-ambiguity and re-ordering of components in the PCA subsamples. EPCA provide a noise-resistant extension of PCA that lends itself naturally to uncertainty quantification. We test EPCA on data corrupt with white noise, sparse noise, and outliers against methods that set the benchmark and show that EPCA performs competitively on all noise domains, with a clear advantage

on datasets containing outliers and orders of magnitude reduction in computational cost compared to other robust PCA methods.

Chapter Seven summarizes our contributions.

Chapter 2

K-Means Clustering

2.1 Introduction to Cluster Analysis

Cluster analysis is used to reveal patterns based on similarities between data points. There are a wide variety of clustering methods, most of which fall into the following categories: hierarchical, hierarchical, density-based, and partitioning [24].

Hierarchical methods find clusters using either a top-down or bottom-up approach. In the bottom-up approach, clusters begin as individual data points and fuse to become one large cluster. For the top-down approach, initially a single cluster comprises all points, and is then successively divided into sub-clusters. In either case, the process creates paths that are cut to stop at the desired similarity level. Because the paths can be cut at any point, there is not just one specific partition given as an output. There is an elegance to the approach, but the methods do not scale well to high-dimensional datasets. Just as more classic approaches, hierarchical clustering is sensitive to noise, which can lead to unreliable results at virtually any step along the all-to-one path.

Density-based methods assume that points in each cluster are drawn from a specific probability distribution and aim to identify the clusters and their respective distribution parameters. A common density-based method is the Expectation Maximization (EM) algorithm [5], which is based on the principle that one should choose the clustering structure and parameters so that the probability of the data being generated by such parameters is maximized. Outliers in the data can affect the estimates of the parameters, but it has been shown that using different probability distributions can diminish their effect [55].

We are particularly interested in partitioning methods, which are those that iteratively

relocate points between k clusters, starting from an initial partition. These methods can be split into two classes: error minimization algorithms and graph clustering. Error minimization algorithms aim to minimize an error criterion, such as the squared Euclidean distance between points and their respective cluster centers. One of the most popular algorithms using this error criterion is k-means. Versions of the the k-means algorithm date back to the mid-1950s, with seminal contributions from Steinhaus [2] and more modern versions developed by Lloyd [7] (published much later in 1982) and Forgy [3] in the mid-1960s. The k-means algorithm is simple, intuitive and can be directly applied without restrictions. Its simplicity and applicability have contributed to its appeal and wide-spread usage; it was named one of the top-10 algorithms in data mining in 2008 [35]. When data is well-separated and contains no outliers, k-means may be able to accurately assign labels to clusters [26]. Unfortunately, real-world data may be compromised by outliers and/or complicated by simultaneous membership to multiple clusters. Under these conditions, k-means is known to perform poorly. The other type of partitioning methods produce clusters via graphs. The edges of the graph connect data points, represented as nodes. The edges take on different weights, depending on the strength of the connection between data points. Some graph-based methods, such as spectral clustering [15], involve the use of another clustering method like k-means, as part of their algorithm. Therefore, they are only as robust to outliers as their component algorithms.

2.2 Classical K-Means Clustering

We first review a fundamental connection between k-means clustering and optimization. The k-means algorithm can be viewed as an alternating minimization approach to solving the following objective function

$$\min_{\mathbf{c},\mathbf{W}} \sum_{j=1}^{k} \sum_{i=1}^{N} w_{j,i} ||\mathbf{x}_i - \mathbf{c}_j||_2^2, \quad \text{where } \mathbf{W}_{:,i} \in \Delta_1. \tag{2.1}$$

The matrix $\mathbf{X} = [\mathbf{x}_1, \cdots \mathbf{x}_N] \in \mathbb{R}^{m \times N}$ contains the data points and the matrix $\mathbf{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_k] \in \mathbb{R}^{m \times k}$ contains the cluster centers [21]. The matrix $\mathbf{W} \in \mathbb{R}^{k \times N}$ contains

auxiliary weights $w_{j,i}$ that map the point-to-cluster relationship. Each column i of $\mathbf{W}$ assigns point $\mathbf{x}_i$ to a cluster whose center is $\mathbf{c}_j$. The constraint $\mathbf{W}_{:,i} \in \Delta_1$ ensures that each column of $\mathbf{W}$ belongs to the 1-capped simplex, i.e. $\sum_{j=1}^{k} w_{j,i} = 1$ for all $1 \leq i \leq N$, as desired. If $w_{j,i} = 0$, point $\mathbf{x}_i$ has no membership to cluster j. Constraining weights to belong to the discrete set $w_{j,i} \in \{0, 1\}$, is a mixed integer problem that is non-smooth and non-convex. It has been shown that the problem of finding the optimal assignment of points to clusters to minimize the objective in Equation 2.1 is NP-hard, even for just two clusters [51].

The simplest approach to minimizing the k-means objective function is Lloyd's (1982) algorithm [7], shown in Algorithm 1. Lloyd's algorithm is guaranteed to converge to at least a local minimum [7]. Once cluster centers are randomly initialized, each point is alternatively assigned to its closest centroid and then cluster centers are updated by taking the mean of all points in an assigned cluster. This iterative process continues until convergence. The approach can be understood as a Gauss-Seidel type method for the nonsmooth, nonconvex problem 2.1. In each iteration, the variables are alternatively minimized as shown in 2.3 and 2.2 until convergence [21].

Algorithm 1 Lloyd's Algorithm

procedure LLOYD($\mathbf{X}, k$) ▷ Input $\mathbf{X} = [\mathbf{x}_1, \cdots \mathbf{x}_n]$ and k

Initialize $\mathbf{C}_0 = [\mathbf{c}_1, \cdots \mathbf{c}_k]$

while not converged **do**

$$w_{j,i}^{q+1} = \begin{cases} 1 & \text{if } ||\mathbf{x}_i - \mathbf{c}_j^{q+1}||_2^2 \leq ||\mathbf{x}_i - \mathbf{c}_t^{q+1}||_2^2 \text{ for } 1 \leq t \leq j \\ 0 & \text{for } j \neq t \end{cases} \tag{2.2}$$

$$\mathbf{c}_j^{q+1} = \frac{\sum_{i=1}^{N} w_{j,i}^{q} \mathbf{x}_i}{\sum_{i=1}^{N} w_{j,i}^{q}} \tag{2.3}$$

return $\mathbf{C}, \mathbf{W}$

While problem 2.1 is essentially combinatorial, we can state an equivalent relaxation of the problem by allowing the auxiliary weights $w_{j,i}$ to instead belong to the continuous interval $[0, 1]$, as opposed to the discrete set $\{0, 1\}$ [21]. If $w_{ij} \neq 0$, then point $\mathbf{x}_i$ has some fractional membership to cluster j.

Solving the k-means objective in a relaxed fashion allows for greater modeling flexibility. In standard k-means, the columns of the weight matrix are projected onto the

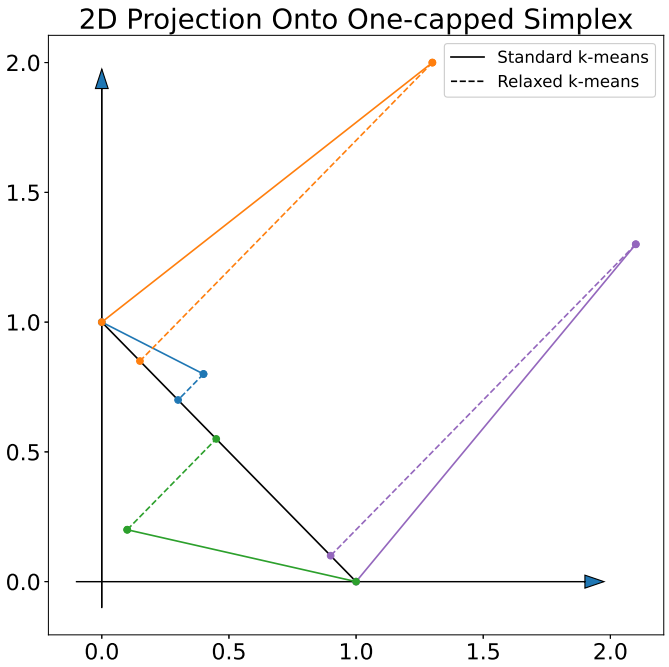


Figure 2.1: Comparison between the behavior of standard and relaxed k-means in projecting two-dimensional points onto the one-capped simplex. In standard k-means, the points are projected onto the vertices of the capped simplex, while in relaxed k-means, points take on a continuum of values.

vertices of the capped simplex each iteration, as shown in 2.2. In the relaxation, the weights can take on a continuum of values in the capped simplex. They can optionally be projected onto the vertices during cluster assignment after convergence. A comparison of the behavior of standard and relaxed k-means in projecting two-dimensional points onto the one-capped simplex is shown in Figure 2.1.

Figure 2.2 gives a glimpse into how the relaxed clustering process works. Point colors existing on the gradient between blue and green represent the continuous cluster weights. In iteration 0, the weights are randomly assigned and cluster centers are initialized at random. By iteration 3, the centers begin to drift apart, and two clusters begin to form. Points on the boundary remain on the gradient between blue and green. In iteration 5, cluster centers begin to stabilize, and in iteration 20, relaxed k-means converges, and all the weights are on the vertices of the 1-capped simplex. The relaxation of weighted k-means allows us to quantify the extent of membership of a point to each cluster at every iteration.

For a fixed cluster assignment, relaxed k-means has a vertex solution that corresponds

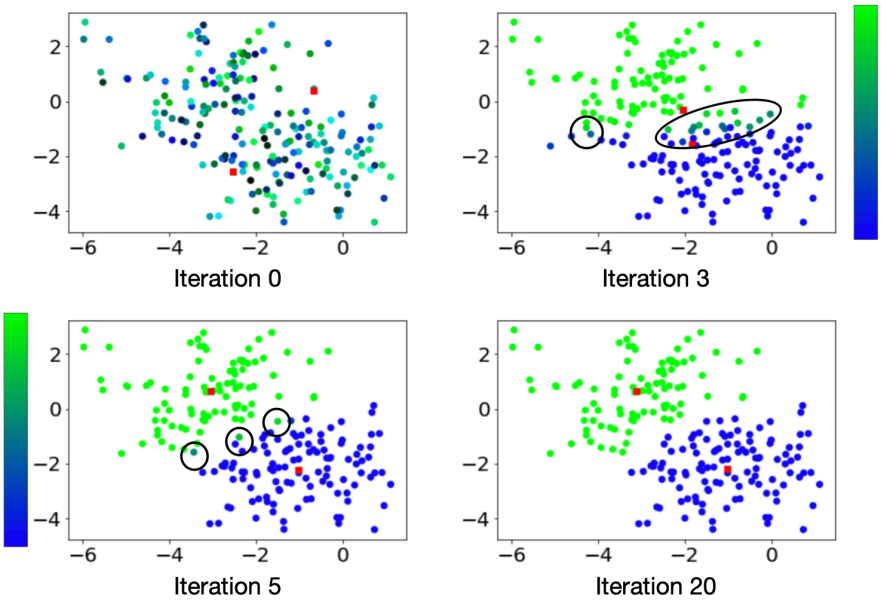


Figure 2.2: Clustering process of relaxed k-means. The colors of the points represent the continuum of weights that assign points to clusters. In iteration 0, all weights and cluster centers are assigned randomly. In iteration 3, two distinct clusters begin to form. In iteration 5, cluster centers begin to stabilize and points on the boundary retain partial membership to both clusters. In iteration 20, relaxed k-means converges to two distinct clusters.

to the combinatorial solution in 2.2. In particular, we can find a solution with the same objective value where each point is restricted to belong to exactly one cluster, to exhibit a solution for 2.1. Problem 2.1 can be solved with a variety of algorithms, as discussed in detail in the next section.

2.3 Solution Methods for k-means Clustering

In general, Equation 2.1 can be solved using iterative minimization methods.

A suitable class of such methods aims to solve a more general problem of the form

$$\min_{\mathbf{x},\mathbf{y}} \Psi(\mathbf{x},\mathbf{y}) \coloneqq f(\mathbf{x}) + g(\mathbf{y}) + H(\mathbf{x},\mathbf{y}), \tag{2.4}$$

where f and g are extended-valued (i.e., allowing the inclusion of constraints) and H is smooth. In the case of k-means clustering, we can take $f = 0$ and write problem 2.1 as

$$\min_{\mathbf{c},\mathbf{W}} \Psi(\mathbf{c},\mathbf{W}) \coloneqq H(\mathbf{c},\mathbf{W}) + g(\mathbf{W}), \tag{2.5}$$

where

$$H(\mathbf{c}, \mathbf{W}) = \sum_{j=1}^{k} \sum_{i=1}^{N} w_{j,i} ||\mathbf{x}_i - \mathbf{c}_j||_2^2, \tag{2.6}$$

and

$$g(\mathbf{W}) = \sum_{i=1}^{N} \delta_{\Delta_1}(\mathbf{W}_{:,i}).$$

The function δ_{Δ_1} denotes the sum of convex indicator functions that model the constraint that each column of $\mathbf{W}$ belongs in Δ_1.

2.3.1 Gauss-Seidel Method

The simplest approach to solving Equation 2.4 is the Gauss-Seidel method. Applied to the clustering problem in Equation 2.5, Gauss-Seidel starts with an initial point $(\mathbf{c}_0, \mathbf{W}_0)$ and generates a sequence $\{(\mathbf{c}^q, \mathbf{W}^q)\}_{q \in \mathbb{N}}$ via the scheme

$$\begin{aligned} \mathbf{c}^{q+1} &\in \arg\min_{\mathbf{c}} \Psi(\mathbf{c}, \mathbf{W}^q), \\ \mathbf{W}^{q+1} &\in \arg\min_{\mathbf{W}} \Psi(\mathbf{c}^{q+1}, \mathbf{W}^q). \end{aligned} \tag{2.7}$$

As shown by Powell [4], the Gauss-Seidel method is guaranteed to converge as long as the minimum of each step in the iteration scheme is uniquely attained. The subproblem for $\mathbf{c}$ is solved through a simple least squares update, derived below,

$$\begin{aligned} \nabla_{\mathbf{c}_j} \left(\sum_{j=1}^{k} \sum_{i=1}^{N} w_{j,i} ||\mathbf{x}_i - \mathbf{c}_j||_2^2 \right) &= 0, \\ \sum_{i=1}^{N} -2 w_{j,i} (\mathbf{x}_i - \mathbf{c}_j) &= 0, \\ \mathbf{c}_j^{q+1} &= \frac{\sum_{i=1}^{N} w_{j,i}^q \mathbf{x}_i}{\sum_{i=1}^{N} w_{j,i}^q}. \end{aligned} \tag{2.8}$$

It can be shown that the subproblem for $\mathbf{W}$ can be minimized uniquely as follows [6],

$$w_{j,i}^{q+1} = \begin{cases} 1 & \text{if } ||\mathbf{x}_i - \mathbf{c}_j^{q+1}||_2^2 \leq ||\mathbf{x}_i - \mathbf{c}_t^{q+1}||_2^2 \text{ for } 1 \leq t \leq j \\ 0 & \text{for } j \neq t \end{cases}. \tag{2.9}$$

This update scheme is the same as Lloyd's algorithm, as seen in Algorithm 1. Lloyd's algorithm is considered to have a "linear" time complexity of $\mathcal{O}(Nmki)$, where N is the number of points, m the dimension of the points, k the number of clusters, and i the number of iterations until convergence. In the worst case, Lloyd's algorithm requires $i = 2^{\times(\sqrt{n})}$ iterations to converge, so that time complexity is actually superpolynomial [25].

The weight update 2.9 ensures that $\mathbf{W}$ is a binary weight matrix, assigning each point to only one cluster. However, our desired constraint on $\mathbf{W}$ is looser, only stating that the entries of each column must sum to 1. Thus, each point $\mathbf{x}_i$ can have fractional membership and belong to more than one cluster. This feature is explored by fuzzy k-means clustering by slightly altering the objective function and the weight update accordingly [6]. We instead seek a weight update that allows for this flexibility directly by considering different optimization methods.

2.3.2 Proximal Alternating Minimization

Proximal Alternating Minimiziation (PAM), suggested by Attouch et al. [39], when applied to the clustering objective, generates a solution sequence $\{(\mathbf{c}^q, \mathbf{W}^q)\}_{q\in\mathbb{N}}$ to Equation 2.5 via a proximal regularization of the Gauss-Seidel scheme. For any function f and real $\nu > 0$, the proximal map is defined by

$$\text{prox}_{\nu f}(\mathbf{x}) \coloneqq \arg\min_{\mathbf{z}} \left\{ f(\mathbf{z}) + \frac{1}{2\nu}||\mathbf{z} - \mathbf{x}||_2^2 \right\}. \tag{2.10}$$

Then, the solution sequence $\{(\mathbf{c}^q, \mathbf{W}^q)\}_{q\in\mathbb{N}}$ for PAM is given by,

$$\begin{aligned} \mathbf{c}^{q+1} &\in \arg\min_{\mathbf{c}} \left\{ \Psi(\mathbf{c}, \mathbf{W}^q) + \frac{\mathbf{e}_q}{2}||\mathbf{c} - \mathbf{c}^q||_2^2 \right\}, \\ \mathbf{W}^{q+1} &\in \arg\min_{\mathbf{W}} \left\{ \Psi(\mathbf{c}^{q+1}, \mathbf{W}^q) + \frac{\mathbf{d}_q}{2}||\mathbf{W} - \mathbf{W}^q||_2^2 \right\}, \end{aligned} \tag{2.11}$$

where $(\mathbf{e}_q)_{q\in\mathbb{N}}$ and $(\mathbf{d}_q)_{q\in\mathbb{N}}$ are any positive sequences. PAM is guaranteed to converge as long as the objective function satisfies the Kurdyka-Łojasiewicz (KL) property as

described by [39]. The subproblem for each component of the $\mathbf{c}$ update is shown below,

$$\begin{aligned} \nabla_{\mathbf{c}_j}\left(\sum_{j=1}^{k}\sum_{i=1}^{N} w_{j,i}||\mathbf{x}_i - \mathbf{c}_j||_2^2 + \frac{\mathbf{e}_q}{2}||\mathbf{c}_j - \mathbf{c}_j^q||_2^2\right) = 0, \\ \sum_{i=1}^{N} -2w_{j,i}(\mathbf{x}_i - \mathbf{c}_j) + \mathbf{e}_q(\mathbf{c}_j - \mathbf{c}_j^q) = 0, \\ \mathbf{c}_j^{q+1} = \frac{\mathbf{e}_q\mathbf{c}_j^q + \sum_{i=1}^{N} w_{j,i}^q\mathbf{x}_i}{\mathbf{e}_q + \sum_{i=1}^{N} w_{j,i}^q}. \end{aligned} \tag{2.12}$$

The subproblem for $\mathbf{W}$ is solved as follows,

$$\begin{aligned} \nabla_{\mathbf{W}_{:,i}}\left(\sum_{j=1}^{k}\sum_{i=1}^{N} \mathbf{W}_{:,i}||\mathbf{x}_i - \mathbf{C}||_2^2 + \frac{\mathbf{d}_q}{2}||\mathbf{W}_{:,i} - \mathbf{W}_{:,i}^q||_2^2\right) = 0, \\ ||\mathbf{x}_i - \mathbf{C}||_2^2 + \mathbf{d}_q(\mathbf{W}_{:,i} - \mathbf{W}_{:,i}^q) = 0, \\ \mathbf{W}_{:,i}^{q+1} = \mathbf{W}_{:,i}^q - \frac{1}{\mathbf{d}_q}||\mathbf{x}_i - \mathbf{C}||_2^2. \\ \text{Since } \mathbf{W}_{:,i} \in \Delta_1,\ \mathbf{W}_{:,i}^{q+1} = \text{proj}_{\Delta_1}\left(\mathbf{W}_{:,i}^q - \frac{1}{\mathbf{d}_q}||\mathbf{x}_i - \mathbf{C}^{q+1}||_2^2\right). \end{aligned} \tag{2.13}$$

The projection $\text{proj}_S(\mathbf{x})$ maps point $\mathbf{x}$ to it's closest point in set S. Thus, updates to each column of $\mathbf{W}$ are simply projections of the gradient descent step onto Δ_1. Now, each point can belong to more than one cluster. The PAM scheme has a time complexity of $\mathcal{O}(Nmki)$, where N is the number of points, m the dimension of points, k the number of clusters, and i the number of iterations required until convergence.

2.3.3 Proximal Alternating Linearized Minimization

A different approach that can be used to solve Equation 2.5, is Proximal Alternating Linearized Minimization (PALM), proposed by Bolte et. al [61]. PALM approximates the solution to PAM by solving the proximal linearization of each subproblem in Equation 2.11, as shown in Equation 2.14. It too converges as long as the objective function satisfies the KL property. The updates for the center and weight matrices are

$$\begin{aligned}\mathbf{c}^{q+1} &\in \arg\min_{\mathbf{c}} \left\{ \langle \mathbf{c} - \mathbf{c}^q, \nabla_{\mathbf{c}} H(\mathbf{c}^q, \mathbf{W}^q) \rangle + \frac{e_q}{2} ||\mathbf{c} - \mathbf{c}^q||_2^2 + f(\mathbf{c}) \right\}, \\ \text{and } \mathbf{W}^{q+1} &\in \arg\min_{\mathbf{W}} \left\{ \langle \mathbf{W} - \mathbf{W}^q, \nabla_{\mathbf{W}} H(\mathbf{c}^{q+1}, \mathbf{W}^q) \rangle + \frac{d_q}{2} ||\mathbf{W} - \mathbf{W}^q||_2^2 + g(\mathbf{W}) \right\}.\end{aligned} \tag{2.14}$$

The iterative scheme in Equation 2.14 can be simply re-written as alternating proximal gradient descent, as shown in Equation 2.15,

$$\begin{aligned}\mathbf{c}^{q+1} &\in \text{prox}_{e_q}^f \left(\mathbf{c}^q - \frac{1}{e_q} \nabla_{\mathbf{c}} H(\mathbf{c}^q, \mathbf{W}^q) \right), \\ \mathbf{W}^{q+1} &\in \text{prox}_{d_q}^g \left(\mathbf{W}^q - \frac{1}{d_q} \nabla_{\mathbf{W}} H(\mathbf{c}^{q+1}, \mathbf{W}^q) \right).\end{aligned} \tag{2.15}$$

The parameters $e_q = \gamma L_1$ and $d_q = \gamma L_2$ are multiples of the Lipschitz constants L_1 and L_2 for the partial gradients $\nabla_{\mathbf{c}} H(\mathbf{c}, \mathbf{W})$ and $\nabla_{\mathbf{W}} H(\mathbf{c}, \mathbf{W})$, respectively. The parameter γ is any constant such that $\gamma > 1$. Since $f(\mathbf{c}) = 0$ and the proximal operator of a constant function is simply the argument of the operator, the $\mathbf{c}$ update is a gradient descent step.

$$\mathbf{c}_j^{q+1} = \mathbf{c}_j^q - \frac{1}{e_q} \sum_{i=1}^{N} -2 w_{j,i} (\mathbf{x}_i - \mathbf{c}_j), \tag{2.16}$$

where $L_1 = 2N$ is the Lipschitz constant such that

$$||\nabla_{\mathbf{c}} H(\mathbf{c}_1, \mathbf{W}) - \nabla_{\mathbf{c}} H(\mathbf{c}_2, \mathbf{W})||_2 \leq L_1 ||\mathbf{c}_1 - \mathbf{c}_2||_2 \text{ for all } \mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^m. \tag{2.17}$$

Thus, $e_q = \gamma 2N$ for some $\gamma > 1$. The subproblem for $\mathbf{W}$ simplifies to have the same update as in Equation 2.13, as shown below,

$$\begin{aligned}\mathbf{W}_{:,i}^{q+1} &= \text{prox}_{d_q}^f \left(\mathbf{W}_{:,i}^q - \frac{1}{d_q} \nabla_{\mathbf{W}_{:,i}} \left(\sum_{i=1}^{N} \sum_{j=1}^{k} \mathbf{W}_{:,i} ||\mathbf{x}_i - \mathbf{C}||_2^2 \right) \right) \\ &= \text{prox}_{d_q}^f \left(\mathbf{W}_{:,i}^q - \frac{1}{d_q} ||\mathbf{x}_i - \mathbf{C}||_2^2 \right) \\ &= \text{proj}_{\Delta_1} \left(\mathbf{W}_{:,i}^q - \frac{1}{d_q} ||\mathbf{x}_i - \mathbf{C}^q||_2^2 \right),\end{aligned} \tag{2.18}$$

where the Lipschitz constant for $\nabla_{\mathbf{W}} H(\mathbf{c}, \mathbf{W})$, L_2, can be any positive constant since

$$||\nabla_{\mathbf{W}} H(\mathbf{c}, \mathbf{W}_1) - \nabla_{\mathbf{W}} H(\mathbf{c}, \mathbf{W}_2)|| = 0. \tag{2.19}$$

Then, $d_q = \gamma$ where $\gamma > 1$. Again, note that this $\mathbf{W}$ update allows each point to belong to more than one cluster. The PALM scheme also has a time complexity of $\mathcal{O}(Nmki)$, where N is the number of points, m the dimension of points, k the number of clusters, and i the number of iterations required until convergence.

2.3.4 Combination Method

Yet another way to solve Equation 2.5 is to use a combination of any of the aforementioned methods. The fastest iterative step for $\mathbf{c}$ will be the direct least-squares update given by the Gauss-Seidel method. However, the $\mathbf{W}$ update using Guass-Seidel does not provide us with the desired flexibility. Therefore, we use either a PALM or PAM update for $\mathbf{W}$, since they are the same. A projection of $\mathbf{W}$ onto Δ_1 allows each point to belong to more than one cluster. If we want to enforce single cluster membership, we determine final assignments as the arg max over each column of $\mathbf{W}$. Algorithm 2 displays our chosen alternative minimization method to optimize the objective function in Equation 2.1. The combination scheme has a time complexity of $\mathcal{O}(Nmki)$, where N is the number of points, m is the dimension of points, k the number of clusters and i the number of iterations required until convergence.

Algorithm 2 Alternating Minimization for k-means

procedure AMKMEANS($\mathbf{X}, k, d_q$) ▷ Input $\mathbf{X} = [\mathbf{x}_1, \cdots \mathbf{x}_n]$, k, and $d_q > 1$
  Initialize $\mathbf{C}_0 = [\mathbf{c}_1, \cdots \mathbf{c}_k]$
  while not converged **do**
    $\mathbf{c}_j^{q+1} = \frac{\sum_i w_{j,i}^q \mathbf{x}_i}{\sum_i w_{j,i}^q}$
    $\mathbf{W}_{:,i}^{q+1} = \text{proj}_{\Delta_1}\left(\mathbf{W}_{:,i}^q - \frac{1}{d_q}||\mathbf{x}_i - \mathbf{C}^{q+1}||_2^2\right)$
  return $\mathbf{C}, \mathbf{W}$

Our approach to solving the k-means objective function gives us greater modeling flexibility. Because we let each $w_{j,i}$ take on a continuum of values between 0 and 1 instead of restricting $\mathbf{W}$ to a binary weight matrix each iteration, we are able to capture more information and in the end find better local minima. However, even with this

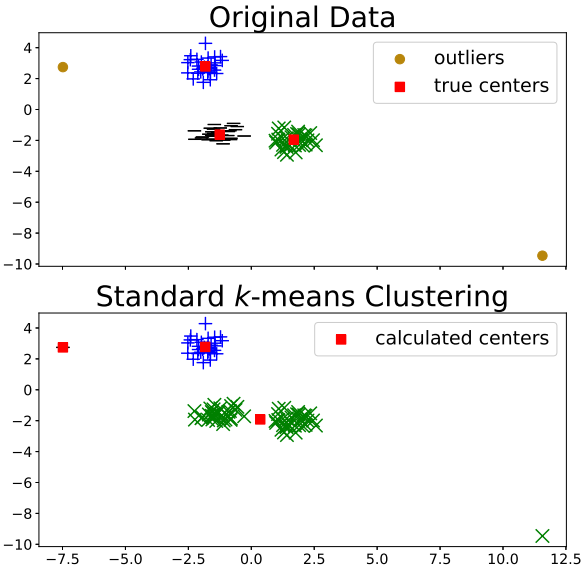


Figure 2.3: (top) Original data consisting of three clusters and two outliers. (bottom) k-means incorrectly assigns a cluster center to the outlier on the right, causing two clusters to be misidentified as one. The outlier on the left skews one cluster center towards the bottom right.

improvement, our solution method does not address the inherent shortcomings of k-means. One of the drawbacks of k-means is sensitivity to outliers and noise. Since points are classified by directly threshholding on the distance from cluster centers, outliers skew center assignment, and in turn point assignment, dramatically. As shown in Figure 2.3 in the top-most cluster, in the best case scenario, a cluster center is inaccurate, but points in the clusters are still classified correctly. In the worst case scenario, as seen in the bottom two clusters, a cluster center is pulled toward the outliers, forcing the remaining points to be completely misclassified. K-means is also known to suffer from sensitivity to cluster center initialization, a tendency to stall at local minima, and an inability to identify non-convex clusters. Even so, k-means is often favored against other methods due to its speed and ability to scale well to large data sets. Therefore, a version of k-means that is robust to outliers and noise is desired.

Chapter 3

Robust K-Means Clustering

3.1 Introduction

Clustering is a fundamental tool in unsupervised learning, used to group objects by distinguishing between similar and dissimilar features of a given data set. One of the most common clustering algorithms is k-means. Unfortunately, when dealing with real-world data many traditional clustering algorithms are compromised by lack of clear separation between groups, noisy observations, and/or outlying data points. Thus, robust statistical algorithms are required for successful data analytics. Current methods that robustify k-means clustering are specialized for either single or multi-membership data, but do not perform competitively in both cases. We propose an extension of the k-means algorithm, which we call *Robust Trimmed k-means* (RTKM) that simultaneously identifies outliers and clusters points and can be applied to either single- or multi-membership data. We test RTKM on various real-world datasets and show that RTKM performs competitively with other methods on single membership data with outliers and multi-membership data without outliers. We also show that RTKM leverages its relative advantages to outperform other methods on multi-membership data containing outliers.

3.2 Related Work

In this section, we review existing approaches to robustify clustering. We focus on methods that build upon the classical k-means algorithm, due to k-means' simplicity, speed, and scalability. These developments also apply to any algorithm that depends on k-means. An improved k-means can be integrated with methods that aim to capture

the non-linearity of real world data by finding the optimal latent space representation and clustering either sequentially or simultaneously. This covers algorithms such as spectral clustering and deep clustering methods, which can be applied to a variety of data distributions on which k-means alone may not perform well.

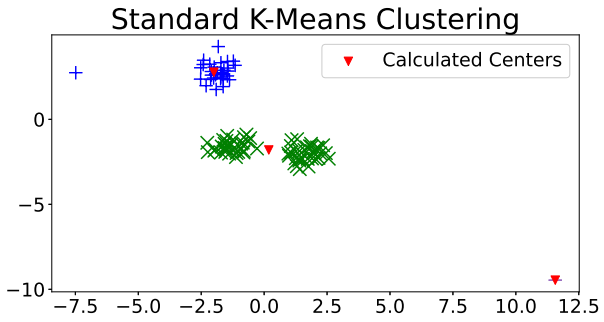
Numerous versions of k-means have been proposed over the years to address the issue of clustering in the presence of outliers. One of the earliest attempts to robustify k-means clustering is trimmed k-means [10], proposed in 1997. Trimmed k-means, detailed in Algorithm 3, works by running the standard k-means algorithm, removing a given percentage α of points with the greatest distance to their cluster centers, updating cluster centers as the mean of the remaining points in respective clusters, and repeating these steps until convergence.

Algorithm 3 Trimmed k-means

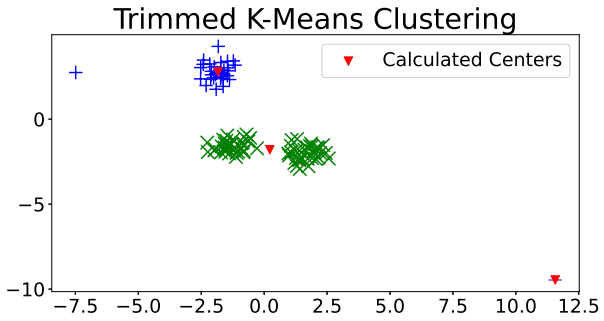
procedure TRIMMEDKMEANS($\mathbf{X}, k, \alpha$) ▷ Input $\mathbf{X} = [\mathbf{x}_1, \cdots \mathbf{x}_n]$, k, and α
  while not converged **do**
    Run k-means Clustering using Algorithm 1
    Trim $\alpha 100\%$ of points with the greatest distance to their cluster centers
    Update each cluster center as the mean of the remaining points in the cluster
  return C

Unfortunately, this method is ultimately ineffective, because it is unable to improve a poor result by the standard k-means algorithm. It is likely that points that should be classified as outliers are masked by the standard k-means clustering. This issue is displayed in Figure 3.1. Figure 3.1a shows the output of standard k-means on the dataset. One center is skewed by an outlier in the upper left-hand corner, and one center is incorrectly assigned to an outlier in the lower right-hand corner. This leaves the remaining two clusters to be misclassified as one. Figure 3.1b shows the improvement offered by trimmed k-means. Although the center that had previously been skewed is improved, the center assigned to the outlier is not readjusted. Once an outlier is incorrectly identified as a cluster center by the standard k-means algorithm, it will never be removed by trimmed k-means. Therefore, a method that directly tackles outliers during clustering is required.

The majority of proposed robust k-means methods perform clustering in stages. In the first stage, some clustering algorithm divides the dataset into clusters, and in the second stage, a measure based on the clusters is applied to the data to identify outliers. An example is the Outlier Removal Clustering (ORC) algorithm, proposed by Hatuamäki



(a)



(b)

Figure 3.1: (a) The result of standard k-means clustering on a dataset with three clusters and two outliers. One cluster's center is skewed by an outlier to its left, while another cluster center is pulled towards an outlier, causing two clusters to be incorrectly identified as one. (b)The result of trimmed k-means on the same dataset. The cluster whose center was previously skewed is adjusted correctly. However, the outlier misidentified as a cluster center cannot be improved.

et. al [20]. ORC first applies standard k-means, and then an outlyingness factor, o_i, is calculated for each data point as the ratio of a point's distance to its closest centroid and the maximum distance from the centroid to any other point. If o_i is beyond some specified threshold value, the point is identified as an outlier and removed from the dataset.

He et. al [17] proposed the cluster-based local outlier factor (CBLOF) as a different measure for identifying outliers. After some clustering algorithm is applied, CBLOF takes into account both the size of the cluster a point belongs to and the distance between the point and its closest cluster. This criteria encourages not only points that are far removed, but also clusters that do not contain many points to be classified as outliers. Other measures of outlierness include the outlier factor of a cluster C_i, OF(C_i), proposed by Jiang et al. [34] and Local Distance-Based Outlier Factor (LDOF), proposed by Zhang et al. [38]. Measure OF(C_i) defines the outlier degree of a cluster as the weighted sum of the distances between C_i and the other clusters. The larger OF(C_i) is, the greater the likelihood that C_i is an outlier cluster. Measure LDOF looks at the ratio between the

k-nearest in-cluster neighbors of a point and the average distance among all points in a cluster. It captures the degree to which a point deviates from its neighborhood system.

Far fewer methods tackle the problem of simultaneously clustering and identifying outliers. Among those that do are Outlier Detection and Clustering algorithm (ODC) [56], k-means $--$ [58], Non-exhaustive, Overlapping k-means (NEO k-means) [67], and k-means clustering with outlier removal (KMOR) [74]. ODC, which removes any data points that are at least p times the average distance away from their centroid, updates centroids as the average of remaining data points, and repeats these steps until cluster assignments no longer change. However, it is important to note that once a point is designated as an outlier and removed, it cannot be used as a normal point again once centers are updated. Similarly, k-means $--$ [58] takes as parameter l the desired number of outliers. Each iteration, all points are ranked in decreasing order by their distance to their nearest cluster center. Then, the top l points are removed and cluster centers are recalculated using the remaining data points. The process repeats until stabilization.

On single membership data containing outliers, KMOR outperforms all of the aforementioned methods in terms of cluster accuracy and outlier detection [74]. KMOR always produces a single-membership assignment. It identifies at most a given number of n_0 outliers by assigning them to a $k + 1^{th}$ cluster based on whether they are further away than γ times the average distance between inliers and their centroids. The parameter γ is difficult to identify when the proportion of outliers is unknown, and can drastically impact the algorithm. Additionally, of all available methods, only NEO k-means, which extends k-means to overlapping clusters with noise, allows points to belong to multiple clusters simultaneously. On multi-membership data without outliers, NEO k-means outperforms similar methods [67]. The algorithm uses a single binary weight matrix to classify points and identify outliers, where parameters σ and β give the user a way to specify the degree of overlap and proportion of outliers, respectively. Note that σ is denoted as α in the original paper. If a point is not assigned to any cluster, it is designated as an outlier. NEO-k-means exhaustively assigns some multiple $(1 + \sigma)$ of the total number points to clusters, where $\sigma \geq 0$. Thus NEO-k-means is unable to identify outliers on datasets containing only a single cluster and cannot be restricted to single-membership on datasets containing outliers.

We propose the Robust Trimmed k-means (RTKM) algorithm, which can be used

to classify either single or multi-membership data in the presence of outliers and noise. In our numerical results, we therefore focus on the natural comparison between RTKM, KMOR, and NEO k-means. We show that RTKM performs competitively with KMOR on single-membership data with outliers and with NEO k-means on multi-membership data without outliers. Moreover, RTKM leverages its advantages in these domains to achieve superior performance on multi-membership data containing outliers. In this way, RTKM can be effectively and competitively applied in multiple contexts.

3.3 Robust Trimmed k-Means

Robust Trimmed k-Means (RTKM) simultaneously classifies points and identifies outliers by minimizing the objective function given in Equation 3.1,

$$\begin{aligned} \min_{\mathbf{c},\mathbf{v},\mathbf{W}} \sum_{i=1}^{N} v_i \sum_{j=1}^{k} w_{j,i} ||\mathbf{x}_i - \mathbf{c}_j||_2^2, \\ \text{where } \mathbf{W} :, i \in \Delta_s, \\ \text{and } \mathbf{v} \in \Delta_{N-[\alpha N]}. \end{aligned} \tag{3.1}$$

As before, $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N] \in \mathbb{R}^{m \times N}$ are the data points, $\mathbf{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_k] \in \mathbb{R}^{m \times k}$ are the cluster centers, and $\mathbf{W} \in \mathbb{R}^{k \times N}$ is a matrix of weights with each column in the s-capped simplex. The variable $\mathbf{v} \in \mathbb{R}^N$ is a vector that identifies outliers and belongs to the $N - [\alpha N]$-capped simplex, where α is a given proportion of expected outliers. Here $[\cdot]$ denotes the nearest integer, where we round up for half integer values. The constraint on $\mathbf{v}$ ensures that $N - [\alpha N]$ points are designated as inliers, since $\sum_{i=1}^{N} v_i = N - [\alpha N]$. Similarly, the constraint on the columns of $\mathbf{W}$ ensures that every point is assigned to at least s clusters, since $\sum_{j=1}^{k} w_{j,i} = s$. This constraint allows for each point $\mathbf{x}_i$ to belong to more than one cluster. To enforce single cluster membership, we set $s = 1$ and make final assignments by calculating the arg max over each column of $\mathbf{W}$ once the algorithm converges.

The objective in Equation 3.1 is approximately solved using Algorithm 4. We alternately minimize the objective with respect to three variables: $\mathbf{W}, \mathbf{c}, \text{and } \mathbf{v}$. We use a Gauss-Seidel update for the centers and Proximal Alternating Minimization (PAM) [39]

updates for both the columns of $\mathbf{W}$ and for $\mathbf{v}$. Algorithm 4 is guaranteed to converge as long as $e_q > 1$ and $d_q > 1$. In practice, we set $e_q = d_q = 1.1$. The least squares update for the centers is derived below,

$$\begin{aligned} \nabla_{\mathbf{c}_j}\left(\sum_{i=1}^{N} v_i \sum_{j=1}^{k} w_{j,i}||\mathbf{x}_i - \mathbf{c}_j||_2^2\right) = 0, \\ \sum_{i=1}^{n} 2v_i w_{j,i}(\mathbf{x}_i - \mathbf{c}_j) = 0, \\ \mathbf{c}_j = \frac{\sum_{i=1}^{N} v_i w_{j,i}\mathbf{x}_i}{\sum_{i=1}^{N} v_i w_{j,i}}. \end{aligned} \tag{3.2}$$

The PAM steps for the columns of the weight variable and our noise point identifying vector $\mathbf{v}$ are derived below. Let f and g be the constraints on the columns of $\mathbf{W}$ and on $\mathbf{v}$ respectively. The parameters d_q and e_q are any constants such that $d_q, e_q > 1$. Our updates are

$$\begin{aligned} \mathbf{W}_{:,i}^{q+1} &= \text{prox}_{d_q}^{f}\left(\mathbf{W}_{:,i}^{q} - \frac{1}{d_q}\nabla_{\mathbf{W}_{:,i}}\left(\sum_{i=1}^{N} v_i \sum_{j=1}^{k} w_{j,i}||\mathbf{x}_i - \mathbf{c}_j||_2^2\right)\right) \\ &= \text{prox}_{d_q}^{f}\left(\mathbf{W}_{:,i}^{q} - \frac{1}{d_q} v_i||\mathbf{x}_i - \mathbf{C}||_2^2\right) \\ &= \text{proj}_{\Delta_1}\left(\mathbf{W}_{:,i}^{q} - \frac{1}{d_q} v_i||\mathbf{x}_i - \mathbf{C}||_2^2\right), \end{aligned} \tag{3.3}$$

and

$$\begin{aligned} \mathbf{v}^{q+1} &= \text{prox}_{\mathbf{c}_k}^{g}\left(\mathbf{v}^{q} - \frac{1}{e_q}\nabla_{v}\left(\sum_{i=1}^{N} v_i \sum_{j=1}^{k} w_{j,i}||\mathbf{x}_i - \mathbf{c}_j||_2^2\right)\right) \\ &= \text{prox}_{\mathbf{c}_k}^{g}\left(\mathbf{v}^{q} - \frac{1}{e_q}\sum_{j=1}^{k} w_{j,i}||\mathbf{x}_i - \mathbf{c}_j||_2^2\right) \\ &= \text{proj}_{\Delta_{N-[\alpha N]}}\left(\mathbf{v}^{q} - \frac{1}{e_q}\sum_{j=1}^{k} w_{j,i}||\mathbf{x}_i - \mathbf{c}_j||_2^2\right). \end{aligned} \tag{3.4}$$

Algorithm 4 Robust Trimmed k-means (RTKM)

procedure RTKM($\mathbf{X}, k, \alpha, s$) ▷ Input $\mathbf{X}$, k, α, and s
  Initialize $\mathbf{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_k]$, $e_q = 1.1$, and $d_q = 1.1$
  while not converged **do**
    $\mathbf{c}_{:,j}^{q+1} = \sum_{i=1}^{N} v_i^q w_{j,i}^q \mathbf{x}_i / \sum_{i=1}^{N} v_i^q w_{j,i}^q$
    $\mathbf{W}_{:,i}^{q+1} = \text{proj}_{\Delta_s}\left(\mathbf{W}_{:,i}^q - \frac{1}{d_q} v_i^q ||\mathbf{x}_i - \mathbf{C}^{q+1}||_2^2\right)$
    $\mathbf{v}^{q+1} = \text{proj}_{\Delta_{N-[\alpha N]}}\left(\mathbf{v}^q - \frac{1}{e_q} \sum_{j=1}^{k} \mathbf{W}_{j,:}^{q+1} ||\mathbf{X} - \mathbf{c}_j^{q+1}||_2^2\right)$
  return $\mathbf{C}, \mathbf{W}, \mathbf{v}$

Notice that the weight update is the projection of the gradient descent step for each column of $\mathbf{W}$ onto Δ_1, and the $\mathbf{v}$ update is the projection of the gradient descent step for $\mathbf{v}$ onto $\Delta_{N-[\alpha N]}$.

The algorithm takes as input the data $\mathbf{X}$, the number of clusters k, and parameters α and s, which control the number of outliers and number of members in each cluster, respectively. Centroids can be initialized using any scheme. If we know the percentage of outliers in the data, we set α equal to that value. Likewise, if we know the cardinality of the data, we set $s = [\text{cardinality}]$, where $[\cdot]$ denotes the closest integer and we round up for half-integer values. Currently, there is no principled approach to estimating these parameters. However, we demonstrate that RTKM performs competitively with other methods on data containing outliers using a range of α values.

3.4 Experiments

We compare the performance of RTKM against other methods that simultaneously perform clustering and outlier detection, specifically KMOR [74] and NEO-k-means [67]. We focus our comparison against KMOR and NEO-k-means, because these methods were shown to outperform others on various datasets. KMOR [74] outperformed ODC [56], k-means$-$ [58] and NEO-k-means [67] on single-membership data containing outliers, while NEO-k-means [67] outperformed six other methods on multi-membership data without outliers. We evaluate the performance of RTKM on three types of datasets: single-membership datasets containing outliers, multi-membership datasets without outliers, and mutli-membership datasets with outliers.

The quality of the cluster assignments is measured using the metric in [67], average F_1

score, which quantifies how well each algorithm finds the ground truth clusters. The F_1 score is defined as

$$F_1 = \frac{TP}{TP + 0.5(FP + FN)}, \tag{3.5}$$

where TP denotes true positives, FP denotes false positives, and FN denotes false negatives. This metric ranges from 0 to 1, with values closer to 1 implying better classification. To calculate the average F_1 score, predicted clusters are matched to ground-truth clusters so that the average F_1 score among all clusters is maximized. Outliers are considered their own cluster for the purpose of calculating the average F_1 score.

The ability to correctly identify outliers is measured using the distance of a classifer on the Receiver Operating Curve graph from the perfect outlier classifier (M_e) as in [74]. The measure M_e is defined as

$$M_e = \sqrt{(FP_{\text{rate}})^2 + (1 - TP_{\text{rate}})^2}, \tag{3.6}$$

,

where TP_{rate} and FP_{rate} denote ratios of true positives to all positives and true negatives to all negatives, as shown below:

$$TP_{\text{rate}} = \frac{TP}{TP + FN}, \tag{3.7}$$

and

$$FP_{\text{rate}} = \frac{FP}{FP + TN}. \tag{3.8}$$

The M_e score depends only on the true and identified outliers in the dataset. The value of M_e ranges from 0 to $\sqrt{2}$, with better outlier classifiers having values closer to 0.

For data containing outliers, we test the performance of each method in terms of both F_1 and M_e score with various values for the expected percentage of outliers to see how sensitive results are to parameter choice. For every value of α, we complete 50 runs of an algorithm with different cluster center initialization each time. Performance metrics are reported as the minimum, maximum, and average M_e and F_1 scores over the 50 runs.

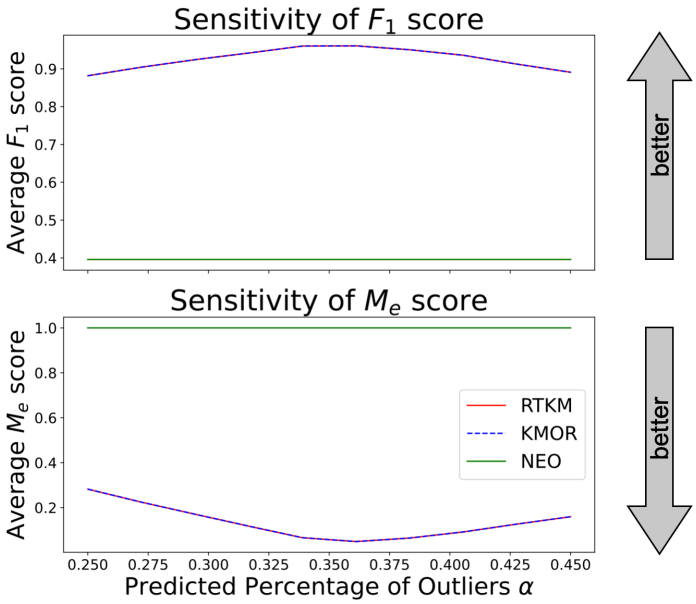


Figure 3.2: Sensitivity of RTKM, KMOR, and NEO-k-means to choice of α on the WBC dataset. RTKM and KMOR exhibit almost identical performance. NEO-k-means is unable to identify any outliers. (Parameters: $k = 1$, RTKM $s = 1$, KMOR $\gamma = 1$, NEO-k-means $\sigma = 0$)

3.4.1 Single-membership Data with Outliers

We begin by evaluating RTKM against KMOR and NEO-k-means on two single-membership datasets containing outliers. The first is the Breast Cancer Wisconsin (WBC) dataset [8] from UCI Machine Learning Repository [73]. The WBC dataset contains 699 instances of tumors with 9 attributes each, all of which are classified as benign or malignant, with the latter being treated as outliers.

Figure 3.2 shows performance metrics for the three algorithms over a range of α values on the WBC dataset. RTKM and KMOR exhibit almost identical performance when given the same parameters, while NEO-k-means, by design, is unable to identify any outliers when $k = 1$.

Next, we test the three algorithms on the shuttle training dataset [73]. The data contains 43,500 records and 7 classes, described by 9 numerical features. The three largest classes contain 99.57% of the data, so we consider these classes to be inliers and the remaining four to be outliers, as in [74]. We again test the sensitivity of the results for RTKM, KMOR, and NEO-k-means to choice of α. Figure 3.3 shows that while $\alpha \leq 0.02$, all three methods perform similarly. Beyond this point, increasing α leads RTKM and NEO-k-means to identify more outliers correctly and achieve much lower average M_e scores compared to KMOR. Unfortunately, the corresponding increase in false positive

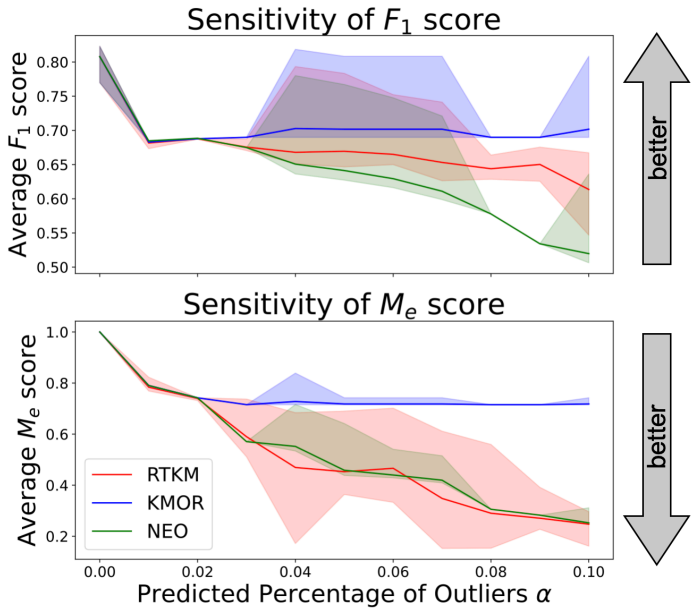


Figure 3.3: Sensitivity of RTKM, KMOR, and NEO-k-means to choice of α on the Shuttle dataset. All three methods perform similarly until $\alpha \geq 0.02$. After, KMOR and NEO-k-means identify significantly more outliers correctly, but the increase in false positives leads to lower average F_1 scores. (Parameters: $k = 14$, RTKM $s = 1$, KMOR $\gamma = 1$, NEO-k-means $\sigma = 0$)

outliers leads to comparatively lower average F_1 scores. In applications where there is a high cost associated with false positive outliers, KMOR may be preferential. Conversely, in applications were the goal is to identify as many outliers as possible, RTKM is superior.

3.4.2 Multi-membership Data without Outliers

Unlike KMOR, RTKM and NEO-k-means are able to classify points as belonging to multiple clusters [67]. The parameters in NEO-k-means determine how many point assignments are made and at most how many outliers are identified. In contrast, RTKM's parameters determine at least how many assignments are made and at least how many outliers are identified.

We evaluate the three methods on the "yeast", "scene", and "emotions" datasets from [46]. The "yeast" dataset contains 2417 instances with 103 numerical attributes. There are 14 classes, and the cardinality is 4.237. The "scene" dataset contains 2407 instances with 294 numeric attributes. There are 6 classes, and the cardinality is 1.074. Finally, "emotions" contains 593 instances with 72 numerical attributes. There are 6 classes, and the cardinality is 1.869. Both "yeast" and "scene" are used for testing MOC [19], fuzzy k-means [6], explicit sparsity constrained clustering (esp) [50], implicit sparsity constrained

Table 3.1: Average F_1 scores of various multi-membership clustering methods on the "yeast", "scene", and "emotions" datasets. NEO-k-means achieves the best F_1 score on "yeast" and "scene". However, RTKM achieves the second highest F_1 score on both datasets, demonstrating its competitiveness. On "emotions", RTKM achieves the highest F_1 score, followed closely by NEO-k-means. ("yeast" parameters: $k = 14, \alpha = 0$, RTKM $s = 4$, KMOR $\gamma = 9$; "scene" parameters: $k = 6, \alpha = 0$, RTKM $s = 1$, KMOR $\gamma = 9$; "emotions" parameters: $k = 6, \alpha = 0$, RTKM $s = 2$, KMOR $\gamma = 9$, NEO-k-means $\sigma = 1$)

	RTKM	KMOR	NEO-k-means	moc	fuzzy	esp	isp	okm	rokm
"yeast"	0.325	0.161	0.366	-	0.308	0.289	0.203	0.311	0.203
"scene"	0.598	0.598	0.626	0.467	0.431	0.572	0.586	0.571	0.593
"emotions"	0.399	0.311	0.373	-	-	-	-	-	-

clustering (isp) [50], OKM [32], and restricted OKM (rokm) [60] in [67]. We borrow performance reports on "yeast" and "scene" for all aforementioned algorithms from [67]. Results on "emotions" are only reported for RTKM, KMOR, and NEO-k-means. Each algorithm is run five times using the same cluster center initialization, and the result that leads to the best objective function value is chosen, as in [67].

Average F_1 scores for each algorithm are shown in Table 3.1. NEO-k-means achieves the highest F_1 score on "scene" and "yeast", but in both cases the F_1 score of RTKM is the second highest of all scores reported. On "emotions", RTKM achieves the highest F_1 score of the three methods. We note that on "scene", KMOR performs just as well as RTKM due to the cardinality of "scene" being so close to one. Contrastingly, on "yeast" and "emotions", KMOR is unable to produce a competitive result due to the higher cardinalities and the fact that the method is not designed to run on multi-membership data.

3.4.3 Multi-membership Data with Outliers

In order to compare RTKM against KMOR and NEO-k-means on multi-membership data containing outliers, we add noise in two different ways to "scene", "yeast", and "emotions" from Section 3.4.2. The first approach is to add Gaussian noise, while the second is to take the average of the data points and add to it some multiple of the average of their standard deviations.

We begin by adding 100 noise points to "scene", so that it contains $\sim 4\%$ outliers. Figures 3.4a and 3.4b demonstrate the performance of the three methods over various α values for the Gaussian and alternative noise models, respectively. On both datasets, RTKM and KMOR score similarly and outperform NEO-k-means in terms of average

F_1 scores. KMOR's competitive performance here is unsurprising, given that the original "scene" data has a cardinality close to 1. Figure 3.4a shows that all three methods perform almost identically in terms of M_e scores on "scene" + Gaussian noise, while Figure 3.4b shows that RTKM and KMOR are able to achieve slightly better outlier detection than NEO-k-means on "scene" + noise.

Next, we add 150 noise points to "yeast" so that the data contains $\sim$ %6 outliers. We again test the sensitivity of the results of all three methods to α for "yeast" with added Gaussian and alternative noise. As seen in Figures 3.5a and 3.5b, of the three methods tested, RTKM achieves the highest average F_1 scores over all values of α, with NEO-k-means following closely behind. KMOR scores poorly for classification accuracy, because of the higher cardinality of the data. In terms of outlier detection, all three methods score similarly when Gaussian noise is added, but on average, RTKM substantially outperforms the other methods with the addition of alternative noise.

Finally, we add 25 noise points to "emotions" so that the data contains $\sim 4\%$ outliers. Figures 3.6a and 3.6b show the sensitivity of each of the methods to α for the addition of Gaussian and alternative noise. On both datasets, RTKM achieves the highest average F_1 scores over all values of α, followed closely by NEO-k-means. Again with the addition of Gaussian noise, all three methods perform similarly in terms of outlier detection, with RTKM scoring slightly better. On "emotions" + noise, RTKM achieves by far the lowest average M_e scores.

On multi-membership data with outliers, RTKM achieves the highest average F_1 scores across all datasets tested. On the lower cardinality "scene"-based data, KMOR classifies points just as well, while on the higher cardinality "yeast"- and "emotions"-based data, KMOR is not competitive and NEO-k-means performs second-best. Unsurprisingly, all three methods perform outlier detection well when the added noise is normally distributed. On "scene" + and "yeast" + Gaussian noise, the average M_e scores are almost indistinguishable, while on "emotions" + Gaussian noise, RTKM achieves only marginally lower M_e scores. It is when alternative noise is added to the data that RTKM's superior ability to detect outliers is pronounced.

Table 3.2: Ranks of RTKM, KMOR, and NEO-k-means in terms of F_1 score.

	RTKM	KMOR	NEO-k-means
WBC	1.5	1.5	3.0
Shuttle	2.0	1.0	3.0
"yeast"	2.5	2.5	1.0
"scene"	2.0	3.0	1.0
"emotions"	1.0	3.0	2.0
"yeast" + Gaussian noise	1.0	3.0	2.0
"scene" + Gaussian noise	1.5	1.5	3.0
"emotions" + Gaussian noise	1.0	3.0	2.0
"yeast" + noise	1.0	3.0	2.0
"scene" + noise	1.5	1.5	3.0
"emotions" + noise	1.0	3.0	2.0
Average Rank	1.45	2.36	2.18

3.5 Nonparametric Statistical Validation of Clustering

We use the Friedman test method [1] to perform a non-parametric statistical validation of the results of RTKM, KMOR, and NEO-k-means. We calculate a representative F_1 and M_e score for each method on every dataset by averaging the values of F_1 and M_e for all runs over all values of α. Table 3.2 lists the ranks of these three methods on all datasets in terms of F_1 score, while Table 3.3 lists the ranks in terms of M_e score.

For F_1 score, the Q-value is below the critical value of the Friedman test with 11 datasets, meaning that the difference of RTKM's performance in terms of classification accuracy is not significant. On the other hand, the Q-value for M_e score is 6.25, which is the critical value of the Friedman test for an α level of 0.05 when the number of datasets is 8. Carrying out a post-hoc Nemenyi test [27] to quantitatively evaluate the differences between methods, we find that RTKM significantly outperforms NEO-k-means, but not KMOR in terms of outlier identification. Although the differences in the F_1 scores of the three methods and the M_e scores of RTKM and KMOR are not deemed significant, the overall performance of RTKM in terms of both F_1 and M_e scores is the best, as evidenced by the lowest average rank of RTKM in both Tables 3.2 and 3.3.

3.6 Conclusions and Future Work

We propose Robust Trimmed k-means (RTKM) as an algorithm for simultaneous point classification and outlier detection that can operate on both single- and multi-membership data. The parameters k and α control the number of clusters and expected percentage

Table 3.3: Ranks of RTKM, KMOR, and NEO-k-means in terms of M_e score.

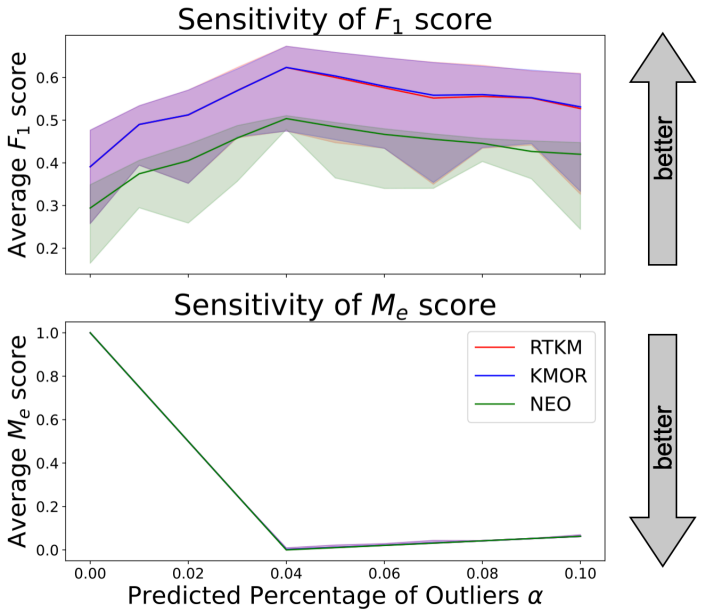
	RTKM	KMOR	NEO-k-means
WBC	1.5	1.5	3.0
Shuttle	1.0	3.0	2.0
"yeast" + Gaussian noise	2.0	2.0	2.0
"scene" + Gaussian noise	2.0	2.0	2.0
"emotions" + Gaussian noise	1.0	2.0	3.0
"yeast" + noise	1.0	2.0	3.0
"scene" + noise	1.5	1.5	3.0
"emotions" + noise	1.0	2.0	3.0
Average Rank	1.38	2.00	2.63

of outliers respectively. The parameter s controls at least how many clusters each point belongs to. Methods such as X-means [13] and G-means [36] have addressed the problem of estimating the number of clusters by running k-means repeatedly with different values of k until some criteria is satisfied. Future work could address how these methods could be altered to use our objective function to estimate k. At the moment, there is no principled approach to estimating the other two parameters α and s. We leave the investigation of such an approach for future work and demonstrate that RTKM remains competitive with existing methods over a range of α values.

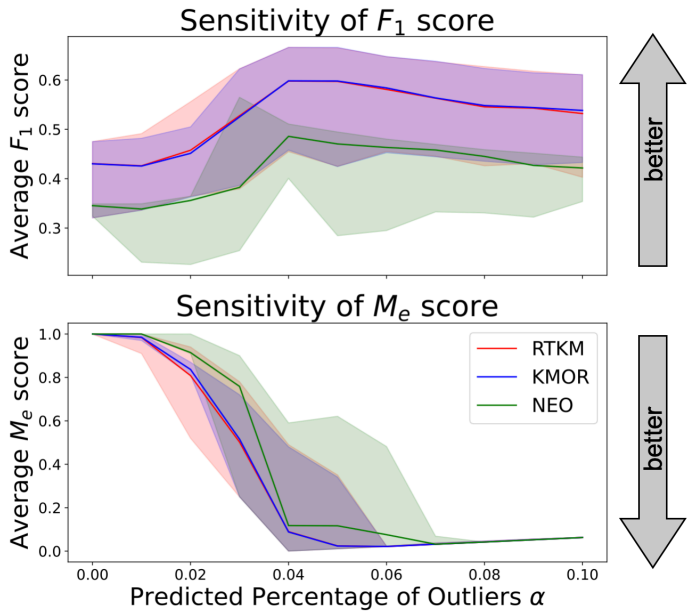
The innovations presented rely on a robust relaxed formulation for the weighted k-means algorithm that allows the classification weight matrix to exist on a continuum of values $[0, 1]$, rather than the binary set $\{0, 1\}$. This relaxation gives the user a way to track the extent of membership of a point to each cluster at every iteration and provides flexibility for multi-cluster membership. We apply the same methodology for outlier detection, thereby avoiding explicitly defining a measure of "outlierness". Relaxation-based formulations have proven to be effective in a number of recent applications [76, 79]. In the context of the current application, relaxation to a continuum of values, coupled with the PAM algorithm, provides a way to search the model space more effectively to discover clusters and outliers.

We test RTKM on three types of data: single-label data with outliers, multi-label data without outliers, and mutli-label data with outliers. While KMOR and NEO-k-means set the benchmark for the first two types of data, respectively, they do not perform well on both. RTKM remains competitive on both types of data, and outperforms both KMOR and NEO-k-means on multi-label data with outliers. On single-label data, we demonstrate that NEO-k-means is, by design, unable to perform when $k = 1$ and when $k > 1$,

RTKM and NEO-k-means achieve improved outlier detection over KMOR at the cost of lower average F_1 scores. We conclude that in applications where the cost of false positive outliers is high, KMOR may be preferred, but in applications where it is important to identify as many outliers as possible, RTKM may be favored. On multi-label data without outliers, RTKM remains competitive against NEO-k-means, achieving a higher F_1 score on two datasets than every other method NEO-k-means is compared against in [67] and even scoring higher in classification accuracy than NEO-k-means on a third dataset. Since KMOR does not have the functionality to make multi-membership assignments, it is unable to remain competitive when the cardinality of a dataset is greater than or equal to 1.5. On multi-label data containing outliers, we show that RTKM consistently scores highest in terms of average classification accuracy over all values of α. Further, when the noise added to the data is normally distributed, all three methods perform outlier detection almost identically well, but when the added noise follows an alternative model, RTKM clearly achieves the lowest average M_e scores. Finally, we use the Friedman test method to statistically validate the results of the clustering methods. We find that RTKM's outlier detection significantly outperforms that of NEO-k-means. Although the differences in the three methods' classification accuracies and the outlier detection of RTKM and KMOR are not deemed significant, the overall performance of RTKM is still superior.

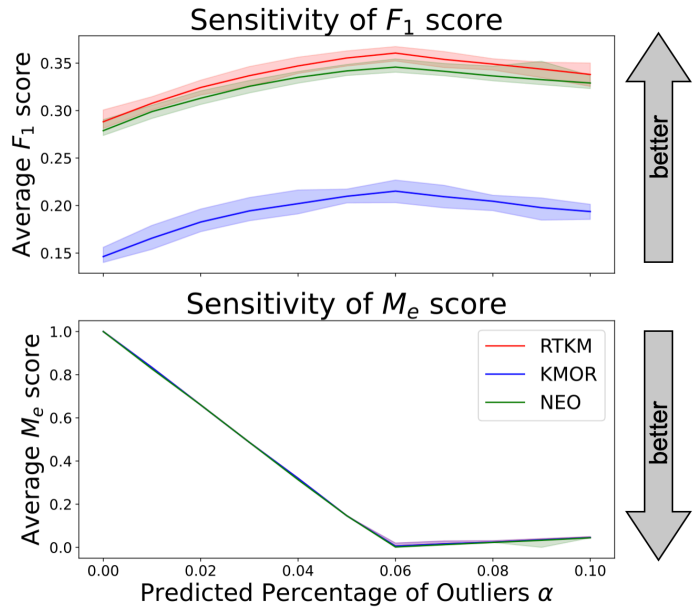


(a) "scene + Gaussian noise"

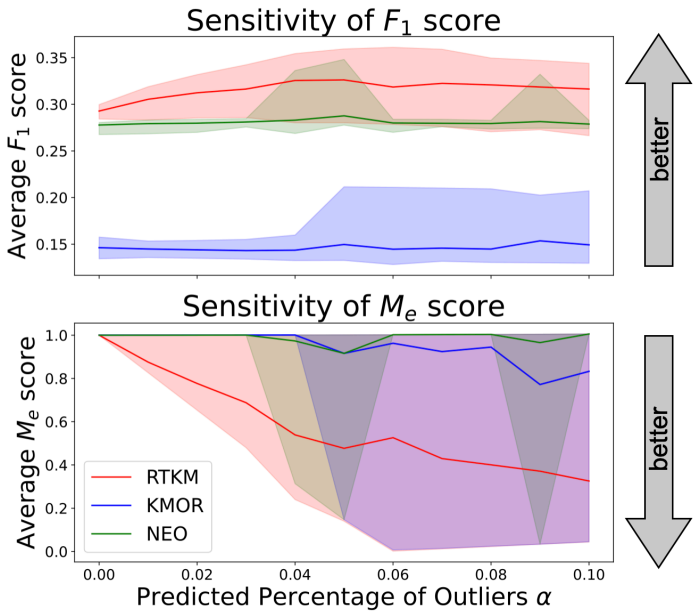


(b) "scene" + noise

Figure 3.4: Sensitivity of RTKM, KMOR, and NEO-k-means to choice of α on the (a) "scene" + Gaussian noise and (b) "scene" + noise datasets. In both (a) and (b), RTKM and KMOR perform similarly well and outperform NEO-k-means in terms of average F_1 scores. In (a), all three methods perform almost identically in terms of outlier detection, while in (b), RTKM and KMOR attain lower average M_e scores than NEO-k-means. (Parameters: $k = 6$, RTKM $s = 2$, KMOR $\gamma = 1$, NEO-k-means $\sigma = 1$)

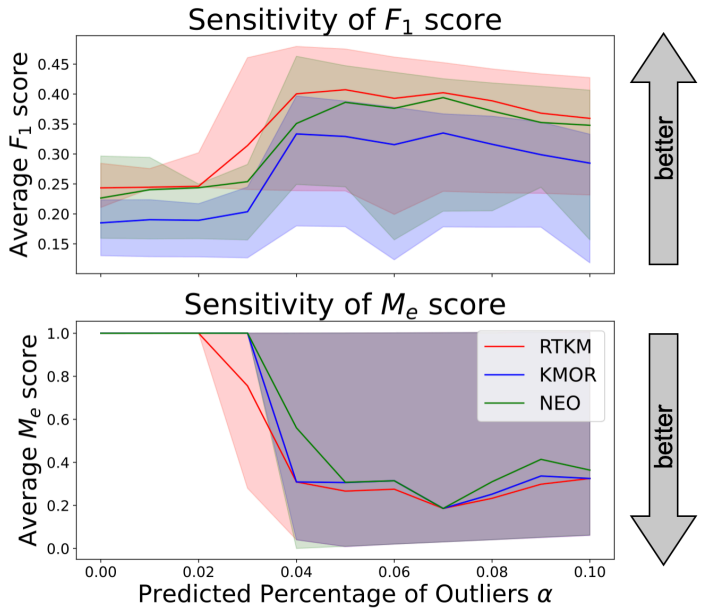


(a) "yeast" + Gaussian noise

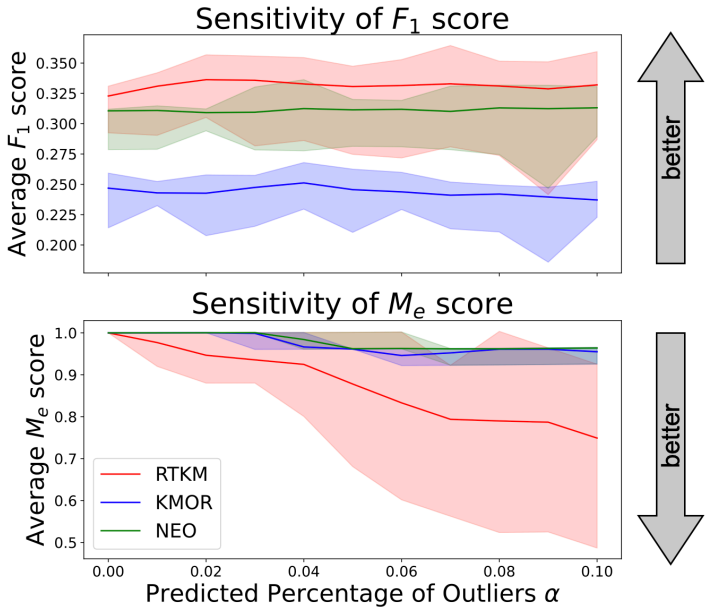


(b) "yeast" + noise

Figure 3.5: Sensitivity of RTKM, KMOR, and NEO-k-means to choice of α on the (a) "yeast" + Gaussian noise and (b) "yeast" + noise datasets. In both (a) and (b), RTKM performs best in terms of F_1 score, followed by NEO-k-means and KMOR. In (a), all three methods perform almost identically in terms of outlier detection, while in (b), RTKM obtains a noticeably lower M_e score. (Parameters: $k = 14$, RTKM $s = 4$, KMOR $\gamma = 1$, NEO-k-means $\sigma = 3$)



(a) "emotions" + Gaussian noise



(b) "emotions" + noise

Figure 3.6: Sensitivity of RTKM, KMOR, and NEO-k-means to choice of α on the (a) "emotions" + Gaussian noise and (b) "emotions" + noise datasets. In both (a) and (b), RTKM performs best in terms of average F_1 scores. In (a), all three methods perform similarly, though RTKM maintains a lower average M_e scores over all values of α, while in (b) RTKM obtains noticeably lower M_e scores on average than either other method. (Parameters: $k = 6$, RTKM $s = 2$, KMOR $\gamma = 1$, NEO-k-means $\sigma = 1$)

Chapter 4

Spatiotemporal k-means

4.1 Introduction

The widespread use of sensor and data acquisition technologies, including IOT, GPS, RFID, LIDAR, satellite, and cellular networks allows for, among other applications, the continuous monitoring of the positions of moving objects of interest. These technologies create rich spatiotemporal data that is found across many scientific and real-world domains including ecologists' studies of collective animal behavior [23], the surveillance of large groups of people for suspicious activity [37], and traffic management [33]. Often, the data collected is large and unlabeled, motivating the development of unsupervised learning methods that can efficiently extract information about object behavior with no human supervision. In this chapter, we propose a method of Spatiotemporal k-means (STKM) clustering that is able to analyze the multi-scale relationships within spatiotemporal data.

Spatial clustering refers to the analysis of static data with features that describe spatial location, such as latitude and longitude, while spatiotemporal clustering is an extension of spatial clustering in which time is added as a data feature, and algorithms have to consider both the spatial and the temporal neighbors of objects in order to extract useful knowledge [30]. There are a handful of spatiotemporal clustering classes, some of which track events or object trajectories, but our focus is on moving clusters, where clusters have a static identity, but their location and content can change over time. The moving cluster problem is especially useful in applications where it is essential to know whether individuals form loose and temporary associations or stable, long-term ones. Applications such as surveillance, transportation, environmental and seismology studies, and mobile

data analysis can be considered within this mathematical framework as surveyed by Ansari et al [78].

The mathematical formulation for the moving clusters problem is significantly more challenging than for stationary, unsupervised clustering. Most approaches first cluster in space and then aggregate the results over time, as opposed to minimizing a unified objective function. Recent findings show that this post-processing approach can lead to erroneous results [63]. Further, the most popular approaches are built upon density-based clustering methods, which are extremely sensitive to hyper-parameter tuning and do not explicitly track cluster centers [81]. Finally, while some existing methods operate well on large data, tracking objects over thousands of time steps or more, we show that they exhibit poor performance in the low-data domain, when dynamics are being inferred from either a small number of individuals or over very short windows of time.

We propose a two phase spatiotemporal, unsupervised clustering method, which we call *spatiotemporal k-means* (STKM), for the moving cluster problem that addresses the shortcomings mentioned above. Phase 1 identifies the loose, temporary associations between data points by outputting an assignment for each point at every time step, with the flexibility for points to change clusters between time steps. The clustering objective function provides a unified formulation over space and time and less hyper-parameter tuning compared to existing methods. It also provides the functionality to directly track cluster paths without any post-processing. In this way, STKM is able to identify long-term point behavior, even in a dynamic environment. Phase 2 can be optionally applied to the cluster assignment histories from Phase 1 to output stable, long-term associations. In fact, Phase 2 can be applied to any method that outputs an assignment for each point at every time step. The combination of Phase 1 and Phase 2 allows us to analyze the multi-scale relationships within spatiotemporal data. STKM outperforms existing methods, with significant performance improvements demonstrated for common evaluation metrics, on the moving cluster problem.

4.2 Related Work

Spatiotemporal data generally record an object state, an event, or a position in space, over a period of time. Ansari et. al [78] provide a taxonomy for spatiotemporal clustering,

dividing approaches into six classes: event clustering, geo-referenced data item clustering, geo-referenced time-series clustering, trajectory clustering, semantic-based trajectory data-mining, and moving clusters. The first two classes output static clusters and consider object similarity with respect to space, time, and possibly other non-spatial attributes. Event clustering focuses specifically on the discovery of events, while geo-referenced data item clustering focuses on the discovery of groups of objects. Some of the most prominent spatiotemporal clustering methods, such as ST-DBSCAN [30] (spatiotemporal density based spatial clustering of application with noise) and ST-OPTICS [68] (spatiotemporal ordering points to identify the clustering structure) belong to the second classification. Both ST-DBSCAN and ST-OPTICS address temporal similarity by retaining only an object's temporal neighbors before assessing the similarity of those neighbors' spatial and non-spatial attributes. Some shortcomings of these methods are that they require four and six input parameters, respectively, heavily influencing the quality of clusters and that they do not provide meaningful cluster centers for analysis. Thus the hyper-parameter tuning of the ten aforementioned parameters becomes critically important for achieving reasonable performance.

The third class of methods highlighted, geo-referenced time series clustering, also produces static clusters, but considers exclusively objects' spatial closeness and time series similarity [78]. An example is a spatiotemporal extension of *fuzzy c-means* (FCM) that incorporates a distance function that takes into account both spatial and temporal distance [48]. This method is attractive due to its clear identification of cluster centers. Another method is *correlation based clustering of big spatiotemporal datasets* (CorClustST), which is based on empirical spatial correlations over time, making its results easily interpretable for scenarios with multiple underlying variables or varying time frames. However, CorClustST is not meant to find an optimal clustering solution, but rather to provide an efficient descriptive tool for spatiotemporal data, making it best used as a data reduction technique before applying another spatiotemporal clustering method [80]. The next two classes of spatiotemporal clustering focus on trajectory clustering, which tries to find objects with similar movement behavior. Spatiotemporal trajectory similarity is most often defined by some combination of geometry, direction, velocity, and co-location in space and time [78].

The algorithms in this paper are concerned with the final classification scheme, moving

clusters. A moving object is a type of spatiotemporal data that changes its spatial location with respect to time and has a unique identifier to trace its movement over time [54]. A moving object is defined by a set of sequences $\langle id, \mathbf{x}, t \rangle$, where the variable id is the unique identifier for each point, t is time, and $\mathbf{x}$ is a vector whose components contain the spatial attributes, i.e. the x and y coordinates [78]. Moving clusters have identities (separate from id above) that do not change over time, although their positions and content may change. The prototypical example is a herd of animals, where individual animals can enter or leave the herd at any given time.

Most approaches to the moving cluster problem first cluster in space and then aggregate the results over time. Kalnis et. al proposed running DBSCAN at every time step and defined a moving cluster criteria to associate clusters in successive time steps [23]. This approach was extended in [33] to the discovery of convoys consisting of at least n points that exist near one another for a minimum q consecutive time steps. Other work identified flocks of objects that stay together for a given window of time [37]. The commonality between these approaches was a requirement for moving clusters to exist in q consecutive time steps. In practice, points can split apart and come back together. Thus, the constraint on successive time steps was relaxed in [40] by proposing the concept of a swarm where a minimum n objects travel together for at least q out of t time steps.

In contrast to methods that consider space and time separately, Chen et. al [63] proposed an extension of DBSCAN that incorporates a novel spatiotemporal distance function. Essentially, points' distances are their spatial distances from one another if they are temporal neighbors and zero otherwise. Their four step clustering process performs even in the presence of noise and missing data. However, like ST-DBSCAN, this method requires extensive hyper-parameter tuning.

Though substantial work has been done to develop various spatiotemporal clustering techniques, the performance of these methods is rarely compared against one another and implementations are not open source. Recognizing that there was no unified and commonly used experimental dataset and protocol, Cakmak et. al proposed a benchmark for detecting moving clusters in collective animal behavior [82]. They generate realistic synthetic data with ground truth, and present state-of-the-art baseline methods. Their implemented algorithms extend spatial clustering methods by first assessing whether a data point is density reachable from another data point with respect to both space and

time and then employing the splitting and merging process for spatiotemporal data by Peca et. al [53].

4.3 Spatiotemporal k-means

Drawing inspiration from approaches that define unique spatiotemporal distance metrics [48, 63], we propose a clustering objective that provides a unified formulation over space and time and predicts cluster membership for each point at every time step. We build upon the k-means algorithm, as opposed to density-based methods, so that cluster centers are explicitly tracked and there are fewer parameters to tune. In a single pass of Phase 1, without any post-processing, point membership and dynamic cluster center paths are output. We provide an optional secondary phase to our method that can extract stable, long-term associations between data points. We compare the results of our method against the state-of-the-art baseline methods on the benchmark data provided by Cakmak et. al [82].

4.3.1 Phase 1: Loose, Temporary Associations

The first phase of our method captures the loose, temporary associations between data points, which tell us at every time step which points are interacting with one another. The outputs of Phase 1 are the location histories of the dynamic cluster centers and a cluster assignment for each point at every iteration, with points having the flexibility to change clusters between time steps. In order to find these short-term associations, we propose a temporal extension of the k-means objective function. We focus on k-means, because of k-means simplicity, speed, and scalablity. Also, unlike density-based methods, k-means explicitly identifies cluster centers, giving us the ability to directly track the movement of our k clusters. Our proposed objective is shown in 4.1,

$$\min_{\mathbf{C},\mathbf{W}} \sum_{i=1}^{N}\sum_{j=1}^{k}\sum_{t=1}^{T} w_{t,j,i}||\mathbf{x}_{t,i} - \mathbf{c}_{t,j}||_2^2 + \frac{\lambda}{n}||\mathbf{c}_{t,j} - \mathbf{c}_{t+1,j}||_2^2, \tag{4.1}$$

$$\text{where } \mathbf{W}_{t,:,i} \in \Delta_1.$$

The matrix $\mathbf{X} = [\mathbf{x}_{t,1}, \cdots \mathbf{x}_{t,N}] \in \mathbb{R}^{T\times m\times N}$ contains the data points and the matrix

$\mathbf{C} = [\mathbf{c}_{t,1}, \cdots, \mathbf{c}_{t,k}] \in \mathbb{R}^{T \times m \times k}$ is made up of the cluster centers. The matrix $\mathbf{W} \in \mathbb{R}^{T \times k \times N}$ contains auxiliary weights $w_{t,j,i}$ that map the point-to-cluster relationship. At time t, column i of $\mathbf{W}_{t,:,:}$ assigns point $\mathbf{x}_{t,i}$ to a cluster whose center is $\mathbf{c}_{t,j}$. Instead of restricting the time frames of the auxiliary weight matrix $\mathbf{W}$ to the discrete set $\{0, 1\}$, we allow $\mathbf{W}_{t,:,i} \in \Delta_1$ so that $\sum_{j=1}^{k} w_{t,j,i} = 1$ for all i and each $w_{t,j,i}$ is allowed to vary over the closed interval $[0, 1]$. This relaxation gives the user a way to track the extent of membership of a point to each cluster at every iteration.

The second term in 4.1 is used to associate cluster centers between time frames, so that clusters maintain their identity over time. The term penalizes cluster centers moving apart between time steps, indirectly discouraging points from switching clusters. The parameter $\lambda \in [0, 1]$ controls the extent of the penalty. This formulation allows us to directly track clusters during the clustering process, rather than by associating clusters between time steps through post-processing as in [23, 33, 37, 53, 82].

Problem 4.1 can be solved using alternating minimization. The centers are updated using the Gauss-Seidel step in 4.2 and derived below,

$$\begin{aligned} \sum_{i=1}^{n} -2w_{t,i,j}(\mathbf{x}_{t,i} - \mathbf{c}_{t,j}) + 2\frac{\lambda}{n}(\mathbf{c}_{t,j} - \mathbf{c}_{t+1,j}) = 0, \\ \mathbf{c}_{t,j}\left(\sum_{i=1}^{n} w_{t,i,j} + \frac{\lambda}{n}\right) = \sum_{i=1}^{n}\left(w_{t,i,j}\mathbf{x}_{t,i} + \frac{\lambda}{n}\mathbf{c}_{t+1,j}\right), \\ \mathbf{c}_{t,j}^{k+1} = \frac{\sum_{i=1}^{n} w_{t,j,i}\mathbf{x}_{t,i} + \lambda\mathbf{c}_{t+1,j}}{\sum_{i=1}^{n} w_{t,j,i} + \lambda}. \end{aligned} \tag{4.2}$$

To better control how quickly the weights are updated, we use the Proximal Alternating Minimization (PAM) approach as shown in 4.3,

$$w_{t,:,i}^{q+1} = proj_{\Delta_1}\left(w_{t,:,i} - \frac{1}{d_q}||\mathbf{x}_{t,i} - \mathbf{c}_{t,j}||_2^2\right). \tag{4.3}$$

.

This update can be thought of as a proximal regularization of the Guass-Seidel scheme [39]. PAM is guaranteed to converge as long as $d_q > 1.0$. In practice, we set $d_q = 1.1$. The procedure for Phase 1 of STKM is summarized in Algorithm 5. Phase 1 has a runtime complexity of $\mathcal{O}(Ntmki)$ where N is the number of points, t is the total number of time steps, m is the dimension of the points, k is the number of clusters, and i is the number

Algorithm 5 Phase 1 Spatiotemporal k-means (STKM)

procedure STKM($\mathbf{X}, k, \lambda$) ▷ Input $\mathbf{X}$, k, λ
  Initialize $\mathbf{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_k]$ and $d_q = 1.1$.
  while not converged **do**
    $\mathbf{c}_{t,j}^{q+1} = \frac{\sum_{i=1}^{n} w_{t,j,i}\mathbf{x}_{t,i} + \lambda \mathbf{c}_{t+1,j}}{\sum_{i=1}^{n} w_{t,j,i} + \lambda}$
    $w_{t,:,i}^{q+1} = proj_{\Delta_1}\left(w_{t,:,i} - \frac{1}{d_q}||\mathbf{x}_{t,i} - \mathbf{c}_{t,j}||_2^2\right)$
  return $\mathbf{C}, \mathbf{W}$

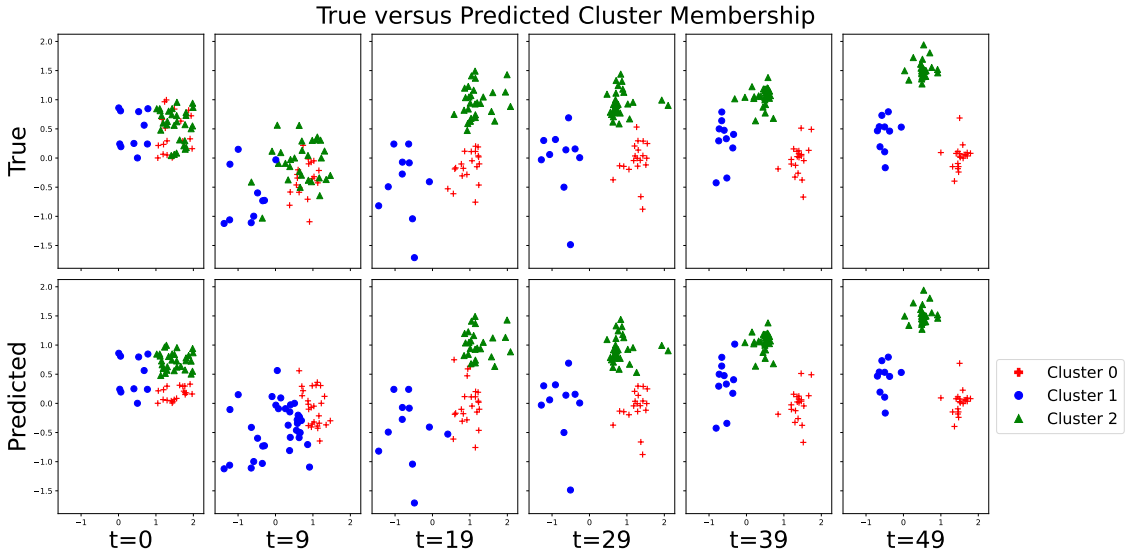


Figure 4.1: True versus predicted cluster membership of moving clusters. The ground truth contains static clusters, while the predicted clusters are dynamic, allowing points to switch clusters between time steps.

of iterations required for convergence.

As previously mentioned, one of the benefits of our method is that point membership and clusters are tracked throughout the clustering process with the matrices $\mathbf{W}$ and $\mathbf{C}$, respectively. Figure 4.1 displays the true versus predicted point membership at select time steps on synthetic data containing three long-term clusters. Whereas the ground truth clusters have static membership, STKM allows points to switch clusters throughout the observation period. Our method finds the loose, temporary associations that occur between data points at every time step. Though these interactions cannot be directly validated, they still provide valuable insight into object behavior. For instance, note that at $t = 9$ STKM only identifies two short-term clusters, suggesting that clusters merge before $t = 9$ and split sometime thereafter.

Further, the matrix $\mathbf{C}$ is used to visualize the paths of the identified dynamic clusters. This ability to directly track clusters is unavailable with any existing spatiotemporal

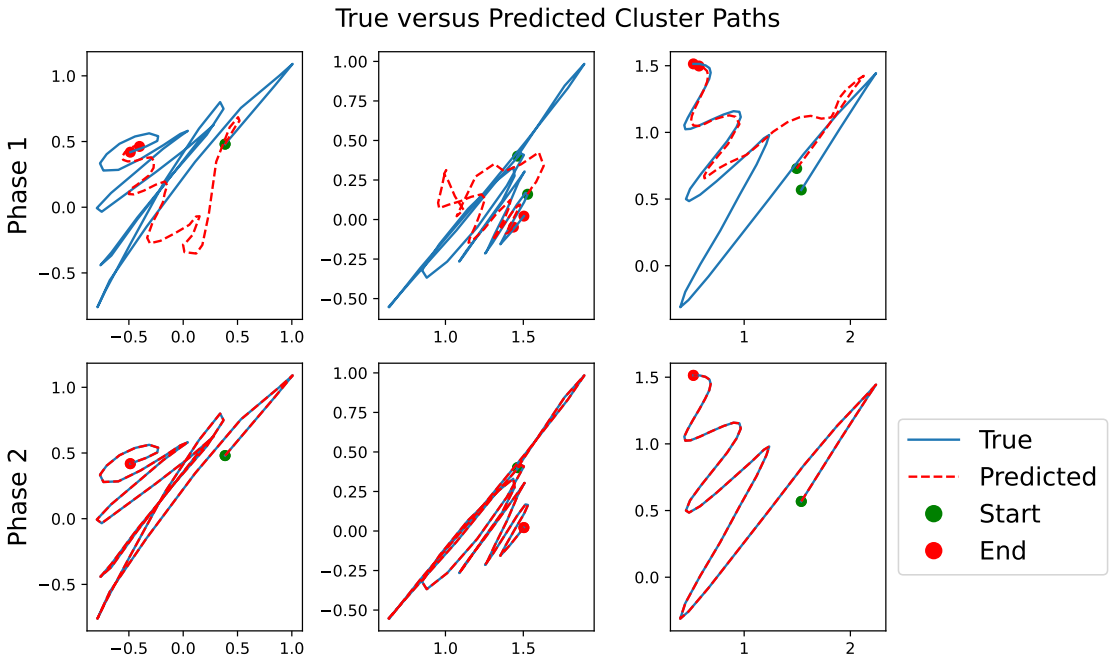


Figure 4.2: True static versus predicted cluster paths for Phase 1 and Phase 2 of STKM. In Phase 1, STKM identifies general trends of cluster movement, even when allowing points to switch clusters over time. In Phase 2, STKM correctly identifies the true static cluster paths.

clustering method without extensive post-processing. The top row of Figure 4.2 shows true versus predicted cluster paths for the same synthetic data as above. We note that the cluster paths are not identified perfectly because the method does not match the data generating mechanism, i.e. we do dynamic prediction on static clusters. Even so, STKM is able to pick up the general trends of cluster movement. This result suggests that even in a dynamic environment, where points are able to change cluster membership, we do not completely lose information about long-term, static cluster behavior.

One of the drawbacks of building upon k-means is that k-means is well known to break down in the presence of outliers and noise. In a dataset, such as the one in the top row of Figure 4.3, we expect outliers to skew cluster centers, and thereby cluster membership significantly. This behavior is exactly what is observed in row two of Figure 4.3 when applying STKM to a dataset with outliers. The outliers are incorrectly assigned as one cluster and the two ground truth clusters are assigned to one large cluster for the majority of time steps. A possible solution to this problem is to replace the Euclidean distance between points and their cluster centers in Equation 4.1 with a more robust distance function, such as the one in Equation 4.4,

$$\sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{t=1}^{T}\frac{v+1}{2}w_{t,i,j}\log\left(v+||\mathbf{x}_{t,i}-\mathbf{c}_{t,j}||_2^2\right)+\frac{\lambda}{n}||\mathbf{c}_t-\mathbf{c}_{t+1}||_2^2. \tag{4.4}$$

Let $v \geq 1$ and use PAM for both the center and the weight updates. The center updates are derived below, while the weight updates remain the same.

$$\mathbf{c}_{t,j}^{q+1} = prox_{d_q}^{f}\left(\mathbf{c}_{t,j}-\frac{v+1}{d_q}\sum_i\frac{-w_{t,i,j}(\mathbf{x}_{t,i}-\mathbf{c}_{t,j})}{v+||\mathbf{x}_{t,i}-\mathbf{c}_{t,j}||_2^2}\right), \tag{4.5}$$

$$\begin{aligned}\mathbf{c}_{t,j}^{q+1} = \arg\min_{\mathbf{a}}\left\{\frac{\lambda}{n}||\mathbf{a}-\mathbf{c}_{t+1,j}||_2^2+\frac{d_q}{2}\left|\left|\mathbf{a}-\mathbf{c}_{t,j}+\frac{v+1}{d_q}\sum_i\frac{-w_{t,i,j}(\mathbf{x}_{t,i}-\mathbf{c}_{t,j})}{v+||\mathbf{x}_{t,i}-\mathbf{c}_{t,j}||^2}\right|\right|_2^2\right\},\\ 2\frac{\lambda}{n}(\mathbf{a}-\mathbf{c}_{t+1,j})+d_q\mathbf{a}-d_q\mathbf{c}_{t,j}+(v+1)\sum_{i=1}^{n}\frac{-w_{t,i,j}(\mathbf{x}_{t,i}-\mathbf{c}_{t,j})}{v+||\mathbf{x}_{t,i}-\mathbf{c}_{t,j}||_2^2}=0,\\ \mathbf{a}=\left(2\frac{\lambda}{n}\mathbf{c}_{t+1,j}+d_q\mathbf{c}_{t,j}-(v+1)\sum_{i=1}^{n}\frac{-w_{t,i,j}(\mathbf{x}_{t,i}-\mathbf{c}_{t,j})}{v+||\mathbf{x}_{t,i}-\mathbf{c}_{t,j}||_2^2}\right)/\left(2\frac{\lambda}{n}+d_q\right).\end{aligned} \tag{4.6}$$

$d_q = (v+1)n$ is the Lipschitz constant of the gradient of $H(\mathbf{W},\mathbf{c})$ wrt $\mathbf{c}$, as shown below,

$$\begin{aligned}&\left|\left|(v+1)\sum_{i=1}^{n}\frac{-w_{t,i,j}(\mathbf{x}_{t,i}-\mathbf{c}_{t,j}^1)}{v+||\mathbf{x}_{t,i}-\mathbf{c}_{t,j}^1||_2^2}-(v+1)\sum_{i=1}^{n}\frac{-w_{t,i,j}(\mathbf{x}_{t,i}-\mathbf{c}_{t,j}^2)}{v+||\mathbf{x}_{t,i}-\mathbf{c}_{t,j}^2||_2^2}\right|\right|_2 \leq\\ \left|\left|(v+1)\sum_{i=1}^{n}-w_{t,i,j}(\mathbf{x}_{t,i}-\mathbf{c}_{t,j}^1)+w_{t,i,j}(\mathbf{x}_{t,i}-\mathbf{c}_{t,j}^2)\right|\right|_2 &= (v+1)\left|\left|\sum_{i=1}^{n}w_{t,i,j}(\mathbf{c}_{t,j}^1-\mathbf{c}_{t,j}^2)\right|\right|_2\\ &\leq (v+1)n||\mathbf{c}_{t,j}^1-\mathbf{c}_{t,j}^2||_2.\end{aligned} \tag{4.7}$$

Using the robust formulation of STKM allows for the two ground truth clusters to retain their individual identities over all time steps, as seen in row three of Figure 4.3. Further post-processing to trim those points furthest from their cluster centers could potentially identify outliers as well. Although a robust distance function prevents outliers and noise from corrupting the clustering results, it also adds to run time and scales poorly with problem size. Therefore, for the remainder of the chapter, we focus on the standard version of STKM and leave the investigation of robust versions to future work.

Another concern of a k-means based approach is handling missing data. The formulation of the objective in Equation 4.1 requires all points to exist at the same time steps

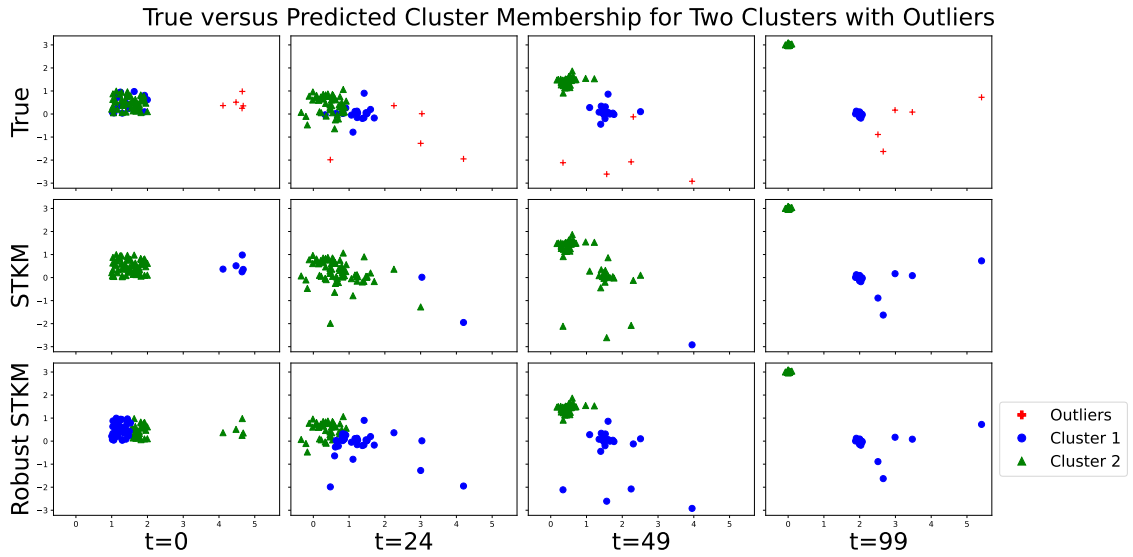


Figure 4.3: True versus predicted cluster membership for two clusters plus outliers. STKM is unable to produce a good result, while Robust STKM retains the identities of both clusters, even in the presence of noise.

across the entire time span of observation. In order to ensure this criteria is satisfied, data can be divided into time intervals, and data missing spatial information in an interval can be augmented using interpolation, while data with multiple spatial coordinates in an interval can be reduced through averaging.

4.3.2 Phase 2: Stable, Long-term Associations

Phase 2 of STKM aims to identify the stable-long term associations between data points. The output of Phase 2 is a single assignment of static long-term clusters containing points that have the most similar spatiotemporal characteristics. Phase 2 outputs the same information as spatiotemporal clustering methods in the event clustering, geo-referenced data item clustering, and geo-referenced time-series clustering classifications. However, because Phase 2 builds upon Phase 1, taking into account short-term information in making decisions about long-term behavior, the long-term clusters predicted by Phase 2 are more accurate than a method from the aforementioned classifications.

Intuitively, points that have similar cluster assignment histories will travel together in the long run. We can apply Phase 2 to any method that outputs an assignment for each point at every iteration. In order to apply Phase 2 to the output from Phase 1, we first need to extract the cluster assignment histories from the weight matrix $\mathbf{W}$. We define the cluster assignment histories as in 4.8, so that $\mathbf{a}_r \in \mathbb{R}^T$ is a vector where each entry

contains the cluster assignment of point r at time t,

$$\mathbf{a}_r = \arg\max_j \mathbf{W}_{t,:,i}. \tag{4.8}$$

Then the cluster assignment similarity measure between two cluster assignment history vectors is simply the magnitude of the intersection of the two vectors, taking the order of assignments into account, over the total number of time steps, as shown in 4.9,

$$sim(\mathbf{a}_r, \mathbf{a}_s) = \frac{|\mathbf{a}_r \cap \mathbf{a}_s|}{T}. \tag{4.9}$$

.

Using this similarity measure, we define a cluster assignment similarity matrix $\mathbf{A}$ where $A_{r,s}$ contains the similarity between point r and point s. We then run agglomerative clustering on the similarity matrix to output k long-term clusters.

In row two of Figure 4.2, we see the result of predicting the static, long-term cluster paths of the moving objects from the previous section. Phase 2 identifies the true, static cluster paths perfectly. By combining the results from Phase 1 and Phase 2, we can form a visualization like the one in Figure 4.4, where the cluster assignment histories of three points from each long-term cluster are displayed. Now, not only do we know which points behave most similarly in the long-run, we also know the interactions of each individual point at every time step. We are able to gain insights such as, the three points tracked in long-term Clusters 0 and 1 spend significant time in the other two clusters, whereas the three points tracked in Cluster 2 spend the vast majority of their time in Cluster 2. This analysis provides valuable information about the multi-scale behavior of the moving objects.

4.4 Experiments

In order to evaluate the performance of our proposed method, we perform experiments on the spatiotemporal benchmark dataset proposed by Cakmak et. al [82]. Recognizing that there was no unified and commonly used experimental dataset and protocol in the field of spatiotemporal clustering, Cakmak et. al proposed a benchmark for detecting moving clusters in collective animal behavior. Their benchmark is based on three collective animal

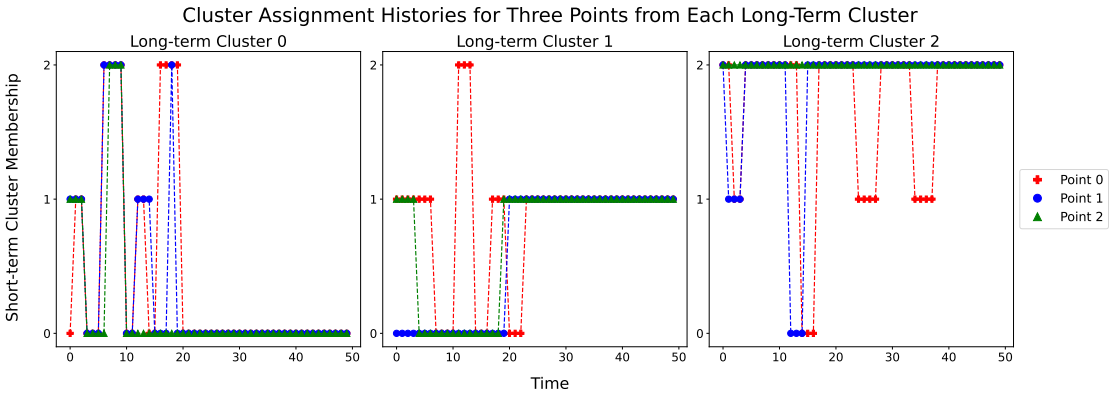


Figure 4.4: Cluster assignment histories for three points from each long term cluster.

behavior models and contains 3,600 spatiotemporal datasets of sizes ranging from 600 up to 520,000, where size is determined by multiplying together number of time steps and number of objects. The datasets track static clusters, where points do not change cluster membership over time [82]. During our evaluation, we focus on the benchmark datasets that have size between 800 and 35,000, of which there are 1,034. We take this approach, because we are particularly interested in the performance of our method in the low-data domain, where either we have very few objects or very few time steps from which to infer behavior.

Cakmak et. al measure clustering quality with the adjusted mutual information (AMI) score and report execution time for a handful of baseline methods. The implemented baseline methods all allow points to switch clusters between time steps, and the reported AMI compares the full cluster assignment histories against the ground truth, which does not allow for point switching. While this comparison provides a way to evaluate the short-term associations, because of the mismatch between the method and the data generating mechanism, we believe that it is more informative to compare the stable, long-term associations derived from the full assignment histories against the ground truth. Therefore, we report both what refer to as the total AMI for the full cluster assignment histories and the long-term AMI for the long-term associations. We calculate total AMI as in Cakmak et. al by comparing the cluster assignment histories to the ground truth static cluster assignments. We obtain the long-term AMI by applying Phase 2 of our method to both our output from Phase 1 and the cluster assignment histories of the baseline methods. As in [82], we divide our data into groups based on size (e.g. 800-3000 data points, 3000-6000 points, etc.). We report results as the median and average of total and long-term AMI for

each range of dataset sizes.

We note that in [82], during the cluster merging process, points that cannot be assigned to a cluster are given the same label. This labeling results in an erroneous association of unassigned points as a single cluster during the calculation of AMI. In order to avoid interpreting unassigned points as a single cluster, we ensure that they are all given unique labels when we calculate total and long-term AMI.

All of the baseline methods have at least four parameters that need to be defined: frame size, frame overlap, ϵ_1, and ϵ_2, which correspond to the number of time steps that belong to a single frame, the number of time steps that frames overlap when associating clusters between frames, and the spatial and temporal distances that define whether a point is density reachable from the current one. In addition, all of the methods except for ST-DBSCAN take as input the true number of clusters k. In their experiments, Cakmak et. al arbitrarily fix frame size to be 100 and frame overlap to be 10. All of the methods use the default value $\epsilon_1 = 0.50$, except for ST-DBSCAN, which searches for $\epsilon_1 \in [0.01, 0.05]$. Grid search is used in [82] to find the optimal remaining parameters that achieve the highest accuracy measure against the ground truth. In a true unsupervised setting, ground truth labels are not available, and we cannot tune parameters to maximize our accuracy. We argue that the performance report of the baseline methods in Cakmak et. al is therefore unrealistic and avoid parameter tuning in our experiments.

In contrast, Phase 1 of our method requires only two parameters: λ, which controls the extent of the penalty that indirectly discourages points from switching clusters, and k, the true number of clusters. Since λ is confined to the range $[0, 1]$, the meaning of its value is intuitive. The same is not true of ϵ_2 from the baseline methods. In order to create an intuitive interpretation of ϵ_2, we define $\epsilon_2 = \alpha t$, where t is the total number of time steps in the data and $\alpha \in [0, 1]$ is some given proportion. This formulation gives us a principled approach to choosing the temporal distance a point is density reachable from the current one, as opposed to arbitrarily choosing a value for each individual data set.

Because we know that the ground truth clusters do not allow points to switch clusters, we set both λ and α fairly high. We run all of the methods on each set of data with $\lambda, \alpha = [0.60, 0.80, 1.00]$. For the baseline methods, we fix the remaining parameters as follows: frame size = 100, frame overlap = 10, $\epsilon_1 = 0.50$ for all of the methods, except for ST-DBSCAN where $\epsilon = 0.05$, and k is set to the true number of clusters. Any other

parameters in the baseline methods are set to their default values. Since we run each method three times using different parameters on each dataset, we obtain metrics for 3,102 runs of each method. We then report the aggregates of the total and long-term AMI for each method on every range of dataset sizes in Figures 4.5-4.7.

4.5 Results

Figure 4.5 presents the results for total AMI, which compares full, variable cluster assignment histories against static ground truth clusters, for various methods on differently sized datasets. The trendlines connecting the medians and averages of AMI on each dataset size in 4.5a are shown in orange and blue, respectively. We note that over all dataset sizes, the boxpolots for all methods span the entire range from 0 to 1, emphasizing the difficulty of the moving object clustering task that comes from evaluating a method that does not match the data generating mechanism. Figures 4.5b and 4.5c provide a closer look at average and median total AMI score trendlines. STKM achieves the highest average total AMI on datasets of size smaller than $20,000$, but maintains the highest median total AMI over all dataset sizes, showing that although STKM achieves higher total AMI scores more often, those scores are on average smaller in magnitude than those achieved by other methods. The boxplots in Figure 4.5a further emphasize this result. The $25th$ percentiles of the STKM boxplots are all higher than other methods, but the $75th$ percentiles are substantially lower on larger datasets. The observed inverse relationship between full AMI and dataset size is expected, as points have more opportunities to change cluster membership over longer spans of time or in datasets with greater spatial density.

Recall that long-term AMI, compares predicted static against ground truth static clusters. We intuitively expect objects that spend the most time transiently clustered together to travel in the same long-term clusters. We therefore expect higher total AMI, which scores the accuracy of our transient associations, to correlate with higher long-term AMI, which scores the accuracy of our static associations. However, Figure 4.6, which show the results of long-term AMI for various methods on differently sized data sets, presents a different story. STKM, ST-Agglomerative, ST-KMeans, and ST-BIRCH score almost identically in terms of their median trendlines, but STKM maintains the highest average long-term AMI over all dataset sizes. Further, the boxplots for STKM in Figure 4.6a have

	STKM	ST-Agglomerative	ST-DBSCAN	ST-KMeans	ST-BIRCH	ST-HDBSCAN
Average Total AMI	**.42**	.40	.33	.38	.38	.30
Average Long-term AMI	**.90**	.73	.37	.77	.76	.44

Table 4.1: Average Total and Long-term AMIs for all methods over all dataset sizes. STKM achieves the highest scores for both accuracy metrics.

the tightest interquartile ranges, demonstrating that STKM has the lowest variability in terms of long-term AMI scores. This result implies that the short-term relationships that STKM detects are more informative than existing methods in identifying long-term point relationships. Figures 4.6b and 4.6c also include the results of baseline ST-DBSCAN, which is a method in the geo-referenced data item clustering classification that produces static cluster assignments. All of the other methods, which use Phase 2 of STKM to generate static cluster assignments, outperform baseline ST-DBSCAN, suggesting that a two phase approach that uses short-term behavior to inform long-term relationships, is more reliable.

When STKM is compared directly against the benchmark methods on a dataset by dataset basis, it achieves the best performance in terms of both total and long-term AMI. STKM scores the highest total AMI on 55% of the data and the highest long-term AMI on 70% of the data. It achieves both the highest total and long-term AMI on 49% of the data. Table 4.1 shows the average of all total AMI and long-term AMI scores for each of the tested methods over all 3,102 datasets. STKM moderately outperforms the other methods in terms of average total AMI and significantly outperforms the other methods in terms of average long-term AMI, highlighting its superior performance. Although STKM demonstrably outputs more informative moving cluster labels and more accurate long-term cluster labels, the trade-off is it's runtime. STKM runs slowest of all methods tested, as seen in Figure 4.7.

4.6 Conclusion

We demonstrate that STKM, a two phase spatiotemporal clustering method, is able to capture the multi-scale behavior of moving object data. Phase 1 returns an assignment for each point at every iteration, and provides us the unique ability to directly track cluster centers without any post-processing. This phase minimizes an objective function, that unlike existing methods, is unified in both space and time and requires many fewer

parameters to run. Phase 2 can be optionally applied to classify each point into a single long-term cluster. Because Phase 2 infers long-term relationships from short-term ones, applying both Phase 1 and Phase 2 results in more accurate predicted static clusters compared to methods that provide exclusively static clusters. The combination of both phases also allows us to explore the individual behavior of points in long-term clusters and draw conclusions about the relationships between clusters.

We demonstrate the competitiveness of STKM against existing spatiotemporal clustering methods on a benchmark dataset proposed by Cakmak et. al [82]. Our comprehensive comparison evaluates the predicted dynamic clusters against ground-truth static clusters using adjusted mutual information score. We refer to this measure as total AMI. STKM achieves the highest average total AMIs on datasets of size smaller than $20,000$ and the highest median total AMIs on all datasets. Averaging over all data sizes, STKM scores the highest average total AMI of all methods it is compared against. We show that the results of long-term AMI, which compares predicted static against ground-truth static clusters, are more representative of performance. We derive long-term clusters by applying Phase 2 of STKM both to the output of Phase 1 and to the outputs of the baseline methods. We show that STKM performs best in terms of average and median long-term AMI over all datasets, suggesting that the short-term relationships predicted by STKM are more informative than those of other methods. The tradeoff in using STKM is a slower runtime.

Overall, STKM demonstrably outperforms existing methods on the moving cluster problem. In the future, we intend to further explore robust versions of STKM for handling noisy spatiotemporal data as well as approaches to estimating the number of clusters k. With ever increasing information from broad applications such as surveillance, transportation, environmental and seismology studies, and mobile data analysis, STKM and other related methods are critical for the unsupervised analysis of spatiotemporal data streams.

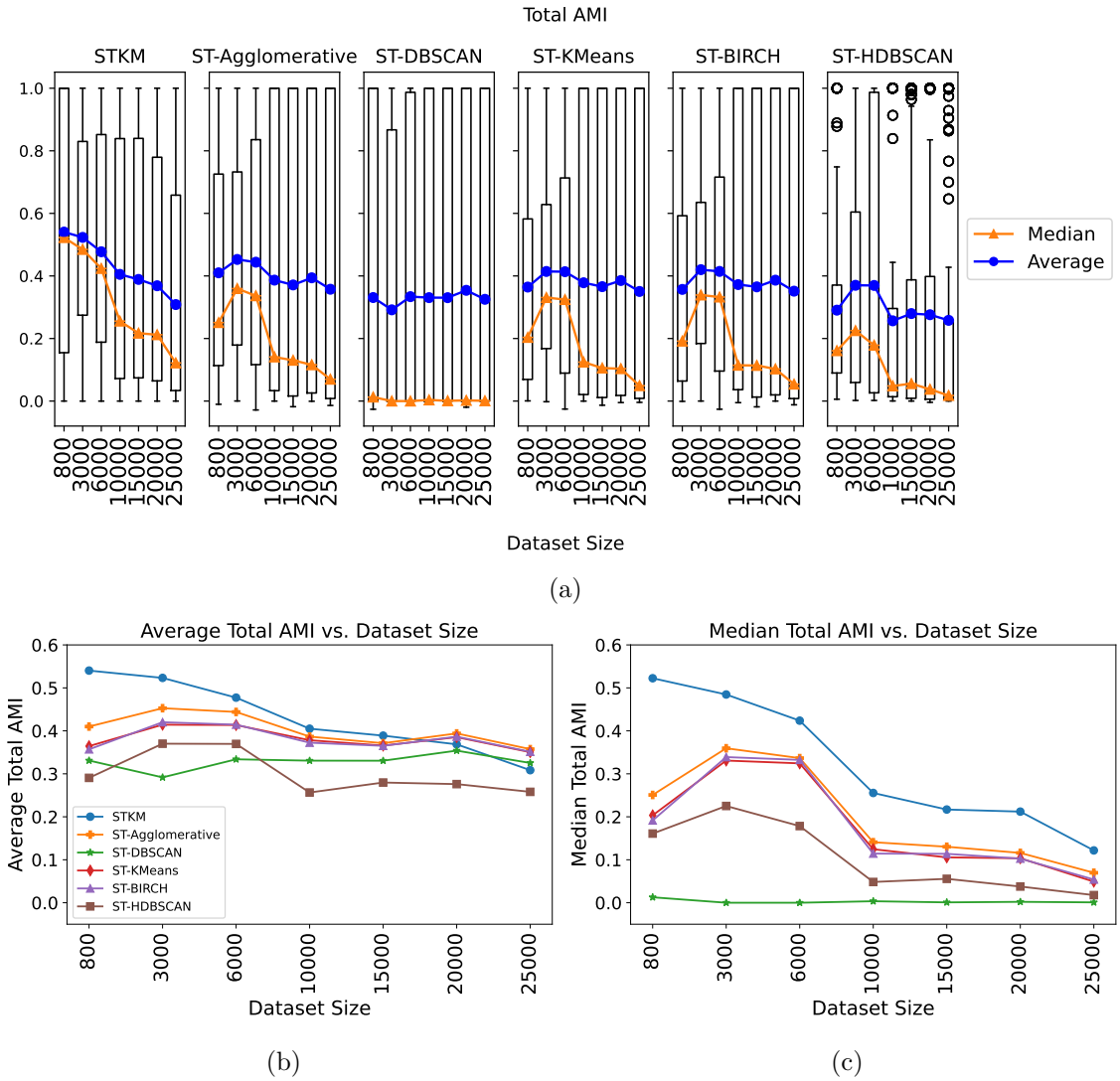


Figure 4.5: (a) Boxplots of total AMI scores for various methods over different dataset sizes. Median scores are shown in orange and average scores in blue. (b) A closeup of the average total AMI trendlines. (c) A closeup of the median total AMI trendlines. Boxplots for all methods and dataset sizes span the entire range $[0.0, 1.0]$, demonstrating the difficulty of evaluating a method that does not match the data generating mechanism. STKM achieves the highest average total AMI on datasets of size $\leq 20,000$, but maintains the highest median total AMI on all dataset sizes. On larger datasets, STKM scores higher in terms of total AMI more often, even though the magnitude of those scores is smaller.

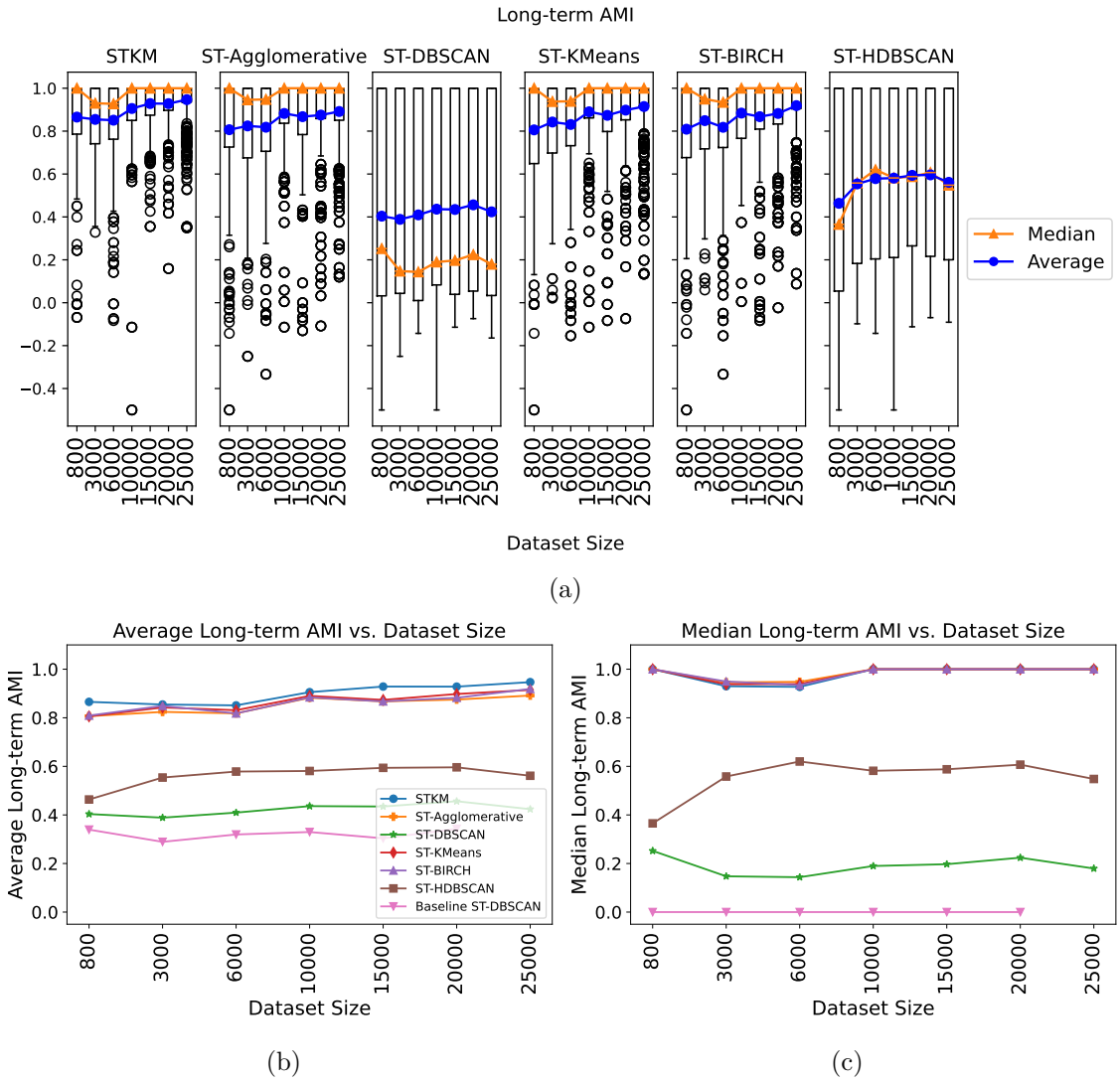
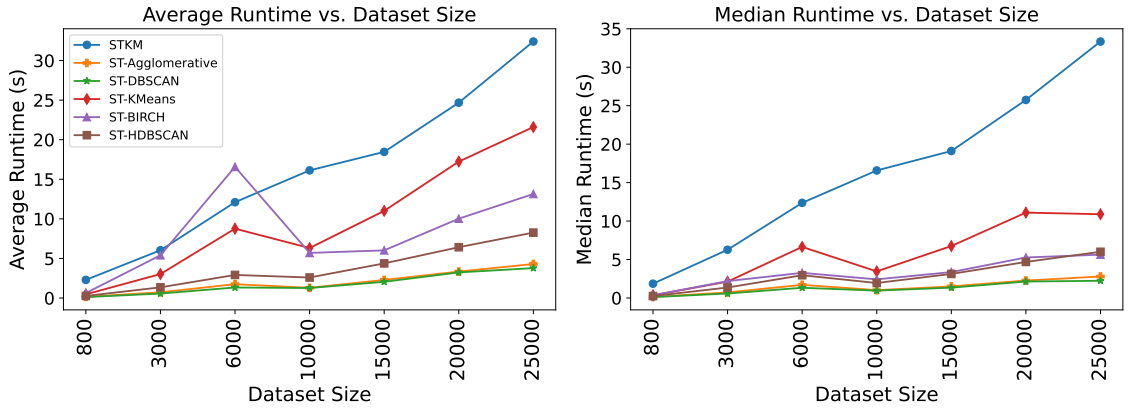


Figure 4.6: (a) Boxplots of long-term AMI scores for various methods over different dataset sizes. Median scores are shown in orange and average scores in blue. (b) A closeup of average long-term AMI trendlines. (c) A closeup of median long-term AMI trendlines. Boxplots for STKM have the smallest interquartile ranges, showing STKM's long-term AMI scores are the least variable. STKM achieves both the highest average and median long-term AMIs on datasets of all sizes. All methods that use short-term information to inform long-term predictions perform better than Baseline ST-DBSCAN, which does not provide information on short-term associations between data points.



(a) (b)

Figure 4.7: STKM has the longest runtime of all methods on datasets of all sizes.

Chapter 5

Dimensionality Reduction

5.1 Introduction

A dramatic increase in computational power, sensor ubiquity, and the availability of finer resolutions in scientific modeling has led to the creation of enormous data sets with fine detail. In response, efficient representations of big data are essential for processing, exploration, and human understanding. Dimensionality reduction operates on the principle that high-dimensional data actually resides in an inherently low-dimensional space and finding this low-dimensional representation leads to important insights.

There are many motivations for dimensionality reduction. In many cases, the features of high-dimensional data exhibit partial redundancy and dependency. In other words, there are more features than required to represent the inherent information of a signal. Dimensionality reduction can be used to extract the independent features that capture most of the information. Even if redundancy is not a concern, extracting the "most important" features of a data matrix and removing irrelevant features helps practitioners both to work with limited computational resources and to understand the underlying structure of the data. Further, projecting high-dimensional data into 2D or 3D space can be extremely beneficial for human visualization. Overall, too many features can complicate data analysis and visualization, and dimensionality reduction helps avoid these pitfalls.

There are two main classes of dimensionality reduction techniques: linear and nonlinear. At their core, linear techniques use linear transformations to shift and stretch the data. Examples include Singular Value Decomposition (SVD), Principal Component Analysis (PCA), and Fisher's linear discriminant analysis (LDA). These methods are a

cornerstone of analyzing high-dimensional data due to their simple geometric interpretations and typically attractive computational properties [64]. They are particularly useful when the data lies in a linear subspace and where the original variables are replaced by a smaller set of underlying variables. Non-linear techniques involve more complicated data transformations and include t-distributed stochastic neighbor embedding (t-SNE), Isomap, and autoencoders. Non-linear methods are generally more powerful than their linear counterparts, but can be slow to optimize and many are non-deterministic, meaning they get different, locally-optimal solutions each time [16].

The focus of our work is linear methods. SVD and PCA are some of the most common linear dimensionality reduction technique used today. In the following sections, we introduce both methods, demonstrate their use, and discuss their shortcomings.

5.2 Singular Value Decomposition

The SVD is motivated by the fact that the image of the unit sphere $\mathbf{a} \in \mathbb{R}^N$ under any matrix $\mathbf{X} \in \mathbb{R}^{m \times N}$ is a hyperellipse, which is simply the surface obtained by stretching the unit sphere in $\mathbb{R}^m$ by some factors $\sigma_1, \cdots, \sigma_m$ in some orthogonal directions $\mathbf{u}_1, \cdots, \mathbf{u}_m \in \mathbb{R}^m$ [83]. The vectors $\{\sigma_i \mathbf{u}_i\}$ are the principal semiaxes of the hyperellipse. Letting

$$||\mathbf{u}_i||_2 = 1,$$

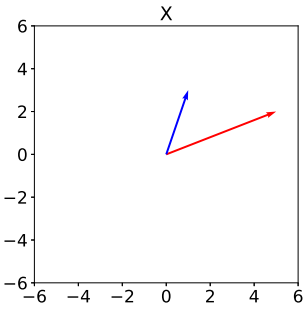
σ_i are then the lengths of these semiaxes and are referred to as the singular values of $\mathbf{X}$. By convention, we assume that the singular values are numbered in descending order so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m$. The left singular vectors of $\mathbf{X}$ are defined as the unit vectors $\{\mathbf{u}_i, \cdots, \mathbf{u}_m\}$ oriented in the directions of the principal semiaxes, and the right singular vectors of $\mathbf{X}$ are the unit vectors $\{\mathbf{v}_1, \cdots, \mathbf{v}_n\}$ that are the preimages of the principal semiaxes, numbered so that

$$\mathbf{X}\mathbf{v}_j = \sigma_j \mathbf{u}_j. \tag{5.1}$$

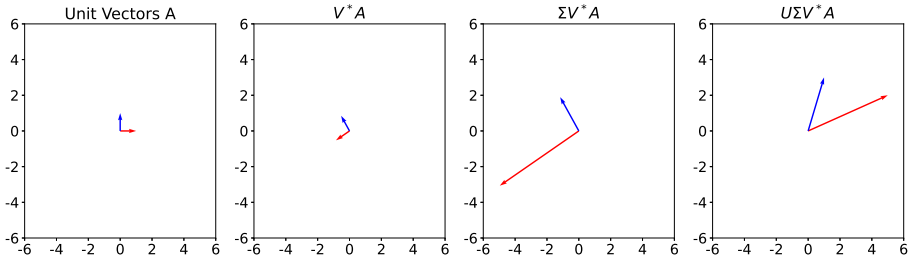
.

The vector equations 5.1 can also be expressed as a matrix equation

$$\mathbf{X}\mathbf{V} = \mathbf{U}\mathbf{\Sigma}, \tag{5.2}$$



(a)



(b)

Figure 5.1: Singular Value Decomposition (SVD) of $\mathbf{X}$. 5.1a Let $\mathbf{X}$ be a matrix containing the vectors $\{\langle 1, 3\rangle, \langle 5, 2\rangle\}$. 5.1b The action of the SVD of $\mathbf{X}$ on the matrix $\mathbf{A}$ containing unit vectors $\{\langle 0, 1\rangle, \langle 1, 0\rangle\}$. The right singular vectors $\mathbf{V}^*$ preserve the unit vectors, the singular values in $\mathbf{\Sigma}$ stretch them to the lengths of the vectors in $\mathbf{X}$, and the left singular vectors $\mathbf{U}$ rotate them to the directions of the vectors in $\mathbf{X}$.

where $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{N \times N}$ are unitary and and $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ is diagonal with positive real entries. Then we can express

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \tag{5.3}$$

and it is clear that the image of the unit sphere in $\mathbb{R}^n$ under the map $\mathbf{X}$ must be a hyperellipse in $\mathbb{R}^m$. The unitary map $\mathbf{V}^*$ preserves the sphere, the diagonal matrix $\mathbf{\Sigma}$ stretches the sphere into a hyperellipse, and the unitary map $\mathbf{U}$ rotates or reflects the hyperellipse without changing its shape [83]. This behavior is showcased in Figure 5.1.

Every matrix $\mathbf{X}$ has an SVD with uniquely determined singular values and left and right singular vectors unique up to complex sign. To calculate the SVD, we find the eigenvalues and eigenvectors of $\mathbf{X}\mathbf{X}^T$. The eigenvectors of $\mathbf{X}\mathbf{X}^T$ make up the columns of $\mathbf{U}$, the eigenvectors of $\mathbf{X}^T\mathbf{X}$ make up the columns of $\mathbf{V}$, and the singular values are the square roots of the eigenvalues of $\mathbf{X}\mathbf{X}^T$. We walk through an example of calculating the SVD of an example matrix below.

Let $\mathbf{X} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$. Then $\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 9 & 0 \\ 0 & 4 \end{bmatrix}$. To find the eigenvalues of $\mathbf{X}^T\mathbf{X}$, we solve the characteristic polynomial of $\mathbf{X}^T\mathbf{X}$, which for a square matrix is simply the determinant, as seen here,

$$\det(X^TX - \lambda I) = (9-\lambda)(4-\lambda) = 0,$$
$$\lambda_1 = 9, \lambda_2 = 4.$$

Since the singular values are simply the square roots of the eigenvalues of $\mathbf{X}^T\mathbf{X}$, we have $\sigma_1 = 3, \sigma_2 = 2$. Next, we proceed by finding the right singular vectors by finding an orthonormal set of eigenvectors of $\mathbf{X}^T\mathbf{X}$,

$$\mathbf{X}^T\mathbf{X} - 9\mathbf{I} = \begin{bmatrix} 0 & 0 \\ 0 & -5 \end{bmatrix}.$$

To find the eigenvector of this matrix corresponding to $\lambda = 9$, we look for a vector for which the following equation is satisfied:

$$\begin{bmatrix} 0 & 0 \\ 0 & -5 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

One potential solution is $\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Thus, $\mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Similarly, to find the eigenvector corresponding to $\lambda = 4$, we find a vector for which the following equation is satisfied:

$$\begin{bmatrix} 5 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

One potential solution is $\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. Thus, $\mathbf{v}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. The vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ both have unit length and are orthogonal to one another, as required. If $\mathbf{V}$ was not already orthonormal, we would use the Gram-Schmidt process to make it so. Then, to find $\mathbf{U}$, we solve $\mathbf{U} = \mathbf{X}\mathbf{V}\mathbf{\Sigma}^{-1}$,

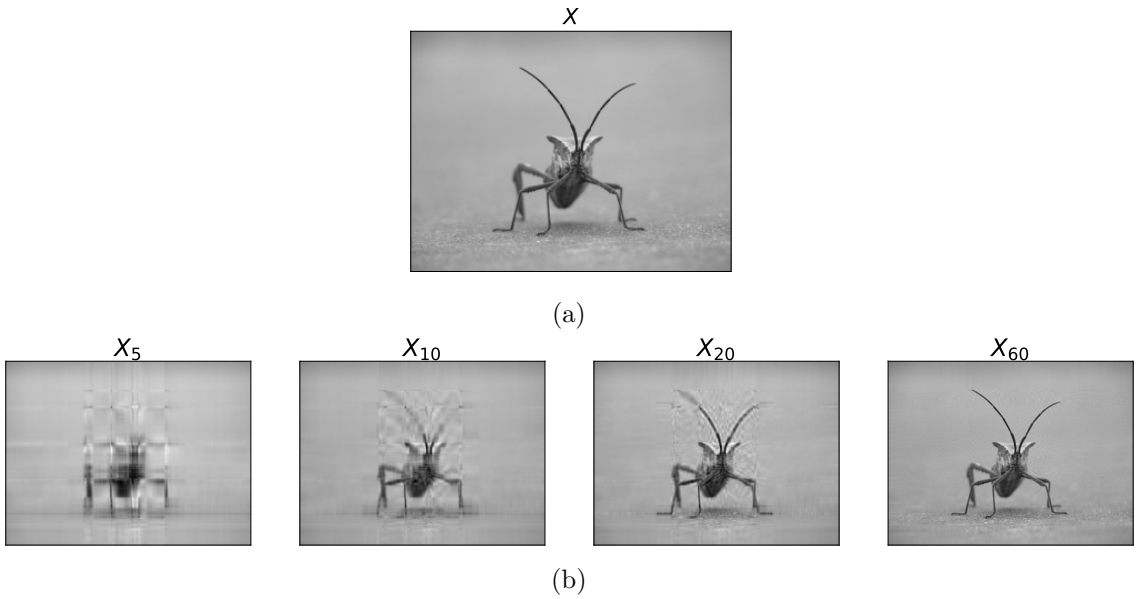


Figure 5.2: (a) The full-rank image $\mathbf{X} \in \mathbb{R}^{375\times500}$. (b)The $\mathbf{X}_k$ k-rank representations of the original image. As the rank increases, the image becomes clearer.

$$\mathbf{U} = \mathbf{X}\mathbf{V}\mathbf{\Sigma}^{-1} = \begin{bmatrix} 3 & 0 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1/3 & 0 \\ 0 & 1/2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Then, the SVD is the following,

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

In order to use the SVD for dimensionality reduction, we choose a subset of the singular values and their corresponding eigenvectors to construct a low-rank approximation. Consider the image in Figure 5.2. This 375×500 matrix $\mathbf{X}$ contains a grayscale number (from 1 to 255) for each pixel in the grid that determines how light or dark that pixel should be. While this image can be contained in memory, one can imagine that a much larger image may need to be compressed. To find a compressed representation we compute the SVD of $\mathbf{X}$ and use the top k singular values and vectors to create $\mathbf{X}_k$ k-rank representations. For example, a rank-k approximation is calculated as

$$\mathbf{X}_k = \mathbf{U}_{:,:k}\mathbf{\Sigma}_{:k,:k}V^*_{:k,:}. \tag{5.4}$$

Figure 5.2 shows how many singular values and vectors are needed to make picture $\mathbf{X}$ recognizable.

5.3 Principal Component Analysis

Principal Component Analysis (PCA) [22] is arguably the most common dimensionality reduction technique used for the analysis of large, multivariate datasets today. The algorithm assumes that the data approximately lies on a low-dimensional linear subspace and seeks the basis vectors that capture the maximum variance in the data.

Let $\mathbf{X} \in \mathbb{R}^{N \times m}$ be a data matrix containing N samples with m features. Define the mean-centered matrix as $\overline{\mathbf{X}} = \mathbf{X} - \mu_{\mathbf{X}}$, where each feature is centered by its mean. The covariance matrix $\mathbf{C} \in \mathbb{R}^{m \times m}$ of $\overline{\mathbf{X}}$ is then defined as

$$\mathbf{C} = \frac{\overline{\mathbf{X}}^T \overline{\mathbf{X}}}{n-1}.$$

Every entry $c_{i,j}$ tells us the covariance between feature columns $\mathbf{x}_i$ and $\mathbf{x}_j$. A positive covariance indicates that features have a positive relationship, a negative covariance indicates a negative relationship, and an entry of zero indicates that features are independent of one another. Since $\mathbf{C}$ is a symmetric matrix, it can be diagonalized as

$$\mathbf{C} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^*,$$

where $\mathbf{V}$ contains the eigenvectors of $\mathbf{C}$ and $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues λ_i on the diagonal. The principal components of $\mathbf{X}$ are defined as $\overline{\mathbf{X}}\mathbf{V}$.

We can derive a relationship between the SVD and PCA as follows. Let $\overline{\mathbf{X}}$ have the singular value decomposition $\overline{\mathbf{X}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$. Then,

$$\mathbf{C} = \frac{\mathbf{V}\mathbf{\Sigma}\mathbf{U}^*\mathbf{U}\mathbf{\Sigma}\mathbf{V}^*}{n-1} = \mathbf{V}\frac{\mathbf{\Sigma^2}}{n-1}\mathbf{V}^*.$$

This shows that the eigenvalues values of $\mathbf{C}$ are related to the principal components of $\overline{\mathbf{X}}$ by

$$\lambda_i = \frac{\sigma_i^2}{n-1},$$

and the principal components

$$\overline{\mathbf{X}}\mathbf{V} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*\mathbf{V} = \mathbf{U}\mathbf{\Sigma},$$

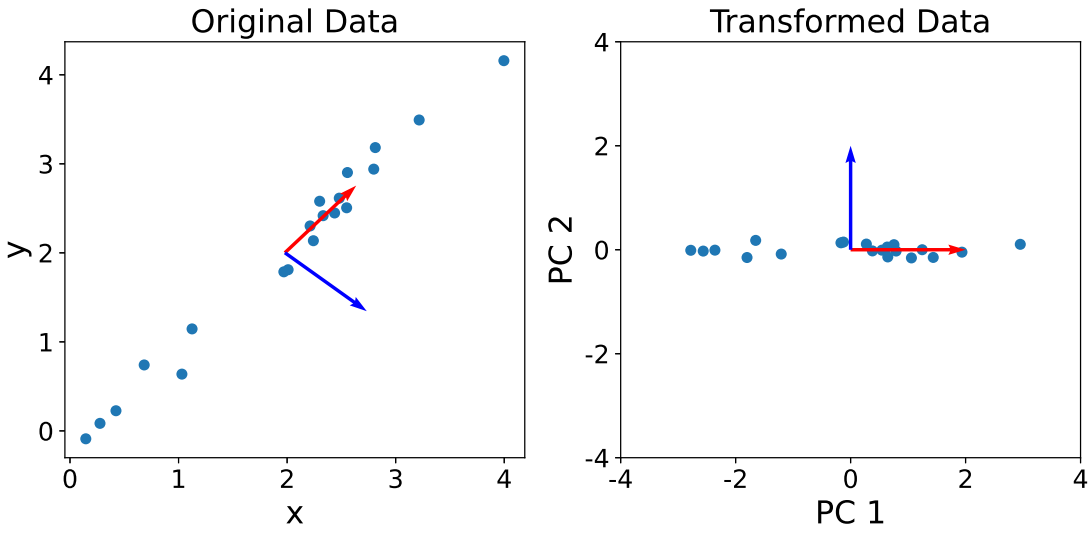


Figure 5.3: PCA carried out on a 2D dataset. The figure on the left shows the principal components superimposed on the original data, while the figure on the right shows the data after projection onto the principal components. The first and second principal components are in the directions of most and second-most variance in the data, respectively.

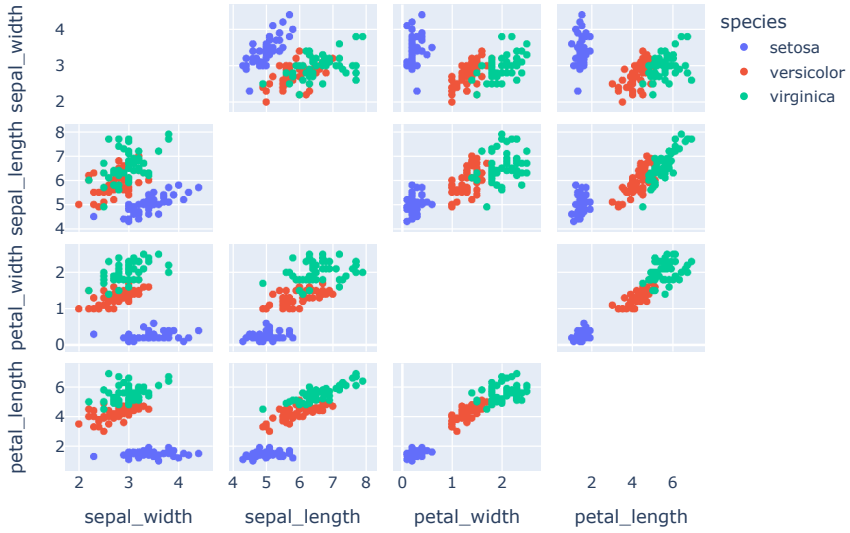
as in SVD.

Figure 5.3 visualizes the principal components on a simple 2D point cloud. The figure on the left shows the principal components superimposed on the original data, while the figure on the right shows the data after projection onto the principal components. It's clear that the first principal component is in the direction of the maximum variance of the data, and the second principal component is in the direction of the second most variance of the data.

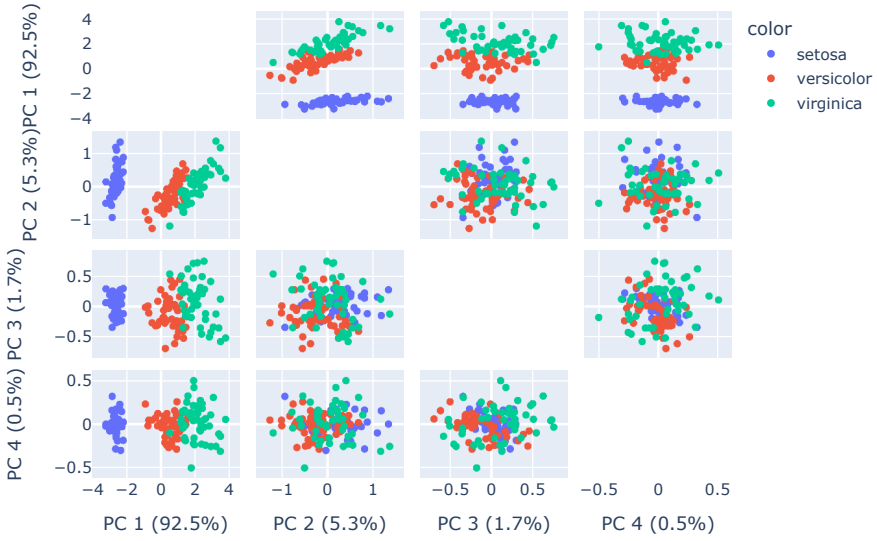
PCA can be used for dimensionality reduction in the same way as the SVD. We choose a subset of the largest eigenvalues and project the data onto the corresponding eigenvectors. Figure 5.4 showcases how PCA can be used for dimensionality reduction on the iris dataset. The iris dataset consists of 150 points with 4 features. Each point is assigned to a class of iris. In Figure 5.4a, the data is plotted against each pair of features. It is clearly difficult to separate the classes well. In Figure 5.4b, the iris dataset plotted against each pair of principal components. The percentages in parentheses correspond to the variance explained by each component. We can see that the iris classes are best separated when the data is plotted against the first and second principal components

Over the years, many extensions of PCA have been developed to address various shortcomings of the method. For example, kernel PCA extends PCA to the nonlinear space

and allows for more complicated data transformations by running PCA on a feature space instead of the inputs themselves [11]. Other extensions have focused on data with missing values. This domain has inspired the development of probabilistic PCA [12], which uses maximum likelihood estimation to to determine the principal factors, sparse PCA [29], where the PCA objective is augmented with a lasso-type penalty term, and other methods that incorporate data imputation [49, 77]. In addition, there has been substantial work done to assess the variability of the principal components and loadings PCA provides [69]. Finally, other extensions have focused on decreasing PCA's sensitivity to gross outliers and noise. One direction of research has been the use of robust estimators of scatter as opposed to sample covariance [18]. Another has been in the development of Robust PCA, which has become the standard method for dimensionality reduction on sparse, noisy data [62]. The following chapter introduces a method that addresses the latter two PCA limitations.



(a)



(b)

Figure 5.4: (a) The iris dataset consists of 150 points with 4 features. Each point is assigned to a class of iris. The dataset is plotted against each pair of features. It is clearly difficult to separate the classes well. (b) The iris dataset plotted against each pair of principal components. The percentages correspond to the variance explained by each component. The iris classes are best separated when the data is plotted against the first and second principal components.

Chapter 6

Ensemble Principal Component Analysis

6.1 Introduction

The focus of this chapter Principal Component Analysis (PCA). We focus on PCA [22], as is arguably the most common dimensionality reduction technique used for the analysis of large, multivariate datasets today. The algorithm assumes that the data approximately lies on a low-dimensional linear subspace and seeks the basis vectors that capture the maximum variance in the data.

Though considered a fundamental tool in data analysis, PCA has two well-known shortcomings: sensitivity to outliers and noise and no clear methodology for the uncertainty quantification of the purely descriptive information it provides. Previous work has focused on each of these problems individually. Robust PCA was developed to ensure PCA's success in the presence of sparse noise [62], and bootstrapping has been explored as a method for estimating the sampling variability of PCA [69]. To our knowledge, there has not been an extensive analysis of bootstrapping for the purpose of developing a noise-resistant PCA algorithm. Gabrys et. al suggested that statistical resampling could be applied to PCA to recover the true components of datasets corrupted with large outliers, but failed to test this concept on a sufficient number of datasets and ignored uncertainty quantification [28]. Drawing inspiration from [28] we propose a method that we call Ensemble Principal Component Analysis that combines bootstrapping with k-means cluster analysis to extend PCA to be both resistant to noise corruption and to provide measures of confidence. We

test EPCA against RPCA and standard PCA on a handful of datasets corrupted with sparse noise, white noise, and outliers. We show that EPCA outperforms competitors on datasets with outliers and performs competitively on datasets with other types of noise. EPCA achieves good performance while also allowing for uncertainty quantification and orders of magnitude reduction in computational cost compared to RPCA.

6.2 Related Work

Despite widespread use, PCA has some well-known drawbacks. The two that we focus on in this paper are 1) sensitivity to outliers and noise and 2) no clear methodology for the uncertainty quantification of the eigenvectors or the variance that each eigenvector explains. These shortcomings have both inspired entire directions of research.

We identify three main classes of noise. The first is white noise, where all entries of a data matrix $\mathbf{X}$ are slightly perturbed. Classical PCA actually gives the optimal estimate of the low-rank approximation of $\mathbf{X}$ when the corruption is caused by additive i.i.d. Guassian noise, and the noise energy is small relative to the dominant singular values of the Singular Value Decomposition (SVD) of $\mathbf{X}$ [44]. The second class is sparse noise, where a small subset of the elements of $\mathbf{X}$ are corrupt with some probability p, and the third class is outliers, where multiplicative noise is applied to a given percentage of rows of $\mathbf{X}$. In all noise domains, PCA will break down under large corruption or if even one entry of $\mathbf{X}$ is grossly corrupted [41].

This sensitivity has motivated the development of noise-resistant extensions of PCA. Directions of research include the use of robust estimators of scatter as opposed to sample covariance [18] and the development of Robust Principal Component Analysis (RPCA). RPCA sets the standard for dimensionality reduction on datasets with sparse noise, and is generally applied for foreground detection [62]. Mathematically, RPCA assumes we observe a data matrix $\mathbf{X}$ of the following form $\mathbf{X} = \mathbf{L}_0 + \mathbf{S}_0 + \mathbf{Z}_0$, where $\mathbf{L}_0$ is low-rank, $\mathbf{S}_0$ is a sparse matrix with most entries being zero, and $\mathbf{Z}_0$ is a noise term containing i.i.d. noise on each entry of the matrix [43, 42]. Generally, RPCA recovers $\mathbf{L}_0$ and $\mathbf{S}_0$ by solving the optimization problem in Equation 6.1 subject to some constraint on the $\mathbf{L}$ and $\mathbf{S}$ matrices [62],

$$\begin{aligned} \min_{\mathbf{L},\mathbf{S}} &||\mathbf{L}||_* + \alpha||\mathbf{S}||_1 \\ \text{s.t. } & X = L + S. \end{aligned} \tag{6.1}$$

.

Here, $||\cdot||_*$ denotes the nuclear norm, $||\cdot||_1$ denotes the sum of the absolute values of matrix entries, and $\alpha > 0$ is a tuning parameter controlling the regularization of $\mathbf{S}$. Equation 6.1 is referred to as the Principal Component Pursuit (PCP) problem, and a handful of methods have been suggested for solving PCP [62, 41]. PCP can recover $\mathbf{L}$ and $\mathbf{S}$ exactly [43], but is limited to the case where the low-rank component is exactly low-rank and the sparse component is exactly sparse. In addition, existing algorithms for solving PCP are computationally expensive and require extensive parameter tuning [62]. Another set of extensions to PCA focus on missing data, where entries of $\mathbf{X}$ are deleted at random. Most approaches specialized for dealing with missingness focus on data imputation [49, 77]. However, missingness can be considered a sub-category of sparse noise, and methods such as RPCA continue to work in this context.

Another drawback of PCA is that it offers no clear method for estimating the sampling variablity of its descriptive information. Though some work has explored analytical, asymptotic confidence intervals (CIs) for principal components, the methods are often either computationally infeasible or require strong assumptions on the data [69]. An alternative approach is using bootstrap-based CIs. Bootstrapping is a way to assess the variability of a statistic of interest through random sampling with replacement. It does not assume any distribution for the estimates of the uncertainties, and can be applied to most statistics [57].

In the context of PCA, boostrapping works by drawing B samples of size n with replacement from the population, running PCA on each sample, and storing the results. The variability of the PCA output from the B boostrap samples is then used to estimate the variability of PCA across different samples of the population [69]. From a theoretical standpoint, boostrapping is generally not useful unless we are interested in inference concerning only a few large eigenvalues, which are well-separated from the bulk and of multiplicity one [71], i,e. data of inherently low-rank. Though bootstrapping has been studied for the uncertainty quantification of PCA, very little work has been done using

boostrapping as a method for robustifying PCA. Gabrys et. al suggested using statistical resampling as a way to recover the principal components of a data matrix corrupt with outliers [28].

Taking inspiration from [28], we propose Ensemble Principal Component Analysis (EPCA) as a method for the dimensionality reduction and analysis of low-rank, noisy data that also lends itself to uncertainty quantification. EPCA ensembles bootstrapping PCA with k-means clustering to aggregate the output. Using k-means clustering allows us to circumvent the challenges associated with component re-ordering and sign ambiguity that arise when running PCA on data subsamples. We test the performance, with respect to runtime and accuracy, of EPCA against classical PCA and Robust PCA on datasets corrupted with sparse noise, white noise, and outliers.

6.3 Ensemble Principle Component Analysis (EPCA)

Given a data matrix $\mathbf{X} \in \mathbb{R}^{N \times m}$, EPCA samples B bags of size n at random with replacement. PCA is run on each of the B samples, the principal components are stored in a matrix $\mathbf{P}^{(j)}$, and the eigenvalues are stored in a matrix $\mathbf{\Lambda}^{(j)}$ for $j \in [1, B]$. The goal is to summarize the results of our B samples to output d dominant modes.

Two challenges are that there is rotational variablity in the principal components found by PCA and that the identified components can be re-ordered in the subsamples [31]. Most boostrapping PCA approaches tackle the first issue by using a Procrustean rotation to match the boostrap PCs to the PCs obtained by running PCA on the entire dataset [9] or by rotating the PCs towards some pre-specified target matrix $\mathbf{T}$ [31]. Instead, we create the matrix

$$\widetilde{\mathbf{P}} = \begin{bmatrix} \mathbf{P}^{(1)} \\ \mathbf{P}^{(2)} \vdots \\ \mathbf{P}^{(B)} \\ -\mathbf{P}^{(1)} \\ -\mathbf{P}^{(2)} \\ \vdots \\ -\mathbf{P}^{(B)} \end{bmatrix}, \tag{6.2}$$

stacking all of the principal components found in the bags and their reflections. In this

way, every principal component is stored along with its reflection, regardless of initial orientation. We also stack the corresponding eigenvalues accordingly, creating

$$\widetilde{\mathbf{\Lambda}} = \begin{bmatrix} \mathbf{\Lambda}^{(1)} \\ \mathbf{\Lambda}^{(2)} \\ \vdots \\ \mathbf{\Lambda}^{(B)} \\ \mathbf{\Lambda}^{(1)} \\ \mathbf{\Lambda}^{(2)} \\ \vdots \\ \mathbf{\Lambda}^{(B)} \end{bmatrix}. \tag{6.3}$$

The next step is to run k-means clustering on $\widetilde{\mathbf{V}}$ to output $2d$ clusters. This approach automatically clusters components that are oriented in the same direction and avoids any challenges associated with re-ordering. We use the normalized cluster centers of our d rotationally unique clusters as our predicted principal components, and the averages of the eigenvalues associated with the members of each cluster as our predicted eigenvalues. We order our final predicted components according to the magnitude of the average predicted eigenvalues. EPCA is visualized in Figure 6.1 and explained in Algorithm 6.

Algorithm 6 Ensemble Principal Component Analysis

(a) Mean center the data. $\overline{\mathbf{X}} = \mathbf{X} - \mu_{\mathbf{X}}$
(b) Select B bags of size n with replacement to create $\overline{X}^{(j)}$ with $j \in [1, B]$.
(c) Run PCA on each of the B bags to output d eigenvalues $\mathbf{\Lambda}^{(j)}$ and d principal components $\mathbf{P}^{(j)}$
(d) Stack all $\mathbf{P}^{(j)}$ and their reflections to create $\widetilde{\mathbf{P}}$. Stack corresponding $\mathbf{\Lambda}^{(j)}$ to create $\widetilde{\mathbf{\Lambda}}$.
(e) Run k-means clustering on $\widetilde{\mathbf{P}}$ with $2d$ clusters to automatically cluster principal components oriented in the same direction.
(f) Output the directionally unique d average principal components, their corresponding average eigenvalues, and the variances of both.

6.4 Uncertainty Quantification

Like all other boostrapping PCA methods [57, 69, 31, 9], EPCA lends itself to uncertainty quantification. There are many approaches to estimate a confidence interval (CI) from the boostrap distribution, but we focus on the percentile method, which uses the central $(1 -$

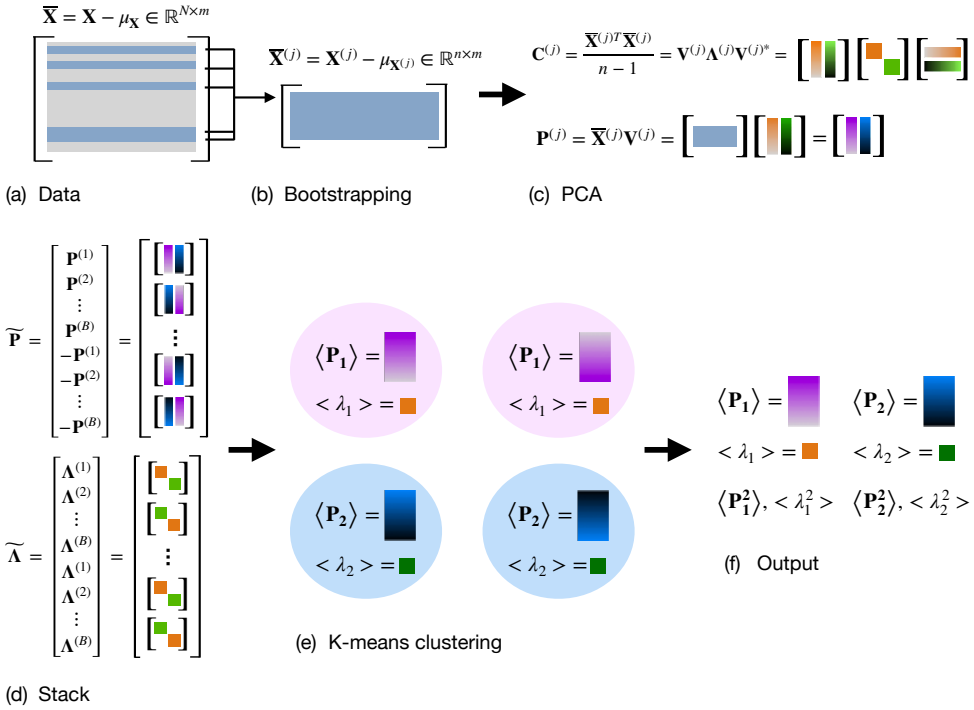


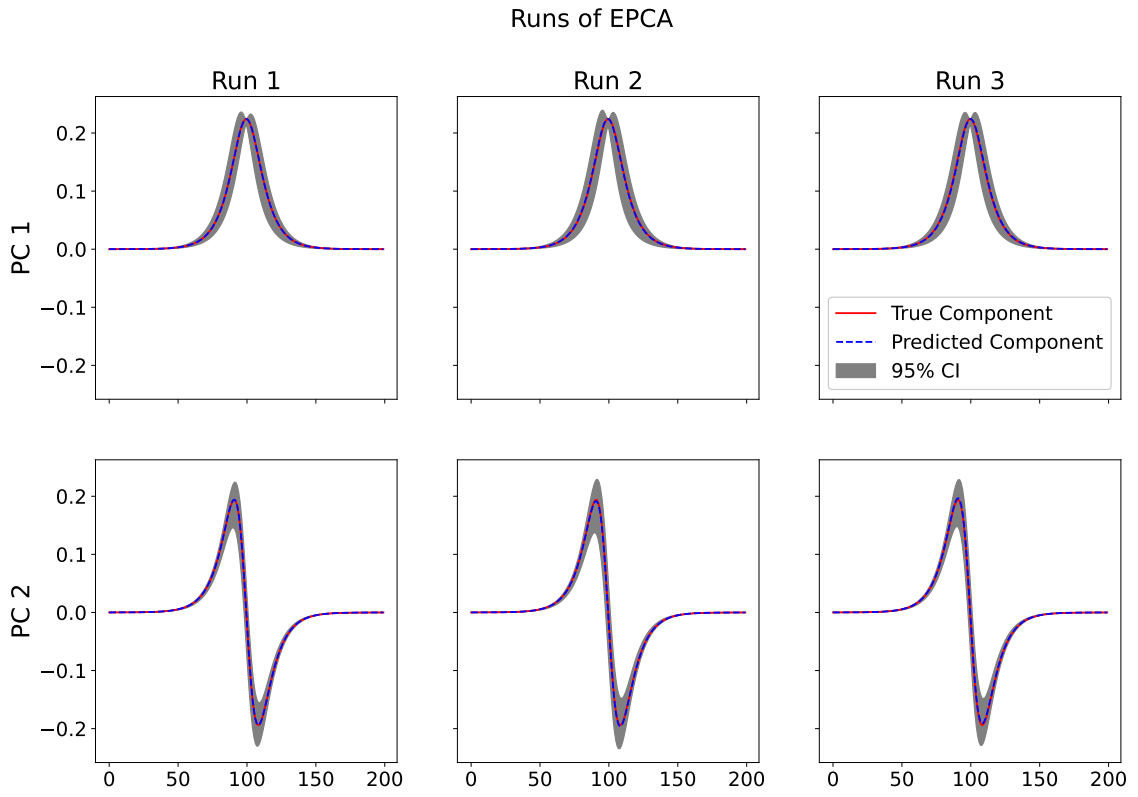
Figure 6.1: Ensemble Principal Component Analysis (EPCA). Given a data matrix with inherently low-rank structure, we sample B bags of size n at random with replacement. We run PCA on and store d principal components $\mathbf{P}^{(j)}$ and their corresponding eigenvalues $\mathbf{\Lambda}^{(j)}$ for each bag. We create $\widetilde{\mathbf{P}}$ by stacking all $\mathbf{P}^{(j)}$, along with their reflections to account for the rotational variability among the principal components. We also stack all $\mathbf{\Lambda}^{(j)}$, in accordance with the order in $\widetilde{\mathbf{P}}$ to create $\widetilde{\mathbf{\Lambda}}$. Next, we run k-means clustering with $2d$ clusters.

2α) part of the cumulative boostrap distribution as the approximate central $100(1-2\alpha)\%$ CI. The lower and upper end-points of the CI are given by $\hat{\theta}^*(\alpha)$ and $\hat{\theta}^*(1-\alpha)$, where $\hat{\theta}^*$ is the mean value of all B boostrap estimates and $\hat{\theta}^*(\alpha)$ represents the 100αth percentile of $\hat{\theta}^*$. The percentile method is transformation respecting, range preserving, and first order accurate [31].

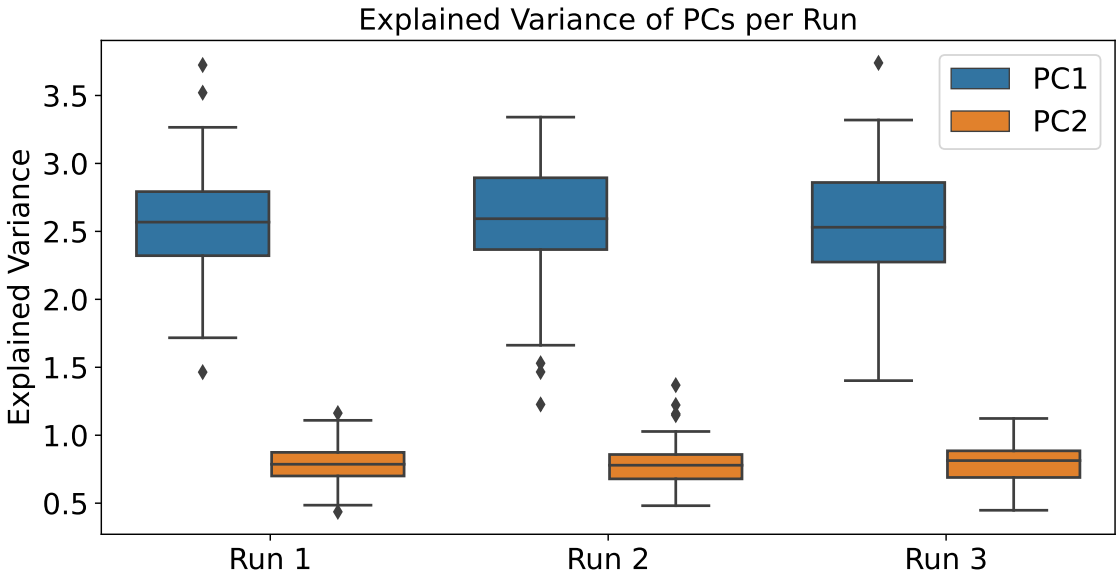
Figures 6.2 and 6.3 show measures of uncertainty quantification for three runs of EPCA on a clean dataset and the same dataset containing 5% outliers of scale 10, respectively. We show the results of three runs, since the results of EPCA vary based on the randomly chosen B bootstrap samples. Figures 6.2a and 6.3a show the 95% CIs for the principal components. The tightness of our CIs gives us an idea of the level of confidence of our output. We note that the CIs are significantly tighter for the clean data, and even though there is greater uncertainty on the corrupted data, EPCA is still able to closely identify the true components in two of the three pictured runs. Figures 6.2b and 6.3b show the distributions of the respective eigenvalues, which tell us the variance explained by each of the principal components. On both the original and corrupt data, the interquartile ranges (IQR) of the explained variances capture the true values, but on the corrupted data, the IQRs are skewed higher and slightly wider. Gross outliers have been removed from the boxplots in Figure 6.3b. Even though the explained variances are more difficult to capture on the outlier data, they at least tell us about the order of the principal components and their separation. In practice, the eigenvalues of the covariance matrix are not used in PCA's projection of data into a lower-dimensional space, so their explicit values are not as important as the principal components themselves.

6.5 Experiments

Classical PCA is known to function well even in the presence of white noise with small variance, but breaks down under large corruption [44]. In response, RPCA was created, specifically to handle sparse noise, and extensive work has been done to improve the runtime and reliability of the method [43]. Even so, RPCA remains time intensive and sensitive to parameter choice. Therefore, we propose EPCA as an alternative for identifying the principal components on noisy data. We test EPCA against RPCA solved using Augmented Lagrange Multipliers [41] and classical PCA on seven datasets with four

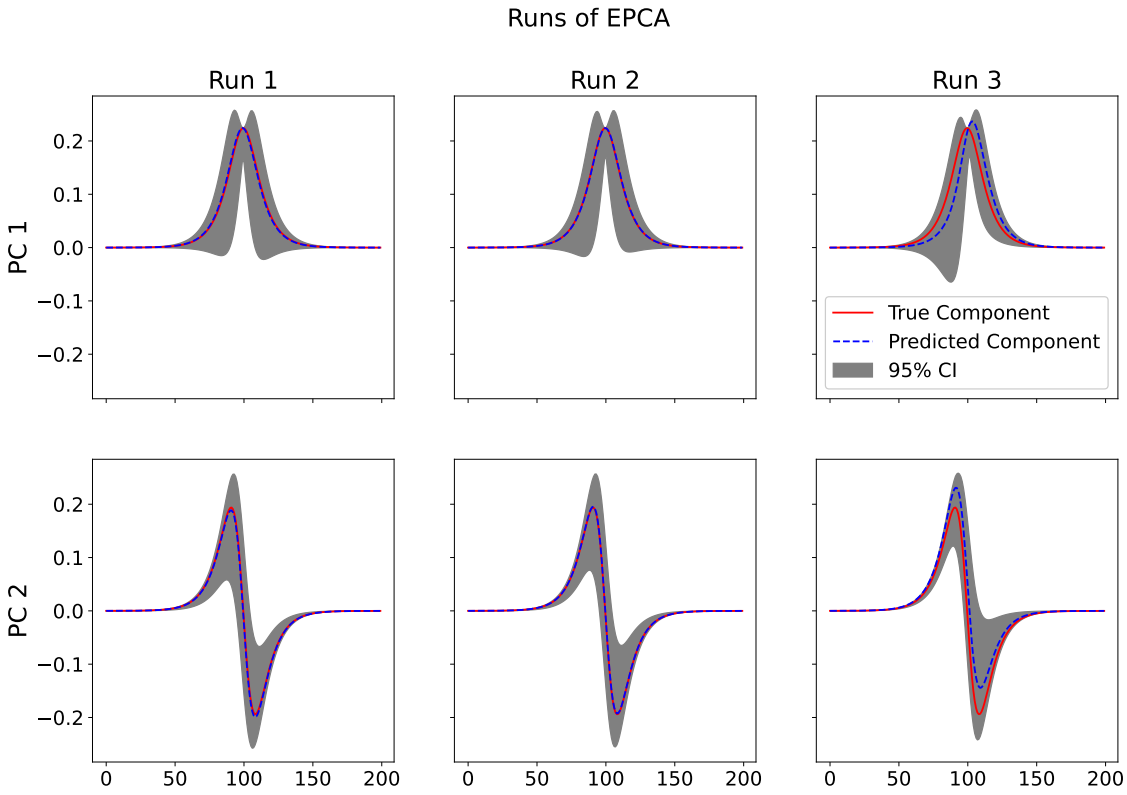


(a)

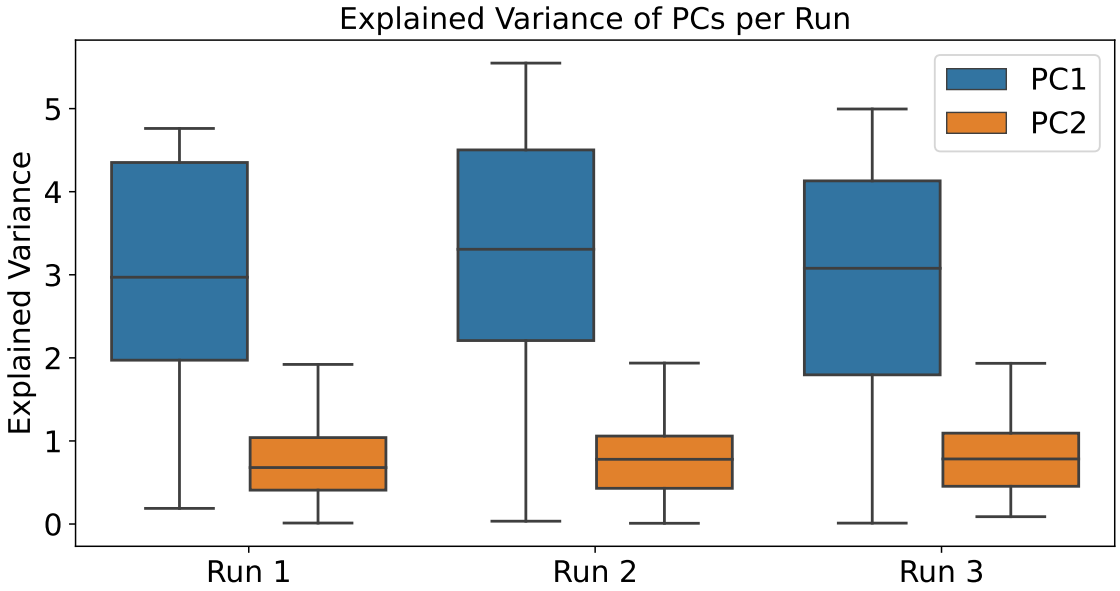


(b)

Figure 6.2: Uncertainty Quantification of three runs of EPCA on wave data. (a) The 95% Confidence Intervals (CIs) associated with the top two principal components. The CIs are tight, suggesting high confidence in the predicted components. (b) Boxplots of the distribution of the explained variance of each of the principal components. The true eigenvalues of the covariance matrix are contained within the interquartile ranges (IQRs) of all three trials.



(a)



(b)

Figure 6.3: Uncertainty Quantification of three runs of EPCA on wave data corrupt with 5% outliers of scale 10. (a) The 95% Confidence Intervals (CIs) associated with the top two principal components. The CIs are wider than that of the clean data. (b) Boxplots of the distribution of the explained variance of each of the principal components. The true eigenvalues of the covariance matrix are contained within the interquartile ranges (IQRs) of all three trials, but the IQRs are skewed upward and wider than those from the clean data. Gross outliers have been removed from the boxplots for visualization purposes.

types of added noise: sparse, Gaussian white, uniform white, and outliers. We include three small, three medium, and one large dataset in our analysis. Our base datasets, in order of increasing size, are iris [45], wine [45], breast cancer Wisconsin (WBC) [45], a synthetic wave, the digits 0 and 1 from MNIST [47], and sea surface temperature (SST) [52].

The first type of noise is sparse noise, where each entry in the data matrix $\mathbf{X}$ is corrupt with probability p. The value of corrupt entries is set to c times the maximum value in $\mathbf{X}$. On datasets with few features, very few points will have corruption. However, as the number of features grows, the number of corrupted points will also grow. With a probability $p = 0.01$, on our smallest dataset, containing points with 4 features, about 96% of the data will remain clean, while on our most high-dimensional dataset, containing points described by 64800 features, it is highly unlikely that any of the points will remain uncorrupted. The next two types of noise are types of white noise, where noise from either the uniform distribution with mean 0 and variance v or the Guassian distribution with mean 0 and variance v are added to $\mathbf{X}$. We ensure the variance is small with respect to the dominant singular value of $\mathbf{X}$, as this is a level of corruption under which classical PCA should still perform well [44]. In practice, we set $v = \frac{\sigma_1}{f}$, where σ_1 is the dominant singular value of $\mathbf{X}$ and f is some variance divisor. Finally, we create outliers by multiplying a randomly selected s% of the rows of $\mathbf{X}$ by an outlier scale of S. We expect RPCA to perform best on sparse data, PCA to perform well even in the presence of low-variance white noise [44], and EPCA to perform best on outlier data.

For all datasets, we perform EPCA with $B = 100$ bags, but vary the size n of the bags. We take n to be larger on datasets with white noise and sparse noise and smaller on datasets with outliers. Since white noise is applied to all entries of $\mathbf{X}$ and sparse noise will impact most entries as $\mathbf{X}$ becomes higher-dimensional, choosing larger bag sizes n helps "mute" the impact of the noise. Contrastingly, since outliers impact only s% of entries, choosing smaller bags helps prevent the selection of an outlier. In RPCA, the parameter controlling the extent of the regularization on the sparse part of $\mathbf{X}$ is set to $\alpha = 0.20$. The remaining parameters for the Augmented Lagrange Multipliers algorithm used to solve RPCA are set to their default values.

Given an initial data matrix $\mathbf{X}$, we calculate the true PCA modes using classical PCA. We then corrupt $\mathbf{X}$ in four ways: sparse noise, uniform white noise, Gaussian white noise,

and outliers. We run PCA, RPCA, and EPCA on each of the corrupted data sets and quantify the percent relative error for the components using Equation 6.4.

$$\%\text{Relative Error} = \frac{||t - p||_2}{||t||_2} \times 100, \tag{6.4}$$

where t are the true components and p the predicted components.

We carry out two sets of experiments. In the first, we test how PCA, EPCA, and RPCA respond to various levels of noise. We take each of our seven datasets, and randomly add noise of a given level to the data five times. The results of PCA and RPCA are deterministic, meaning that on a given dataset with set parameters, these methods will produce only a single result. In contrast, the results of EPCA are stochastic due to the randomness in the bagging procedure, and the method can produce a different prediction on each run. Therefore, we run PCA and RPCA one time each on the five corrupted datasets and average their respective errors to get an average percent relative error for that particular level of corruption. Since the results of EPCA are non-deterministic, we run EPCA five times on each of the five datasets, and average all twenty-five runs together to get the average error for that corruption. We repeat this process for the seven datasets, and average performance over all datasets to get our final results. For sparse noise, we vary both the probability p of an entry being corrupt and the scale c of the corruption. For both normal and uniform white noise, we vary the scale of the variance divisor f. Finally, for outliers, we vary both the percentage s of corrupt rows and the scale S.

In the second set of experiments, we compare the performance of PCA, RPCA, and EPCA on datasets with fixed levels of corruption to test variability in performance. For each of our seven datasets and each type of noise, we repeat random corruption and evaluation 100 times. In our analysis, we consider only the result produced by EPCA on the first run.

As we saw in Section 6.4, the eigenvalues of the covariance matrix in EPCA are skewed on noisy data. However, the eigenvalues are not involved in PCA's low-dimensional mapping, only in ordering the predicted components. Therefore, we consider only the error in the principal components in our experiments.

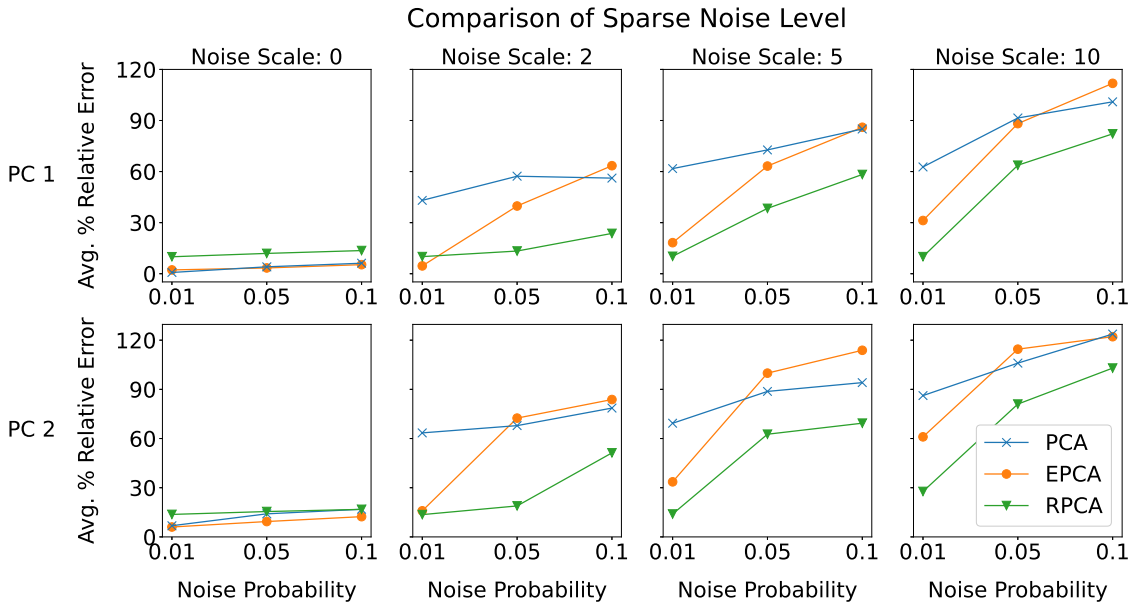


Figure 6.4: Average percent relative error of predicted principal components from PCA, EPCA, and RPCA for data corrupted with sparse noise of different probabilities and scales.

6.6 Results

Figure 6.4 compares the average performance of PCA, EPCA, and RPCA over datasets corrupted with various levels of sparse noise, determined by some combination of the probability p of an entry being corrupt and the scale c of the corruption. As we expect, RPCA performs best on sparse noise, most noticeably as the probability and scale of the noise increases. The exception is when the noise scale is set to 0, simulating missing data. It is possible that at a noise scale of 0, the value of α controlling the regularization of the sparse part of the data matrix in RPCA was not tuned optimally. Recall that we set $\alpha = 0.20$ for all experiments. When $p = 0.01$, EPCA consistently performs second-best. When $p = 0.05$, PCA performs second-best in terms of error in the first principal component and slightly worse than EPCA in terms of the error in the second principal component. Finally when $p = .10$, EPCA generally performs worst of the three methods. Ultimately, RPCA is the preferred choice on sparse data, but EPCA is a competitive choice when our noise probability is very small.

Figure 6.5 explores the performance of the three methods on data with added Gaussian and uniform white noise of different variances $v = \frac{\sigma_1}{f}$. When the variance divisor f is very large, PCA performs best in terms of the average percent relative error in the first and second principal components. As the variance divisor becomes smaller, PCA, EPCA, and

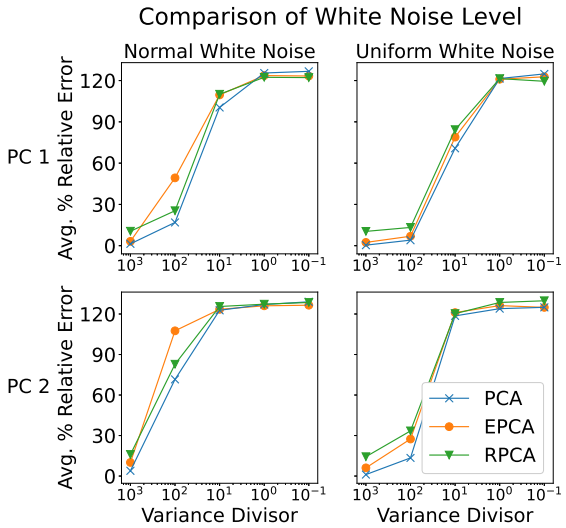


Figure 6.5: Average percent relative error of predicted principal components from PCA, EPCA, and RPCA for data corrupted with Gaussian and uniform white noise of various variances.

RPCA begin to perform similarly to one another. With one exception, when $f = 10^2$ on Gaussian white noise data, EPCA again performs second-best.

Next, Figure 6.6 displays the performance of PCA, EPCA, and RPCA on data corrupted with outliers at various percentages s and scales S. We observe that for the first principal components, when the percentage of outliers is 15% or lower, EPCA consistently achieves the lowest average percent relative error, regardless of outlier scale. The exception is 1% noise of scale 2, when PCA achieves the lowest error in the first principal component. As the percentage of outliers increases, EPCA begins to perform worse than PCA and RPCA. For the second principal component, EPCA outperforms the competitors in the majority of cases, regardless of outlier scale, across all outlier percentages. The exception again is 1% noise of scale 2, as well as 25% noise of scale 5. We conclude that EPCA will generally outperform PCA and RPCA on outlier data, when the percentage of outliers remains small, regardless of the scale of those outliers.

Next, we investigate the performance of PCA, EPCA, and RPCA over datasets with fixed levels of corruption. All datasets are formed by adding one of our four types of noise to seven base datasets 100 times. For sparse noise, we choose to corrupt each entry of our data matrix $\mathbf{X}$ with probability $p = .01$ and noise scale $c = 2$. For white noise, we choose our variance divisor to be $f = 1000$, so that the variance is set to $v = \frac{\sigma_1}{1000}$ for each data matrix $\mathbf{X}$. Finally, we add 5% outliers of scale 5 to each dataset.

The first aspect of performance we consider is runtime. For each of our seven corrupted

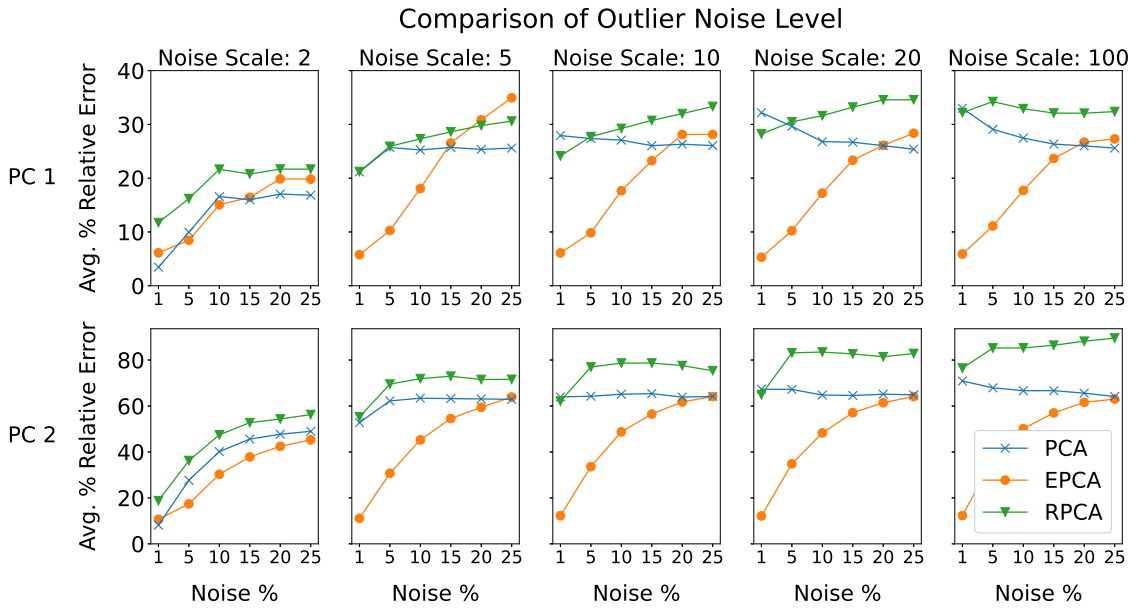


Figure 6.6: Average percent relative error of predicted principal components from PCA, EPCA, and RPCA for data corrupt with outliers of varied percentages and magnitudes.

datasets, we create boxplots of the spread of the runtime of the 400 runs of each method, as seen in Figure 6.7. For MNIST, we summarize runtime for digits 0 and 1 together. PCA consistently achieves the fastest runtime for all data. On the smallest datasets, iris ($\mathbf{X} \in \mathbb{R}^{140\times4}$) and wine ($\mathbf{X} \in \mathbb{R}^{178\times13}$), RPCA runs slightly faster than EPCA, and on the WBC data ($\mathbf{X} \in \mathbb{R}^{569\times30}$) RPCA runs slightly slower than EPCA. On all three datasets, RPCA and EPCA run in time on the same order of magnitude, while PCA runs one to two orders of magniutde faster. On the medium-sized datasets wave ($\mathbf{X} \in \mathbb{R}^{6000\times200}$) and MNIST 0 and 1 ($\mathbf{X} \in \mathbb{R}^{5923\times784}$, $\mathbf{X} \in \mathbb{R}^{6742\times784}$), RPCA runs two orders of magnitude slower than EPCA, but PCA and EPCA run on the same order of magnitude. Finally, on the largest SST data ($\mathbf{X} \in \mathbb{R}^{1726\times64800}$), EPCA is unable to provide an output, timing out after 120s, while EPCA and RPCA run on the same order of magnitude. We conclude that on small datasets, RPCA and EPCA have similar runtime. However, unlike RPCA, which is infeasible to run on larger datasets, our method EPCA scales no worse than classical PCA.

The second aspect of performance that we consider is percent relative error in the predicted first and second components. Figure 6.8 shows boxplots of error for each of the three methods for 100 runs of each type of data corruption over our seven datasets. Outliers in the boxplots have been removed for easier visualization. As expected, on datasets where sparse noise is added, RPCA is able to identify the true principal components with the

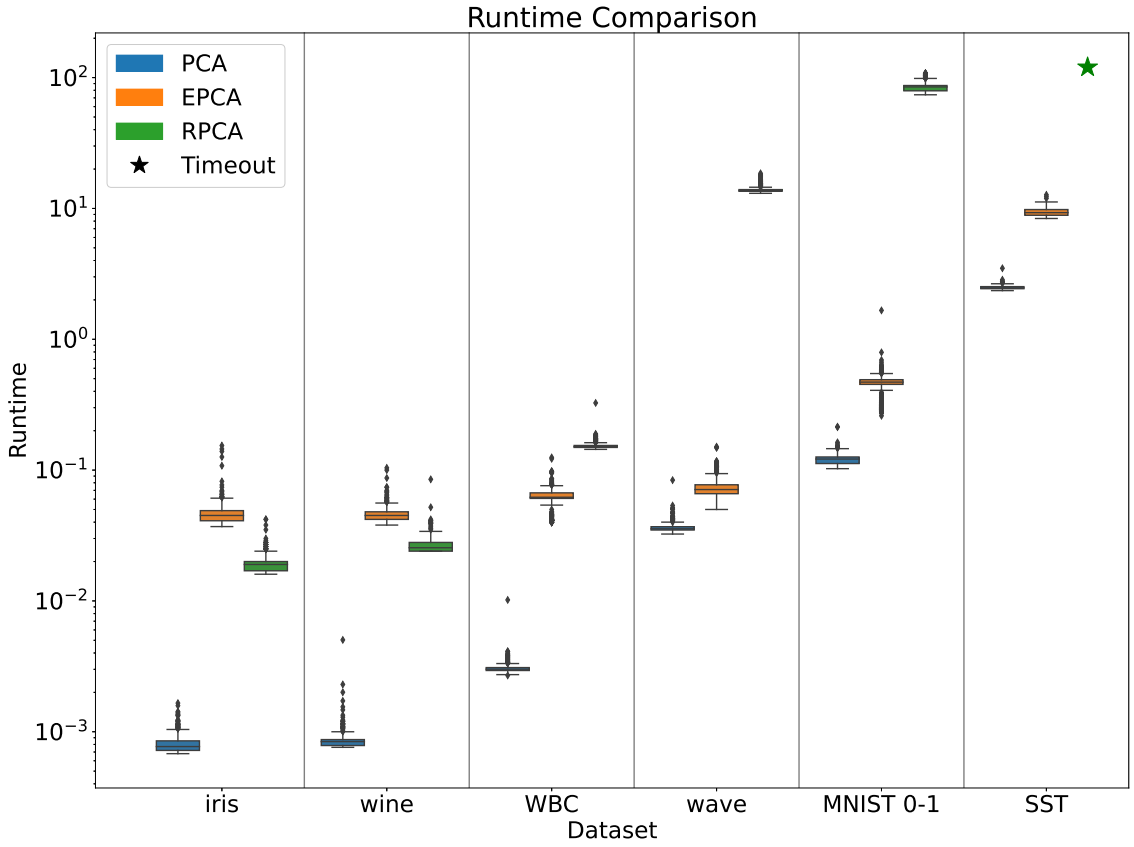


Figure 6.7: Runtime Comparison of PCA, EPCA, and RPCA. For each of our thirteen corrupted datasets, we create boxplots of the spread of the runtime of 400 runs of each method. For MNIST, we summarize runtime for all ten digits together. PCA consistently achieves the fastest runtime for all data. On the smallest datasets, iris, wine, and WBC, RPCA and EPCA run in time on the same order of magnitude, while PCA runs one to two orders of magnitude faster. On the medium-sized datasets wave and MNIST, RPCA runs two orders of magnitude slower than EPCA, but PCA and EPCA run on the same order of magnitude. Finally, on the largest SST data, EPCA is unable to provide an output, timing out after 120s, while EPCA and RPCA run on the same order of magnitude.

lowest median percent relative errors and the least variability in its results, as evidenced by tighter interquartile ranges (IQRs). Recall that this performance unfortunately comes at the cost of a much higher runtime. Though EPCA performs with significantly more error than RPCA, we note that EPCA achieves a similar median error to PCA, as well as a tighter IQR for error in the first PC and both a lower median error and slightly tighter IQR for the second PC.

For uniform white noise, classical PCA outperforms the other methods with both the smallest IQRs and lowest median errors for both principal components. EPCA performs second best in both categories. On data with normal white noise, PCA performs best and EPCA second-best in terms of median error in both PCs. RPCA has the tightest IQR for the first PC, and EPCA for the second.

Finally, on datasets containing outliers, EPCA outperforms the other methods significantly, achieving a lower median percent relative error for both principal components. EPCA also achives the tightest IQR for the first PC and the second-tightest for the second PC. The results of Figure 6.8 are summarized in Table 6.1, which ranks the median percent relative error in both principal components for the three methods in each data domain, and provides average ranks of performance. Though not shown, for all types of data corruption, the boxplots associated with EPCA contain many outliers. This is unsurprising, as the results of EPCA are non-deterministic. It is possible that the outliers in the EPCA boxplots are the results of boostrapping procedures that contained bags with high levels of noise.

6.7 Conclusion

We propose Ensemble Principal Component Analysis (EPCA) as a scalable extension of PCA that operates well in the presence of various types of noise and lends itself naturally to uncertainty quantification. Our innovative ensembling of boostrapping and k-means clustering allows us to automatically handle the challenges of principal component reordering and sign ambiguity in boostrapping PCA. We test the performance of EPCA against RPCA, which is specialized for datasets courrupt with sparse noise, and classical PCA, which should operate well even in the presence of low variance white noise. Further, we demonstrate that EPCA scales no worse than classical PCA and orders of magnitude

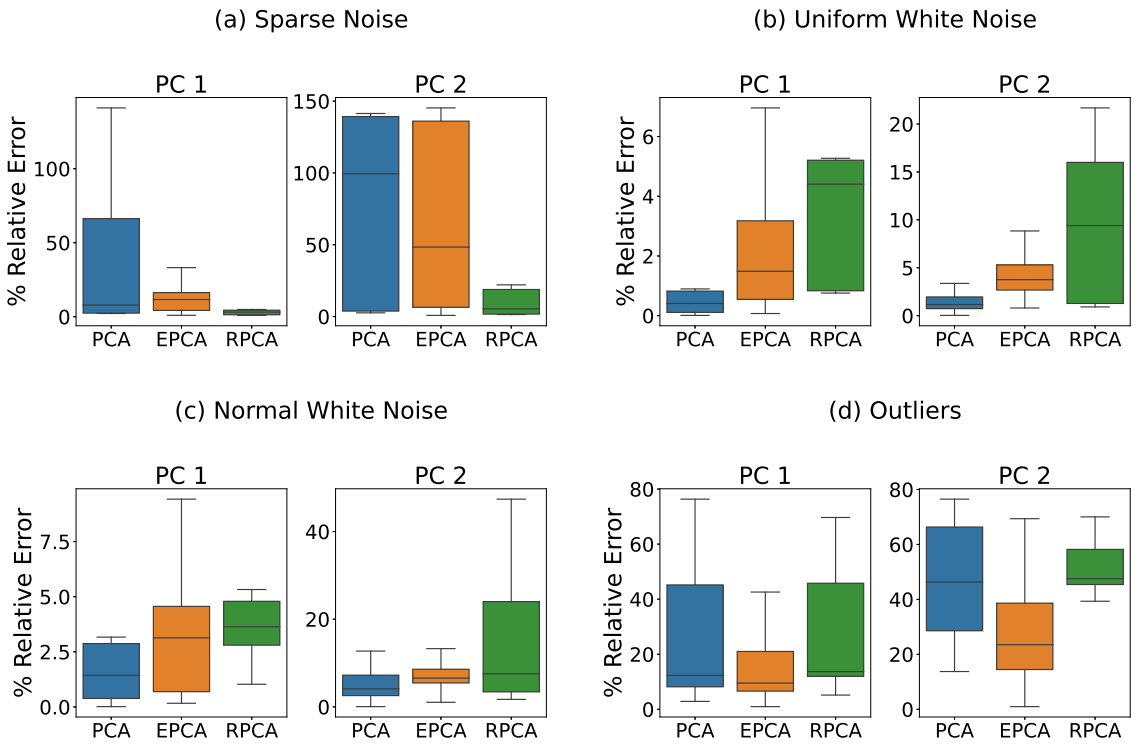


Figure 6.8: Summary of the performance of PCA, EPCA, and RPCA on noisy data folowing 100 trials of random corruption and evaluation. (a) For data corrupt with sparse noise ($p = 0.01$, $c = 2$), RPCA outperforms the competitors with both lower and tighter IQRs for both principal components. RPCA performs second best with lower median error than PCA. (b) and (c) For data corrupt with uniform and Gaussian white noise ($v = \frac{\sigma_1}{1000}$), PCA performs best with tighter and lower IQRs, followed closely by EPCA. For data with outliers ($s = 5$, $S = 5$), EPCA significantly outperforms the competitors with both lower and tighter IQRs for both principal components.

better than RPCA with respect to computational runtime.

Overall, EPCA is unable to outperform RPCA on sparsely corrupted data or classical PCA on data with added white noise with respect to percent relative error in the principal components. However, EPCA performs second-best in both noise domains. On datasets containing gross outliers, EPCA significantly outperforms both PCA and EPCA and maintains this performance advantage, regardless of the scale of the outliers, as long as the percentage of outliers remains low.

Although EPCA has a noise domain where it performs best, the method remains useful for other types of data corruption as well. An added bonus of using EPCA is that unlike classical PCA or RPCA, confidence intervals for the components and the explained variance of those components can be derived naturally. Finally, the benefits of EPCA come without a large computational cost. EPCA is significantly faster than RPCA and scales no worse than PCA on large datasets. The runtime of EPCA is is influenced by the number B and size n of bootstrapped samples, but since PCA is a subroutine of EPCA, EPCA is unlikely to outperform it. However, each PCA subroutine is independent of the rest, and the potential for the parallelization of EPCA should be explored as future work.

Table 6.1: Ranks of EPCA, RPCA, and classical PCA in terms of Median % Relative Error in the principal components after corruption and evaluation 100 times in each data domain. We find that RPCA performs best on data with sparse noise, and PCA performs best on data white noise. In both aforementioned domains, EPCA performs second-best, and further performs best on outlier data.

	EPCA PC1	RPCA PC1	PCA PC1	EPCA PC2	RPCA PC2	PCA PC2
Sparse Noise						
iris	1.0	3.0	2.0	1.0	2.0	3.0
wine	1.0	2.0	3.0	2.0	1.0	3.0
breast cancer	2.0	1.0	3.0	2.0	1.0	3.0
wave	2.0	1.0	3.0	2.0	1.0	3.0
mnist 0	2.0	1.0	3.0	2.0	1.0	3.0
mnist 1	2.0	1.0	3.0	2.0	1.0	3.0
sea surface	1.0	3.0	2.0	1.0	3.0	2.0
Average Rank	**1.57**	**1.71**	**2.71**	**1.71**	**1.43**	**2.86**
Uniform White Noise						
iris	2.0	3.0	1.0	2.0	3.0	1.0
wine	2.0	3.0	1.0	2.0	3.0	1.0
breast cancer	2.0	3.0	1.0	2.0	3.0	1.0
wave	2.0	3.0	1.0	2.0	3.0	1.0
mnist 0	2.0	3.0	1.0	2.0	3.0	1.0
mnist 1	2.0	3.0	1.0	2.0	3.0	1.0
sea surface	2.0	3.0	1.0	2.0	3.0	1.0
Average Rank	**2.0**	**3.0**	**1.0**	**2.0**	**3.0**	**1.0**
Gaussian White Noise						
iris	2.0	3.0	1.0	2.0	3.0	1.0
wine	2.0	3.0	1.0	2.0	3.0	1.0
breast cancer	2.0	3.0	1.0	2.0	3.0	1.0
wave	2.0	3.0	1.0	2.0	3.0	1.0
mnist 0	2.0	3.0	1.0	2.0	3.0	1.0
mnist 1	2.0	3.0	1.0	2.0	3.0	1.0
sea surface	2.0	3.0	1.0	2.0	3.0	1.0
Average Rank	**2.0**	**3.0**	**1.0**	**2.0**	**3.0**	**1.0**
Outliers						
iris	1.0	2.0	3.0	1.0	2.0	3.0
wine	1.0	2.0	3.0	1.0	3.0	2.0
breast cancer	1.0	3.0	2.0	1.0	3.0	2.0
wave	1.0	3.0	2.0	1.0	3.0	2.0
mnist 0	1.0	3.0	2.0	1.0	3.0	2.0
mnist 1	3.0	2.0	1.0	1.0	3.0	2.0
sea surface	1.0	3.0	2.0	1.0	3.0	2.0
Average Rank	**1.28**	**3.0**	**2.57**	**1.0**	**2.86**	**2.14**

Chapter 7

Conclusion

In today's day and age, data science and machine learning are inescapable. They permeate the way we structure experiments, conduct data analysis, and provide actionable decisions. The adoption of machine-learning methods can be found in every field of science and technology, including health care, manufacturing, education, financial modeling, policing, and marketing[65]. Diverse machine learning methods have been developed to cover the needs of these various applications, and three main paradigms have emerged: supervised, reinforcement, and unsupervised learning. Neither of the two former paradigms can operate without extensive labeled data, and that is what the third paradigm aims to address. Specifically, unsupervised learning aims to discover the underlying trends and patterns of unlabeled data without human supervision through methods like cluster analysis, dimensionality reduction, and anomaly detection [24]. The basis of this dissertation is that working with real data means working with corrupt and noisy data, and machine learning algorithms and statistical methods require robustification in order to stabilize their performance and predictive power [14]. Our specific algorithmic innovations pertain to developing robust clustering algorithms, as well as noise-agnostic methods for dimensionality reduction.

We began in Chapter Two with a review of classical cluster analysis techniques, which aim to automatically and efficiently group objects that are similar and distinguish objects that are dissimilar based on some measure of similarity [24]. We focused on k-means clustering, due to its popularity, simplicity, and scalability [35]. We introduced the fundamental connection between k-means clustering and optimization and walked through different approaches for the minimization of the objective function. We showed that re-

gardless of solution method, k-means inherently struggles with a sensitivity to outliers.

The following chapter directly addressed k-means' shortcomings with respect to noise-resistance. Beginning with a review of existing approaches for the development of a robust k-means, Chapter Three culminated in the proposal of a method we call Robust Trimmed k-means (RTKM), based on a relaxed formulation for the weighted k-means algorithm. RTKM provides a framework for the simultaneous identification of outliers and point clustering in a single objective function, and the algorithmic design gives the method flexibility to be applied to single- or multi-membership data. RTKM avoids the pitfalls of explicitly defining measures of "outlierness," and provides a way to search the model space more effectively to discover clusters and outliers. We tested RTKM on numerous real-world datasets and showed that RTKM performs competitively with methods that set the standard for single membership data with outliers and multi-membership data without outliers. We also showed that RTKM leverages its relative advantages to outperform these same methods on multi-membership data containing outliers.

We continued in Chapter Four by extending k-means to the spatiotemporal domain through a method we call Spatiotemporal k-means (STKM). Whereas spatial clustering refers to the analysis of static data with features that describe spatial location, spatiotemporal clustering adds time as a data feature, and algorithms have to consider both the spatial and the temporal neighbors of objects in order to extract useful knowledge [30]. We introduced the moving cluster problem, where clusters have a static identity, but their location and content can change over time, and reviewed prior work in this domain. We discussed the deficiencies of existing methods, and showed how our method, STKM, addresses them. STKM provides a two-phase approach for the analysis of the multi-scale relationships in spatiotemporal data. Phase 1 summarizes the short term interactions between moving objects and offers an objective function that is unified over space and time and involves the tuning of only two parameters, as opposed to methods that operate separately in space and time and are extremely sensitive to hyper-parameter tuning. Phase 2 can then be optionally applied to output long-term associations between objects. We evaluated STKM against baseline methods on a recently developed benchmark dataset and showed that STKM outperforms existing methods, particularly in the low-data domain, with significant performance improvements demonstrated for common evaluation metrics on the moving cluster problem. Further, we discussed the potential for a noise-resistant

version of STKM that replaces the Euclidean distance between points and their cluster centers in the objective function with a more robust distance metric.

Chapter Five saw us pivot from clustering methods to dimensionality reduction and offered a discussion as to why dimensionality reduction is essential for data processing, exploration, and human understanding. We reviewed the fundamental dimensionality reduction techniques of the Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). We provided geometric interpretations of these methods and demonstrated their use for both data compression and separation. Finally, we discussed their shortcomings, again with respect to performance in the presence of outliers and noise.

Showcasing the last of our contributions, Chapter Six introduced Ensemble Principal Component Analysis (EPCA) as a scalable, noise-resistant extension of PCA that ensembles boostrapped PCA with k-means clustering and lends itself naturally to uncertainty quantification. Drawing inspiration from previous work that utilized bootstrapping for PCA uncertainty quantification and the numerous existing robust PCA extensions, we proposed boostrapping as a method for robust data analysis. EPCA's use of k-means clustering allows it to automatically handle the challenges of principal component reordering and sign ambiguity that arise from bootstrapping. In addition, EPCA involves the tuning of only two parameters and boasts a runtime on the same scale as classical PCA, an attractive alternative to much more computationally intensive robust PCA approaches. We tested EPCA against its competitors on data with four types of corruption, and demonstrated that EPCA performs competitively on all noise domains, with a particular advantage on outlier corrupt data.

Overall, we have provided a comprehensive review of unsupervised learning techniques in the context of noisy data, explored their limitations, and have offered our own contributions to the field. We introduced three methods: Robust Trimmed k-means for flexible, outlier-resistant clustering, Spatiotemporal k-means (STKM) for summarizing the multiscale relationships in moving object data, and Ensemble Principal Component Analysis (EPCA) for scalable, noise-resistant, and explainable dimensionality reduction. It is inarguable that the development of robust machine learning techniques is essential for forward progress. We hope that our innovations have contributed to that cause.

Acknowledgements

I am very grateful to my advisors Nathan Kutz and Aleksandr Aravkin for their guidance and continued support. I would also like to thank the remaining members of my committee, Thomas Trogdon and Sam Burden, for overseeing my defense. Finally, I extend my deepest gratitude to my parents Iwona and Wojciech, my brother Tomek, my partner Tom, and my friends and peers, who have taken time to provide feedback on my work. (mic drop)

Bibliography

[1] Milton Friedman. "The use of ranks to avoid the assumption of normality implicit in the analysis of variance". In: *Journal of the american statistical association* 32.200 (1937), pp. 675–701.

[2] Hugo Steinhaus. "Sur la division des corps matériels en parties". In: *Bull. Acad. Polon. Sci* 1.804 (1956), p. 801.

[3] Edward W Forgy. "Cluster analysis of multivariate data: efficiency versus interpretability of classifications". In: *biometrics* 21 (1965), pp. 768–769.

[4] Michael JD Powell. "On search directions for minimization algorithms". In: *Mathematical programming* 4.1 (1973), pp. 193–201.

[5] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22.

[6] James C Bezdek. "A convergence theorem for the fuzzy ISODATA clustering algorithms". In: *IEEE transactions on pattern analysis and machine intelligence* 1 (1980), pp. 1–8.

[7] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[8] Olvi L Mangasarian and William H Wolberg. *Cancer diagnosis via linear programming*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 1990.

[9] Luis Milan and Joe Whittaker. "Application of the parametric bootstrap to models that incorporate a singular value decomposition". In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 44.1 (1995), pp. 31–49.

[10] Juan Antonio Cuesta-Albertos, Alfonso Gordaliza, Carlos Matrán, et al. "Trimmed k-means: An attempt to robustify quantizers". In: *The Annals of Statistics* 25.2 (1997), pp. 553–576.

[11] Sebastian Mika et al. "Kernel PCA and de-noising in feature spaces". In: *Advances in neural information processing systems* 11 (1998).

[12] Michael E Tipping and Christopher M Bishop. "Probabilistic principal component analysis". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3 (1999), pp. 611–622.

[13] Dan Pelleg, Andrew W Moore, et al. "X-means: Extending k-means with efficient estimation of the number of clusters." In: *Icml*. Vol. 1. 2000, pp. 727–734.

[14] Peter J Huber. "John W. Tukey's contributions to robust statistics". In: *Annals of statistics* (2002), pp. 1640–1648.

[15] Andrew Y Ng, Michael I Jordan, and Yair Weiss. "On spectral clustering: Analysis and an algorithm". In: *Advances in neural information processing systems*. 2002, pp. 849–856.

[16] Vin Silva and Joshua Tenenbaum. "Global versus local methods in nonlinear dimensionality reduction". In: *Advances in neural information processing systems* 15 (2002).

[17] Zengyou He, Xiaofei Xu, and Shengchun Deng. "Discovering cluster-based local outliers". In: *Pattern Recognition Letters* 24.9-10 (2003), pp. 1641–1650.

[18] S Van Aelst and G Willems. "PCA based on multivariate MM-estimators with fast and robust bootstrap". In: *Preprint. MR2085872* (2004).

[19] Arindam Banerjee et al. "Model-based overlapping clustering". In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005, pp. 532–537.

[20] Ville Hautamäki et al. "Improving k-means by outlier removal". In: *Scandinavian Conference on Image Analysis*. Springer. 2005, pp. 978–987.

[21] Joshua Zhexue Huang et al. "Automated variable weighting in k-means type clustering". In: *IEEE transactions on pattern analysis and machine intelligence* 27.5 (2005), pp. 657–668.

[22] Ian Jolliffe. “Principal component analysis”. In: *Encyclopedia of statistics in behavioral science* (2005).

[23] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. “On discovering moving clusters in spatio-temporal data”. In: *International symposium on spatial and temporal databases*. Springer. 2005, pp. 364–381.

[24] Oded Maimon and Lior Rokach. “Data mining and knowledge discovery handbook”. In: (2005).

[25] David Arthur and Sergei Vassilvitskii. “How slow is the k-means method?” In: *Proceedings of the twenty-second annual symposium on Computational geometry*. 2006, pp. 144–153.

[26] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[27] Janez Demšar. “Statistical comparisons of classifiers over multiple data sets”. In: *The Journal of Machine Learning Research* 7 (2006), pp. 1–30.

[28] Bogdan Gabrys, Bruno Baruque, and Emilio Corchado. “Outlier resistant PCA ensembles”. In: *Knowledge-Based Intelligent Information and Engineering Systems: 10th International Conference, KES 2006, Bournemouth, UK, October 9-11, 2006. Proceedings, Part III 10*. Springer. 2006, pp. 432–440.

[29] Hui Zou, Trevor Hastie, and Robert Tibshirani. “Sparse principal component analysis”. In: *Journal of computational and graphical statistics* 15.2 (2006), pp. 265–286.

[30] Derya Birant and Alp Kut. “ST-DBSCAN: An algorithm for clustering spatial–temporal data”. In: *Data & knowledge engineering* 60.1 (2007), pp. 208–221.

[31] Marieke E Timmerman, Henk AL Kiers, and Age K Smilde. “Estimating confidence intervals for principal component loadings: a comparison between the bootstrap and asymptotic results”. In: *British Journal of Mathematical and Statistical Psychology* 60.2 (2007), pp. 295–314.

[32] Guillaume Cleuziou. “An extended version of the k-means method for overlapping clustering”. In: *2008 19th International Conference on Pattern Recognition*. IEEE. 2008, pp. 1–4.

[33] Hoyoung Jeung, Heng Tao Shen, and Xiaofang Zhou. "Convoy queries in spatio-temporal databases". In: *2008 IEEE 24th International Conference on Data Engineering*. IEEE. 2008, pp. 1457–1459.

[34] Sheng-yi Jiang and Qing-bo An. "Clustering-based outlier detection method". In: *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*. Vol. 2. IEEE. 2008, pp. 429–433.

[35] Xindong Wu et al. "Top 10 algorithms in data mining". In: *Knowledge and information systems* 14.1 (2008), pp. 1–37.

[36] Zhonghua Zhao et al. "G-means: a clustering algorithm for intrusion detection". In: *International Conference on Neural Information Processing*. Springer. 2008, pp. 563–570.

[37] Marcos R Vieira, Petko Bakalov, and Vassilis J Tsotras. "On-line discovery of flock patterns in spatio-temporal data". In: *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. 2009, pp. 286–295.

[38] Ke Zhang, Marcus Hutter, and Huidong Jin. "A new local distance-based outlier detection approach for scattered real-world data". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2009, pp. 813–822.

[39] Hédy Attouch et al. "Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality". In: *Mathematics of operations research* 35.2 (2010), pp. 438–457.

[40] Zhenhui Li et al. *Swarm: Mining relaxed temporal moving object clusters*. Tech. rep. ILLINOIS UNIV AT URBANA-CHAMPAIGN DEPT OF COMPUTER SCIENCE, 2010.

[41] Zhouchen Lin, Minming Chen, and Yi Ma. "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices". In: *arXiv preprint arXiv:1009.5055* (2010).

[42] Zihan Zhou et al. "Stable principal component pursuit". In: *2010 IEEE international symposium on information theory*. IEEE. 2010, pp. 1518–1522.

[43] Emmanuel J Candès et al. "Robust principal component analysis?" In: *Journal of the ACM (JACM)* 58.3 (2011), pp. 1–37.

[44] Xinghao Ding, Lihan He, and Lawrence Carin. "Bayesian robust principal component analysis". In: *IEEE Transactions on Image Processing* 20.12 (2011), pp. 3419–3430.

[45] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[46] Grigorios Tsoumakas et al. "Mulan: A Java Library for Multi-Label Learning". In: *Journal of Machine Learning Research* 12 (2011), pp. 2411–2414.

[47] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[48] Hesam Izakian, Witold Pedrycz, and Iqbal Jamal. "Clustering spatiotemporal data: An augmented fuzzy C-means". In: *IEEE transactions on fuzzy systems* 21.5 (2012), pp. 855–868.

[49] Julie Josse and François Husson. "Handling missing values in exploratory multivariate data analysis methods". In: *Journal de la Société Française de Statistique* 153.2 (2012), pp. 79–99.

[50] Haibing Lu et al. "Overlapping clustering with sparseness constraints". In: *2012 IEEE 12th International Conference on Data Mining Workshops*. IEEE. 2012, pp. 486–494.

[51] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. "The planar k-means problem is NP-hard". In: *Theoretical Computer Science* 442 (2012), pp. 13–21.

[52] NASA Earth Observations. *Sea Surface Temperature*. 2012. URL: `https://psl.noaa.gov/repository/entry/show?entryid=4abf55c8-335b-4117-b595-d7ce3d242f4f`.

[53] Iulian Peca et al. "Scalable Cluster Analysis of Spatial Events." In: *EuroVA@ EuroVis* 6 (2012), pp. 19–23.

[54] Hadi Fanaee Tork. "Spatio-temporal clustering methods classification". In: *Doctoral symposium on informatics engineering*. Vol. 1. Faculdade de Engenharia da Universidade do Porto Porto, Portugal. 2012, pp. 199–209.

[55] Miin-Shen Yang, Chien-Yo Lai, and Chih-Ying Lin. "A robust EM clustering algorithm for Gaussian mixture models". In: *Pattern Recognition* 45.11 (2012), pp. 3950–3961.

[56] Mohiuddin Ahmed and Abdun Naser Mahmood. "A novel approach for outlier detection and clustering improvement". In: *2013 IEEE 8th Conference on Industrial Electronics and Applications (iciea)*. IEEE. 2013, pp. 577–582.

[57] Hamid Babamoradi, Frans van den Berg, and Åsmund Rinnan. "Bootstrap based confidence limits in principal component analysis—A case study". In: *Chemometrics and Intelligent Laboratory Systems* 120 (2013), pp. 97–105.

[58] Sanjay Chawla and Aristides Gionis. "k-means–: A unified approach to clustering and outlier detection". In: *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM. 2013, pp. 189–197.

[59] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[60] Chiheb-Eddine ben N'Cir, Guillaume Cleuziou, and Nadia Essoussi. "Identification of non-disjoint clusters with small and parameterizable overlaps". In: *2013 International Conference on Computer Applications Technology (ICCAT)*. IEEE. 2013, pp. 1–6.

[61] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. "Proximal alternating linearized minimization for nonconvex and nonsmooth problems". In: *Mathematical Programming* 146.1-2 (2014), pp. 459–494.

[62] Thierry Bouwmans and El Hadi Zahzah. "Robust PCA via principal component pursuit: A review for a comparative evaluation in video surveillance". In: *Computer Vision and Image Understanding* 122 (2014), pp. 22–34.

[63] Xi Chen et al. "Clustering dynamic spatio-temporal patterns in the presence of noise and missing data". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.

[64] John P Cunningham and Zoubin Ghahramani. "Linear dimensionality reduction: Survey, insights, and generalizations". In: *The Journal of Machine Learning Research* 16.1 (2015), pp. 2859–2900.

[65] Michael I Jordan and Tom M Mitchell. "Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245 (2015), pp. 255–260.

[66] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[67] Joyce Jiyoung Whang, Inderjit S Dhillon, and David F Gleich. "Non-exhaustive, overlapping k-means". In: *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM. 2015, pp. 936–944.

[68] KP Agrawal et al. "Development and validation of OPTICS based spatio-temporal clustering technique". In: *Information Sciences* 369 (2016), pp. 388–401.

[69] Aaron Fisher et al. "Fast, exact bootstrap principal component analysis for p¿ 1 million". In: *Journal of the American Statistical Association* 111.514 (2016), pp. 846–860.

[70] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[71] Noureddine El Karoui and Elizabeth Purdom. "The bootstrap, covariance matrices and PCA in moderate and high-dimensions". In: *arXiv preprint arXiv:1608.00948* (2016).

[72] David Donoho. "50 years of data science". In: *Journal of Computational and Graphical Statistics* 26.4 (2017), pp. 745–766.

[73] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: `http://archive.ics.uci.edu/ml`.

[74] Guojun Gan and Michael Kwok-Po Ng. "K-means clustering with outlier removal". In: *Pattern Recognition Letters* 90 (2017), pp. 8–14.

[75] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[76] Peng Zheng et al. "A unified framework for sparse relaxed regularized regression: Sr3". In: *IEEE Access* 7 (2018), pp. 1404–1423.

[77] Ziwei Zhu, Tengyao Wang, and Richard J Samworth. "High-dimensional principal component analysis with heterogeneous missingness". In: *arXiv preprint arXiv:1906.12125* (2019).

[78] Mohd Yousuf Ansari et al. “Spatiotemporal clustering: a review”. In: *Artificial Intelligence Review* 53.4 (2020), pp. 2381–2423.

[79] Kathleen Champion et al. “A unified sparse optimization framework to learn parsimonious physics-informed models from data”. In: *IEEE Access* 8 (2020), pp. 169259–169271.

[80] Marc Hüsch, Bruno U Schyska, and Lueder von Bremen. “CorClustST—Correlation-based clustering of big spatio-temporal datasets”. In: *Future Generation Computer Systems* 110 (2020), pp. 610–619.

[81] Panthadeep Bhattacharjee and Pinaki Mitra. “A survey of density based clustering algorithms”. In: *Frontiers of Computer Science* 15.1 (2021), pp. 1–27.

[82] Eren Cakmak et al. “Spatio-temporal clustering benchmark for collective animal behavior”. In: *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Animal Movement Ecology and Human Mobility*. 2021, pp. 5–8.

[83] Lloyd N Trefethen and David Bau. *Numerical linear algebra*. Vol. 181. Siam, 2022.